UNIVERSITY OF CALIFORNIA, SAN DIEGO

Development and Dissemination of Computational Methods for Genome-scale Modeling

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

in

Bioengineering

by

Ali Ebrahim

Committee in charge:

Professor Bernhard Palsson, Chair
Professor Nuno Bandeira
Professor Christian Metallo
Professor Glenn Tesler
Professor Kun Zhang

2016

The Dissertation of Ali Ebrahim is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____

_____

Chair

University of California, San Diego

2016

DEDICATION

My heart breaks for the refugees who have been driven from their homes in Syria. Though it must mean very little, I dedicate this thesis to them, and I will try to apply some of what I have learned to help give them some of the opportunities I have had.

EPIGRAPH

What I cannot **create**,

I do not **understand**.

—*Richard P. Feynman*

I like to call it

low-input,

high-throughput,

no-output biology.

—*Sydney Brenner*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I been very lucky to recieve excellent mentorship while in graduate school. I have learned so much from my advisor Dr. Palsson, who has given me so many opportunties in his lab. I know it's not often one is given the chance to work so closely with a world-class scientist and engineer, and I am very grateful to have recieved Dr. Palsson's expert guidance for so many years. In the same vein, I would like to thank the rest of my committee members Drs. Bandeira, Metallo, Tesler, and Zhang for their time and guidance in preparing my thesis research.

The research presented here also could not have been done without the ridiculous amount of support I got from my colleagues in the Palsson lab. Without Joshua Lerman and Teddy O'Brien, the ME models that formed the basis for a lot of my research wouldn't even exist. In addition to being excellent friends, the two of them very graciously showed me the ropes in working with these extremely complicated models. Colton Lloyd was an invaluable collaborator, and words can not express how grateful I am for his help, especially in the final push to finish the final chapter of this thesis. Without his diligence and tenacity, even on the most difficult and annoying problems, this process would have taken so much longer. I also owe a lot to Aarash Bordbar, as both a friend and mentor. As an older graduate student, Aarash always went out of his way to provide me with very useful advice on how to navigate the waters of graduate school. He also frequently forced me to take breaks to watch Warriors games and get food, which I am certain preserved my sanity. I would also like to take the time to remind him that he in fact does bear a resemblance to Klay Thompson. I would also like to thank Jon Monk for being my office mate for so many years, and tolerating my presence and questionable taste in music (and of course for

answering my never-ending questions). BΩΠ for life. Zachary King has been both a pleasure to work with, and a fellow geek to debate the finer points of command-line text editors, even if he is mistaken in his stubborn belief that EMACS is superior to vim. Gabriela Ines Guzmán Lopez Aguado has been a great collaborator and friend as well. She even taught me how to do resequence DNA back in my experimental days, which must have required an unfathomable level of patience. Ryan LaCroix is a great friend, and I'm glad he has finally seen the light of the greatness of Python. I know our adventures together are far from over. I would also like to thank my fellow members of team "D", Haythem Latif and Harish Nagarajan. When I was new in lab, both were excellent mentors who helped me get started, and together we did all kinds of things to so many omics datasets. Mallory Embree was also a valuable mentor and friend. Even though she jokingly boasted about her "special" ability to prevent us younger graduate students from pestering her, in fact she was one of the most approachable friends I made in graduate school to turn to for advice. I would like to thank Karsten for constantly finding things to make fun of me about, and always keeping me on my toes. I owe Adam Feist a great deal for the incredible amount of advice he always gave me, and for letting me spend time with his labradoodle Rosie. I would also like to thank Justin Tan, Bin Du, Jennifer Levering, Steven Federowicz, Laurence Yang, Jahir Gutierrez, Daniel Zielinski, Nathan Lewis, and Joanne Liu for both their friendship and advice throughout graduate school. I would also like to thank Ambros Gleixner for his patience getting me started with ill-scaled linear programs, and for being such a gracious host to me in Berlin. Similarly, I would like to thank Matthias Miltenberger for always promptly merging my patches and fixing bugs for my particular use cases. I would also like to thank Dr. Maarten Chrispeels,

Shelley Glenn Lee, Johnnie Lyman, Stuart Douglas and the rest of my fellow Socrates fellows for making my NSF Socrates Fellowship year a very enjoyable one. Finally, I would like to thank all of my co-authors. It was a pleasure working with all of them, and I am sure all of our manuscripts are stronger for it.

