Data Integration with Constraint-based Genome-scale Models

A Thesis
Presented to
The faculty of the School of Engineering and Applied Science
University of Virginia

In Partial Fulfillment
of the requirements for the Degree
Doctor of Philosophy

by
Paul Anthony Jensen
December 2013

The thesis has been read and approved by the examining committee:

Jason Papin, Ph.D., Advisor
Kevin Janes, Ph.D.
Jeffrey Saucerman, Ph.D.
Daniel Burke, Ph.D.
Stephen Patek, Ph.D.

Accepted for the School of Engineering and Applied Science by

James H. Aylor, Dean
December 2013

Contents

List of Figures

List of Tables

Abstract

Genome-scale modeling is a powerful tool for quantifying relationships between genetic, metabolic, and phenotypic factors. The mechanistic detail of these models presents unique opportunities in metabolic engineering, drug discovery, and disease comprehension. Despite recent advances, several challenges remain for the genome-scale modeling field. Existing models focus almost exclusively on metabolism, and few studies integrate other biochemical systems. Models describe all functionality encoded in an organism's genome, and tailoring models to specific genetic states is a difficult, highly specialized process. Large-scale models require equally large datasets for validation, yet few comprehensive datasets exist (especially for lesser-studied organisms).

This thesis focuses on developing methods to overcome the limitations of current genome-scale modeling techniques. Herein, I describe novel methods to

- assemble, visualize, and simulate large models
- design instrumentation to rapidly produce genome-scale datasets for model validation
- improve algorithms to leverage high-throughput expression data and refine models for a particular condition

Together, these advances provide a framework for contextualizing high-throughput data with genome-scale modeling. Special attention is given to the interpretation of functional genomics data through the enzymatic architecture underlying the metabolic network.

Acknowledgements

This work was possible thanks to the support of many mentors, colleagues, family, and friends. I am deeply indebted to

- My advisor Jason Papin for his unwavering professional and personal support.
- The members of the Papin lab, especially Jennie Bartell, Kevin D'Auria, Matt Biggs, Anna Blazier, and Bonnie Dougherty.
- My thesis committee – Kevin Janes, Jeff Saucerman, Steve Patek, and Dan Burke – for their time, advice, and patience.
- Cheryl Borgman for knowing anything I don't.
- Brian Helmke for his mentoring with teaching.
- Several undergraduate students for their help in the lab, especially Tom Moutinho and Kyla Lutz.
- My family and friends for their support during my education.
- My wife Karin, for everything.

Chapter 1

Introduction

1.1 Metabolic modeling

Stanley Falkow famously quipped that "the goal of every bacterium is to become bacteria". Metabolism lies at the center of microbe's struggle to produce biomass for the succeeding generation from the chemicals available in the environment. Intense study over the past century has made metabolism arguably the most well characterized cellular subsystem.

Metabolic researchers have historically been early adopters of quantitative techniques. The development of the Michaelis-Menten model of enzyme kinetics in 1913 [1] was among a series of seminal developments in mathematical enzymology. Biochemists applied mechanistic models to examine metabolic reactions largely in isolation for the next fifty years [2]. A 1963 paper by Higgins marks a transition from single enzyme to pathway models as focus shifted from reactions to small reaction networks [3]. This transition was given a strong theoretical footing with the seminal development of metabolic control analysis (MCA) by Kacser and Burns in 1973 [4] and Heinrich and Rapoport in 1974 [5]. For the following two decades, the focus of metabolic modeling remained largely unchanged – MCA models of individual pathways were built and analyzed using kinetic data from *in vitro* enzyme assays [6].

Kinetic modeling's dominance over metabolic research during much of the 20th century reflects the technological limitations of era. Enzyme assays were well established and widely accepted, while the fields of genomics and proteomics were in their infancy. For the last time in history, it was easier for scientists to characterize the kinetics of a known enzyme than to discover a new one.

The genome sequencing revolution upset the biological information balance. Researchers suddenly had access to the complete "parts list" of the bacteria cell. The race to assemble a complete model of metabolism was on, but a new mathematical framework was necessary. The genome of the most well-studied bacteria, *Escherichia coli*, contained hundreds of enzymes with completely uncharacterized kinetics. A stoichiometric model could be assembled from a list of reactions and genome annotations, but the paucity of kinetic details prevented a genome-wide extension to MCA. Instead, researchers turned to constraint-based framework designed by Fell and Small [7]. While kinetic models use rate laws to drive the conversion of reactants to products, constraint-based models use reactions to define a space of feasible transformations across the entire network [8]. The development of this framework along with the parallel assembly of *in silico* metabolic networks became the field of constraint-based reconstruction and analysis, or COBRA, modeling. From only reaction stoichiometry and basic thermodynamic constraints, COBRA models can predict growth phenotypes in a range of environmental and genotypic states [9].

COBRA models are structural models, derived entirely from the list of genes encoded by an organism's genome [10]. The emergent phenomena of COBRA models are not derived from complex dynamics or nonlinearities, but instead from the model's scale. While MCA provides a detailed model of a small system, COBRA methods rely on a simpler model of a large system. Both models are mechanistic, albeit with different resolutions.

Despite the successful application of COBRA models to many fields [11], the current framework is being surpassed by technological advances. Functional genomics provides insight on the dynamics of gene and protein expression. The regulatory structure of enzymes, pathways, and metabolic flux is encoded in a cell's transcriptome and proteome; decoding this structure requires a systematic approach to reconciling data with our prior knowledge of the network. A genome-scale model could provide the necessary scaffold for interpreting high-throughput data. However, the structure of COBRA models ignores or oversimplifies many of the dynamic features of the enzymatic network. Overcoming these limitations is necessary for COBRA modeling to meet the demands of the functional genomics revolution.

In this thesis, I present several extensions to the COBRA framework with a focus on data integration. I also describe a suite of software tools to improve model construction, visualization, and validation. The overall goal of my work is to improve the correspondence between the mathematical representation of a metabolic network and biological system it describes.

1.2 Formalism

1.2.1 Stoichiometric mass balances

A stoichiometric model describes a set of r reactions that transform m metabolites. For COBRA models, a metabolite is defined by both a chemical species and its localization. Extracellular and intracellular glucose, for example, are represented by distinct metabolites in the model. The independent variables in stoichiometric models are the fluxes v_i through each reaction i.

The reaction-centered formalism for COBRA models differs from kinetic modeling frameworks. Kinetic models consider the concentrations of metabolites as independent variables. The fluxes in kinetic models are computed from the metabolite concentrations and a parameterized rate law for each reaction. COBRA models compute fluxes directly without any kinetic parameters. The results are not dependent on, and contain little information about, the metabolite concentrations.

At steady state, the pool of each metabolite remains at a constant level. The constraint implies that the net production of a metabolite per unit time must be matched by the net consumption. If a metabolite is consumed by (is a reactant of) a set C of reactions and produced by (is a product of) a set P of other reactions, the steady-state assumption is written

$$\sum_{p \in P} a_p v_p = \sum_{c \in C} a_c v_p \tag{1.1}$$

where a_i is the stoichiometric coefficient of the metabolite in reaction i and v_i is the steady state flux through reaction i. Equation (1.1) describes a linear mass balance constraint on a single metabolite. We can collect these constraints for all m metabolites and r reactions in a single matrix equation

$$Sv = 0 \tag{1.2}$$

where v is a vector of reaction fluxes and S is an $m \times r$ stoichiometic matrix. The magnitude of each element s_{ij} of S is the stoichiometric coefficient of metabolite i in reaction j. If metabolite i is a product of reaction j, then $s_{ij} > 0$. If metabolite i is consumed by reaction j, then $s_{ij} < 0$. (If metabolite i is does not participate in reaction j, then $s_{ij} = 0$.)

Any flux distribution v satisfying (1.2) is mass balanced. However, not all flux distributions are biologically feasible. Some reactions may be irreversible, requiring the corresponding elements of v to be nonnegative. Other thermodynamic and biochemical constraints may limit the upper and lower bounds of v to a physiologically reasonable range. We can represent the reversibility and feasibility constraints by requiring that v be bounded by two constant vectors l and u with the equation

$$l \leq v \leq u \tag{1.3}$$

It is important to note that the addition of (1.3), or any other constraints, to (1.2) does not violate the mass balance, since any v that satisfies (1.2) and (1.3) by definition satisfies (1.2), and is therefore mass balanced.

The stoichiometric constraints in (1.2) prevent the accumulation or depletion of any metabolites in the model. However, a microbe's metabolism is continuously consuming extracellular nutrients and producing biomass. We need a way to allow metabolites to enter and exit the model without violating equation (1.2). The most common approach is to include *exchange reactions* in the model.[1] An exchange reaction has a single reactant and no products. By convention, a positive flux through an exchange reaction removes the metabolite from the model. A negative flux through an exchange reaction adds the metabolite to the model.

1.2.2 The objective function and Flux Balance Analysis

In practice, the number of metabolites (equations) is less than the number of reactions (variables) so the system of equations in (1.2) and (1.3) is underdetermined.[2] Often there exist infinitely many flux vectors that satisfy (1.2) and (1.3) [12]. We must have a procedure for selecting a physiologically relevant flux distribution from the model's solution space. COBRA models use a biological *objective function* to select flux distributions [13]. The metabolic networks of many micro-organisms have evolved to maximize of yield of biomass production given constraints on metabolite availability [14].

COBRA models define an objective reaction that produces a unit of biomass by consuming a weighted sum of biomass precursors (ATP, amino acids, lipids, carbohydrates, etc.). Choosing a feasible flux distribution in the metabolic network is accomplished by optimizing the flux through the objective reaction (v_{obj}) subject to mass balance and feasibility constraints.[3] The optimization is expressed as the following linear program

$$\begin{aligned} \max \quad & v_{\text{obj}} \\ \text{subject to} \quad & Sv = 0 \\ & l \leq v \leq u \end{aligned} \tag{1.4}$$

[1]An alternative approach to augmenting the mass balance is to convert equation (1.2) to an inhomogeneous system $Sv = b$. If element $b_i < 0$, then $|b_i|$ units of metabolite i must be removed from the system. Conversely, if $b_i > 0$, then b_i units must be added. Fixing the metabolites in this manner requires that all exchanges match exactly the prescribed values in b. Often only a subset of the exchange rates are known, and those that are known have been measured with uncertainty. For these reasons, the exchange reaction approach is favorable, since the exchange reactions can be confined to a range of values by equation (1.3).

[2]Sometimes the number of metabolites m in a model is greater than the number of reactions r. However, almost always $\text{rank}(S) < r$, so the system is underdetermined. Adding the constraints in equation (1.3) does not fully determine the system, since these are inequality constraints.

[3]The biomass reaction in many COBRA models will produce a biomass pseudo-metabolite. In these cases, the exchange reaction removing the biomass is used as the objective function. This abstraction allows models to several biomass reactions with varying composition. The biomass reaction for a particular simulation can be selected by setting appropriate bounds l and u without changing the mathematical objective function.

Finding solutions to problem (1.4) is known as *Flux Balance Analysis* (FBA), the central tool of COBRA modeling [15]. Any flux vector that solves (1.4) is known as a *flux distribution*. The flux through the objective reaction, or the *objective value* (v^*_{obj}), is the key result of an FBA analysis. Strictly interpreted, the objective value is the yield of biomass relative to the input fluxes. However, numerous studies have demonstrated that the objective value is also proportional to the growth rate of during exponential phase for many microbes [16].[4] Thus, it is common to refer to the objective value as the *growth rate*, although this nomenclature should be reserved for exponentially-dividing microbes.

Most algorithms for analyzing COBRA models (including the methods developed in this thesis) are extensions to the FBA problem [18]. It is convenient to define the following set of constraints that yield flux distributions that are nearly optimal for the FBA problem

$$\text{nearFBA}(\epsilon) \equiv \begin{cases} Sv = 0 \\ l \leq v \leq u \\ v_{\text{obj}} \geq \epsilon\, v^*_{\text{obj}} \end{cases} \tag{1.5}$$

If a flux distribution satisfies the nearFBA set of constraints, then the flux through the objective reaction in this distribution must be within ϵ of the objective value v^*_{obj} of the FBA problem. Algorithms that include the nearFBA constraints require that any feasible solution allows the model to produce nearly optimal yields of biomass, where “nearly” is quantified by the parameter ϵ.

1.2.3 Degeneracy and Flux Variability Analysis

A stoichiometric matrix and reaction bounds will produce a unique objective flux through the FBA algorithm. By contrast, infinitely many flux distributions can produce the same objective flux for the same FBA problem [12]. Said another way, the scalar objective value v^*_{obj} is unique for an FBA problem, but the flux vector v is not.

To understand why multiple flux distributions exist for a single FBA problem, consider the small reaction network in Figure 1.1. After transport into the system, metabolite A is converted to C, the sole component of biomass. Removal of C is therefore the cellular objective. Metabolite B is produced by either of two independent pathways: 1.) direct conversion of A to B (reaction R1), or 2.) conversion of A to an intermediate, I, and subsequent conversion of I to B (reactions R2 and R3). If we restrict the input of A to 10 flux units and solve the FBA problem, we find that the optimal objective value is 10 flux units (every unit of A is converted to C and removed). This objective value is unique, while the full flux distribution in the network is not. Without thermodynamic or kinetic details, we cannot specify the ratio of fluxes in the two pathways that produce B. The software used to solve the FBA problem will arbitrarily choose a flux distribution. This flux distribution is based on the numerical, not biological, structure of the network. Erroneously assigning biological significance to exact values in an FBA-derived flux distribution is the most common abuse of the modeling framework.

The flux distribution from an FBA problem is degenerate as a whole. However, the model's constraints may entirely or partially specify the fluxes through individual reactions. In the example network shown in Figure 1.1, the flux through reaction R4 is always 10 flux units in any FBA solution. Any solution with less flux through R4 would have a lower flux through the objective reaction and would not be an optimal solution. Unlike reactions R1, R2, and R3, the value of R4

[4]Notable exceptions to these data are fermenting yeast. However, the correspondence between yield and growth rate appears to hold in models of yeast when oxygen uptake is constrained [17].

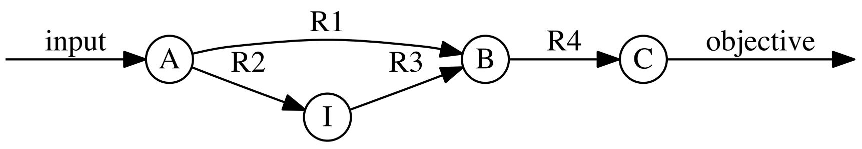


Figure 1.1: Sample network with degenerate flux distributions.

in a flux distribution does offer direct biological insight. We have no certainty in the flux values for the former reactions, and we know exactly the relative flux through R4.

To quantify the degeneracy of each reaction in a COBRA model, we can compute the minimum and maximum allowable flux through each reaction in every flux distribution that satisfies FBA optimality. Rather than examine the entire (potentially infinite) set of optimal flux distributions, we can cast the range-finding problem as an optimization. For each reaction i, the minimum flux v_i in any optimal solution is found by minimizing v_i while requiring the entire flux distribution is optimal. The optimality requirement can be enforced with the nearFBA constraints (equation (1.5)), allowing us to compute the flux range with the following algorithm:

$$\text{range}(v_i) = [\min v_i, \max v_i] \text{ subject to nearFBA}(1) \tag{1.6}$$

The range of allowable fluxes for a set of network constraints is termed the reaction's *flux variability*. The algorithm in (1.6) is called Flux Variability Analysis (FVA) [19]. A reaction's flux variability may correlate with essentiality, network centrality, and druggability [20]. In our example network (Figure 1.1), inhibiting the enzyme catalyzing reaction R4 may be an effective strategy to reduce growth, since any deviation in R4's flux would necessarily decrease the objective flux. By contrast, singly targeting any of reactions R1, R2, or R3 may not have an effect on growth, since optimal flux distributions may exist with low flux through these reactions.

1.2.4 Gene associations

Reactions and fluxes do not exist as physical entities in cells; they cannot be directly manipulated through experimentation. Instead, only metabolites and the enzymes that transform them can be perturbed to study or engineer biological systems. Our description of COBRA models has neglected the important mapping between genes, the enzymes they encode, and the reactions they catalyze. These relationships are essential for translating between a model's flux predictions and a cell's genetic state.

Enzyme-catalyzed reactions require a set of gene products for any appreciable transformation. The mapping between genes and reactions is not one-to-one (Figure 1.2). Isozymes can independently catalyze reactions. Other enzymes require multiple protein subunits to form a functional enzymatic complex.

Any reaction associated with more than one gene product is said to have a *complex* GPR association. Complex associations are common for metabolic reactions. A COBRA reconstruction for the yeast *Saccharomyces cerevisiae* [17] contains 810 enzyme-catalyzed metabolic reactions; 231 (28.5%) of these reactions have complex GPR associations. The most complex GPR in this

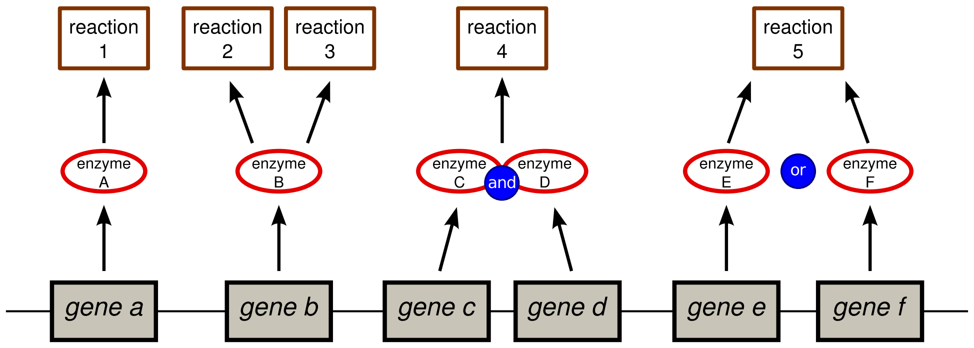


Figure 1.2: Gene-protein-reaction relationships are not one-to-one. Reaction 1 is catalyzed by enzyme A. Enzyme B is “promiscuous”, catalyzing either reaction 2 or reaction 3. Both enzymes C and D are required for catalysis of reaction 4. Reaction 5 can be catalyzed independently by either enzyme E or enzyme F. The GPR associations for reactions 4 and 5 are complex by definition.

model involves the products of 18 open reading frames. The entire set of GPR rules contains 340 instances of isozyme-like behavior (two proteins both able to fully catalyze a reaction) and 279 different complexes of protein subunits. The nonlinear mapping between genes and reactions adds complexity to the metabolic network. Incorporating GPR relationships into COBRA simulations reveals the interplay between genetic states and cellular objectives.

To formalize the GPR associations, each reaction in COBRA model includes a logical GPR *rule*. The rule is a Boolean expression stating the gene product(s) necessary to catalyze the reaction. In COBRA models, each gene is modeled by a binary variable g_k. If $g_k = 1$, the gene is expressed (“on”), and the gene product is available to catalyze the associated reactions. If $g_k = 0$, the gene is “off”, and no enzyme is available. For standard FBA analyses, genes can occupy only these two states (on/off). There are no intermediate states denoting levels of transcription or translation. If a gene is expressed, enough enzyme is available to catalyze any feasible flux satisfying the FBA constraints.[5]

Each reaction’s GPR rule is a binary function $R_j(g)$ that combines the genes g with logical operators and and or. For example, the rule $R_j(g) = g_a \textsf{ and } g_b$ indicates that genes a and b encode protein subunits. The rule $R_j(g) = g_a \textsf{ or } g_b$ identifies genes a and b as isozymes.

FBA simulations on COBRA models with gene associations follow a two-step procedure. The process begins with a genetic state describing the subset of genes in the model that are expressed for a particular set of model parameters (environment, mutations, etc.). In the first step, the rules for each reaction are evaluated in the context of the genetic state. A reaction is removed from the model if its rule is not satisfied.[6] In the second phase, the “sub-model” containing all reactions with satisfied rules is solved by FBA.

[5]Several methods have been proposed to use the continuous reaction bounds to simulate varying levels of gene expression. The methods to date have not allowed the full complexity of the GPR associations to be included in a model. I will present a formalism to overcome these limitations in Chapter 5.

[6]In practice, if $R_j(g) = 0$, the flux bounds are set as $l_j = u_j = 0$. This restriction has the same effect as removing the reaction from the stoichiometric matrix. Most linear programming solvers will remove these zero-bounded variables from the model during preprocessing.

1.3 Outline

Separating GPR evaluation and flux optimization into two sequential stages prevents algorithms from fully interrogating the relationship between the enzymatic network and the reaction network. A framework exists to consolidate the GPR logic and reaction stoichiometry into a single optimization problem [21]. In Chapter 2, I generalize this framework to support a wide class of regulatory interactions. I also present a software package to streamline the development of integrated metabolic-regulatory models.

The accuracy of COBRA models has recently been improved through the integration of high-throughput expression data. However, extensions of FBA require that such data be discretized *a priori* into sets of genes or proteins that are either "on" or "off". This procedure requires selecting relatively subjective expression thresholds, often requiring several iterations and refinements to capture the expression dynamics and retain model functionality. In Chapter 3, I present a method for mapping expression data from a set of environmental, genetic or temporal conditions onto a metabolic network model without the need for arbitrary expression thresholds. Metabolic Adjustment by Differential Expression (MADE) uses the statistical significance of changes in gene or protein expression to create a functional metabolic model that most accurately recapitulates the expression dynamics.

Metabolic reaction maps allow visualization of genome-scale models and high-throughput data in a format familiar to many biologists. However, creating a map of a large metabolic model is a difficult and time consuming process. In Chapter 4, I describe a software package, MetDraw, to fully automate the map drawing process for metabolic models containing hundreds to thousands of reactions. MetDraw also overlays high-throughput "omics" data directly on the generated maps.

Approximating genes as binary variables limits COBRA models to simulations of complete gene knockouts. Examining the effects of more subtle changes in gene expression and enzyme activity requires a new framework wherein continuous variations in the enzymatic networks are coupled to the reaction network. Such a framework is developed in Chapter 5. The formalism I present preserves the nonlinear mapping between genes and reactions while retaining the computation efficiency of linear programming. I also demonstrate how this new framework improves correspondence between COBRA model predictions and experimental data.

All models depend on experimental data for validation. Genome-scale models require high-throughput, multi-dimensional datasets with thousands of independent observations. To complement my computational work, I present a novel device for phenotypic screening in Chapter 6. This chapter is self-contained and may be read independently.

Chapter 2

Tools for integrating genome-scale models, expression data, and regulatory networks

2.1 Introduction

Because of the complexity of the gene-protein-reaction (GPR) mappings, early extensions to FBA were reaction, rather than gene, centric. For example, the OptKnock [22] algorithm removed reactions from a FBA model to design a strain of *E. coli* with optimal production of a metabolic byproduct. Ideally, OptKnock would operate by removing genes, not reactions, since it is not straightforward to independently remove reactions from a biological system without genetic manipulations. An optimization using genes as decision variables would require a method for encoding the GPR logic into a set of linear inequalities. This encoding was developed as SR-FBA [21] using a mixed integer optimization approach for GPR logic and other Boolean regulatory rules. An SR-FBA-based approach was later used to develop OptORF, a method to design microbial strains through gene knockouts and overexpression [23]. Other gene-centric, FBA-related algorithms have been developed, each using a variation of the SR-FBA method [24, 25, 26]. However, a general software platform for coupling GPR rules of arbitrary complexity with a COBRA model using mixed integer programming is not available. Such a tool would speed the development of new algorithms by removing the need for researchers to re-implement this complex process.