I have so many people to thank for their all their help throughout my life. The first person on this list has to be my mother. I still fondly remember childhood backyard science experiments, and her patience and encouragement throughout my life have been invaluable. My brothers Senan and Hassaan have also been incredibly supportive. Even though they are my younger brothers (a fact which I have never let them forget), I find myself constantly turning to them for their advice and wisdom. I have no clue when they got more than me, but I'm happy that they at least share it generously.

I would also like to thank my figurative "bro"s and friends. There are far too many of them to list, so even for those I don't mention directly, I hope they know how grateful I am anyways. I would like to thank Flora Li for her patient tolerance of my hackneyed attempts at humor, and for constantly motivating me (usually by relentless teasing about how long I had been in graduate school). She always seemed to know how to cheer me up when things were at their most stressful. Even though Dan Pipe-Mazo was only in San Diego for a year, in that time he became one of my best friends, and I cherish our Sunday tradition of watching football and wandering to the farmer's market. I would like to thank Sam Brotherton and Briana Goodale for always being great hosts wherever they were, helping me blow off a lot of steam (and of course, Sam for helping me get a job at Google). I would like to thank Eythan for tolerating me as a roommate for so many years. We both learned how to live like

adult human beings together, and that was a pretty special journey. Similarly, I would like to thank Julian Warchall and Steven Okai for being excellent roommates at Mt. Brolympus. If I had to pick a group of rambunctious ECE kids to live with, I really don't think a better combination exists, unless there is a version of Steven Okai that also watches "Once Upon a Time." Also, at heart, Julian is a very special kind of lion. I don't know what I am going to do in the future when my Leilani's and Rigoberto's visitation frequency drops to zero. Christopher Dewan has also been a very supportive friend throughout the years. I have always admired his depth of knowledge of topics he is passionate about, and I especially appreaciate his help learning to navigate the world of job interviews and salary negotiations. I would like to thank Stephany Lai for being a fellow photography nerd at the San Diego Zoo and Safari Park, and also for her years of friendship. Sujitha Martin has also been a wonderful friend over the years, and especially during my first year when we were both new to San Diego. Elisa Walsh and Margaret Chiu have also both been wonderful and supportive friends for so many years, and put a lot of time into pretty much teaching me what biology was when I was an undergraduate. I have also not regretted any of the calories from the delicious baked goods Elisa always seemed to provide. I would also like to thank Elaine To for her years of friendship and support through some of the hardest years of grad school.

Finally, I would like to inform Kelvin Fang that I have judged him to be a first-rate scoundrel. Neither words nor numbers nor pictures can effectively convey the foulness of his company, and I can imagine no reason for any sane human being to voluntarily spend time in his immediate presence. I vow to continue to support the "Screw Kelvin Day" holiday, to raise awareness for the misery he inflicts on those around him, especially during games of "Smallworld," which he always seems to win

by a statistically significant margin (a fact which I am by no means bitter over).

Borrowed chapters

The text of chapter 1 was borrowed, in whole or in part, from the following manuscript: Ebrahim, A., Lerman, J.A., Palsson, B.O., and Hyduke, D.R. (2013). COBRApy: COnstraints-Based Reconstruction and Analysis for Python. BMC Syst. Biol. 7, 74.

The text of chapter 2 was borrowed, in whole or in part, from the following manuscript: Ebrahim, A., King, Z.A., Jamshidi, N., Palsson, B.O., Lewis, N.E., Drager, A. (In preparation). Distribution and Portability of Validated Genome-Scale Constraint-Based Models.

The text of chapter 3 was borrowed, in whole or in part, from the following manuscript: Kim, D., Ebrahim, A., Seo, S., Palsson, B.O. (In preparation). Model-driven elucidation of transcriptional regulatory network in bacteria.

The text of chapter 4 was borrowed, in whole or in part, from the following manuscript: Ebrahim, A., Brunk, E., Tan, J., O'Brien, J., Kim, D., Szubin, R., Lerman, J.A., Lechner, A., Sastry, A., Bordbar, A., Feist, A.M., Palsson, B.O. (under review) Multi-omic data integration enables discovery of hidden biological regularities. Nature Communications

The text of chapter 5 was borrowed, in whole or in part, from the following manuscript: Ebrahim, A., Lloyd, C., Monk, J., Yang, L., O'Brien, E.J., Liu, J.K., Palsson, B.O. (In preparation). Construction of Genome-scale models of Metabolism and Gene Expression for the *Escherichia coli* family.

VITA

2016	Ph. D. in Bioengineering, University of California, San Diego
2010	B. S. with honors in Chemistry, California Institute of Technology
2016	Software Engineer, Google, Mountain View, CA
2015	Programming Consultant, San Diego, CA
2012-2013	NSF Socrates Fellow, National City, CA
2011-2013	Gradaute Teaching Assistant, University of California, San Diego
2009	Undergraduate Research Fello, NASA Jet Propulsion Laboratory, La Canada Flintridge, CA
2008-2010	Undergraduate Teaching Assistant, California Institute of Technology
2007-2008	Undergraduate Researcher, Jacqueline K. Barton Lab, Pasadena, CA

PUBLICATIONS

King, Z.A., Lu, J., Dräger, A., Miller, P., Federowicz, S., Lerman, J.A., Ebrahim, A., Palsson, B.O., and Lewis, N.E. (2015). BiGG Models: A platform for integrating, standardizing and sharing genome-scale models. Nucleic Acids Res.

Ebrahim, A., Almaas, E., Bauer, E., Bordbar, A., Burgard, A.P., Chang, R.L., Dräger, A., Famili, I., Feist, A.M., Fleming, R.M., et al. (2015). Do genome-scale models need exact solvers or clearer standards? Mol. Syst. Biol. 11, 831.

Yang, L., Tan, J., O'Brien, E.J., Monk, J.M., Kim, D., Li, H.J., Charusanti, P., Ebrahim, A., Lloyd, C.J., Yurkovich, J.T., et al. (2015). Systems biology definition of the core proteome of metabolism and expression is consistent with high-throughput data. Proc. Natl. Acad. Sci. U. S. A. 112, 10810–10815.

King, Z.A., Dräger, A., Ebrahim, A., Sonnenschein, N., Lewis, N.E., and Palsson, B.O. (2015). Escher: A Web Application for Building, Sharing, and Embedding Data-Rich Visualizations of Biological Pathways. PLoS Comput. Biol. 11, e1004321.

Guzman, G.I., Utrilla, J., Nurk, S., Brunk, E., Monk, J.M., Ebrahim, A., Palsson, B.O., and Feist, A.M. (2015). Model-driven discovery of underground metabolic functions in Escherichia coli. Proc. Natl. Acad. Sci. U. S. A. 112, 929–934.

LaCroix, R.A., Sandberg, T.E., O'Brien, E.J., Utrilla, J., Ebrahim, A., Guzman, G.I., Szubin, R., Palsson, B.O., and Feist, A.M. (2015). Use of adaptive laboratory evolution to discover key mutations enabling rapid growth of Escherichia coli K-12 MG1655 on glucose minimal medium. Appl. Environ. Microbiol. 81, 17–30.

Sandberg, T.E., Pedersen, M., LaCroix, R.A., Ebrahim, A., Bonde, M., Herrgard, M.J., Palsson, B.O., Sommer, M., and Feist, A.M. (2014). Evolution of Escherichia coli to 42 °C and subsequent genetic engineering reveals adaptive mechanisms and novel mutations. Mol. Biol. Evol. 31, 2647–2662.

Bordbar, A., Nagarajan, H., Lewis, N.E., Latif, H., Ebrahim, A., Federowicz, S., Schellenberger, J., and Palsson, B.O. (2014). Minimal metabolic pathway structure is consistent with associated biomolecular interactions. Mol. Syst. Biol. 10, 737.

Federowicz, S., Kim, D., Ebrahim, A., Lerman, J., Nagarajan, H., Cho, B.-K., Zengler, K., and Palsson, B. (2014). Determining the control circuitry of redox metabolism at the genome-scale. PLoS Genet. 10, e1004264.

Nagarajan, H., Sahin, M., Nogales, J., Latif, H., Lovley, D.R., Ebrahim, A., and Zengler, K. (2013). Characterizing acetogenic metabolism using a genome-scale metabolic reconstruction of Clostridium ljungdahlii. Microb. Cell Fact. 12, 118.