In this chapter, I will present a software toolbox to automate, simplify, and expand a mathematical framework for COBRA models with integrated genetic constraints. Rather than focusing only on GPR integration, I will develop techniques for integrating general, multi-level Boolean equations with FBA problems. This larger class of equations can be used to describe many molecular interactions, including a wide class of regulatory effects.

Parts of this chapter are adapted from: Jensen PA, Lutz KA, Papin JA. TIGER: Toolbox for integrating genome-scale metabolic models, expression data, and transcriptional regulatory networks. *BMC Systems Biology.* 2001, 5:147.

2.2 Transcriptional regulatory networks

The accuracy of COBRA models has been improved through the addition of transcriptional regulatory networks (TRNs) [27, 28]. The TRN is a set of rules that relate the expression states of metabolic genes to various genetic and environmental cues. Because of the paucity of kinetic details available to describe these relationships, genome-scale models also represent regulatory and environmental cues in a binary, "on" or "off" format. This approach allows TRNs to be described with Boolean logic.

The first genome-scale TRNs were applied to models of *Escherichia coli* [27] and *Saccharomyces cerevisiae* [28] metabolism. The rules were written in standard Boolean format, where each Boolean variable is given by an explicit function of the other variables. This method creates two significant problems. First, the TRN uses the absence or presence of metabolites in the extracellular environment to calculate which genes (and, subsequently, reactions) will be active. However, certain metabolic pathways secrete byproducts into the extracellular space, thereby changing the environment. Studies with the *E. coli* and *S. cerevisiae* TRNs used an iterative approach [29] – applying the TRN to the metabolic network in a starting environment, determining which metabolites would be secreted, and then repeating the process in the new environment until the environment no longer changes between iterations. A more straightforward approach would be to solve the TRN and metabolic networks simultaneously by formulating both problems in a single optimization.

A second obstacle with TRN integration is that the explicit rule formulation used by previous studies [29] can over-constrain the metabolic model. (In explicit rules, each gene's state can be calculated unambiguously from the state of all other genes and metabolites.) Consider the following subnetwork of the iMH805 TRN for *S. cerevisiae* [28]:

$$\begin{tikzcd} \mathit{mig1} \arrow[r, -|] & \mathit{mth1} \arrow[d] & \\ & \mathit{rgt1} & \text{gln-L} \arrow[l, -|] \end{tikzcd} \tag{2.1}$$

Transcription factor *mth1* is repressed by *mig1* and promotes expression of *rgt1*. Extracellular L-glutamine (gln-L) represses *rgt1* expression. The original iMH805 study required this set of constraints be described with the following set of explicit rules [28]:

$$\begin{aligned} \textsf{not}\ \text{MIG1} &\Leftrightarrow \mathit{mth1} \\ \text{MTH1}\ \textsf{and}\ (\textsf{not}\ \text{gln-L}) &\Leftrightarrow \mathit{rgt1} \end{aligned}$$

An implicit representation of (2.1) is

$$\begin{aligned} \text{MIG1} &\Rightarrow \textsf{not}\ \mathit{mth1} \\ \text{MTH1} &\Rightarrow \mathit{rgt1} \\ \text{gln-L} &\Rightarrow \textsf{not}\ \mathit{rgt1} \end{aligned}$$

The three transcription factors and one metabolite in these rules can be arranged in $2^4 = 16$ possible states. As shown in Table 2.2, only four of the sixteen states are feasible for the explicit rules. The implicit formulation of the same system allows four new states and makes one of the explicit states infeasible. This example illustrates that two mathematical descriptions of the same biological process can lead to distinct model predictions. The model developer should be free to choose the rule formulation that best encompasses the underlying biology. However, implicit rules require simultaneous solution with a metabolic model and are often more difficult to parse into a

State				Feasibility	
mig1	*mth1*	*rgt1*	gln-L	Explicit	Implicit
0	0	0	0		●
0	0	0	1		●
0	0	1	0		●
0	0	1	1		
0	1	0	0		
0	1	0	1	●	
0	1	1	0	●	●
0	1	1	1		
1	0	0	0	●	●
1	0	0	1	●	●
1	0	1	0		●
1	0	1	1		
1	1	0	0		
1	1	0	1		
1	1	1	0		
1	1	1	1		

Table 2.1: Feasible states for explicit and implicit rules describing the transcriptional regulatory network (2.1). Rules are taken from the *S. cerevisiae* regulatory network model iMH805 [28].

mixed integer linear program. As a result, previous TRN integration studies have relied solely on explicit rules to describe regulatory interactions [29]. A software platform that can correctly parse both explicit and implicit rules would ease the development of large TRN models.

2.3 Objectives

Software suites have been developed to enable COBRA analyses. Packages such as CellNetAnalyzer [30], the BioMet Toolbox [31], and the COBRA Toolbox [32] implement several useful algorithms for studying COBRA models and TRNs. However, to date, no single software platform has been developed to 1.) convert COBRA models and TRNs into integrated optimization problems, 2.) analyze these integrated models with existing algorithms to incorporate high-throughput expression data, and 3.) allow users to easily develop new algorithms for the integrated models.

To overcome these limitations, we present a Toolbox for Integrating Genome-scale metabolism, Expression, and Regulation (TIGER). TIGER automatically converts a list of implicit or explicit GPR and TRN rules into a set of linear inequalities; these equations are integrated with an existing COBRA model. The software allows rules to be written in a generalized Boolean format, enabling TRN logic to more accurately reflect the underlying biology. We demonstrate how this increased expressivity can overcome inconsistencies in existing TRN models.

2.4 Implementation

The primary functions of TIGER are shown in Figure 2.1. TIGER converts a GPR and additional regulatory rules into an equivalent mixed integer linear program (MILP). The MILP constraints

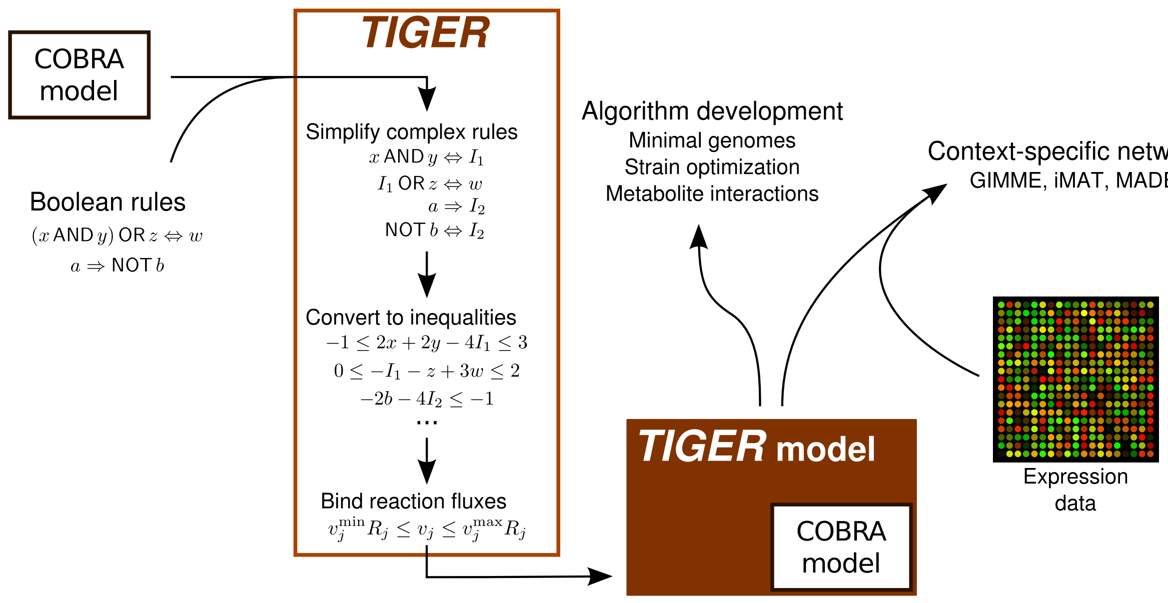

COBRA model
Boolean rules
$(x \text{ AND } y) \text{ OR } z \Leftrightarrow w$
$a \Rightarrow \text{NOT } b$
TIGER
Simplify complex rules
$x \text{ AND } y \Leftrightarrow I_1$
$I_1 \text{ OR } z \Leftrightarrow w$
$a \Rightarrow I_2$
$\text{NOT } b \Leftrightarrow I_2$
Convert to inequalities
$-1 \leq 2x + 2y - 4I_1 \leq 3$
$0 \leq -I_1 - z + 3w \leq 2$
$-2b - 4I_2 \leq -1$
...
Bind reaction fluxes
$v_j^{\min} R_j \leq v_j \leq v_j^{\max} R_j$
Algorithm development
Minimal genomes
Strain optimization
Metabolite interactions
Context-specific networks
GIMME, iMAT, MADE
TIGER model
COBRA model
Expression data

Figure 2.1: TIGER platform overview. TIGER converts Boolean rules to MILPs. Rules are first simplified by substitution and converted into a system of linear inequalities. The rules are optionally attached to a COBRA model by coupling indicators of reaction participation (R_i) to the reaction flux v_i. In addition to serving as a platform for developing new algorithms, TIGER models can be integrated with high-throughput expression data to generate context-specific models using variations on the GIMME, iMAT, and MADE algorithms.

are added to a COBRA metabolic model to create a TIGER model that combines metabolism, GPR associations, and transcriptional regulation. This integrated model serves as a platform for applying many gene-centric extensions to FBA, including algorithms that incorporate "omics" data for model refinement. In this section, we describe how the rules parsed by TIGER are constructed, and how they are converted to an MILP. Sample files depicting a COBRA model, GPR, and TRN are provided in the "`test/samples`" directory of the TIGER distribution.

2.4.1 Creating rules

These GPR relationships can be described as a Boolean *expression* using the standard operators and and or. For example, a reaction that is catalyzed by either of two isozymes, the second of which is composed of two subunits, would have a GPR of the form "isozyme$_1$ or (isozyme$_{2a}$ and isozyme$_{2b}$)". Two expressions are joined with an implication operator ($\Rightarrow$ or $\Leftrightarrow$, corresponding to "if" and "if and only if"), to form a *rule*. For GPR associations, rules are formed as "GPR $\Leftrightarrow$ *reaction*", where *reaction* is an indicator variable that constrains the flux through a reaction to be zero when the GPR expression is false.

TIGER expressions allow additional features to describe logical relationships that are more complex than those typically found in GPRs. The not operator allows logical negation, which is often used to construct rules for transcriptional repression. Expressions can also contain *conditionals* that compare the numerical values of individual variables. If a gene g was known to be expressed when glucose uptake is greater than 10 flux units, this relationship could be represented by the rule "glc_ex $> 10 \Rightarrow g$", where "glc_ex" is the glucose exchange reaction in the metabolic model. Any two expressions of arbitrary complexity can be combined as a rule and parsed by TIGER.

The grammar used by TIGER for rules was designed to resemble logical operations in common programming languages and to be compatible with the GPRs of widely-used COBRA models. A complete description of the TIGER syntax appears in Appendix A.

Some transcriptional regulators, such as the response of *crp* to cAMP in *E. coli*, display multiple levels of activity and cannot be easily described with Boolean logic [27]. Rather than require users to create several variables describing each state of activation, TIGER allows multilevel variables. If a transcription factor t activates target genes g_{low} at low levels of expression and g_{high} at high levels of expression, then this relationship could be described with the rules

$$(t = 1) \Rightarrow g_{\text{low}} \tag{2.2}$$

$$(t = 2) \Rightarrow g_{\text{high}} \tag{2.3}$$

where $t = 0$, 1, and 2 corresponds to no, low, and high expression. Logical operators have a different interpretation when applied to multilevel variables. If proteins x and y form a promoter complex for expression of gene z, then the corresponding rule for z expression would be "x and $y \Rightarrow z$", since both x and y are required for z transcription. If x and y were multilevel, one would assume that z expression would be proportional to the promoter subunit in lower abundance, since this species would limit the amount of complete promoter complex that could be formed. Thus, the and operation often corresponds to a minimization:

$$x \text{ and } y \equiv \min\{x, y\} \tag{2.4}$$

The or operation would be used in situations where either factor can independently induce expression. In this situation, the species in higher abundance determines the target gene's transcription level. TIGER implements the multilevel or as a maximization:

$$x \text{ or } y \equiv \max\{x, y\} \tag{2.5}$$

The not operator can have two interpretations when applied to multilevel variables:

$$\text{not } x \equiv \begin{cases} x > 0 & \text{pseudo-binary} \\ \bar{x} - x & \text{inversion, } x \in \{0, \ldots, \bar{x}\} \end{cases} \tag{2.6}$$

where $x \in \{0, ..., \bar{x}\}$. The first case (pseudo-binary) regards any nonzero value as true, regardless of the number of levels the variable may occupy. The second case (inversion) requires that the value of x and the quantity not x always sum to the maximum value that x can occupy. In this case, not x is a measure of how far x is from its upper bound. Users are able to select the pseudo-binary or inversion representation depending on which interpretation is a better approximation of the biological context. For example, consider a gene/repressor relationship $R \Rightarrow$ not G, where the repressor R can take on three biologically distinct levels – "off," "low," and "high". If both "low" and "high" levels of R prevent any expression of G, then the pseudo-binary not operator would be appropriate as G is off whenever R is not "off". However, if G also has the same three levels of expression, then the inversion interpretation of the not operator is more appropriate. This choice implies that

$$R = \text{off} \quad \rightarrow \quad G = \text{high} \tag{2.7}$$

$$R = \text{low} \quad \rightarrow \quad G = \text{low} \tag{2.8}$$

$$R = \text{high} \quad \rightarrow \quad G = \text{off} \tag{2.9}$$

Any variable in a TIGER model can be declared with multiple levels. Such declarations are made when rules are added to a model using the `add_rule` function in two ways: 1.) setting the default upper bound to all variables to any integer greater than one, or 2.) providing a list of variable names and a set of upper and lower bounds.

2.4.2 Rule simplification

Simple Boolean rules can be represented by systems of linear inequalities of integer variables [33]. A general Boolean rule can be converted by the following procedure to a set of simple rules before conversion to an MILP.

We define an "atomic" expression as either a variable (x) or a negated variable ($\mathsf{not}\ x$). If a not operator appears before an expression that is not atomic, TIGER applies DeMorgan's laws to move the negation onto atomic expressions (e.g., $\mathsf{not}\ (x\ \mathsf{and}\ y)$ becomes the equivalent expression $(\mathsf{not}\ x)\ \mathsf{or}\ (\mathsf{not}\ y)$). A simple rule then conforms to one of the following patterns

$$x \quad (\Rightarrow \mid \Leftrightarrow) \quad z \tag{2.10}$$

$$x\ \mathsf{and}\ y \quad (\Rightarrow \mid \Leftrightarrow) \quad z \tag{2.11}$$

$$x\ \mathsf{or}\ y \quad (\Rightarrow \mid \Leftrightarrow) \quad z \tag{2.12}$$

$$x\ \langle op \rangle\ y \quad (\Rightarrow \mid \Leftrightarrow) \quad z \tag{2.13}$$

where x, y, and z are atomic, and $\langle op \rangle$ is a conditional operator ($\leq$, $\geq$, etc.). Non-simple rules are converted to simple rules through a series of recursive substitutions. For example, the rule

$$(x\ \mathsf{or}\ (\mathsf{not}\ y))\ \mathsf{and}\ z \Rightarrow w \tag{2.14}$$

is not simple, since the expression $x\ \mathsf{or}\ (\mathsf{not}\ y)$ is not atomic. By defining an indicator variable I, which is true if and only if the expression $x\ \mathsf{or}\ (\mathsf{not}\ y)$ is true, equation (2.14) can be written as two simple rules:

$$x\ \mathsf{or}\ (\mathsf{not}\ y) \Leftrightarrow I \tag{2.15}$$

$$I\ \mathsf{and}\ z \Rightarrow w \tag{2.16}$$

The bounds of I are determined by the bounds of x and y. If $x \in \{0, \ldots, \bar{x}\}$ and $y \in \{0, \ldots, \bar{y}\}$, then $I \in \{0, \ldots, \bar{I}\}$, where

$$\bar{I} = \begin{cases} \max\{\bar{x}, \bar{y}\} & \text{for } x\ \mathsf{or}\ y \\ \min\{\bar{x}, \bar{y}\} & \text{for } x\ \mathsf{and}\ y \end{cases} \tag{2.17}$$

Thus, if x and y are binary, $\bar{x} = \bar{y} = 1$, so I is binary as well.

TIGER applies the above substitutions recursively, creating indicator variables as necessary until all rules are simple. Each simple rule is converted to a set of linear inequalities that are added as constraints to the model structure. If a variable name already appears in the model, TIGER assumes that these variables represent the same quantity and thus allows new rules to be added to an existing model without recompiling previous rules. At the same time, TIGER creates variables to substitute for negated variables. For efficiency, TIGER ensures that only one negated variable is created for each original variable, regardless of the number of times the negated expression appears in the set of simple rules. Details of the conversion between simple rules and inequalities, along with methods for handling conditionals, are provided in Appendix A.

2.4.3 Reaction coupling

If the GPR expression for a reaction is not satisfied, the reaction is not allowed to carry flux. To enforce this relationship during an optimization, a set of discrete variables R_i are defined, where $R_i = 0$ if the GPR for reaction i is not satisfied, and $R_i > 0$ otherwise. To enforce the GPR's effect on flux, TIGER adds the constraint

$$v_i^{\min} R_i \leq v_i \leq v_i^{\max} R_i \tag{2.18}$$

where v_i is the flux through the ith reaction, with lower and upper bounds $v_i^{\min}$ and $v_i^{\max}$.

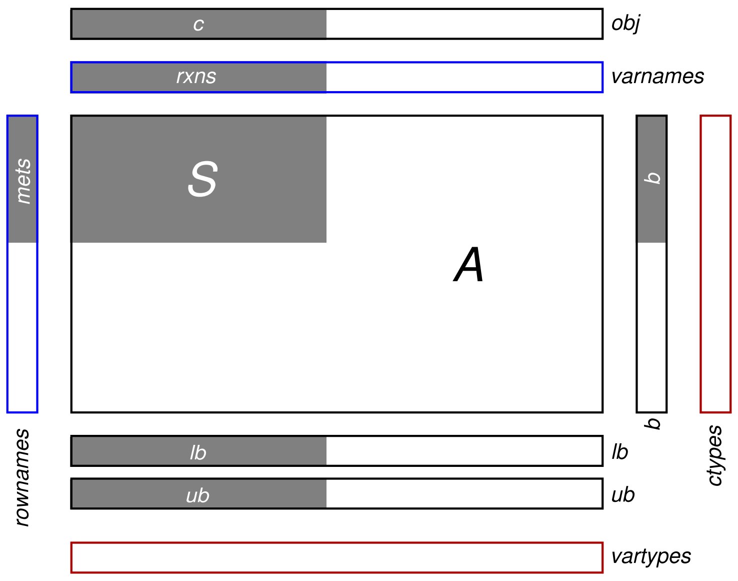


Figure 2.2: Structure of TIGER models. TIGER models are represented as Matlab structures. Boxes indicate size and orientation of the fields. Black text denotes TIGER field names. Gray areas contain data from the COBRA model, with white text indicating the relevant COBRA field names. Border color indicates data type: black → double-precision matrix, blue → cell array of strings, red → character array.

2.4.4 Model structure

TIGER models are represented as Matlab structures. The layout of this structure is shown in Figure 2.2. The structure contains fields `obj`, `A`, `b`, `lb`, and `ub` that correspond to the values in the following MILP problem:

$$\begin{aligned}&\min\ obj'x\\ &\text{subject to}\\ &\qquad Ax\,(\leq \mid = \mid \geq)\,b\\ &\qquad lb \leq x \leq ub\end{aligned}$$

The type of (in)equality for each constraint in `A` is determined by the character vector `ctype`. The type of variable for each entry in x is specified by the field `vartype`, where ‘`c`’, ‘`b`’, and ‘`i`’ denote continuous, binary, and general integer variables. Reaction fluxes are continuous variables, while all other variables are either binary or integer depending on the corresponding upper bound. The fields `rownames` and `varnames` contain descriptive names of the constraints and variables, stored as cell arrays of strings. Functions in TIGER allow variables to be interchangeably referenced by their name, column index, or through Matlab's logical indexing features.

The format for TIGER models is designed for compatibility with the model structure for the COBRA Toolbox [32]. TIGER can use a COBRA Toolbox model as a starting point for converting a genome-scale reconstruction; therefore, any model in a file format supported by the COBRA Toolbox (SBML, Simpheny, etc.) can be converted to a TIGER model.

2.4.5 Accessing the MILP solver

TIGER uses a custom Matlab class CMPI (for Common Mathematical Programming Interface) to create and solve mathematical programming problems. CMPI defines a consistent structure for MILP (and mixed-integer quadratic programming, MIQP) problems, providing independence from the underlying MILP solver software. TIGER currently supports the CPLEX, Gurobi, and GLPK (via GLPK Mex) software packages, all of which are freely available for academic use. Porting TIGER to use a new solver requires modifying only the CMPI method `solve_mip` to specify the new interface. CMPI also provides a standardized method for configuring common solver parameters (maximum solution time, optimality and feasibility tolerances, etc.).

Previous work has indicated that the computation time of some FBA-related algorithms, such as Flux Variability Analysis [19], can be reduced by saving information about the problem structure between calls to the MILP solver [34]. CMPI provides a method, `solve_multiple_milps`, to preserve the solver state between successive calls to the CPLEX optimizer and reduce runtime in this manner. (Gurobi and GLPK currently do not support this feature in their Matlab interfaces.) If the CPLEX optimizer is not installed, CMPI will automatically make successive calls to the installed optimizer. While this removes the potential speed increase from using solver restarts, it allows TIGER code to remain solver independent and portable.

2.4.6 Using TIGER

TIGER source code and installation instructions are available online at `http://bme.virginia.edu/csbl/downloads/` or `http://csbl.bitbucket.org/tiger`. All functions in the toolbox are documented using Matlab's "help" facilities. Complete documentation and a step-by-step tutorial are also available on the TIGER website. The software includes a testing suite to verify the installation. These tests contain examples that build a TIGER structure from a simple COBRA model, add a set of TRN rules, call a MILP solver, and display the solution.

2.5 Results and Discussion

2.5.1 Refining integrated models for *Saccharomyces cerevisiae*

TIGER was used to couple the 1266 reactions in iND750 [17], a genome-scale model of *Saccharomyces cerevisiae* metabolism, with 750 metabolic genes. The resulting TIGER model contained 4498 constraints in 3214 variables. A model of *S. cerevisiae* transcriptional regulation, iMH805 [28], was added. The additional 805 rules contributed 1057 constraints and 562 variables to the TIGER model. The conversion took 53.66 s for iND750 and 20.31 s to add the TRN using an Intel 3.2 GHz i7-quad core processor running Linux.

As mentioned above, previous methods for integrating TRNs involve an iterative process, alternating between calculating gene states from a given environment and determining an environment based on metabolic byproducts [29]. However, the multiple layers of trascriptional regulation may require several iterations of this method to reach a stable gene state. The number of iterations to reach a stable state varies by environment and cannot easily be determined *a priori* [35]. In fact, some feedback mechanisms in TRNs may lead to a stable cycle of gene activation/inactivation rather than a single gene state. TIGER solves the TRN and FBA problems simultaneously, so the resulting gene state is always stable (or an optimal state inside a stable cycle).