Schmidt, B.J., Ebrahim, A., Metz, T.O., Adkins, J.N., Palsson, B.Ø., and Hyduke, D.R. (2013). GIM3E: condition-specific models of cellular metabolism developed from metabolomics and expression data. Bioinformatics 29, 2900–2908.

Ebrahim, A., Lerman, J.A., Palsson, B.O., and Hyduke, D.R. (2013). COBRApy: COnstraints-Based Reconstruction and Analysis for Python. BMC Syst. Biol. 7, 74.

Gong, Y., Ebrahim, A., Feist, A.M., Embree, M., Zhang, T., Lovley, D., and Zengler, K. (2013). Sulfide-driven microbial electrosynthesis. Environ. Sci. Technol. 47, 568–573.

Gorodetsky, A.A., Ebrahim, A., and Barton, J.K. (2008). Electrical detection of TATA binding protein at DNA-modified microelectrodes. J. Am. Chem. Soc. 130, 2924–2925.

ABSTRACT OF THE DISSERTATION

Development and Dissemination of Computational Methods for Genome-scale Modeling

by

Ali Ebrahim

Doctor of Philosophy in Bioengineering

University of California, San Diego, 2016

Professor Bernhard Palsson, Chair

Constraint-based modeling of metabolism at the genome scale has existed as a successful field for two decades. However, genome-scale modeling has entered a new era, with models of ever-increasing size which account for every known reaction in the metabolic and gene expression networks, known as ME models. These models are able to integrate for both metabolic costs (operational expenses) and gene expression costs (capital expenses) in a quasi-econometric model of growing bacterial cells. This modeling type presents new challenges but also brings exciting novel capabilities to constraint-based modeling.

An initial challenge due to models of this size and scale is simply software. Existing software tools for constraint-based modeling simply do not scale to handle models at the size of ME models. Additionally, they make assumptions about the model structure which prevent them from modeling the nonlinearities present in ME models. To address this, a new software package, COBRApy was developed to meet these needs. Unfortunately, interoperability between constraint-based modeling frameworks was already difficult, often resulting in different modeling results. To address this shortcoming, a web-based model validator was deployed to aid the constraint-based modeling community in ensuring all models compute identically with different tools. Finally, a new framework was written on top of COBRApy to build and simulate ME models quickly, accurately, and reproducibly.

To exploit the descriptive nature of ME models, new algorithms were developed to gain increased biological insight from modeling simulations. To improve the understanding of the transcriptional regulatory network in *E. coli*, a method was developed to predict transcription factor activity from ME simulations, which were successfully used to guide experimental design in identifying novel transcription factor binding sites. To improve the ability of the *E. coli* ME model to directly predict differential gene expression, a novel method was developed to estimate ME model parameters from high-throughput omics data. Finally, the ME model of the *E. coli* K-12 MG1655 strain was extended to build strain-specific ME models for 40 different *E. coli* strains.

Chapter 1

COBRApy: COnstraints-Based Reconstruction and Analysis for Python

1.1 Abstract

1.1.1 Background

COnstraint-Based Reconstruction and Analysis (COBRA) methods are widely used for genome-scale modeling of metabolic networks in both prokaryotes and eukaryotes. Due to the successes with metabolism, there is an increasing effort to apply COBRA methods to reconstruct and analyze integrated models of cellular processes. The COBRA Toolbox for MATLAB is a leading software package for genome-scale analysis of metabolism; however, it was not designed to elegantly capture the complexity inherent in integrated biological networks and lacks an integration framework

for the multiomics data used in systems biology. The openCOBRA Project is a community effort to promote constraints-based research through the distribution of freely available software.

1.1.2 Results

Here, we describe COBRA for Python (COBRApy), a Python package that provides support for basic COBRA methods. COBRApy is designed in an object-oriented fashion that facilitates the representation of the complex biological processes of metabolism and gene expression. COBRApy does not require MATLAB to function; however, it includes an interface to the COBRA Toolbox for MATLAB to facilitate use of legacy codes. For improved performance, COBRApy includes parallel processing support for computationally intensive processes.

1.1.3 Conclusion

COBRApy is an object-oriented framework designed to meet the computational challenges associated with the next generation of stoichiometric constraint-based models and high-density omics data sets.