Applying large-scale TRNs to COBRA models may result in infeasible models, i.e., models unable to produce any biomass. This is often due to a small number of rules that turn off reactions that are essential for biomass production. Previous work has developed techniques for finding which

		Iteration				
	Start	1	2	3	4	Rule
O_2[e]	1	1	1	1	1	
glucose[e]	1	1	1	1	1	
hap1	0	1	1	1	1	O_2[e] or not ROX1
rox1	0	0	1	1	1	O_2[e] and HAP1
erg11	1	1	1	0	0	O_2[e] and HAP1 and (not ROX1)
erg11	1	1	1	1	1	O_2[e] and HAP1 and (high_o2 or not ROX1)

Table 2.2: Gene states for *erg11* regulation. To start, all transcription factors are assumed to be "off", and all metabolic genes are "on". The extracellular environment contains only glucose (glucose[e]), oxygen (O_2[e]), and essential salts and minerals. Each iteration calculates the next gene state based on the current state of metabolites in the environment and transcription factors. After three iterations, expression of *erg11* turns off, unless a modified rule is used that account for high oxygen uptake. The modified rule (shown below the double line) is more consistent with the results in Turi & Loper [36] Rules are taken from the *S. cerevisiae* regulatory network model iMH805 [28].

rules create the model infeasibility [25]. TIGER includes the function `find_infeasible_rules` to identify rules that prevent feasible solutions to the resulting MILP. Given a model and a set of rules that prevent a feasible solution, `find_infeasible_rules` creates a MILP that preserves the logic of the rules but allows each rule to be artificially satisfied. The objective of this MILP is to minimize the number of rules that must be artificially satisfied while finding a feasible solution for the model. (Details of this process are available in Appendix A.)

Analysis by TIGER reported that the combined iND750/iMH805 metabolic and TRN network was unable to produce biomass under aerobic conditions in a glucose minimal media. Since *S. cerevisiae* is well-known to grow in this environment, I used TIGER's `find_infeasible_rules` function to identify the following three rules that prevented growth:

$$\mathrm{O_2[e]}\ \mathsf{or}\ (\mathsf{not}\ \mathrm{ROX1}) \Leftrightarrow \mathit{hap1} \tag{2.19}$$

$$\mathrm{O_2[e]}\ \mathsf{and}\ \mathrm{HAP1} \Leftrightarrow \mathit{rox1} \tag{2.20}$$

$$\mathrm{glucose[e]}\ \mathsf{and}\ \mathrm{HAP1}\ \mathsf{and}\ (\mathsf{not}\ \mathrm{ROX1}) \Leftrightarrow \mathit{erg11} \tag{2.21}$$

The product of gene *erg11*, lanosterol 14α-demethylase, is an essential enzyme for the production of ergosterol, a main sterol in *S. cerevisiae* [36]. This gene is essential in the iND750 metabolic model and must remain "on" during aerobic growth on glucose. However, the stable gene state for *erg11* in the above rules is always "off" after three iterations, as described in Table 2.2. Because the iMH805 study used the results of the second iteration as the final gene state, this inaccuracy was unnoticed.

Rule (2.21) was originally derived from Turi & Loper [36]. Re-examination of this manuscript revealed that while ROX1 represses *erg11*, complete repression is only observed under low oxygen conditions. To incorporate these findings, we create an indicator "high_o2" that is true if and only if the cell uptakes more than 10% of the maximum oxygen consumption rate. This relationship is expressed as

$$\mathrm{high\_o2} \Leftrightarrow \mathrm{EX\_o2(e)} < -0.244 \tag{2.22}$$

where "EX_o2(e)" is the iND750 name for the oxygen exchange reaction, and the maximum oxygen uptake rate for growth on glucose is 2.44 mol/(g dry cell weight)/h (negative flux through exchange reactions indicate uptake by the cell). Rule (2.21) for *erg11* expression was re-written to only exhibit ROX1 repression under low oxygen conditions:

$$\text{glucose[e]} \ \mathsf{and}\ \text{HAP1}\ \mathsf{and}\ (\text{high\_o2}\ \mathsf{or}\ \mathsf{not}\ \text{ROX1}) \Leftrightarrow \textit{erg11} \tag{2.23}$$

The set of refined rules (2.20,2.21,2.23) reproduces the correct growth phenotype in aerobic glucose conditions. This example demonstrates a three-step procedure for refining existing TRN models using TIGER: 1.) apply the existing TRN to a COBRA model, 2.) use the `find_infeasible_rules` function to identify rules that cause the model to differ from a known phenotype, and 3.) re-examine the evidence for these rules and make appropriate modifications. As shown in the previous example, new biological information can often be incorporated into existing rules using TIGER's support for complex logical expressions.

2.6 Conclusions

I have presented TIGER, a software platform for converting generalized Boolean and multilevel rules to mixed-integer linear programs, and coupling these rules to genome-scale models of metabolism. The flexibility of TIGER's generalized rule format allows for a more accurate description of biological processes such as catalysis by isozymes and multi-meric proteins, metabolic flux control, and transcriptional regulation. These features were used to identify and correct inconsistencies within an existing TRN model of *Saccharomyces cerevisiae*. I have also demonstrated how TIGER can be used as a starting point for implementing and improving existing algorithms for genome-scale analysis.

In addition to adding implementations of other gene-centric algorithms to TIGER, I am exploring methods to improve the solution efficiency of the generated MILP. Possible strategies include exploiting indicator constraints, specially-ordered-sets (SOS), and other solver optimizations through CMPI.

Chapter 3

Creating context-specific metabolic models from gene expression data

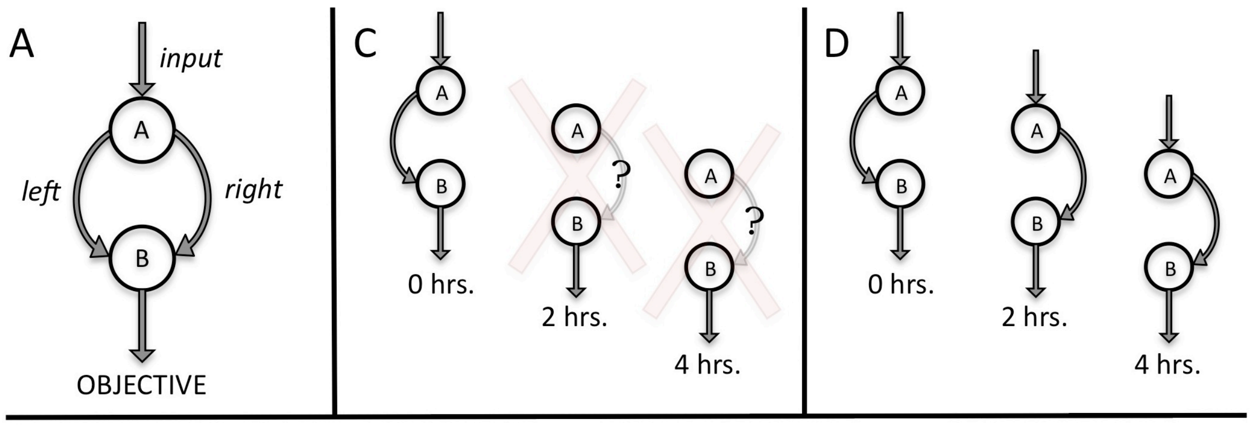
3.1 Introduction

Genome-wide transcriptional regulatory networks (TRNs) can be used to predict the likely "on" or "off" state of metabolic genes as a function of environmental factors. Unfortunately, complete TRNs are only available for a limited number of organisms (including *Escherichia coli* [27] and *Saccharomyces cerevisiae* [28]). Other reconstructions must instead rely on experimental measurements (typically from gene expression microarrays) to determine the appropriate gene states, but this approach suffers from two complications. First, classifying reactions as "on" or "off" requires setting a threshold in the continuous expression levels of the associated genes; such thresholds cannot be accurately determined from measured mRNA levels alone. Recent evidence indicates that such a correlation between transcript levels and metabolic reaction activity does exist [37]. However, determining a specific threshold above which microarray measurements imply physiologically appreciable activity in the corresponding metabolic reaction is an unresolved problem. Second, these thresholds are not independent, since functioning metabolic reconstructions require a minimum set of reactions to be active in each condition, regardless of any apparent change in the underlying gene expression data. Especially when considering decreases in expression levels, the choice to fit these data by inactivating the corresponding reaction must be balanced with the functional effects of removing a reaction from the metabolic network.

An early attempt at reconciling gene expression data with a FBA model was the GIMME algorithm [38]. Using a set of user-supplied thresholds for the transition of each gene from "on" to "off," GIMME iteratively re-activated "off" reactions (by turning on genes below their threshold) until a functioning model was obtained. While GIMME did produce functioning FBA models, the method required the *a priori* determination of expression thresholds.

A more recent approach [39] used the expression thresholds in previously published databases to classify reactions as either highly, moderately, or lowly active. After choosing a threshold for the minimum flux through a highly active reaction, a mixed-integer linear program (MILP) was formulated to calculate a steady-state flux distribution that maximized the number of highly (lowly) expressed reactions that carried at least (not more than) the threshold flux. All such reactions were

Parts of this chapter are adapted from: Jensen PA, Papin JA. Functional integration of a metabolic network model and expression data without arbitrary thresholding. *Bioinformatics*. 2011, 27(4):541-7.



B

Measured Gene Expression*

	0 hrs.	2 hrs.	4 hrs.
input	223	158	162
left	174	52	48
right	23	88	102

(**arbitrary units*)

Difference

	0 - 2 hrs.	2 - 4 hrs.
input	-1 (*p*=0.02)	0 (*p*=0.04)
left	-1 (*p*=0.01)	0 (*p*=0.06)
right	1 (*p*=0.03)	1 (*p*=0.09)

MADE Binary Approximation

	0 hrs.	2 hrs.	4 hrs.
input	1	1	1
left	1	0	0
right	0	1	1

Figure 3.1: Results of MADE on a prototypic system. (A) After transport facilitated by the gene product of *input*, metabolite A is converted to B by isozyme products of *left* or *right* before being removed through the objective reaction. (B) Measured expression data for each gene in the system, and the corresponding changes in expression (-1 and 1 indicate decreases and increases, respectively, and 0 represents no significant change). (C) Challenges with functional integration. Although *input* expression decreases, it must always remain "on" for a functioning model. It is not clear from the expression data whether *right* should be first expressed at 2 or 4 hours. (D) MADE finds the best functional model that reproduces the observed expression changes. The resulting model indicates that A is transformed to B by the product of *left* at early times. After 2 hours the flux is re-directed through the reaction catalyzed by *right*'s product.

assumed to be active (inactive). Although this approach was successfully applied to a reconstruction of the human metabolic network, the same technique would not necessarily guarantee that the set of active reactions could produce the necessary objective flux in microorganisms with a more defined objective function. Additionally, determining the state of each reaction does not uniquely determine the expression state of each gene in the reconstruction. (An active reaction, for example, could be activated if either of two genes, each encoding an isozyme, were expressed.)

In this chapter, I present a method for *de novo* determination of a functional, binary expression state for all genes in a metabolic reconstruction using experimentally-derived expression data. My algorithm, Metabolic Adjustment by Differential Expression (MADE), uses an optimization-based approach to create a sequence of binary expression states that reflect the most statistically significant changes in the series of gene expression measurements; it does not require the use of arbitrary expression thresholds for each gene. The optimization is constrained to produce only functioning models by simultaneously imposing a minimum value on the system's metabolic objective. MADE is formulated as a single MILP problem using the TIGER software described previously.

3.2 Approach

MADE relies on expression data from two or more conditions to determine the best-fitting gene states. (Again, "conditions" can be any set of environmental, genomic, or temporal data.) To illustrate, consider the simple metabolic system in Figure 3.1a. The product of the gene *input* facilitates the transport of metabolite A into the system, after which it undergoes transformation into B, catalyzed by either of two isozymes – the products of genes *left* and *right*. Production of B is assumed to be the metabolic objective. This example system receives an external stimulus, and the transcriptional response is measured for each gene at 0, 2, and 4 hours, as summarized in Figure 3.1b.

As indicated by the expression data, the expression levels of genes *input* and *left* decrease between 0 and 2 hours; they remain at this lower level of expression until the 4 hour timepoint. Expression of the enzyme Right appears to increase between each set of timepoints. The challenges of converting these data into a binary gene state are illusted in Figure 3.1c. As a first approximation, having *input* and *left* "on" at 0 hours and "off" at 2 and 4 hours matches the overall expression pattern. However, the gene product of *input* is always necessary for a functioning model, so neither of the resulting 2 or 4 hour gene states is feasible. Finding a binary representation for the gene *right* is also not straightforward. The continuous expression levels increase during each transition, but a binary approximation can only transition from "off" to "on" once. It is unclear whether *right* should be actively expressed beginning at 2 or 4 hours.

Figure 3.1d demonstrates the results of the MADE approach. Consistent with the measured expression, *left* turns "off" between 0 and 2 hours, remaining in that state for the final timepoint. Although *input* expression decreases significantly, MADE leaves the gene "on" for the entire time course because of its essentiality. It is assumed that while mRNA expression of *input* has decreased by 2 hours, the functional abilities of the gene product must be preserved to facilitate transport of a necessary metabolite. Finally, MADE activates *right* transcription at 2 rather than 4 hours, using the greater statistical significance of the increase between 0 and 2 hours as evidence for this approximation. The resulting model is functional at all timepoints and suggests that the metabolic system initially routes flux through the Left-catalyzed pathway and transitions to the Right-catalyzed pathway after 2 hours.

3.3 Methods

3.3.1 MADE Representation

Given a sequence of measured mean expression levels $\{e_1, e_2, \ldots, e_n\}$ for a gene, a sequence of differences $\{d_{1\to2}, d_{2\to3}, \ldots, d_{n-1\to n}\}$ can be defined as

$$d_{i\to i+1} = \begin{cases} +1 & e_i < e_{i+1} \\ 0 & e_i = e_{i+1} \\ -1 & e_i > e_{i+1} \end{cases} \tag{3.1}$$

For example, the value of $d_{1\to2}$ signifies that the mean expression of the gene significantly increases (+1), decreases (-1), or remains constant (0) between conditions 1 and 2. Statistical significance is determined by $p_{1\to2}$, the p-value corresponding to the confidence of assigning the gene to one of the cases in Equation 3.1.

MADE finds a sequence of binary expression states $\{x_1, x_2, \ldots, x_n\}$, $x_i \in \{0, 1\}$ such that the differences between successive states (x_{i+1} - x_i) most closely match the corresponding differences in the mean expression levels $d_{i \to i+1}$. Because the binary representation of gene expression assumed by FBA models cannot account for all possible expression patterns, MADE uses the statistical significance of the differences to create the most probable approximation. For example, consider a gene with mean expression levels (over four timepoints) of {10, 11, 65, 109}. If the calculated differences were {0, +1, +1} with p-values {0.02, 0.048, 0.0012}, an exact binary representation of this sequence is not possible, since the gene must increase over two consecutive intervals. MADE would choose the binary expression sequence {0, 0, 0, 1} over {0, 0, 1, 1}, since the increase between timepoints 3 and 4 is more statistically significant than the increase between timepoints 2 and 3.

More formally, binary expression sequence returned by MADE is computed as the

$$\arg\min_{x \in X} \sum_{i=1}^{n-1} w(p_{i \to i+1}) |d_{i \to i+1} - (x_{i+1} - x_i)| \tag{3.2}$$

where n is the number of conditions, $X \subset \{0,1\}^n$ is the set of all gene expression states that allow functioning FBA models, and $w(p_{i \to i+1})$ is a weighting function that assigns larger weights to more significant p-values (for example, $w(p) = -\log p$). The absolute value in the summand measures the discrepancy between the binary transitions and expression differences. Binary sequences that show an increase or decrease when the measured expression indicates no change (or vice-versa) incur a "penalty" of w; binary sequences that increase when expression data decrease (or vice-versa) are penalized by $2w$.

3.3.2 Formulating the MILP problem

For each transition between conditions, the binary variables x representing the expression state of the metabolic genes are partitioned into three sets I, D, and C, corresponding to increasing, decreasing, or constant expression between the two conditions. The optimization objective f for this transition is the weighted sum

$$\begin{aligned} f_{i \to i+1}(x) = &\sum_{x \in I} w(p_{x_{i \to i+1}})(x_{i+1} - x_i) \\ &+ \sum_{x \in D} w(p_{x_{i \to i+1}})(x_i - x_{i+1}) \\ &- \sum_{x \in C} w(p_{x_{i \to i+1}}) \Delta_{x_i, x_{i+1}} \end{aligned} \tag{3.3}$$

where $\Delta_{x_i, x_{i+1}}$ is a binary variable that takes the value 0 when $x_i = x_{i+1}$ and 1 otherwise; this definition is enforced by adding the Boolean constraint $\Delta_{x_i, x_{i+1}} = (x_i \text{ XOR } x_{i+1})$.

The weights $w(p)$ were calculated as $(-\log p)$, but using other functions that mapped smaller p-values to a higher weight, e.g. $w(p) = 1-p$, did not significantly change the results. The logarithmic transformation was chosen to minimize ill-conditioning for highly significant changes. Using a unit weighting function $w(p) = 1$ for all p-values would cause MADE to match the greatest number of transitions without regard for the significance of the changes. The p-values were determined by t-test on the normalized array data in accordance with standard microarray analysis procedures using the GeneSpring software package (Agilent Technologies, Palo Alto, CA, USA). The objective function reaches its theoretical maximum when all calculated expression states match the observed expression changes.

MADE maximizes the sum of the objective function values across all conditions subject to a regulated FBA formalism:

$$\max \sum_{i=1}^{n-1} f_{i \to i+1}(x)$$

subject to

$$\begin{aligned} Sv &= 0 && \text{(a)} \\ lb \leq v &\leq ub && \text{(b)} \\ v_{\text{obj}} &\geq v_{\min} && \text{(c)} \\ N \begin{pmatrix} v \\ x \end{pmatrix} &= b && \text{(d)} \end{aligned}$$

Constraints (a) and (b) represent the mass balance, stoichiometric, and thermodynamic constraints of FBA. The metabolic objective (e.g. biomass production) is required to be above a minimum value in (c) to ensure a viable flux distribution. Minimum values of 0.1-0.3 of the optimal objective flux have been used to ensure viability in FBA-based bacterial strain design [23]. The system of equations (d) contains an integer programming formalism for the GPR rules and a set of constraints that couple these rules to the metabolic fluxes v. These constraints were formulated using SR-FBA, as described above. A separate set of constraints (a-d) appear for each experimental condition.

MADE was implemented in TIGER version 1.0.1 using the genome-scale *S. cerevisiae* metabolic model iND750 [17]. The final MILP problem was solved with the Gurobi Optimizer (Gurobi Optimization, Houston, TX, USA) on an Intel dual-core desktop computer running Linux. The MILP gap for all problems converged to less than 0.5% in under 600 seconds.

3.4 Results

Expression data gathered from the glucose to glycerol shift in the yeast *S. cerevisiae* are ideal candidates for MADE application. Yeast treats glucose as its preferred carbon source, as evidenced by the repression of genes catabolizing all other carbon sources in glucose-rich media [40]. After nearly all available glucose has been exhausted, yeast begin a drastic shift in metabolic activity to initiate respiration of ethanol, the fermentation product of glucose. The effects of this "diauxic shift" extend beyond central carbon metabolism, as the organism also slows production of amino acids and ribosomal-related proteins. The demand for these metabolic byproducts is decreased by the drastically slowed growth rate. Overall, nearly 1700 genes show differential expression in this process, accounting for over 25% of the yeast genome [41]. (For review of the diauxic shift, see [42])

While the metabolic effects of the diauxic shift are well-studied, the transition from glucose- to glycerol-based growth is less understood. Roberts and Hudson used a series of microarrays to measure expression changes at 15, 30, and 60 minutes following a shift from glucose- to glycerol-rich media [43]. The results indicated changes among expression levels in a number of gene ontologies and several individual genes. Integrating these data with a metabolic model would enable the analysis of pathway-level and functional shifts in the metabolic flux distribution.

I used MADE to integrate these data with iND750, a fully-compartmentalized, genome-scale model of *S. cerevisiae* metabolism [17]. A total of four environmental conditions were considered: growth to early log phase in YPD (glucose-rich media); and growth on YPG (glycerol-rich media) for 15, 30, and 60 minutes after the switch from YPD. Expression states were calculated for 743 metabolic genes in the model (a sample of these results is shown in Figure 3.2; complete results are

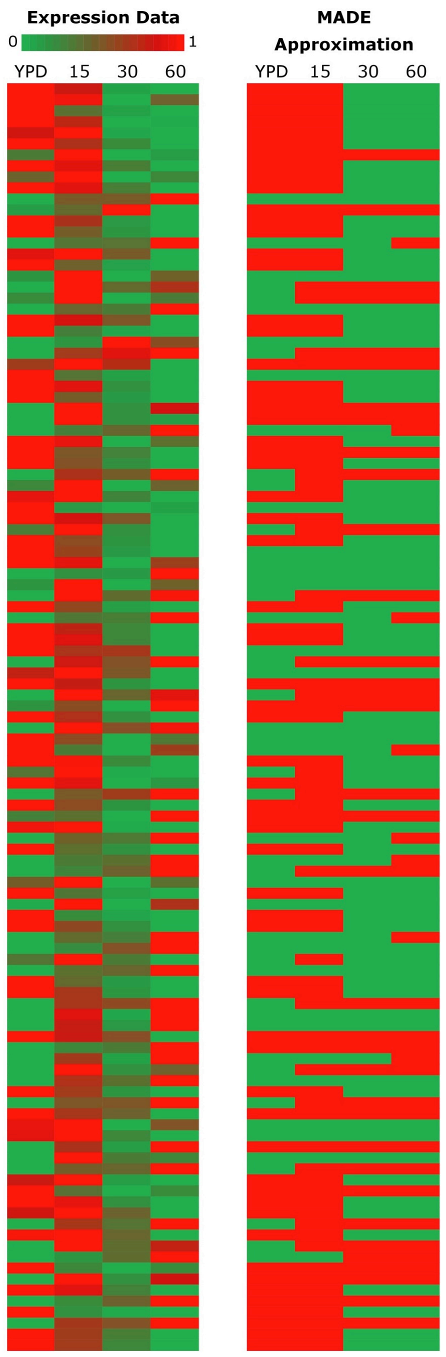


Figure 3.2: MADE results for 100 randomly selected genes from the *S. cerevisiae* model. The left heatmap indicates the normalized expression level across the four timepoints; the right heatmap displays the binary approximation calculated by MADE. Overall, 98.8% of the feasible transitions were correctly matched.