1.2 Background

Constraint based modeling approaches have been widely applied in the field of microbial metabolic engineering [FP08, KRaSN12] and have been employed in the analysis [LBY$^+$03, HJT$^+$07, THL10] and, to a lesser extent, modeling of transcriptional [CP02, GJPP09, TJFP09] and signaling [HP10] networks. And, we've recently

developed a method for integrated modeling of gene expression and metabolism on the genome scale [LHL+12].

The popularity of these approaches is due, in part, to the fact that they facilitate analysis of biological systems in the absence of a comprehensive set of parameters. Constraints-based approaches focus on employing data-driven physicochemical and biological constraints to enumerate the set of feasible phenotypic states of a reconstructed biological network in a given condition. These constraints include compartmentalization, mass conservation, molecular crowding [VBD+08], thermodynamic directionality [HBH07], and transcription factor activity [GCJJPG+08]. More recently, transcriptome data have been used to reduce the size of the set of computed feasible states [CBZ+09, FZF+11, BMN+12, HLP13]. Because constraints-based models are often underdetermined they may provide multiple mathematically-equivalent solutions to a specific question – these equivalent solutions must be assessed with experimental data for biological relevance [MS03].

We have previously published the COBRA Toolbox [SQF+11] for MATLAB to provide systems biology researchers with a high-level interface to a variety of methods for constraint-based modeling of genome-scale stoichiometric models of cellular biochemistry. The COBRA Toolbox is being increasingly recognized as a standard framework for constraint-based modeling of metabolism [MvRTB12]. While the COBRA Toolbox has gained widespread use and become a powerful piece of software, it was not designed to cope with modeling complex biological processes outside of metabolism or for integrated analyses of omics data, and requires proprietary software to function. To drive COBRA research through this avalanche of omics and model increasingly complex biological processes [LHL+12], we have developed an

object-oriented implementation of core COBRA Toolbox functions using the Python programming language. COBRA for Python (COBRApy) provides access to commonly used COBRA methods in a MATLAB-free fashion.

1.3 Implementation

The core capabilities of COBRApy are enabled by a set of classes (Figure 1.1) that represent organisms (Model), biochemical reactions (Reaction), and biomolecules (Metabolite and Gene). The core code is accessible through either Python or Jython (Python for Java). COBRApy contains: (1) cobra.io: an input/output package for reading / writing SBML [HFB+04] models and reading / writing COBRA Toolbox MATLAB structures. (2) cobra.flux_analysis: a package for performing common FBA operations, including gene deletion and flux variability analysis (refs). (3) cobra.topology: a package for performing structural analysis – the current version contains the reporter metabolites algorithm of Patil & Nielsen [PN05]. (4) cobra.test: a suite of unit tests and test data. (5) cobra.solvers: interfaces to linear optimization packages. And, (6) cobra.mlab: an interface to the COBRA Toolbox for MATLAB.

1.4 Results and discussion

COBRApy is a software package for constraints-based modeling that is designed to accommodate the increasing complexity of biological processes represented with COBRA methods. Like the COBRA Toolbox, COBRApy provides core COBRA modeling capabilities in an extendible and accessible fashion. However, COBRApy employs an object oriented programming approach that is more amenable to represent-