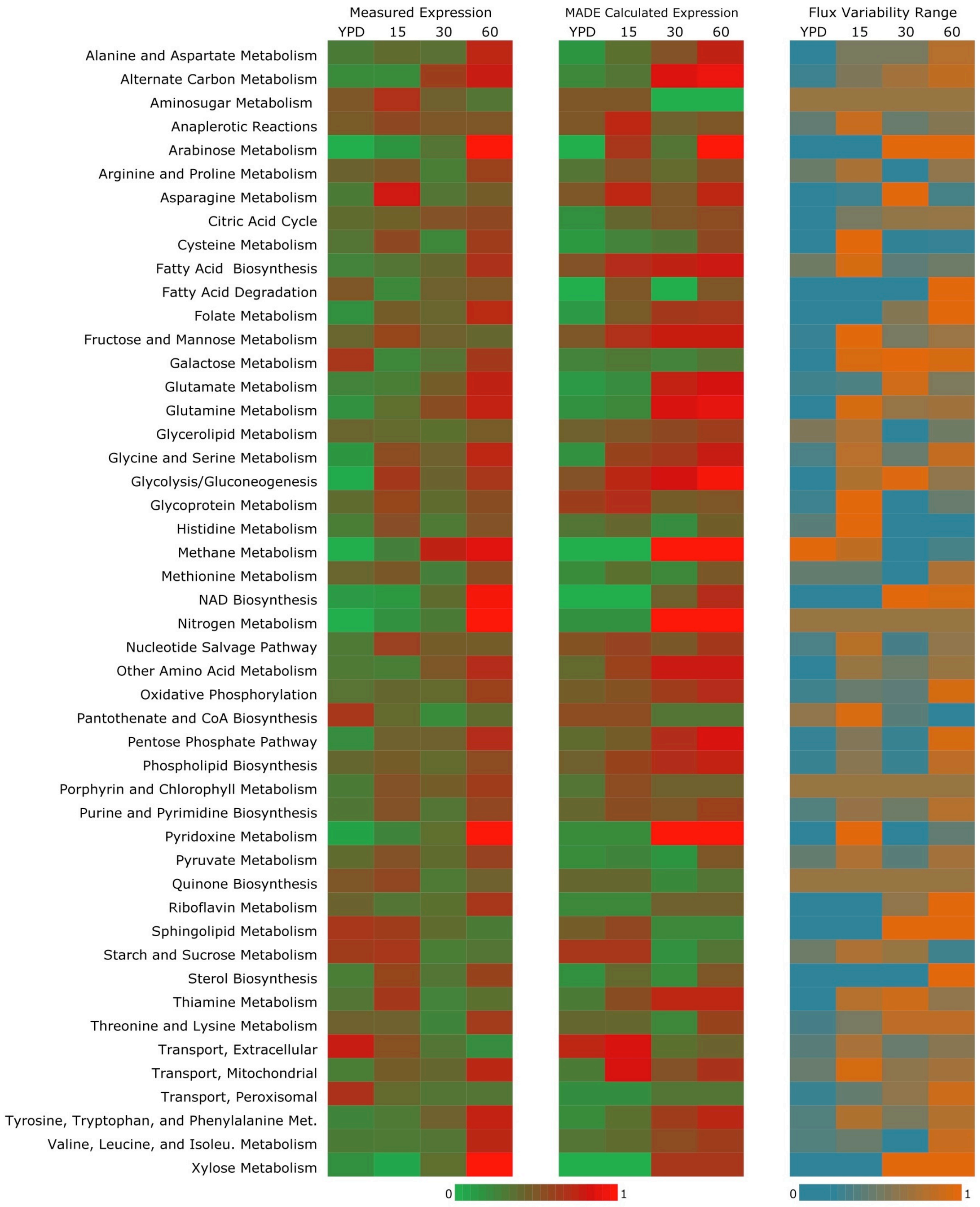


Figure 3.3: Average gene expression and flux variability by metabolic subsystem in *S. cerevisiae* after a transition from glucose- to glycerol-based respiration. Normalized expression levels were averaged across all genes associated with the subsystem. For comparison, the same approach was applied to the binary expression states calculated by MADE. The range of flux variability is the difference between the maximum and minimum allowable fluxes through a reaction while preserving an optimal objective value. These ranges were averaged for reactions in a subsystem and normalized – "1" indicates the largest average range for reactions in the subsystem; "0" is the smallest range.

shown in the Supplementary Data). Figure 3.3 shows the overall gene expression levels grouped by functional pathways. The expression levels for genes associated with each reaction in the pathway were averaged as a measure of overall subsystem-related transcriptional activity. The MADE-derived binary approximations for several subsystems follow the continuous expression patterns, although changes in measured expression often appear more subtle than switches between "on" and "off" states. Specifically, expression levels in the fatty acid and glutamate subsystems increase immediately after the glycerol transition, and membrane/ion transport activity decreases near the end of the transition period. All of these findings are consistent with published conclusions [43].

3.4.1 Flexibility of Metabolic Subsystems

The models returned by MADE also recapitulate several of the expected functional behaviors of the glucose/glycerol shift. I used Flux Variability Analysis to characterize the metabolic flexibility in each subsystem. The heatmap in Figure 3.3 shows how the average range of admissible fluxes for each reaction in a subsystem changes during the shift. High values indicate that the flux through a subsystem can vary significantly while still achieving optimal growth. Lower values correspond to tight metabolic regulation.

Yeast grown with an ample glucose supply strongly repress catabolism of all other carbon sources [40]. When glucose is removed, a genome-wide transcriptional adjustment re-activates pathways involved in transport and utilization of other nutrients. The MADE-generated models reflect this shift. Cells grown in YPD display the smallest range in flux variability, corresponding to a tightly regulated metabolic state that is highly optimized toward glucose utilization. After the shift to glycerol, the flux variability increases as the organism increases its metabolic versatility.

3.4.2 Incorporating a Transcriptional Regulatory Network

The differential expression initiated by the glucose to glycerol switch is due in part to transcriptional regulation. In order to refine the MADE-generated models, I coupled the iND750 model with iMH805, a transcriptional regulatory network previously assembled from primary literature [28]. The network consists of 436 Boolean rules capturing expression relationships among 82 nutrients, 55 transcription factors, and 750 metabolic genes. The network was converted to a MILP system using the SR-FBA formalism, and the resulting equations were added to the GPR constraint set – no further modification of the MADE algorithm was necessary to calculate best fit expression states for each of the 55 transcription factors. The addition of the TRN changed the MADE prediction for 122 (16.4%) of the metabolic genes (see Supplementary Data). As expected, the network did help refine the MADE prediction: for example, *ENO2*, a phosphopruvate hydratase, is known to be expressed in glycerol-based media (SGD). Without the TRN, MADE predicted that only the related gene *ENO1* would be expressed. With the TRN, MADE correctly identified both *ENO1* and *ENO2* as being "on."

3.4.3 Accuracy of MADE

As previously mentioned, the binary variables representing gene expression in the FBA framework cannot describe all possible expression patterns. Using the prototypic model in Figure 3.1 as an example, the total number of comparisons is six (3 genes $\times$ 2 transitions). However, the number of feasible comparisons is only five, since the binary description of *right* expression cannot increase twice consecutively. Therefore, when measuring the ability of MADE's results to reproduce the expression change dynamics, it is useful to also calculate the percent of feasible transitions matched. The number of feasible matches can be calculated by running MADE without the minimum metabolic objective constraint. This allows MADE to fit gene expression states without regard for overall model functionality.

The fermentive/glycolytic shift data contained 3715 transitions, of which 2991 were feasible. MADE matched 98.8% of the feasible transitions (83.6% of all transitions). It is important to note that the 15.4% of transitions that are infeasible are a consequence of the binary representation of gene expression in FBA. Additionally, the assumption in MADE that the statistical significance of differential gene expression correlates with biological significance does not always hold. Also, some of the correctly matched transitions may not accurately reflect the true metabolic behavior. The transitions in gene expression data are not necessarily a completely accurate representation of the actual expression state due to errors in the experimental data.

3.5 Conclusions

The Metabolic Adjustment by Differential Expression (MADE) algorithm integrates gene expression data and a metabolic model without *a priori* determination of activity thresholds. The method attempts to match most closely the genes that exhibit the most statistically significant changes in expression levels. The resulting gene states always produce functioning models and match the direction of differential expression with high accuracy. Results from the glucose/glycerol shift in *Saccharomyces cerevisiase* indicate that MADE-derived models also recapitulate the overall behavior of the actual biological system.

Chapter 4

Automated visualization of genome-scale metabolic models

4.1 Introduction

The number of genome-scale metabolic models has increased greatly in recent years [44]. An accompanying expansion in available algorithms for analyzing these models in the context of high-throughput data [18] has created the need for tools to visualize large models and datasets. Cytoscape [45] and similar graph-drawing software can visualize arbitrary biological networks. However, the resulting node-and-edge graphs are visually distinct from the more familiar "metabolic map" layout where lines show the flow of reactants coming together before branching into products. Only a few genome-scale metabolic reconstructions include a manually curated visualization. However, producing these maps is difficult and time-consuming, and the maps are rarely updated as the model is revised.

KEGG [46] and other metabolic pathway databases provide visualizations of indexed reactions, but these maps exclude reactions not in the database (such as transport or species-specific reactions). Metabolic modeling software packages such as the COBRA Toolbox [47] and CellNetAnalyzer [30] can overlay flux and gene expression data on reaction maps. While both packages allow users to modify existing maps, no software package is capable of assembling a complete genome-scale metabolic map *de novo* from a set of reactions.

4.2 Features

In this chapter, I describe MetDraw, an online tool and software package for automatically generating a reaction map for genome-scale metabolic reconstructions. MetDraw also allows users to visualize metabolomic, reaction flux, and gene/protein expression data directly on the resulting maps. Maps are created from SBML model files and exported as SVG images. This widely-accept image format allows users to easily customize details of the final maps with image editing software. Although the map creation process is completely automated, several features allow users to control the drawing output.

Parts of this chapter are adapted from: Jensen PA, Papin JA. MetDraw: Automated visualization of genome-scale metabolic models and high-throughput data. *under revision.*

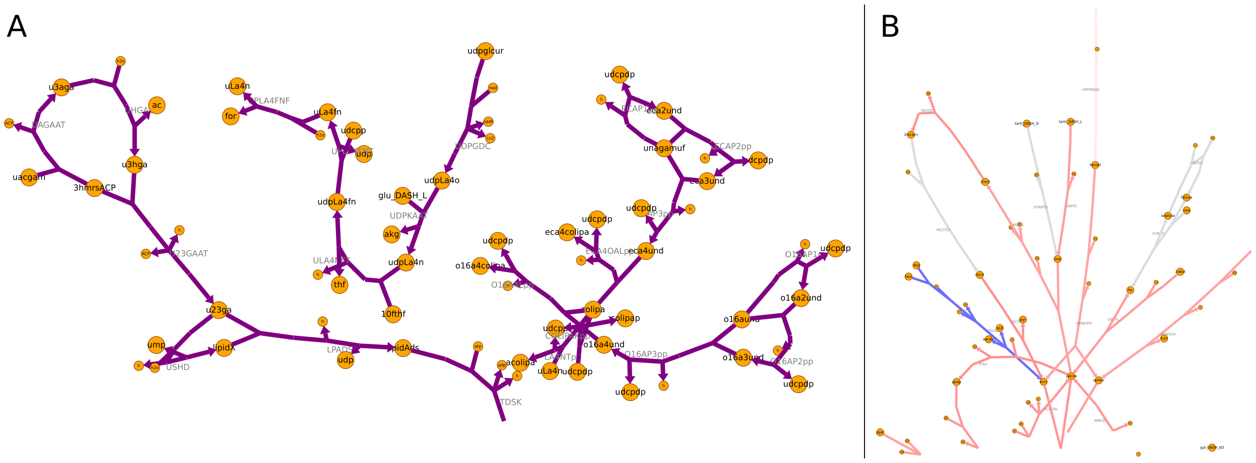


Figure 4.1: Visualization of the *E. coli* metabolic model described in [48]. The map was created by MetDraw from the SBML model file without additional user input. A. Pathways from lipopolysaccharide biosynthesis and recycling pathways, enlarged to show detail. B. Visualization of Gibbs free energy change for reactions (ΔG_r) in the alternative carbon metabolism subsystem. Coloring was applied by MetColor. Blue (red) indicates positive (negative) ΔG_r.

4.2.1 Input

MetDraw begins with a valid SBML file of the metabolic reconstruction. The SBML file can be uploaded to the MetDraw website (`http://www.metdraw.com`), or the software can be run on a local computer. MetDraw allows compartmentalized models with transport reactions spanning multiple compartments. For optimal layouts, the SBML file should also contain subsystem assignments in the "notes" section of several reactions. These designations are used to partition the smaller subgraphs that can be visualized more easily.

4.2.2 Layout Overview

Metabolites are designated as "major" or "minor" depending on the number of reactions involving each species. Major metabolites are drawn once in each compartment with arrows denoting how the species is produced or consumed by each reaction. Minor, or currency, metabolites are those that appear in many reactions, e.g. high-energy phosphates, water, protons, and common metabolic cofactors. Minor metabolites are redrawn for each reaction rather than being drawn once per subsystem and shared by multiple reactions. The removal of minor metabolites reduces much of the visual clutter caused by these highly connected species. MetDraw identifies minor metabolites by thresholding metabolite/reaction participation counts. Optionally, MetDraw will export a list of reaction counts for each metabolite and allow the user to designate the minor metabolites manually.

If subsystem designations are available for any reactions, MetDraw partitions the corresponding reactions. Very small subsystems or subsystems that share nearly all reactions with other subsystems are merged to preserve information flow in the final map. Subsystems and unclassified reactions are then placed into compartments. Transport reactions, reactions with reactants in more than one compartment, are identified and separated for layout across the compartment boundaries.

The final layout uses the widely-used Graphviz [49] software. MetDraw converts the reactions to a series of edges and nodes using the Graphviz DOT language. Unlike other graph drawing programs for biological networks, MetDraw inserts invisible nodes and edges and uses multiple graph elements to create a final layout that more closely resembles a classical biochemical map.

Even after identifying minor metabolites and partitioning the reactions by subsystem, the resulting graph may still contain several overlapping edges that clutter the map. MetDraw attempts to alleviate these problem areas by identifying major metabolites that are more highly connected than other metabolites in the same subsystem. These metabolites are "cloned" and redrawn several times in the subsystem to spread the layout and remove overlapping reactions. The cloned metabolites are connected with a dashed line to aid viewing.

Compartments are bounded by a box and labeled in the final image. Subsystems can either be bounded in a similar manner or left free for tighter packing of the compartment. Transport reactions are added to visualize mass flow between compartments.

4.2.3 Reaction and subsystem classification

The SBML file assigns each metabolite to a compartment. MetDraw requires that compartments provide sufficient "inner" and "outer" attributes to properly nest the compartments. (For example, it must be possible to determine that the nucleus is inside the cytoplasm, which is itself inside the extracellular compartment.)

Reactions containing only metabolites from a single compartment are assigned to that compartment. Reactions with metabolites in two or more compartments are labeled as *exchange reactions*. Exchange reactions are assigned to the innermost compartment of any of its metabolites. Exchange reactions are rendered separately from other reactions. This rendering is controlled by the `SHOW_EXCHANGES` parameter.

Subsystem assignments are very helpful when rendering a map with MetDraw. Subsystems are rendered separately, reducing clutter and edge intersections. The SBML standard does not provide a mechanism for classifying reactions into subsystems. Instead, metabolic models usually encode this information in the "notes" tag for a reaction. MetDraw searches the notes tag for the pattern "SUBSYSTEM:" to identify a reaction's subsystem. Reactions without a subsystem designation are *orphan* reactions. Orphan reactions are rendered as a separate subsystem in each compartment.

4.2.4 Minor metabolite identification

Many "minor" or "currency" metabolites (water, ATP, cofactors, etc.) participate in hundreds of reactions. Drawing these metabolites as only one node per subsystem would create many overlapping edges denoting participation in several reactions. Instead, it is preferable to draw these metabolites separately for each reaction. By default, any metabolite that participates in at least `MINOR_MET_FRACTION` of the total number of reactions in the model is considered a minor metabolite. To designate individual metabolites as major or minor, a listing of metabolites and reaction counts can be generated with the `--count_mets` option to MetDraw. After selecting the minor metabolites, this file can be passed to a subsequent MetDraw run with the `--mets` argument.

4.2.5 Metabolite cloning

Some major metabolites are highly connected in a subsystem. Drawing these metabolites only once per subsystem could create a large, unintelligible cluster. However, treating these metabolites as minor species would remove the important connectivity information from the network map. Instead, MetDraw offers another solution, *metabolite cloning*. If a major metabolite participates in at least `CLONE_LEVEL` reactions in a single subsystem, it is re-drawn (cloned) for each reaction in that subsystem. If the parameter `LINK_CLONES` is true, the clones are connected with a dashed line to represent their connectivity. (The parameters of these links are controlled by the `CLONE_LINK_ATTRS`

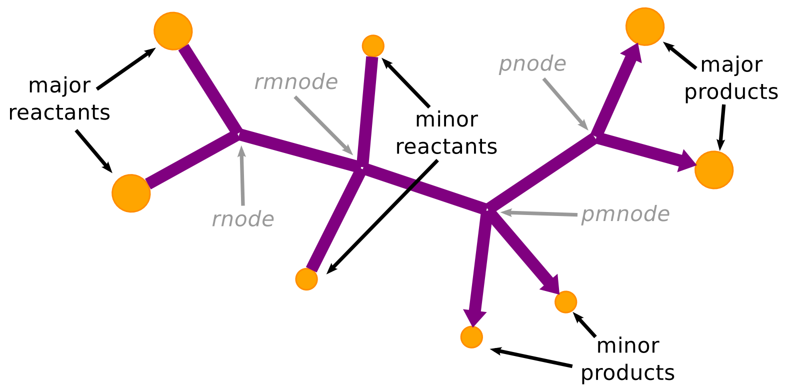


Figure 4.2: MetDraw rendering of a reaction with multiple major and minor reactants and products. Gray labels identify invisible nodes inserted by MetDraw.

parameter.) By default, the clone linking-edges are weights much less than reaction edges, preventing a tight cluster of clones and improving the visualization of the subsystem. To turn off metabolite cloning, set `CLONE_LEVEL` to a very large number.

4.2.6 Reaction layout

After the reactions have been assigned and partitioned, and the metabolites have been classified and cloned, MetDraw begins creating a DOT representation of the network. The DOT representation is rendered by Graphviz to give a pathway map of the model.

MetDraw uses a series of invisible nodes and edges to render maps in a pathway style that is more similar to standard metabolic maps than other graph-theoretic layouts. Five invisible node types are added. For large reactions (those with two or more major and minor reactants and products), the major reactants are connected to a node named `rnode`, the minor reactants to `rmnode`, the major products to `pnode`, and the minor products to `pmnode`. (The actual names appearing in the DOT file will be appended with the reaction ID to distinguish among reactions.) A sample reaction is shown in Figure 4.2 with the default nodes and connectivity.

For smaller reactions, or when MetDraw is run with the `COMPACT` parameter, some of the invisible nodes are excluded. Reactions with only one major and one minor reactant connect both metabolites to `rnode`. In some cases, both reactants and products are connected to a common node `cnode`. The connectivity for larger reactions depends on the number of major reactants (M_r), minor reactants (m_r), major products (M_p), and minor products (m_p). The final layout is determined by the following rules:

1. Start with the first rule in the Table 4.1, and continue until the first match.

2. If $M_r = 0$ and $m_r = 0$, insert an invisible node so $M_r = 1$.

3. If $M_r = 0$ and $m_r > 0$, treat the minors as majors ($M_r \to m_r$, $m_r \to 0$).

4. If $M_r + m_r < M_p + m_p$, or when a pattern is not specified, switch the products and reactants (with `rnode` $\to$ `pnode` and `rmnode` $\to$ `pmnode`).

Reaction Parameters					Connected to...			
M_r	m_r	M_p	m_p	compact	M_r	m_r	M_p	m_p
1	0	≤ 1	0		M_p[1]			
1	≤ 1	≤ 1	0		`rnode`		`rnode`	
> 1	1			true	`cnode`	`cnode`		
> 1	1			false	`rnode`	`rmnode`		
> 1	> 1				`rnode`	`rmnode`		

Table 4.1: Connectivity rules for reactions based on the number of major reactants (M_r), minor reactants (m_r), major products (M_p), and minor products (m_p).

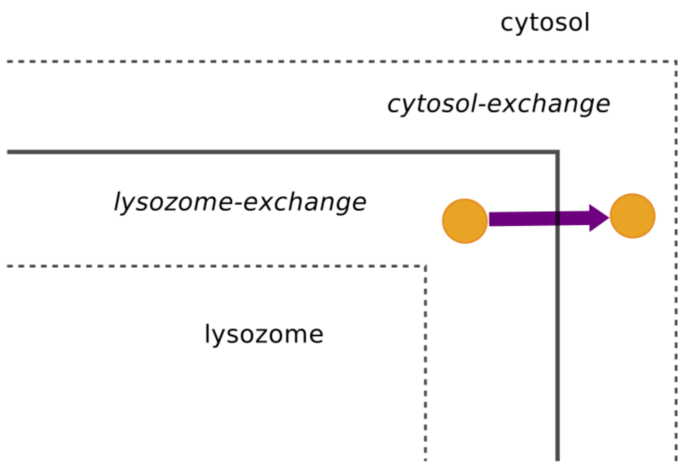


Figure 4.3: Nested compartments to position exchange reactions. The reaction shown transports a metabolite from the lysozome into the cytosol. The dashed lines do not appear in the final visualization. The line surrounding the "lysozome-exchange" compartment forms the visual boundary for the lysozome compartment.

4.2.7 Transport and exchange reaction layout

Compartments (and subsystems if `CLUSTER_SUBSYSTEMS` is true) are rendered as "subgraphs" in Graphviz. Graphviz renders each subgroup independently; these subgraphs are subsequently repositioned in the final layout, but nodes and edges inside each subgraph are not repositioned to improve their location relative to nodes outside the subgraph. This rendering scheme creates a problem for transport and exchange reactions where reactants and products are contained in separate compartments. To render these reactions, MetDraw creates two additional subgraphs around the inner compartment. The first subgraph contains the metabolites for the inner compartment; the second contains metabolites for the outer compartments. As shown in Figure 4.3, the compartment boundary that appears in the final map is actually the border between these additional subgraphs. All other boundaries are invisible.

The `SHOW_EXCHANGES` parameter can be used to hide all exchange reactions. The line and font properties for subgraph boundaries are controlled by the `COMPARTMENT_FONTSIZE`, `SUBSYSTEM_FONTSIZE`, and `SUBSYSTEM_BORDER_STYLE` parameters.

4.2.8 Output and Data Visualization

By default, MetDraw exports an SVG image of the reconstruction. This format is compatible with MetDraw's data visualization features. MetDraw can also export visualization in any other format supported by Graphviz, including PDF, PNG, and EPS. Final changes to the layout can be made with a vector graphic editing program. Users can freely add graphical elements or move metabolites and reactions; the resulting images can still be used to visualize data with MetDraw.

MetDraw allows visualization of fluxomic, metabolomic, gene/protein expression data on metabolic maps. MetDraw accepts a text file containing numerical data for each metabolite or reaction over several conditions. These data are normalized, applied to a colormap, and overlayed on a previously generated metabolic map. If several conditions are given in the same data file, MetDraw creates a separate image for each condition using the same color scale. These images can be combined for visual comparisons and to animate transient conditions.

Many metabolic models contain gene associations for a subset of the model's reactions. These mappings can be used to visualize gene (or protein) expression data on a map of metabolic reactions. MetDraw creates a gene-protein-reaction (GPR) mapping file by extracting gene associations from the "notes" field of the SBML file. The associations are stored in a file with a ".gpr" suffix. This filename can be provided as an argument to MetColor via the `--gprfile` parameter. When a gene name appears in the data file for MetColor, the value is mapped to the corresponding reaction using the GPR. If a reaction's GPR contains multiple genes, the gene expression values are averaged to color the corresponding reaction.

4.3 Implementation

MetDraw is written in Python 2.7 and runs on Linux, Microsoft Windows, and Mac OS X. MetDraw requires Graphviz 2.28 or later. An online interface is available at `http://www.metdraw.com`. The web server is built with Node.js 0.92 and provides asynchronous access through the Jade tempting engine.

4.4 Conclusions

MetDraw provides the first fully-automated pipeline for creating metabolic reaction maps. The software attempts to create maps resembling common pathway layouts without user input. Users can customize the appearance of the map by adjusting layout parameters or editing the final map. The accompanying program MetColor allows streamlined visualization of flux, metabolite, and gene expression data.

Chapter 5

Linking enzyme architectures and reaction networks

5.1 Introduction

Among the most simplifying assumptions of the FBA framework is the representation of genes as binary variables. If a gene is "off", the associated reactions cannot carry any flux. If a gene is "on", the corresponding reactions can carry any physiologically feasible flux. It is therefore assumed that enzymes encoded by "on" genes are always expressed at a level that is never flux limiting. This assumption contradicts observed correlations between enzyme abundance and reaction flux [50]. In reality, gene expression varies continuously and exerts control over the flux through a pathway.

The first attempt at incorporating continuous gene expression data as constraints on reaction fluxes in a COBRA model followed a two-step procedure [51]. First, expression values were combined using the GPR to compute an upper bound for each reaction. The GPR rules were applied by substituting a max operator to replace and, and a min operator instead of or. For example, if a reaction's GPR rule is g_1 and g_2, the corresponding constraint on reaction flux is $v \leq \min\{E(g_1), E(g_2)\}$, where $E(g)$ is the expression of gene g, usually derived from the log-intensity of a microarray probeset [52]. The upper bound on flux implies that flux is limited by the least expressed subunit of an enzymatic complex. Replacing the or operator with max in GPR rules for reactions with isozymes implies the flux is limited only by the abundance of the most highly expressed enzyme.

The approach reinforces observations that flux through metabolic pathways can be approximated with log-linear kinetic laws as functions of enzyme abundance [53]. In bacteria, the close correlation between gene and protein expression for metabolic enzymes [54] supports the use of gene expression as a surrogate for enzyme abundance – the former being much easier to measure with microarray and RNA sequencing technology. Although post-translational modifications do control flux distributions at some metabolic branch points [55], predicting flux distributions in a COBRA model from gene expression alone has proven useful as a first approximation [56].

However, the flux-bounding formalism described above has two key limitations. First, the flux bounds for each reaction are computed independently. Many enzymes catalyze separate reactions through the same catalytic site [57]. Enzyme "promiscuity" creates a dependency between enzyme abundance and the fluxes through all reactions catalyzed by the same enzyme. This coupling is not reflected when flux bounds are computed independently. Second, the mapping from expression values to reactions must be computed before the FBA optimization. The expression of each gene is not included as a decision variable in the FBA problem, limiting the questions that can be asked with the model. This limitation is analogous to the limitations of the two-stage FBA procedure before the invention of SR-FBA and its extensions in TIGER.

In this chapter, I present a new formalism for representing genome-scale metabolic models – Flux/Activity Linked Constraints (FALCON). FALCON extends COBRA models to include continuous variables representing the activity of each enzyme. The logic of the GPR associations is preserved, and reactions catalyzed by the same enzyme are coupled appropriately. I demonstrate how FALCON allows the direct interrogation of the enzymatic network and how this integration improves predictions about network properties.

5.2 FALCON formalism

Building an FALCON model begins with a constraint-based model amenable to analysis by FBA. Each reaction/GPR pair in the FBA model is converted to a set of activity-linked reactions (ALRs). Figure 5.2 describes the ALR conversion for some simple reaction/GPR pairs. If the reaction $A \to B$ is catalyzed by enzyme e, a new species, $\mathit{Activity}(e)$, is introduced to represent the activity of e, as shown in Figure 5.2a. We use "activity" to represent the catalytic activity required to allow one unit of flux through the reaction. The ALR for this reaction becomes $A + \mathit{Activity}(e) \to B$. The stoichiometry constrains the production of B by requiring both one unit of A and one unit of activity from enzyme e.

If, as in Figure 5.2b, the reaction $A \to B$ requires activity from two enzymes to carry flux (e.g., if two protein subunits are required for a functioning enzyme complex), the corresponding ALR is $A + \mathit{Activity}(e_1) + \mathit{Activity}(e_2) \to B$. Notice that if the activity of either e_1 or e_2 is constrained to be zero, the reaction cannot carry flux, as if one the pseudo-reactants $\mathit{Activity}(e_1)$ or $\mathit{Activity}(e_2)$ is not supplied. By contrast, the reaction in Figure 5.2c can be catalyzed by either of two enzymes. In this situation, a separate ALR is constructed for each isozyme: $A + \mathit{Activity}(e_1) \to B$, and $A + \mathit{Activity}(e_2) \to B$. To ensure that these separate reactions preserve the thermodynamic flux constraints in the original FBA problem, we add a constraint requiring that the fluxes through each ALR sum to the flux of the original FBA reaction.

The FALCON procedure can be applied to enzymes that participate in more than one reaction. Consider the example in Figure 5.2d, where a promiscuous enzyme e catalyzes both the reactions $A \to B$ and $C \to D$. The corresponding ALRs both include $\mathit{Activity}(e)$ as a reactant ($A + \mathit{Activity}(e) \to B$ and $C + \mathit{Activity}(e) \to D$). If a constraint is placed on the maximum level of e's activity, a tradeoff is created in the optimization problem. Allowing more flux through the $A \to B$ reaction requires a decrease in the flux through the $C \to D$ reaction, since both reactions draw from the same pool of e activity. Analogous to FBA's use of optimization to determine the flux distribution through a reaction network, the solution of an FALCON model reveals an activity distribution that maximizes the metabolic objective. (And, since the activity is coupled directly to the reaction fluxes, a corresponding optimal flux distribution is calculated simultaneously for FALCON models.)

The FALCON procedure in the proceeding examples can be generalized to all reactions with arbitrarily-complex GPR mappings through the following algorithm:

1. If the reaction is reversible, re-write the reaction as the difference between two irreversible reactions representing the forward and reverse fluxes.

2. Factor the GPR into the disjunctive normal form – a set of clauses composed of conjunctions (e_1 & e_2 & ...) that are joined by or operators. Each clause corresponds to a set of proteins that are sufficient to catalyze the reaction.

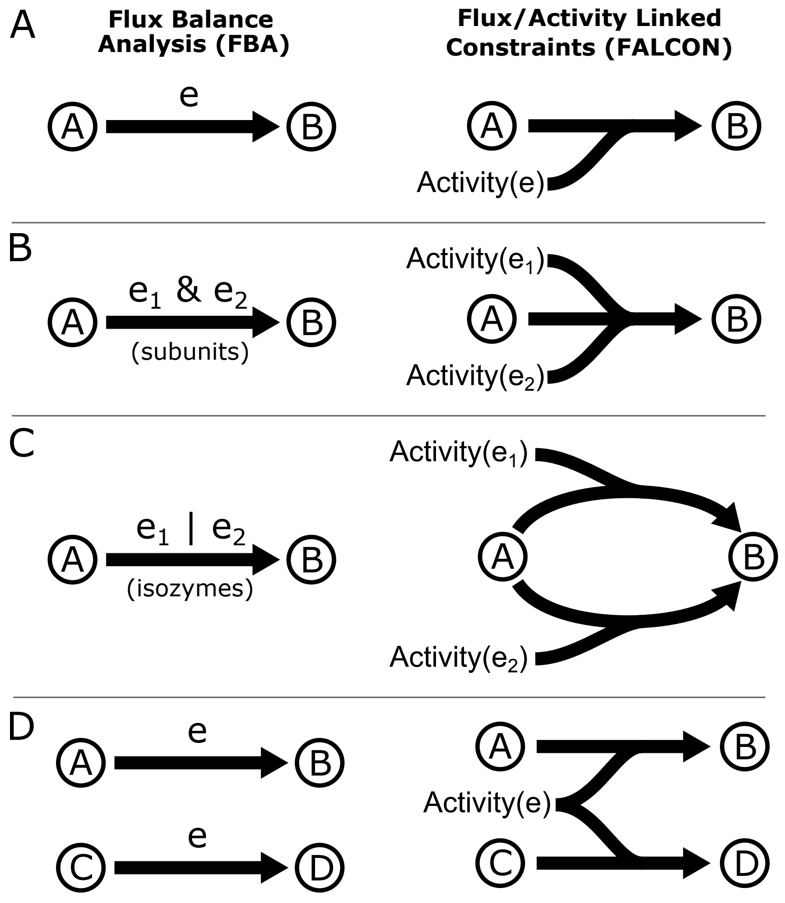


Figure 5.1: FALCON ALR equivalents for simple GPR/reaction pairs. A. Single enzyme. B. Two required enzymes. C. Two independent isozymes. D. One enzyme catalyzing two reactions.

3. For each clause, construct an ALR using the activity of each enzyme in the clause as a pseudo-reactant.

4. Enforce the bounds on the original reaction flux (v_{orig}) by adding the constraint

$$v_{\text{orig}} = \sum_{i \in F} v_i - \sum_{i \in R} v_i \tag{5.1}$$

where F is the set of ALRs for the forward reaction, and R is the set of ALRs for the reverse reaction. This constraint ensures that no matter how many ALRs needed to link the enzymes in a GPR to a reaction, the sum of all fluxes through the ALRs will not exceed the flux bounds on the original reaction.

5. Require that only forward or reverse ALRs can carry flux at the same time through the constraints

$$\sum_{i \in F} v_i \leq v_{\text{orig}}^{\max} I_f \tag{5.2}$$

$$\sum_{i \in R} v_i \leq v_{\text{orig}}^{\min} (1 - I_f) \tag{5.3}$$

where I_f is a binary indicator variable. I_f is equal to one if a positive flux exists in the forward direction.

5.3 Computational complexity and efficiency

The complexity of FBA problems is determined by the complexity of the underlying linear program. FBA requires one constraint for each metabolite and one variable for each reaction in the metabolic network. Because the size of a factored Boolean GPR may be exponentially larger than the original rule [58], I wanted to verify that FALCON models of currently reconstructed organisms do not contain exponentially more constraints or variables than the FBA model from which they were derived. I generated FALCON models for six well-characterized constraint-based reconstructions. The models included three species of bacteria (*E. coli* [59], *P. aeruginosa* [60], and *P. putida* [61]) and two eukaryotic microbes (the yeast *Saccharomyces cerevisiae* [17] and the algae *Chlamdymonas reinhardtii* [62]). As shown in Figure 5.2a, the number of constraints and variables added to create an FALCON model scales linearly with the size of the original FBA model.

The computer runtime to find an optimal solution to a MILP can vary significantly for problems of similar complexity. Unlike linear programs, there is no way to estimate the solution time of a MILP from problem size alone. To ensure that FACON problems are not intractable, I performed simulations to compare the solution time of a FALCON-based model to an FBA model with genes integrated by SR-FBA. Because FALCON models include continuous variables describing enzyme activity and SR-FBA uses binary variables to represent gene state, a direct comparison is not possible without modifications to either formalism. Two modifications to FALCON models allow the direct application of any SR-FBA-based algorithm (GIMME, MADE, etc.). First, a set of binary indicator variables are added to represent the on/off state of each gene. Second, the activity of each enzyme is bounded by the gene indicator, i.e. $\mathit{Activity}(e_i) \leq Mg_i$, where M is a constant larger than the maximum possible activity in the model. This constraint requires that "off" genes have zero activity, but allows "on" genes to carry any feasible activity. In essence, this modified FALCON framework (termed binary-FALCON) is analogous to SR-FBA with the Boolean logic of the GPR encoded by continuous variables instead of binary inequalities.

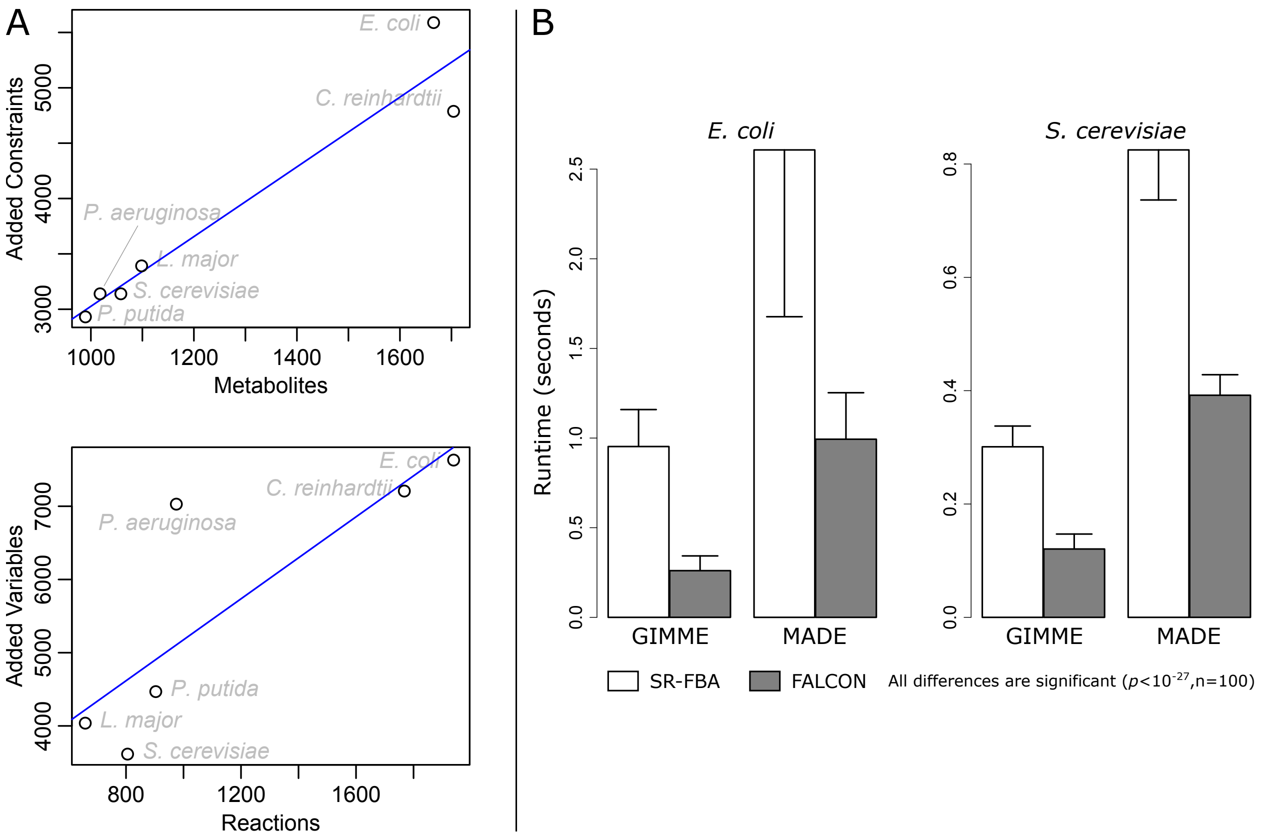


Figure 5.2: Size and complexity of FALCON models. A. The number of variables and constraints in FALCON models is proportional to the size of the underlying COBRA model. B. Average runtime to solve GIMME and MADE problems with the *E. coli* model [59] and random gene expression data. The binary-FALCON formalism was used to allow direct comparison with SR-FBA.

It is important to note that when compared to FALCON, the binary-FALCON representation of a COBRA model will contain thousands of additional constraints and binary variables. The runtime of a binary-FALCON model will always be longer than a FALCON model and therefore is an upper bound on the FALCON performance. This tradeoff is necessary to allow a direct comparison with SR-FBA. Using the *E. coli* metabolic model [59], I performed 100 simulations of the GIMME and MADE algorithms using randomized gene expression data. As shown in Figure 5.2b, the binary-FALCON models solved significantly faster for both algorithms. It is difficult to link runtime with model structure in MILPs, but the differences in these simulations are most likely attributable to the substitution of binary intermediate variables in SR-FBA for continuous ALR fluxes in FALCON.

5.4 Identifying correlated enzyme sets

Many of the FBA-derived algorithms rely on sampling flux distributions in COBRA models [18]. Such algorithms require reaction fluxes to be continuous variables, and are therefore not applicable to the Boolean gene associations underlying the metabolic network. With FALCON, any reaction-centric algorithm can be applied to sample the activity distribution of a metabolic network. The ability to analyze enzyme interactions directly creates new opportunities for interrogating metabolism on a systems level.

One successful method for identifying connectivity in reaction networks is the computation of correlated reaction sets, or "cosets" [63]. At steady state, a reaction in a coset carries flux *if and only if* all other reactions in the coset carry nonzero flux. One example of a coset is a series of reactions in an unbranched, linear pathway. Only two flux distributions are possible at steady state – no flux through any reaction, or a net flux through all reactions in the pathway. If only part of the linear pathway carried flux, intermediate metabolites would accumulate inside the pathway, violating the steady state assumption.

Cosets have been widely used to identify pathways for metabolic engineering [64]. Because reactions in cosets function together, it is hypothesized that enzymes associated with the reactions must be modulated as a group to increase pathway flux. Indeed, a previous study revealed that genes associated with a coset are more likely to show correlated expression than randomly chosen metabolic genes [65]. Like all coset studies, this work mapped genes to cosets by creating the union of all genes associated to at least one reaction in a coset [64]. Mapping genes to cosets *a posteriori* ignores the subtle complexity of the GPR mappings.

I hypothesized that sets of genes identified by sampling enzyme activity distributions will show stronger correlation in expression than sets of genes derived from mapping the GPR rules to reactions in a coset. For example, consider the conversion of 2-phosphoglycerate (2-PG) to phosphoenolpyruvate (PEP). This reaction is the next-to-last step in upper glycolysis in the yeast *S. cerevisiae* and is catalyzed by the product of either of two enolase genes, ENO1 and ENO2 [66]. If the enolase reaction were included in a coset, both ENO1 and ENO2 would be added to the gene list for the coset. However, ENO1 and ENO2 are isozymes, each capable of independently catalyzing the production of PEP [67]. Expression of ENO1 and ENO2 are not correlated, and it is well know that the two genes are activated at distinct times during the cell cycle [68]. If an activity-based sampling were performed, ENO1 or ENO2 would not be found in the same enzyme set, since either enzyme can be active without activity from the other.

Using a FALCON model, it is possible to directly sample activity distributions and identify correlated enzyme sets, or CORES. Sampling is accomplished with a simple Monte Carlo procedure:

1. The feasible bounds for each enzyme's activity are computed by applying an FVA-like algo-

rithm on a FALCON model. The activity of each enzyme is maximized and minimized in two separate optimizations. The objective values of these optimizations are the activity bounds.

2. A random activity distribution is created by uniformly sampling within the bounds of each enzyme.

3. Least-squares constrained optimization is used to find the feasible activity distribution with the shortest Euclidian distance from the random activity distribution. By constraining the optimization with the FALCON model, we are assured that the activity distribution corresponds to a mass balanced flux distribution.

After repeatedly sampling the FALCON model, a correlation matrix is calculated from the activity distributions. Sets of enzymes with high ($r > 0.95$) correlation coefficients are partitioned into CORES.

I calculated cosets and CORES for a genome-scale metabolic model of the bacterium *Pseudomonas aeruginosa*, model iMO1086 [60]. Mapping genes in the GPR associations to reactions in the cosets produced a list of gene sets for comparison to the enzymes in the CORES. The sampling identified 58 CORES and 48 cosets in the iMO1086 model. The 48 gene-mapped cosets are not a subset of the 58 CORES, although 33 of the CORES and cosets contain at least one gene in common.

To test my hypothesis that CORES would include more strongly correlated genes, I performed a meta-analysis of 120 microarray datasets for *P. aeruginosa* downloaded from the Gene Expression Omnibus (GEO). Spearman correlation coefficients were calculated for all pairs of genes in the iMO1086 model. Figure 5.3a shows the average correlation coefficient between all pairs of genes in a CORE or coset. The correlation between any two randomly selected genes from the iMO1086 model is 0.05. The average correlation coefficient between genes mapped to a CORE is significantly higher than the correlation between genes in a coset. As shown in Figure 5.3b, the average correlation in CORES does not depend as strongly on the size of the set. The collection of iMO1086 cosets, however, includes several large groups of poorly correlated genes. When the CORES and cosets are aligned by pairing sets that share genes, CORES usually produce stronger correlations (Figure 5.3b). This indicates that CORES are not identifying completely new sets with correlated genes, but modifying the composition of sets to remove lowly-correlated genes.

5.5 Conclusions

I have developed the FALCON framework for including enzyme activities as continuous variables in COBRA models. FALCON preserves the logic of the GPR rules for each reaction and couples reactions catalyzed by the same enzyme. FALCON models contain integral constraints and are proportional to the size of the original COBRA model. A binary extension to the FALCON framework allows compatibility with all SR-FBA algorithms, and simulations indicate the binary-FALCON models will solve faster than equivalent SR-FBA models. This observation has led to a "fast GPR" extension to the TIGER toolbox where models are built with a hybrid SR-FBA/binary-FALCON strategy for improved performance.

FALCON allows any algorithm using reaction fluxes as decision variables to be applied directly to the enzymatic network. Comparisons between cosets and the FALCON-derived CORES demonstrates how sampling enzyme activities directly improves correspondence with correlations in gene expression data. Importantly, these improvements require only a change in the mathematical formulation of the model. FALCON models are built from the GPR associations already present

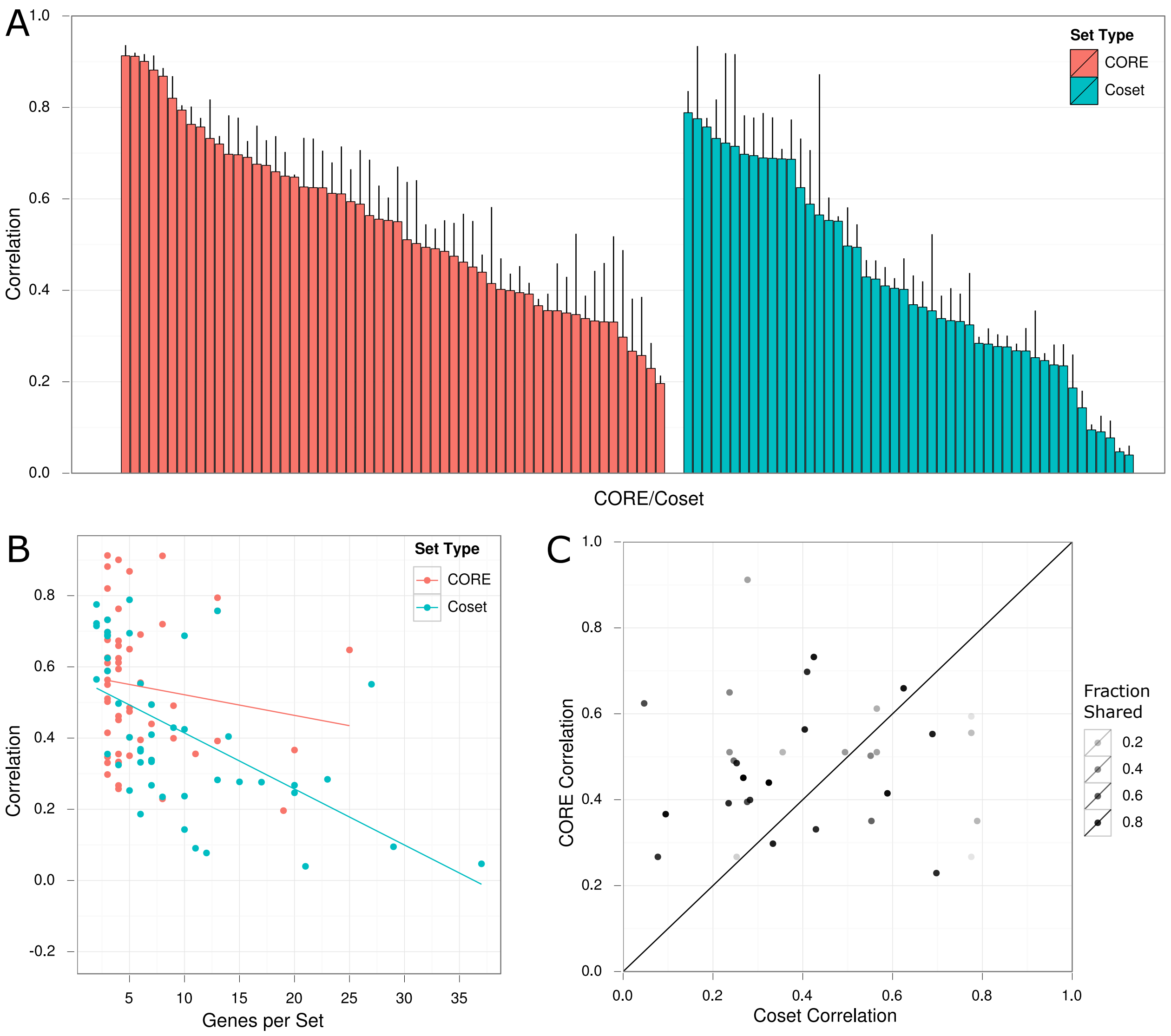


Figure 5.3: Comparisons between CORES and coset-derived gene sets. A. Average Spearman correlation coefficient between two genes in the same set. B. Comparison of average gene pair correlation in aligned CORES and cosets. Circle color indicates the fractional overlap in genes between the CORE and coset. C. Dependence of average gene pair correlation on set size.

in a COBRA model. The improvements seen in the iMO1086 case study were essentially “free”, requiring no additional effort or information from the modeler.

Chapter 6

A miniaturized device for high-throughput phenotypic screening

6.1 Introduction

COBRA models predict biomass yield given a metabolic environment and genotypic state [15]. Validating these models requires data on a microbe's growth rate under a wide range of environments and genetic perturbations. Two assays are commonly used to generate validation data. Phenotypic microarrays (PMs) are multiwell plates with a different metabolic environment in each well [69]. The most widely used PMs are manufactured by Biolog and contain nearly 1000 environments in a set of 96 well plates. Cells in PMs are incubated with a tetrazolium-based clear solution [70]. When the microbes respire, some of the tetrazolium is reduced to formazan, a purple dye. The dye provides a visual readout of cellular respiration [70]. Measuring the effects of genetic perturbations is usually accomplished by screening collections of single gene mutants. Ordered arrays of single gene mutants have been produced for several model bacteria, either by gene deletion [71] or transposon insertion mutagenesis [72, 73, 74]. In a typical screen, multiwell plates containing a defined media are inoculated with the mutant collection. The plates are incubated for several hours, and an optical density measurement is used to score each mutant as "grow" or "no grow" [72].

Despite the widespread use of PMs and mutant collections for validating COBRA models, the readouts of these assays are not optimal. COBRA models predict the rate of biomass production, and biomass synthesis is a more demanding phenotype than cellular respiration. COBRA models can easily make grow/no grow predictions to match mutant array screens. However, several gene deletion strains have reduced fitness, but are still able to grow [75]. These defects would not be identified by a grow/no grow screen, and focusing only on conditionally essential genes unnecessarily reduces the validation data collected from a screen. In both cases, measuring cellular growth by tracking optical density (OD) over time would be a more useful metric. A culture's OD is proportional to its dry cell weight, so the rate of change in OD is proportional to the biomass production rate predicted by the COBRA model [60].

Benchtop plate readers measure OD by absorbance of a fixed wavelength of light. For kinetic measurements over the entire growth cycle, specialized plate readers with internal incubators, shakers, and gas exchangers are necessary to produce a controlled metabolic environment. For

Parts of this chapter are the the subject of US Patent Application No. 61/710,961.

screens involving thousands of environments or strains, reading one microtiter plate at a time is prohibitively low-throughput. Robotic plate changers can be attached to plate readers to cycle multiple plates during one experiment; however, plate changers are not an optimal solution due to their high cost and the need to incubate, shake, and control the environment of the entire robotic assembly. Additionally, reading multiple plates sequentially limits the temporal resolution of the experiment, as it takes minutes to insert, read, and remove each plate.

In this chapter, I describe the development and validation of a novel, miniature plate reader for low-cost, high-throughput phenotypic screening. This "minireader" can measure optical density in PMs, mutant arrays, or any other microbial assay in 96 well plates. I also provide data highlighting the advantages of measuring full growth curves over grow/no grow screening.

6.2 Growth profiling of a *P. aeruginosa* mutant array.

Transposon mutant libraries exist for the PA14 strain of the human pathogen *P. aeruginosa* [72]. As part of a larger project in the Papin lab, I began growth profiling all 5400 mutants in the PA14 array. These data would be used to validate a PA14 COBRA model generated from the existing iMO1086 model of PAO1 strain [60]. (The PA14 model building is currently in progress, led by Jennifer Bartell and Anna Blazier, with help from Nelitza Martinez Vega and Moira Smith.) The protocol for the screen is outlined in Figure 6.1. Twelve multiwell plates were place in a benchtop incubator. Each well contained a PA14 transposon mutant cultured in a synthetic cystic fibrosis medium (SCFM) [76]. Plates were moved one at a time into a Tecan 200 Infinite Pro plate reader for three replicate OD measurements. Readings were recorded every 40 minutes for six hours from pre-exponential to early stationary phase. The data were compiled and growth rates were estimated by linear regression.

The growth rates for 2688 PA14 mutants are shown in Figure 6.2. Approximately 1.4% (37/2688) of the interrupted genes are conditionally essential – the mutant grows in rich media (lysogeny broth), but cannot grow in the defined SCFM (lactate, glucose, 17 amino acids, and salts) [76]. In addition, another 4.6% (123/2688) mutants show a fitness of less than 0.8, i.e., their growth rate is reduced by at least 20%. These growth-reduced mutants would not be identified by a grow/no grow screen. Measuring growth rate rather than growth/no growth generated a 330% increase in the number of data points for validating genome-scale models.

An *in vitro* growth screen generates hypotheses regarding the fitness contribution of individual genes. For example, the PA14 library screen revealed that mutations in the histidine degradation pathway decrease growth rate. The last step of this pathway, the hydrolysis of N-formimino-glutamate to formate and L-glutamate, is catalyzed by either of two enzymes in *P. aeruginosa* – PA5091 (*hutG*), or PA3175 [77]. Previous studies have verified the mechanism of both enzymatic routes, but the relative importance of the two competing enzymes has not been examined [77]. In our screen, the PA3175 mutant grew at a relative rate of 0.87, while the PA5091 mutant showed a relative growth rate of only 0.41. Degradation by PA5091 appears to be the predominant branch of the histidine utilization pathway in SCFM, although neither enzyme is able to fully compensate for the deletion of the other.

The growth-impaired mutants identified by this screen are not evenly distributed across functional subsystems. Figure 6.3 shows the fraction of mutants in each PseudoCAP classification with a defect. (PseudoCAP classifications are an ontology system curated by the *Pseudomonas* community [78].) Nearly one third of all genes involved in cell division show defective growth. The most overrepresented metabolic ontologies are nucleotide biosynthesis (20%) and cofactor synthesis (18%). Genes in central metabolism and carbon metabolism were surprisingly not well represented

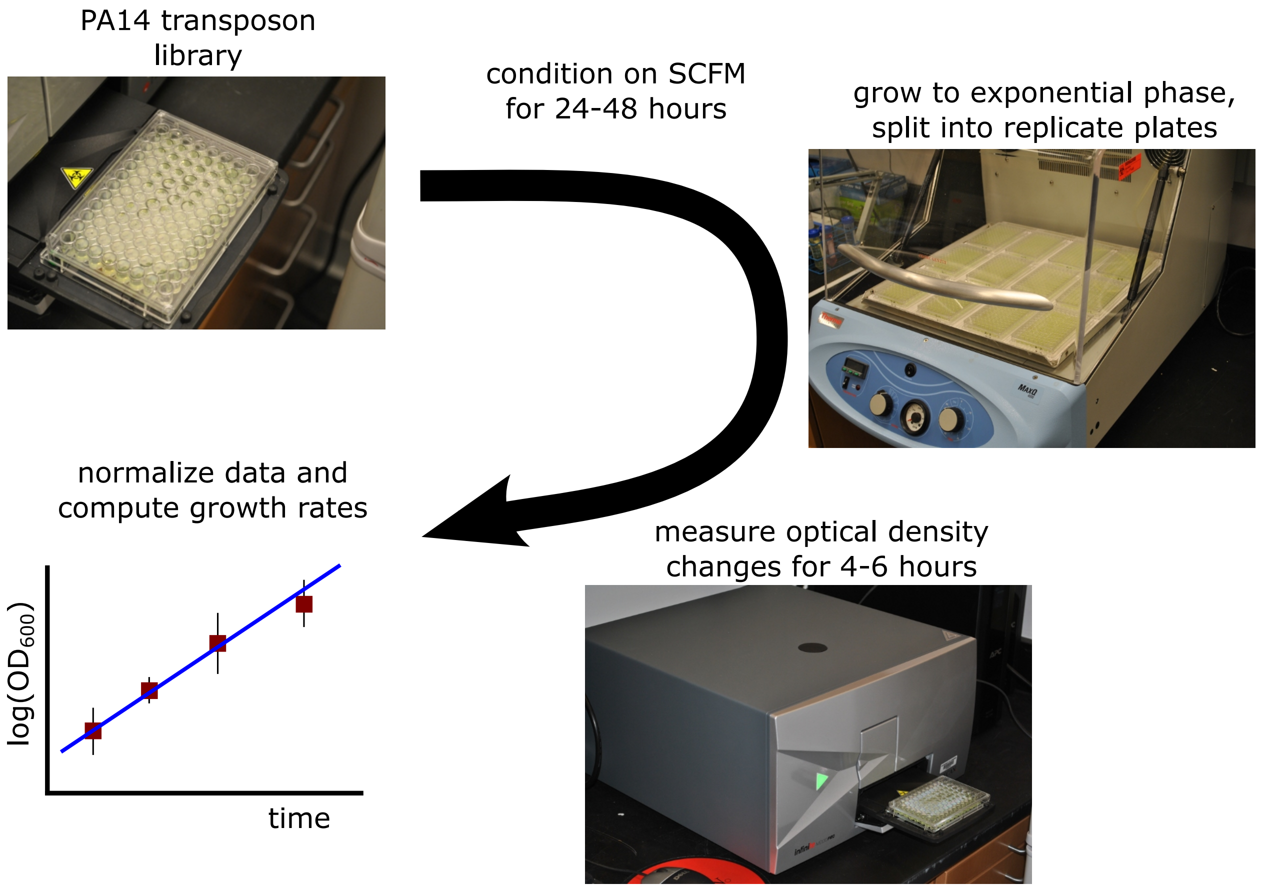

PA14 transposon library
condition on SCFM for 24-48 hours
grow to exponential phase, split into replicate plates
measure optical density changes for 4-6 hours
normalize data and compute growth rates
log(OD600)
time

Figure 6.1: Workflow for manual phenotypic screening. Plates were moved by hand through the reading cycle. Data collection and analysis was completed offline after the experiment completed.

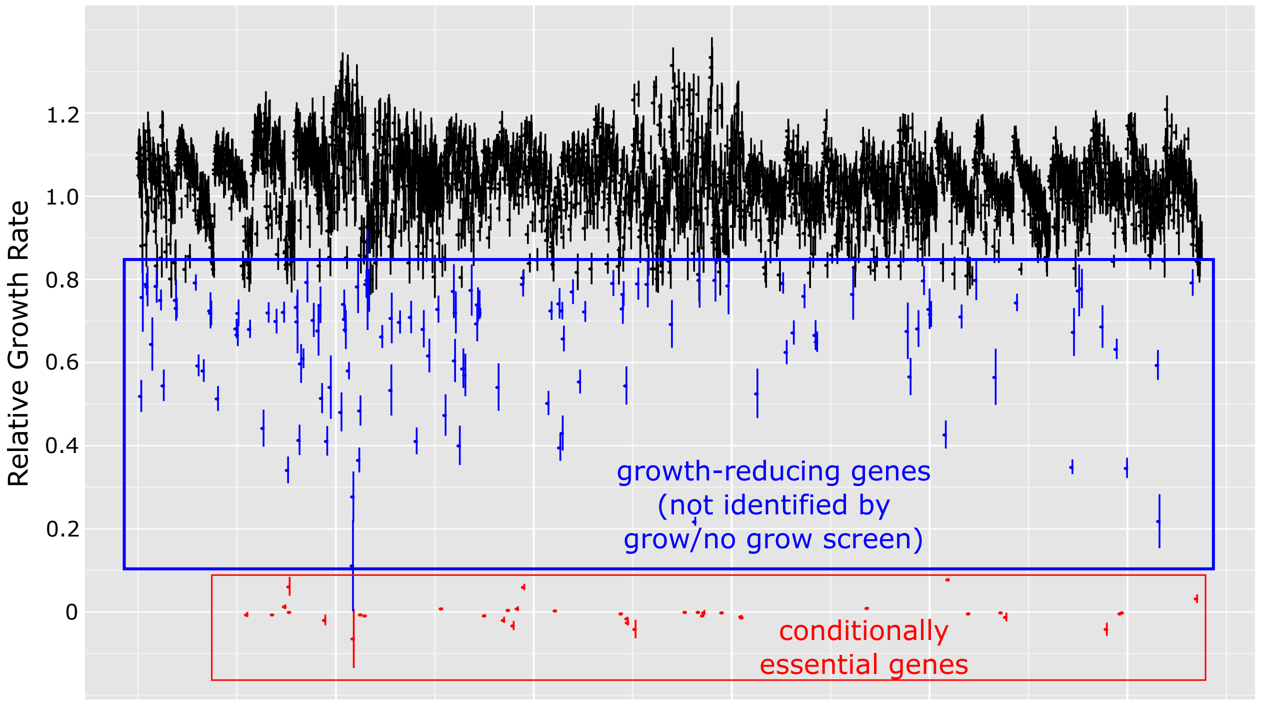


Figure 6.2: Normalized growth rates of PA14 mutants. A normalized growth rate of 1.0 corresponds to wild type growth (no fitness defect). Blue points are growth reduced by at least 20%. Red points are conditionally essential and showed no growth in SCFM. Error bars represent SEM for 3 replicates.

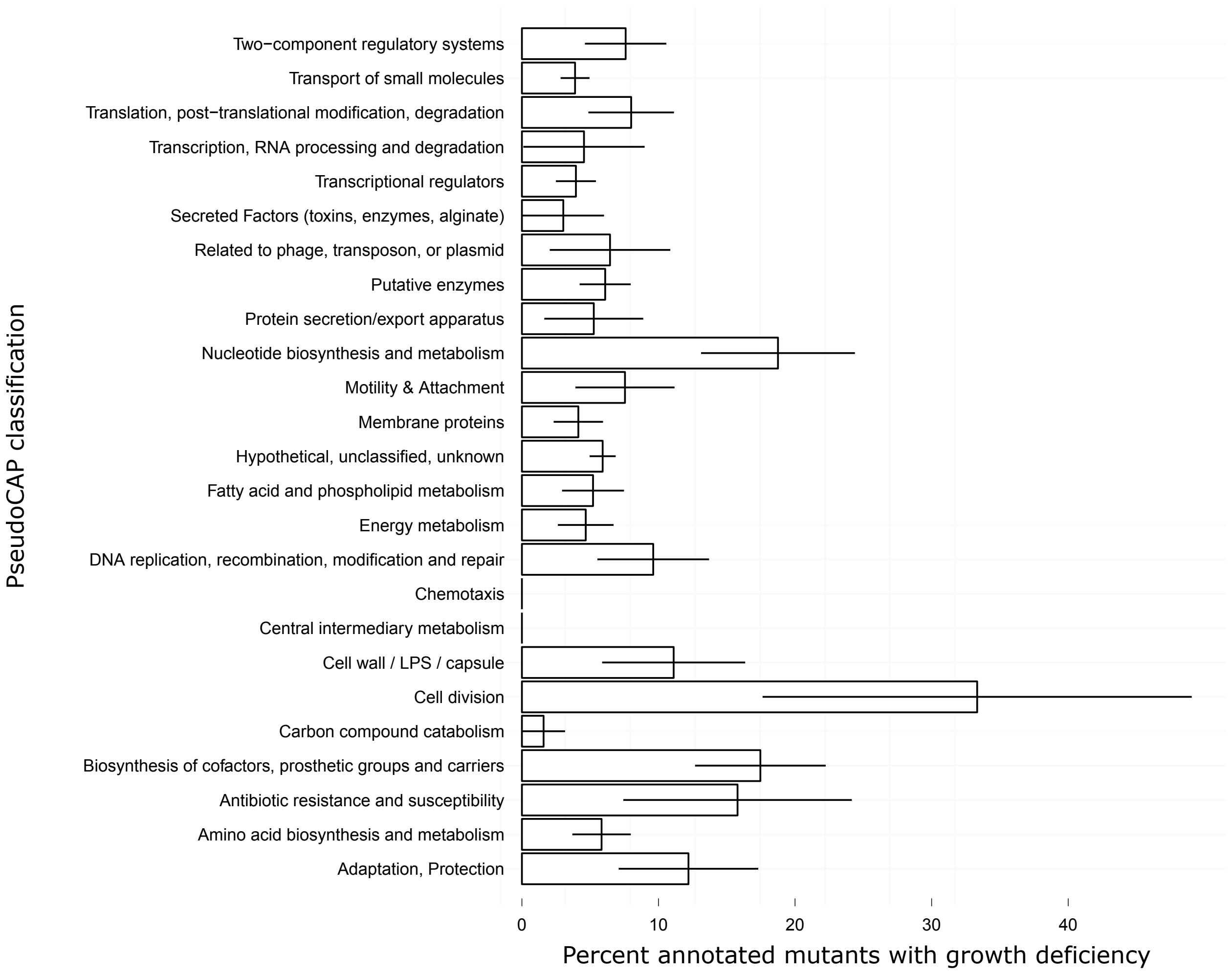


Figure 6.3: PseudoCAP distribution of fitness defective PA14 mutants. Horizontal bars indicate the fraction of mutants screened that were defective in each category. Error bars represent standard error of the proportion.

in the defective list. Their absence is mostly likely attributable to the catabolic versatility of *P. aeruginosa* and the presence of multiple carbon sources in SCFM.

Although this screen produced a validation dataset for the PA14 modeling project, the data contain some technical issues. The mutants in Figure 6.2 are ordered chronologically. The data display a periodicity caused by batch effects as the plates are moved from the incubator to the plate reader. This is most likely attributable to condensation forming on the plate during reading, which skews the OD of the wells at the end of the plate. Troubles with condensation are not unique to our plate reader or incubator setup; other groups using plate moving robots struggle with the same problem[1].

Another problem with the plate-swapping protocol appears as highly-variable rates in the middle of Figure 6.2. Because reading each plate takes at least three minutes, the fastest all 12 plates can be read is every 36 minutes. For fast-growing bacteria like *P. aeruginosa*, 36 minute readings place only three or four readings in exponential phase. If timed correctly four reads are captured, but mis-timing the reads can leave only three usable points. As seen in Figure 6.2, estimating

[1]Tecan, Inc., personal communication

the exponential growth rate with only three points significantly increases the variability of the measurements.

6.3 Developing a novel plate reader

Measuring OD requires a relatively simple electronic circuit. A light-emitting diode (LED) produces light focused on a peak emission wavelength, usually 600 nm. The light passes through the sample where it is partly absorbed by the suspended cells. The transmitted light is translated to a proportional voltage by a phototransistor (PT). The absorbed light is calculated as the difference between the emitted and transmitted light. OD is defined as the logarithm of the absorbance (relative to a transparent standard).

Most plate readers measure fluorescence, luminescence, and other optical properties in addition to OD. Fluroescence in particular requires expensive laser and photomultiplier equipment to detect faint emissions from samples. The optics equipment in plate readers are mounted to a robotic arm that aligns the emitter/receiver pair sequentially over the wells in a microtiter plate. The robotics and optical hardware comprises the majority of the cost and size of the plate reader. By focusing only on measuring OD in 96 wells plates, much of the hardware in a plate reader can be removed. LED/PT pairs are inexpensive, especially those components with peak wavelengths in the infrared (IR) range. (IR LEDs and PTs are found in many consumer electronics and are therefore mass produced at low cost.) I hypothesized that replacing the robotics in a plate reader with a fixed array of 96 LED/PT pairs would produce a small, low-cost OD reader.

A prototype "minireader" is shown in Figure 6.4a. A printed circuit board (PCB) holds IR LEDs and PTs spaced to match a 96 well plate.[2] Besides offering support for the optoelectronic components, the PCB is engineered with two cost-saving features. First, rather than connect each LED/PT to a separate amplifier and analog-to-digital convertor (ADC), the PCB uses a grid of analog multiplexers to connect all pairs to a single anode and cathode. These two wires are connected to a single amplifier/ADC chip, and seven digital logic lines selectively connect each well to the sensing hardware. Second, the layout of the PCBs is identical for both the LED and PT (bottom and top) boards. The LED and PT circuits differ, but these differences have been moved to a common controller board that houses the sensor hardware. The dual-use PCBs require only one template for manufacturing, significantly reducing manufacturing cost.

The LED and PT boards are connected by ribbon cables to a controller board (Figure 6.4b). This board contains

- An integrated analog front end (AFE). The AFE (Texas Instruments LMP90100) contains a 24-bit ADC with pV resolution, corresponding to nano-OD theoretical sensitivity. The AFE also includes automatic background correction to adjust for temperature variation and programmable gain to allow calibration by software.
- A 16-bit micro controller (Arduino Micro). The microcontroller coordinates well switching, reads the voltage from the AFE, and stores or transmits data.
- A voltage regulator, allowing the minireader to be powered by battery or DC current from a wall-mounted power supply.
- Communication links. A category 5 ("ethernet style") cable provides serial connectivity. A XBee wireless chip can be added for cable-free monitoring.

[2]Spacing in a microtiter plates is standardized by a working group of the Society for Laboratory and Screening to facilitate interoperability among plates and plate readers.

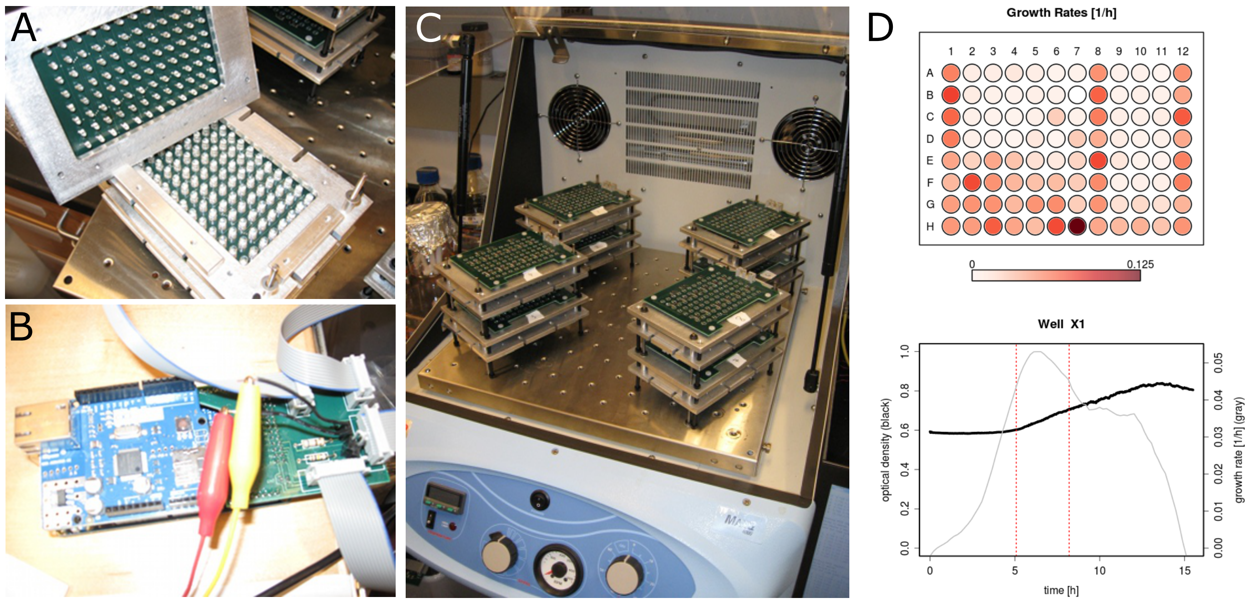


Figure 6.4: Miniaturized plate reader for phenotypic screening. A. Fixed-position IR LED/PT pairs measure optical density at 840nm. Each reader is slightly larger than a 96 well plate. B. A controller board links optoelectronics, hardware sensors, and communication links. C. Eight readers stacked in a benchtop incubator. D. Companion web-based analysis software provides visualizations of entire plates (top) or single wells (bottom).

- Data storage. A microSD card provides long-term storage and a backup cache to prevent data loss during transmission.

The multiplexed PCB design greatly reduces board complexity and cost, but requires that each well be read sequentially. Switching and reading wells on the minireader is fast enough to avoid any issues with the sampling rate. Because the minireader has no moving parts, the time to read each well is a combination of the multiplexer switching time (10 ns) plus the read time of the ADC (100 μs). The entire plate can be read in under 0.1 s, far shorter than the timescale for bacterial growth.

Multiple minireader units can be placed in a single benchtop incubator (Figure 6.4c). Simultaneous OD reading during incubation removes the batch effects observed during plate transfer. By placing the entire reader in the incubator, the environment can be controlled without additional equipment. The minireader can fit inside an anaerobic chamber, on a shaker to improve aeration, or inside a BSL containment suite. The battery power allows the minireader to function in resource-limited laboratories and during remote fieldwork studies.

Data from the controller board is transmitted to an embedded computer in the device. The computer hosts a web server broadcasted over a standalone wireless access point. Users can connect to the minireader's wireless network with an internet browser for real-time data visualization and export (Figure 6.4d). The server computes instantaneous growth rates for each well and reports the maximum growth rate during exponential phase.

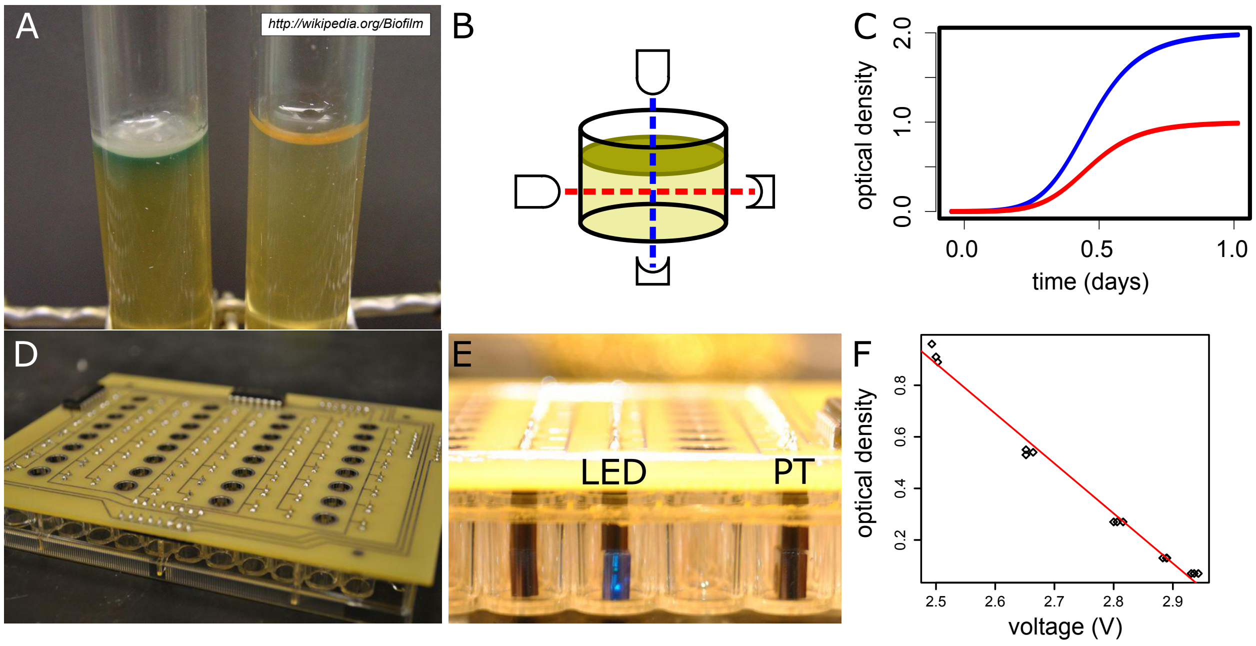


Figure 6.5: A novel device for quantifying biofilm production rate. A. Optically dense biofilms on the aerobic surface of PAO1 cultures [81]. B. A dual-axis measurement strategy. Emitter/detector pairs are place either above and below or in adjacent wells of a microtiter plate. C. Expected optical density transients for horizontal (red) and vertical (blue) sensors. Optical density increases faster along the vertical axis due to BLS accumulation on the aerobic interface. D. Prototype biofilm reader atop a 96 well plate. E. Cutaway view of a 96 well plate. LED/PT pairs are suspended from the PCB and read across the sample well. F. Calibration curve for horizontal OD readings.

6.4 Future work: biofilm monitoring

A hallmark of many bacterial infections is the production of a thick, mucoid biofilm. Previous studies have used genome-scale metabolic models to explore biofilm production capabilities [79], but few data exist to validate the model predictions. Existing assays for biofilm growth quantify cellular attachment rather than production rate of biofilm and biofilm-like substance (BLS) [80]. I have developed a high-throughput assay for quantifying the production rate of biofilm and BLS. My method is based on the observation that biofilm largely accumulates on the top of a bacterial culture at the aerobic interface (Figure 6.5a). The compact biofilm layer has significantly higher optical density than the rest of the culture. I have adapted my plate readers to measure optical density along both the horizontal and vertical axes (Figure 6.5b). As the biofilm forms at the top of the well, the optical density measured by the vertical sensors will increase faster than the optical density measured horizontally (Figure 6.5c). The difference between these optical density curves is proportional to the total biofilm produced at any time, and the accumulation of the integrated difference is proportional to the biofilm production rate. Screening a mutant library for biofilm production with the device will generate four growth parameters for each mutant: 1.) exponential growth rate, 2.) time of biofilm initiation, 3.) biofilm and BLS production rate, and 4.) total biofilm and BLS produced.

Figure 6.5d shows a prototype of the biofilm monitor. A PCB holding the horizontal LED/PT pairs sits atop a 96 well plate. Holes in the PCB allow vertical OD readings when the plate/PCB combination is placed in a minireader. The sideview LED/PT pairs are suspended from the PCB into wells neighboring the sample well (Figure 6.5e). Despite reading through four curved surfaces, the horizontal LED/PT pair shows excellent linearity over the OD range of interest for bacterial cultures (Figure 6.5f).

While the majority of the BLS and total biofilm mass accumulates at the aerobic interface, some biofilm formation occurs on the walls and bottom of the culture vessel. Biofilm on the walls can decrease the biofilm predicted by my proposed two-axis method. I will quantify the biofilm attached to the vessel walls by traditional crystal-violet staining protocols [82]. These data will be used to normalize the biofilm predictions for each well.

6.5 Conclusions

The minireader device for growth screening provides a low-cost, scalable platform for phenotypic profiling. The prototypes described in this chapter cost under $500 US to produce, over tenfold less expensive than the smallest commercially-available plate reader. The 16 minireaders in the Papin lab can complete a full phenotypic microarray or PA14 mutant library screen in under one week. The entire minireader set costs less than the least expensive Tecan plate reader, not including the robotic system necessary to achieve a comparable throughput.

Although the initial design focused on measuring OD, any absorbance assay could be incorporated by changing the LED/PT pairs. We have not thoroughly investigated building a fluorescent minireader, since the 96 pairs of fluorescent optoelectronics would most likely be prohibitively expensive.

Most importantly, the ability to assay a full growth curve, rather than respiration or simple growth, provides a wealth of new data for validating COBRA models. The pilot screen of the PA14 mutant library demonstrates the added benefits of continuous growth monitoring.

6.6 Acknowledgements

This chapter summarize years of work by several researchers. I have the fortune of leading an excellent team, and I wish to acknowledge their numerous contributions to this project.

Tom Moutinho designed the mechanical frame for the first prototypes and the layout for the revised controller board. He assembled many of the devices with help from Matt Biggs and Matt Jenny. Matt Biggs also wrote networking software for the first controller boards. Bonnie Dougherty wrote much of the microcontroller and communication code for the second generation prototypes. She currently manages a collaboration with SPARK Engineering and the UVA Coulter Foundation to commercialize the technology. David Chen organized focus groups to gather feedback on initial designs. Several researchers from UVA attended these sessions, including members of my thesis committee. The project is funded by a Coulter Foundation grant to Jason Papin and Erik Hewlett.

Jennie Bartell dared to prepare one of the first full-scale test screens with the readers, and was gracious enough to continue speaking to me after all the data were lost in a software crash at 3am. Katherine Estep spent weeks moving plates between an incubator and a plate reader, providing part of the motivation for this work.

Chapter 7

Summary

COBRA modeling has expanded into widespread applications, including metabolic engineering, infectious disease, and ecology [11]. The mathematical framework underlying these analyses has remained largely unchanged during the past two decades. The view of the genome as a "parts list" for metabolic reactions ignores the dynamic role gene play in regulating cell function. My work has focused on updating the COBRA formalism to give genetic networks equal status in the metabolic model. My goal was to prepare the COBRA framework for new roles in the era of functional genomics.

Identifying inconsistencies in the *S. cerevisiae* regulatory network and the results of the CORES analysis highlights an important consideration for large models. Often, the quality of a model's predictions are assumed to be a function of the input data and the knowledge of the underlying biology. My results suggest that a mathematical framework can also impact the accuracy of a model and its ability to represent a biological system. The computational details of a model are important, and we should look beyond mathematical convenience when choosing an appropriate framework.

I believe the frameworks and algorithms in this thesis share a common strength in their universality and ease of use. TIGER can add logical rules of any complexity to a COBRA model, allowing rule formulations that best represent the underlying biology. MADE removes the need to assign arbitrary thresholds to expression data and is compatible with any model that can be assembled with TIGER. MetDraw is the first fully-automated software for mapping an entire COBRA model. Any COBRA model can be reformulated as a FALCON model, allowing all algorithms designed for reaction networks to be directly applied to the enzymatic architecture. In all cases, I have focused on providing solutions at no additional cost to the modeler. As a result, the software I have published so far has been downloaded hundreds of times, and I have corresponded with research groups on four continents who are using the tools. I hope my work continues to find wide use outside of my own research interests.

One important application of COBRA models is the search for novel antibiotics. Microbial metabolism provides a link between cellular growth rate, biofilm secretion, and virulence factor production. Pathogens must dynamically allocate metabolic resources among these processes during the course of an infection. Rather than study these processes independently, a metabolic model can explore the tradeoffs in these subsystems.

My work offers new tools to address the complexity of modeling infection-causing microbes. MADE and TIGER allow genome-scale models to be tailored to a single strain or clinical isolate. Pathogenic bacteria display enormous genetic and phenotypic diversity, and capturing these differences with a mechanistic model is an important step towards a "personalized medicine" approach to model-guided drug discovery. An *in silico* drug screen is fast and inexpensive, allowing an enormous number of strain-specific models to be screened within hours. By contrast, the cost of large-scale experimental drug screening limits traditional drug discovery approaches to a few well-characterized reference strains.

The scalability of computerized drug screening creates opportunities for combinatorial screening to identify synergistic drug pairs. Ideally, models could identify sets of drug targets that are likely to synergize. This goal is difficult to achieve with a standard COBRA model due to the binary representation of essential genes. In models with binary gene-reaction mappings, removing an essential gene results in a no-growth condition. There is no room for synergy in such conditions, as the pathogen is already dead. Finding genes that interact with essential genes requires the continuous gene-reaction mapping provided by FALCON. The CORES algorithm for FALCON models identifies genes that are functionally coupled to other drug targets. These gene sets may be promising leads for synergistic gene pairs.

The power of computational drug screening can shift the drug discovery bottleneck from target identification to target validation. Experimentally testing "hits" in multiple strains using many combinations of antibiotics requires large phenotypic screens. In contrast to drug discovery screens, validation screens typically require high resolution and accuracy. The miniaturize plate readers I developed are ideally suited for this task. These devices provide the necessary experimental throughput for validating predictions from genome-scale models, demonstrating how computational and experimental technologies must advance in parallel.

Appendix A

Supplementary Computational Methods

Complete TIGER Grammar

This section describes the syntax used for all rules that can be parsed by TIGER. The grammar was designed to resemble Boolean operations in common programming languages.

Expressions are composed of *atoms* and *numerics* joined by *operators*. Atoms are either

- Identifiers composed of a sequence of any characters except `~!&|()<>=` or whitespace. Identifiers are case sensitive and cannot overlap with an operator (see below).
- Quoted strings of any characters. Either single (`'`) or double (`"`) quotes may be used. Quotation marks appearing inside a quoted string may be escaped with a backslash character (\).

Numerics are integer or floating point numbers that appear in conditionals. Scientific notation is allowed in the forms '`1e-10`' or '`-2E5`'. Some metabolic models use gene identifiers that could be interpreted as numbers, e.g. the Entrez gene identifier. TIGER allows two solutions to this potential ambiguity. First, numbers in quotation marks are interpreted as atoms. Second, the `parse_string` function allows numeric parsing to be turned off for all identifiers.

Two atoms or numerics can also be compared using the comparison operators `=`, `>`, `<`, `>=`, `<=`, and `~=`. The expressions are called *conditionals* and correspond to a value of one (true) if the condition is satisfied, and a zero (false) otherwise.

Atoms and conditionals are joined by the Boolean `AND`, `OR`, and `NOT` operators to create *expressions* of arbitrary complexity. Two expressions are joined by the `=>` (if) or `<=>` (if and only if) operators to form a *rule*.

The TIGER operators obey the following order of precedence:

highest	`NOT`
	`=`, `>`, `<`, `>=`, `<=`, and `~=`
	`AND`, `OR`
lowest	`=>`, `<=>`

All operators are left-associative. Parentheses can be used to change the order of operations in rules. Without parenthesis, the following expressions are parsed as shown

`a AND b AND c`	$\rightarrow$	`(a AND b) AND c`
`a > 2 OR b`	$\rightarrow$	`(a > 2) OR b`
`NOT a AND b`	$\rightarrow$	`(NOT a) AND b`
`NOT a <= 3`	$\rightarrow$	`(NOT a) <= 3`
`a AND NOT b OR c`	$\rightarrow$	`(a AND (NOT b)) OR c`

The `add_rule` function allows two different interpretations of the `not` operator when applied to a multilevel variable. The *binary* form considers any nonzero value of x to be true, while the *inverted* form requires that $x + \textsf{not } x = x^{\max}$. The differences are described in the following table for $x \in \{0, 1, 2, 3\}$:

x	not x, binary	not x, inverted
0	1	3
1	0	2
2	0	1
3	0	0

Notice that for binary variables, both forms are equivalent.

In order to increase the number of existing COBRA models that can be parsed by TIGER without modification, synonyms are allowed for the following operators:

Operator	Other allowed forms
`AND`	`and`, `&`, `&&`
`OR`	`or`, `|`, `||`
`NOT`	`not`, `~`, `!`
`=>`	`->`, `==>`, `-->`, `if`, `IF`
`<=>`	`<->`, `<==>`, `<-->`, `iff`, `IFF`
`=`	`==`
`~=`	`!=`, `<>`

The following formal grammar incorporates the above information and describes all rules that can be parsed and translated by the TIGER platform.

$$
\begin{array}{rcl}
\textit{rule} & \rightarrow & \textit{expr}\ (\Rightarrow \mid \Leftrightarrow)\ \textit{expr} \\
\textit{expr} & \rightarrow & \textsf{not}\ \textit{expr} \\
 & \mid & \textit{expr}\ \textsf{and}\ \textit{expr} \\
 & \mid & \textit{expr}\ \textsf{or}\ \textit{expr} \\
 & \mid & \textit{cond} \\
 & \mid & \textit{id} \\
\textit{cond} & \rightarrow & \textit{id}\ \textit{op}\ (\textit{id} \,|\, \textit{number}) \\
\textit{op} & \rightarrow & (\leq \mid < \mid = \mid \sim= \mid > \mid \geq) \\
\textit{number} & \rightarrow & \texttt{[+-]!\textbackslash d+\textbackslash .!\textbackslash d*([eE]+[+-]!\textbackslash d+)!} \\
\textit{id} & \rightarrow & \texttt{[\^{}\~{}!\&|()<>=]+} \\
 & \mid & \texttt{"([\^{}"]|\textbackslash ")+"} \\
 & \mid & \texttt{'([\^{}']|\textbackslash ')+'}
\end{array}
$$

Converting Rules to Linear Inequalities

As described in the main text, TIGER allows generalized rules of the form

$$f(x, y, \ldots) \quad (\Rightarrow \mid \Leftrightarrow) \quad g(x, y, \ldots) \tag{A.1}$$

We define an atomic expression as either x or $\mathsf{not}\ x$, where x is a discrete variable. If $g(x, y, \ldots)$ is not atomic, we define an indicator variable I such that

$$f(x, y, \ldots)(\Rightarrow \mid \Leftrightarrow)\, I \tag{A.2}$$

$$g(x, y, \ldots) \Leftrightarrow I \tag{A.3}$$

Using the recursive substitution procedure outlined in the text, rules (A.2) and (A.3) can be reduced to one of the following forms:

$$x \quad (\Rightarrow \mid \Leftrightarrow) \quad I \tag{A.4}$$

$$x \;\mathsf{and}\; y \quad (\Rightarrow \mid \Leftrightarrow) \quad I \tag{A.5}$$

$$x \;\mathsf{or}\; y \quad (\Rightarrow \mid \Leftrightarrow) \quad I \tag{A.6}$$

$$x \,\langle op \rangle\, y \quad (\Rightarrow \mid \Leftrightarrow) \quad I \tag{A.7}$$

where x, y, and I are all atomic. We now describe how each of these cases is converted into a set of linear inequalities.

Binary variables

If x and y are binary, the following transformations are used:

$$\begin{aligned}
x \Rightarrow I \quad &\to \quad I \geq x \\
x \Leftrightarrow I \quad &\to \quad I = x \\
x \;\mathsf{and}\; y \Rightarrow I \quad &\to \quad 2x + 2y - 4I \leq 3 \\
x \;\mathsf{and}\; y \Leftrightarrow I \quad &\to \quad \begin{cases} x \;\mathsf{and}\; y \Rightarrow I \\ 2x + 2y - 4I \geq -1 \end{cases} \\
x \;\mathsf{or}\; y \Rightarrow I \quad &\to \quad -x - y + 3I \geq 0 \\
x \;\mathsf{or}\; y \Leftrightarrow I \quad &\to \quad \begin{cases} x \;\mathsf{or}\; y \Rightarrow I \\ -x - y + 3I \leq 2 \end{cases}
\end{aligned}$$

Multilevel or continuous variables

The following transformations are used when either x or y are positive integers or continuous real values. If both x and y are discrete ($x \in \{0, \ldots, x^{\max}\}$, $y \in \{0, \ldots, y^{\max}\}$), I will be discrete and valued in the set $\{0, \ldots, I^{\max}\}$. If either x or y is continuous with $x \in [x^{\min}, x^{\max}]$ and $y \in [y^{\min}, y^{\max}]$, then I is a continuous variables with bounds $[I^{\min}, I^{\max}]$.

The rules $x \Rightarrow I$ and $x \Leftrightarrow I$ follow the same transformations as in the binary case. The junction operators are interpreted as

$$\begin{aligned}
x \;\mathsf{or}\; y \Rightarrow I \quad &\to \quad I \geq \max\{x, y\} \\
x \;\mathsf{or}\; y \Leftrightarrow I \quad &\to \quad I = \max\{x, y\} \\
x \;\mathsf{and}\; y \Rightarrow I \quad &\to \quad I \geq \min\{x, y\} \\
x \;\mathsf{and}\; y \Leftrightarrow I \quad &\to \quad I = \min\{x, y\}
\end{aligned}$$

These transformations are implemented with the following inequalities:

$$x \text{ or } y \Rightarrow I \quad \rightarrow \quad \begin{cases} I \geq x \\ I \geq y \end{cases}$$

$$x \text{ or } y \Leftrightarrow I \quad \rightarrow \quad \begin{cases} x \text{ or } y \Rightarrow I \\ x > y \Leftrightarrow I_{\text{aux}} \\ I \leq x + (x^{\max} - x^{\min})(1 - I_{\text{aux}}) \\ I \leq y + (y^{\max} - y^{\min})I_{\text{aux}} \end{cases}$$

$$x \text{ and } y \Rightarrow I \quad \rightarrow \quad \begin{cases} x > y \Leftrightarrow I_{\text{aux}} \\ x - (x^{\max} - x^{\min})I_{\text{aux}} \leq I \\ y - (y^{\max} - y^{\min})(1 - I_{\text{aux}}) \leq I \end{cases}$$

$$x \text{ and } y \Leftrightarrow I \quad \rightarrow \quad \begin{cases} x \text{ and } y \Rightarrow I \\ I \leq x \\ I \leq y \end{cases}$$

where the axillary variable I_{aux} is binary. The bounds on I are

	$I^{\min}$	$I^{\max}$
x or $y \Rightarrow I$	$\max\{x^{\min}, y^{\min}\}$	$\max\{x^{\max}, y^{\max}\}$
x or $y \Leftrightarrow I$	$\max\{x^{\min}, y^{\min}\}$	$\max\{x^{\max}, y^{\max}\}$
x and $y \Rightarrow I$	$\min\{x^{\min}, y^{\min}\}$	$\max\{x^{\max}, y^{\max}\}$
x and $y \Leftrightarrow I$	$\min\{x^{\min}, y^{\min}\}$	$\min\{x^{\max}, y^{\max}\}$

Conditionals

Consider the simple rule

$$\sum_i \phi_i x_i \geq k \Rightarrow I \tag{A.8}$$

that is a linear inequality in terms of variables x_i with constant coefficients ϕ_i. We convert this expression to an equivalent set of rules

$$\sum_i \phi_i x_i + s = k \tag{A.9}$$

$$s \leq 0 \Rightarrow I \tag{A.10}$$

where s is a continuous *slack variable* that is strictly positive if and only if $\sum_i \phi_i x_i < k$. We can represent rule (A.10) with the inequality $I + s > 0$. Because MILP solvers do not allow strict inequalities, this constraint is implemented as $I + s \geq \epsilon$, where ϵ is a very small, positive number.

The upper and lower bounds, u and l, on the quantity $\sum_i \phi_i x_i$ are

$$u = \sum_i \max\{\phi_i x_i^{\max}, \phi_i x_i^{\min}\}$$
$$l = \sum_i \min\{\phi_i x_i^{\max}, \phi_i x_i^{\min}\}$$

Thus, the bounds $s \in [k - u, k - l]$ are sufficient to allow the slack variable s to always satisfy the constraint (A.9).

To represent the rule

$$\sum_i \phi_i x_i \geq k \Leftrightarrow I \tag{A.11}$$

we begin by following the above procedure and add constraints (A.9) and (A.10). Then, to enforce that $I \Rightarrow s \leq 0$, we require that

$$s \leq (k - l)(1 - I) \tag{A.12}$$

This completes the "if and only if" implication.

Finally, we show that all conditionals parsed by TIGER can be represented in terms of rules (A.8) and (A.11) through the following substitutions:

$$\begin{aligned}
x > y &\quad\rightarrow\quad \textsf{not } (-x \geq -y) \\
x < y &\quad\rightarrow\quad \textsf{not } (x \geq y) \\
x \leq y &\quad\rightarrow\quad -x \geq -y \\
x = y &\quad\rightarrow\quad (x \geq y) \textsf{ and } (-x \geq -y) \\
x \neq y &\quad\rightarrow\quad \textsf{not } ((x \geq y) \textsf{ and } (-x \geq -y))
\end{aligned}$$

The difference operator

Some algorithms require the inclusion of the nonlinear absolute value operator, often to measure the difference between two variables, $|x - y|$. TIGER includes the `add_diff` function that adds a variable $d = |x - y|$. If x and y are binary, then

$$d = x \textsf{ XOR } y = (x \textsf{ or } y) \textsf{ and } (\textsf{not } (x \textsf{ and } y)) \tag{A.13}$$

If either x or y is continuous or multilevel, then d is constructed as the maximum of $x - y$ and $y - x$. This is represented by the following constraints:

$$\begin{aligned}
f^+ &= x - y, \quad f^+ \in [x^{\min} - y^{\max}, x^{\max} - y^{\min}] \\
f^- &= y - x, \quad f^- \in [y^{\min} - x^{\max}, y^{\max} - x^{\min}] \\
d &= \max\{f^+, f^-\} \\
&= f^+ \textsf{ or } f^-
\end{aligned}$$

Finding Infeasible Rules

Consider a feasible TIGER model M and a set of rules R that, when added to M, make the resulting problem infeasible. We can find a minimal set of rules in R that are inconsistent though the following procedure.

Each rule in R is of the form

$$p \Rightarrow q \tag{A.14}$$

or

$$p \Leftrightarrow q \tag{A.15}$$

We create a set S of *artificially satisfiable rules* by taking each rule in R and adding a set of Boolean variables as follows:

$$\begin{aligned}
p \Rightarrow q &\quad\rightarrow\quad p \Rightarrow q \textsf{ or } r_i \\
p \Leftrightarrow q &\quad\rightarrow\quad p \textsf{ or } l_i \Leftrightarrow q \textsf{ or } r_i
\end{aligned}$$

These rules are artificially satisfiable since setting all variables l_i and r_i to true will satisfy every rule in R. If all l_i and r_i are set to false, then the original logic in R is preserved, i.e. $R = S|_{l_i, r_i = 0}$. A MILP is then formulated as

$$\begin{aligned} &\min \sum_i l_i + \sum_i r_i \\ &\text{subject to} \\ &\qquad M, S \end{aligned}$$

Any rule containing a true l_i or r_i in the optimal solution of this problem is returned by TIGER as part of the set of infeasible rules. The `find_infeasible_rules` function also reports whether each infeasible rule was artificially satisfied on the left (l_i true) or right (r_i true) side.

Appendix B

Experimental Methods

PA14 mutant screening

Plates inoculated from frozen stocks of the PA14 transposon library [72] were grown overnight in lysogeny broth (Sigma) at 37^{o}C. A 96 well plate containing 200μl SCFM [76] was inoculated with a pinning tool and allowed to grow overnight at 37^{o}C. The following morning, three plates were inoculated with 10μl culture into 200μl SCFM. Plates were incubated 37^{o}C with shaking at 225 RPM. OD reading were taken every 40 minutes in a Tecan Infinite Pro 200 plate reader (600nm, 5 flashes/well). OD readings over 0.3 were discarded due to the onset of stationary phase.

Bibliography

[1] L. Michaelis, M. L. Menten, K. A. Johnson, and R. S. Goody, "The original Michaelis constant: translation of the 1913 Michaelis-Menten paper.," *Biochemistry*, vol. 50, pp. 8264–9, Oct. 2011.

[2] D. Fell, "Increasing the flux in metabolic pathways: A metabolic control analysis perspective," *Biotechnology and bioengineering*, vol. 58, pp. 121–4, Apr. 1998.

[3] J. Higgins, "Analysis of sequential reactions.," *Annals of the New York Academy of Sciences*, vol. 108, pp. 305–21, May 1963.

[4] H. Kacser and J. A. Burns, "The control of flux.," *Symposia of the Society for Experimental Biology*, vol. 27, pp. 65–104, Jan. 1973.

[5] R. Heinrich and T. A. Rapoport, "A linear steady-state treatment of enzymatic chains. General properties, control and effector strength.," *European journal of biochemistry / FEBS*, vol. 42, pp. 89–95, Feb. 1974.

[6] A. Cornish-Bowden, *Fundamentals of Enzyme Kinetics.* Portland Press Limited, 1995.

[7] D. A. Fell and J. R. Small, "Fat synthesis in adipose tissue. An examination of stoichiometric constraints.," *The Biochemical journal*, vol. 238, pp. 781–6, Sept. 1986.

[8] A. Varma and B. O. Palsson, "Metabolic capabilities of Escherichia coli: I. synthesis of biosynthetic precursors and cofactors.," *Journal of theoretical biology*, vol. 165, pp. 477–502, Dec. 1993.

[9] J. S. Edwards, R. U. Ibarra, and B. O. Palsson, "In silico predictions of Escherichia coli metabolic capabilities are consistent with experimental data.," *Nature biotechnology*, vol. 19, pp. 125–30, Feb. 2001.

[10] A. Varma and B. O. Palsson, "Stoichiometric flux balance models quantitatively predict growth and metabolic by-product secretion in wild-type Escherichia coli W3110.," *Applied and environmental microbiology*, vol. 60, pp. 3724–31, Oct. 1994.

[11] D. McCloskey, B. O. Palsson, and A. M. Feist, "Basic and applied uses of genome-scale metabolic network reconstructions of Escherichia coli.," *Molecular systems biology*, vol. 9, p. 661, Jan. 2013.

[12] S. Bordel, R. Agren, and J. Nielsen, "Sampling the solution space in genome-scale metabolic networks reveals transcriptional regulation in key enzymes.," *PLoS computational biology*, vol. 6, p. e1000859, Jan. 2010.

[13] A. M. Feist and B. O. Palsson, "The biomass objective function.," *Curr Opin Microbiol*, vol. 13, pp. 344–349, June 2010.

[14] S. S. Fong and B. O. Palsson, "Metabolic gene-deletion strains of Escherichia coli evolve to computationally predicted growth phenotypes.," *Nature genetics*, vol. 36, pp. 1056–8, Oct. 2004.

[15] J. D. Orth, I. Thiele, and B. O. Palsson, "What is flux balance analysis?," *Nat Biotechnol*, vol. 28, pp. 245–248, Mar. 2010.

[16] S. Schuster, T. Pfeiffer, and D. A. Fell, "Is maximization of molar yield in metabolic networks favoured by evolution?," *Journal of theoretical biology*, vol. 252, pp. 497–504, June 2008.

[17] N. C. Duarte, M. J. Herrgå rd, and B. O. Palsson, "Reconstruction and validation of Saccharomyces cerevisiae iND750, a fully compartmentalized genome-scale metabolic model," *Genome Res*, vol. 14, pp. 1298–1309, July 2004.

[18] N. E. Lewis, H. Nagarajan, and B. O. Palsson, "Constraining the metabolic genotype-phenotype relationship using a phylogeny of in silico methods.," *Nat Rev Microbiol*, vol. 10, pp. 291–305, Apr. 2012.

[19] R. Mahadevan and C. H. Schilling, "The effects of alternate optimal solutions in constraint-based genome-scale metabolic models.," *Metab Eng*, vol. 5, pp. 264–276, Oct. 2003.

[20] Y. Bilu, T. Shlomi, N. Barkai, and E. Ruppin, "Conservation of expression and sequence of metabolic genes is reflected by activity across metabolic states.," *PLoS computational biology*, vol. 2, p. e106, Aug. 2006.

[21] T. Shlomi, Y. Eisenberg, R. Sharan, and E. Ruppin, "A genome-scale computational study of the interplay between transcriptional regulation and metabolism," *Mol Syst Biol*, vol. 3, p. 101, 2007.

[22] 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*, vol. 84, pp. 647–657, Dec. 2003.

[23] J. Kim and J. L. Reed, "OptORF: Optimal metabolic and regulatory perturbations for metabolic engineering of microbial strains," *BMC Syst Biol*, vol. 4, p. 53, 2010.

[24] P. A. Jensen and J. A. Papin, "Functional integration of a metabolic network model and expression data without arbitrary thresholding," *Bioinformatics*, vol. 27, pp. 541–547, Feb. 2011.

[25] D. Barua, J. Kim, and J. L. Reed, "An automated phenotype-driven approach (GeneForce) for refining metabolic and regulatory models," *PLoS Comput Biol*, vol. 6, no. 10, p. e1000970, 2010.

[26] P. F. Suthers, A. Zomorrodi, and C. D. Maranas, "Genome-scale gene/reaction essentiality and synthetic lethality analysis.," *Mol Syst Biol*, vol. 5, p. 301, 2009.

[27] 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*, vol. 429, pp. 92–96, May 2004.

[28] M. J. Herrgå rd, B.-S. Lee, V. Portnoy, and B. O. Palsson, "Integrated analysis of regulatory and metabolic networks reveals novel regulatory mechanisms in Saccharomyces cerevisiae," *Genome Res*, vol. 16, pp. 627–635, May 2006.

[29] M. W. Covert, C. H. Schilling, and B. Palsson, "Regulation of gene expression in flux balance models of metabolism.," *J Theor Biol*, vol. 213, pp. 73–88, Nov. 2001.

[30] S. Klamt, J. Saez-Rodriguez, and E. D. Gilles, "Structural and functional analysis of cellular networks with CellNetAnalyzer.," *BMC Syst Biol*, vol. 1, p. 2, 2007.

[31] M. Cvijovic, R. Olivares-Hernández, R. Agren, N. Dahr, W. Vongsangnak, I. Nookaew, K. R. Patil, and J. Nielsen, "BioMet Toolbox: genome-wide analysis of metabolism.," *Nucleic Acids Res*, vol. 38, pp. W144—-W149, July 2010.

[32] 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 Protoc*, vol. 2, no. 3, pp. 727–738, 2007.

[33] H. W. Kuhn, A. W. Tucker, and G. B. Dantzig, *Linear inequalities and related systems*, vol. no. 38. Princeton: Princeton University Press, 1956.

[34] S. Gudmundsson and I. Thiele, "Computationally efficient flux variability analysis.," *BMC Bioinformatics*, vol. 11, p. 489, 2010.

[35] 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*, vol. 2, p. e101, Aug. 2006.

[36] T. G. Turi and J. C. Loper, "Multiple regulatory elements control expression of the gene encoding the Saccharomyces cerevisiae cytochrome P450, lanosterol 14 alpha-demethylase (ERG11).," *J Biol Chem*, vol. 267, pp. 2046–2056, Jan. 1992.

[37] J. F. Moxley, M. C. Jewett, M. R. Antoniewicz, S. G. Villas-Boas, H. Alper, R. T. Wheeler, L. Tong, A. G. Hinnebusch, T. Ideker, J. Nielsen, and G. Stephanopoulos, "Linking high-resolution metabolic flux phenotypes and transcriptional regulation in yeast modulated by the global regulator Gcn4p," *Proc Natl Acad Sci U S A*, vol. 106, pp. 6477–6482, Apr. 2009.

[38] S. A. Becker and B. O. Palsson, "Context-specific metabolic networks are consistent with experiments," *PLoS Comput Biol*, vol. 4, p. e1000082, May 2008.

[39] T. Shlomi, M. N. Cabili, M. J. Herrgå rd, B. O. Palsson, and E. Ruppin, "Network-based prediction of human tissue-specific metabolism," *Nat Biotechnol*, vol. 26, pp. 1003–1010, Sept. 2008.

[40] J. R. Dickinson and M. Schweizer, *The Metabolism and Molecular Physiology of Saccharomyces cerevisiase.* CRC Press, 2nd ed., 2004.

[41] J. L. DeRisi, V. R. Iyer, and P. O. Brown, "Exploring the metabolic and genetic control of gene expression on a genomic scale," *Science*, vol. 278, pp. 680–686, Oct. 1997.

[42] M. Carlson, "Glucose repression in yeast," *Curr Opin Microbiol*, vol. 2, pp. 202–207, Apr. 1999.

[43] G. G. Roberts and A. P. Hudson, "Transcriptome profiling of Saccharomyces cerevisiae during a transition from fermentative to glycerol-based respiratory growth reveals extensive metabolic and structural remodeling," *Mol Genet Genomics*, vol. 276, pp. 170–186, Aug. 2006.

[44] M. A. Oberhardt, B. O. Palsson, and J. A. Papin, "Applications of genome-scale metabolic reconstructions," *Mol Syst Biol*, vol. 5, p. 320, 2009.

[45] P. Shannon, A. Markiel, O. Ozier, N. S. Baliga, J. T. Wang, D. Ramage, N. Amin, B. Schwikowski, and T. Ideker, "Cytoscape: a software environment for integrated models of biomolecular interaction networks.," *Genome Res*, vol. 13, pp. 2498–2504, Nov. 2003.

[46] M. Kanehisa, S. Goto, Y. Sato, M. Furumichi, and M. Tanabe, "KEGG for integration and interpretation of large-scale molecular data sets.," *Nucleic Acids Res*, vol. 40, pp. D109—-D114, Jan. 2012.

[47] J. Schellenberger, R. Que, R. M. T. Fleming, I. Thiele, J. D. Orth, A. M. Feist, D. C. Zielinski, A. Bordbar, N. E. Lewis, S. Rahmanian, J. Kang, D. R. Hyduke, and B. O. Palsson, "Quantitative prediction of cellular metabolism with constraint-based models: the COBRA Toolbox v2.0.," *Nat Protoc*, vol. 6, pp. 1290–1307, Sept. 2011.

[48] J. D. Orth, T. M. Conrad, J. Na, J. A. Lerman, H. Nam, A. M. Feist, and B. O. Palsson, "A comprehensive genome-scale reconstruction of Escherichia coli metabolism–2011," *Mol Syst Biol*, vol. 7, p. 535, 2011.

[49] E. R. Gansner and S. C. North, "An open graph visualization system and its applications to software engineering," *SOFTWARE - PRACTICE AND EXPERIENCE*, vol. 30, no. 11, pp. 1203–1233, 2000.

[50] K. Kochanowski, U. Sauer, and V. Chubukov, "Somewhat in control-the role of transcription in regulating microbial metabolic fluxes.," *Current opinion in biotechnology*, vol. 24, pp. 987–93, Apr. 2013.

[51] C. Colijn, A. Brandes, J. Zucker, D. S. Lun, B. Weiner, M. R. Farhat, T.-Y. Cheng, D. B. Moody, M. Murray, and J. E. Galagan, "Interpreting expression data with metabolic flux models: predicting Mycobacterium tuberculosis mycolic acid production.," *PLoS computational biology*, vol. 5, p. e1000489, Aug. 2009.

[52] J. Pevsner, *Bioinformatics and Functional Genomics.* Wiley-Blackwell, 2009.

[53] K. Smallbone, E. Simeonidis, N. Swainston, and P. Mendes, "Towards a genome-scale kinetic model of cellular metabolism.," *BMC systems biology*, vol. 4, p. 6, Jan. 2010.

[54] R. C. Taylor, B.-J. M. Webb Robertson, L. M. Markillie, M. H. Serres, B. E. Linggi, J. T. Aldrich, E. A. Hill, M. F. Romine, M. S. Lipton, and H. S. Wiley, "Changes in translational efficiency is a dominant regulatory mechanism in the environmental response of bacteria.," *Integrative biology : quantitative biosciences from nano to macro*, vol. 5, pp. 1393–406, Nov. 2013.

[55] A. P. Oliveira, C. Ludwig, P. Picotti, M. Kogadeeva, R. Aebersold, and U. Sauer, "Regulation of yeast central metabolism by enzyme phosphorylation.," *Molecular systems biology*, vol. 8, p. 623, Jan. 2012.

[56] B. R. B. Haverkorn van Rijsewijk, A. Nanchen, S. Nallet, R. J. Kleijn, and U. Sauer, "Large-scale 13C-flux analysis reveals distinct transcriptional control of respiratory and fermentative metabolism in Escherichia coli.," *Molecular systems biology*, vol. 7, p. 477, Mar. 2011.

[57] B. Szappanos, K. Kovács, B. Szamecz, F. Honti, M. Costanzo, A. Baryshnikova, G. Gelius-Dietrich, M. J. Lercher, M. Jelasity, C. L. Myers, B. J. Andrews, C. Boone, S. G. Oliver, C. Pál, and B. Papp, "An integrated approach to characterize genetic interaction networks in yeast metabolism.," *Nature genetics*, vol. 43, pp. 656–62, July 2011.

[58] F. M. Brown, *Boolean Reasoning: The Logic of Boolean Equations (Dover Books on Mathematics)*. Dover Publications, 2012.

[59] D. McCloskey, J. A. Gangoiti, Z. A. King, R. K. Naviaux, B. A. Barshop, B. O. Palsson, and A. M. Feist, "A model-driven quantitative metabolomics analysis of aerobic and anaerobic metabolism in E. coli K-12 MG1655 that is biochemically and thermodynamically consistent.," *Biotechnology and bioengineering*, Oct. 2013.

[60] M. A. Oberhardt, J. Pucha\lka, K. E. Fryer, V. A. P. M. dos Santos, and J. A. Papin, "Genome-scale metabolic network analysis of the opportunistic pathogen Pseudomonas aeruginosa PAO1.," *J Bacteriol*, vol. 190, pp. 2790–2803, Apr. 2008.

[61] J. Puchaka, M. A. Oberhardt, M. Godinho, A. Bielecka, D. Regenhardt, K. N. Timmis, J. A. Papin, and V. A. P. Martins dos Santos, "Genome-scale reconstruction and analysis of the Pseudomonas putida KT2440 metabolic network facilitates applications in biotechnology.," *PLoS computational biology*, vol. 4, p. e1000210, Oct. 2008.

[62] R. L. Chang, L. Ghamsari, A. Manichaikul, E. F. Y. Hom, S. Balaji, W. Fu, Y. Shen, T. Hao, B. O. Palsson, K. Salehi-Ashtiani, and J. A. Papin, "Metabolic network reconstruction of Chlamydomonas offers insight into light-driven algal metabolism.," *Molecular systems biology*, vol. 7, p. 518, Jan. 2011.

[63] A. P. Burgard, E. V. Nikolaev, C. H. Schilling, and C. D. Maranas, "Flux coupling analysis of genome-scale metabolic network reconstructions," *Genome Res*, vol. 14, pp. 301–312, Feb. 2004.

[64] Y. Xi, Y.-P. P. Chen, C. Qian, and F. Wang, "Comparative study of computational methods to detect the correlated reaction sets in biochemical networks.," *Briefings in bioinformatics*, vol. 12, pp. 132–50, Mar. 2011.

[65] O. Rokhlenko, T. Shlomi, R. Sharan, E. Ruppin, and R. Y. Pinter, "Constraint-based functional similarity of metabolic genes: going beyond network topology.," *Bioinformatics (Oxford, England)*, vol. 23, pp. 2139–46, Aug. 2007.

[66] J. Strathern, ed., *The Molecular Biology of the Yeast Saccharomyces: Metabolism and Gene Expression (Cold Spring Harbor monograph series)*. Cold Spring Harbor Laboratory Pr, 1982.

[67] "The Saccharomyces Genome Database, http://www.yeastgenome.org."

[68] A. A. Carmen, P. K. Brindle, C. S. Park, and M. J. Holland, "Transcriptional regulation by an upstream repression sequence from the yeast enolase gene ENO1.," *Yeast (Chichester, England)*, vol. 11, pp. 1031–43, Sept. 1995.

[69] BIOLOG, "BIOLOG Phenotype MicroArrays Panels PM-M1 to PM-M14," 2007.

[70] Biolog, "Biolog Redox Dye Mixes," tech. rep., 2007.

[71] T. Baba, T. Ara, M. Hasegawa, Y. Takai, Y. Okumura, M. Baba, K. A. Datsenko, M. Tomita, B. L. Wanner, and H. Mori, “Construction of Escherichia coli K-12 in-frame, single-gene knock-out mutants: the Keio collection.,” *Molecular systems biology*, vol. 2, p. 2006.0008, 2006.

[72] N. T. Liberati, J. M. Urbach, S. Miyata, D. G. Lee, E. Drenkard, G. Wu, J. Villanueva, T. Wei, and F. M. Ausubel, “An ordered, nonredundant library of Pseudomonas aeruginosa strain PA14 transposon insertion mutants,” *Proc Natl Acad Sci U S A*, vol. 103, pp. 2833–2838, Feb. 2006.

[73] P. D. Fey, J. L. Endres, V. K. Yajjala, T. J. Widhelm, R. J. Boissy, J. L. Bose, and K. W. Bayles, “A genetic resource for rapid and comprehensive phenotype screening of nonessential Staphylococcus aureus genes.,” *mBio*, vol. 4, pp. e00537–12, Jan. 2013.

[74] L. A. Gallagher, E. Ramage, R. Patrapuvich, E. Weiss, M. Brittnacher, and C. Manoil, “Sequence-Defined Transposon Mutant Library of Burkholderia thailandensis.,” *mBio*, vol. 4, Jan. 2013.

[75] T. van Opijnen, K. L. Bodi, and A. Camilli, “Tn-seq: high-throughput parallel sequencing for fitness and genetic interaction studies in microorganisms.,” *Nature methods*, vol. 6, pp. 767–772, 2009.

[76] K. L. Palmer, L. M. Aye, and M. Whiteley, “Nutritional cues control Pseudomonas aeruginosa multicellular behavior in cystic fibrosis sputum.,” *J Bacteriol*, vol. 189, pp. 8079–8087, Nov. 2007.

[77] R. Martí-Arbona, C. Xu, S. Steele, A. Weeks, G. F. Kuty, C. M. Seibert, and F. M. Raushel, “Annotating enzymes of unknown function: N-formimino-L-glutamate deiminase is a member of the amidohydrolase superfamily.,” *Biochemistry*, vol. 45, pp. 1997–2005, Feb. 2006.

[78] “Pseudomonas genome database – http://www.pseudomonas.com.”

[79] G. Sigurdsson, R. M. T. Fleming, A. Heinken, and I. Thiele, “A systems biology approach to drug targets in Pseudomonas aeruginosa biofilm,” *PLoS One*, vol. 7, no. 4, p. e34337, 2012.

[80] C. L. Haley, J. A. Colmer-Hamood, and A. N. Hamood, “Characterization of biofilm-like structures formed by Pseudomonas aeruginosa in a synthetic mucus medium,” *BMC Microbiol*, vol. 12, p. 181, 2012.

[81] Wikipedia, “PAO biofilm.”

[82] L. M. Junker and J. Clardy, “High-throughput screens for small-molecule inhibitors of Pseudomonas aeruginosa biofilm development,” *Antimicrob Agents Chemother*, vol. 51, pp. 3582–3590, Oct. 2007.