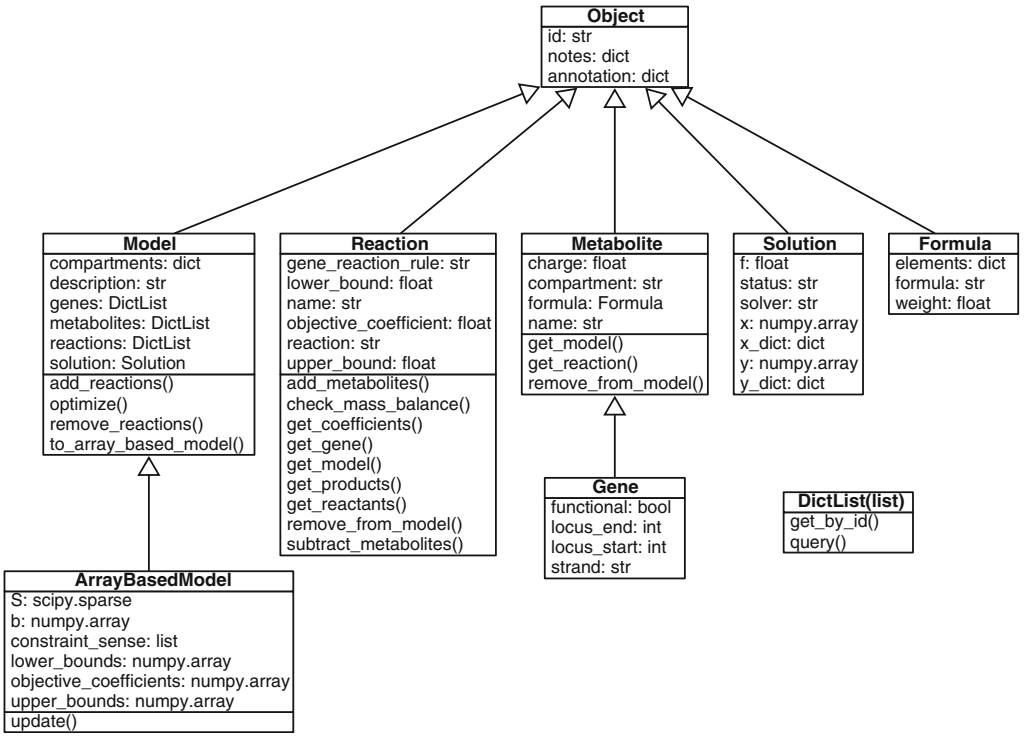


Figure 1.1: Core classes in COBRA for Python with key attributes and methods listed. Additional attributes and methods are described in the documentation.

ing increasingly complex models of biological networks. Moreover, COBRApy inherits numerous benefits from the Python language, and allows the integration of models with databases and other sources of high-throughput data. Additionally, COBRApy does not require commercial software for commonly used COBRA operations whereas the COBRA Toolbox depends on MATLAB. As the COBRA Toolbox is in wide use, it will likely be used as a development and analysis platform for years to come. To take advantage of legacy and future modules written for the COBRA Toolbox, COBRApy includes a module for directly interacting with the COBRA Toolbox (cobra.mlab) and support for reading and writing COBRA Toolbox MATLAB structures (cobra.io.mat).

In recent years, a number of software packages have been developed that employ stoichiometric constraint-based modeling approaches [LKCL14], such as Cell Net

Analyzer [KvK11], FASIMU [HHG+11], PySCeS-CBM [ORH05], the Raven Toolbox [ALS+13], and the Systems Biology Research Tool [WW08]. While there is overlap in functionality between some of packages and COBRApy (Table 1), the other packages do not currently support the next generation models of metabolism and expression (ME-Models) [LHL+12] nor integration with the COBRA Toolbox for MATLAB. It is worth noting that the other software packages often contain a rich variety of functionality that is targeted towards other research topics, such as modeling signaling networks [KvK11]. COBRApy continues the COBRA Toolbox's tradition of providing an interactive / programmable framework for constraints-based modeling and is a new initiative of The openCOBRA Project. Software downloads, tutorials, forums, and detailed documentation are available at http://opencobra.github.io.

Table 1.1: Features of available constraints-based programming packages

Package	GUI	FBA	FVA	M	ME	SBML	Design	Language
Cell Net Analyzer	+	+	+	+		+	+	MATLAB
COBRA toolbox		+	+	+		+	+	MATLAB
COBRApy		+	+	+	+	+	+	Python
fasimu		+	+	+		+		bash
PySCeS-CBM		+	+	+		+		Python
Raven	+	+		+		+		MATLAB
Sys Bio Res Tool		+	+	+				Java

1.4.1 Core classes: model, metabolite, reaction, & gene

The core classes of COBRApy are Model, Metabolite, Reaction, and Gene. The Model class serves as a container for a set of chemical Reactions, including associated Metabolites and Gene products (Figure 1.2a). Within a Model, Metabolites are modified by one or more Reactions that may be spontaneous or catalyzed by one or more Genes (Figure 2b). The underlying genetic requirements for a Reaction to

be active in a Model are supplied as a Boolean relationship [19], where each gene is referred to by a unique identifier. During the construction of a Model, the Model and the Reactions, Metabolites, and Genes are explicitly aware of each other. For example, given a Metabolite, it is possible to use the get_reaction() method to determine in which Reactions this Metabolite participates. Then the genes associated with these Reactions may be accessed by the Reaction.get_gene() method.

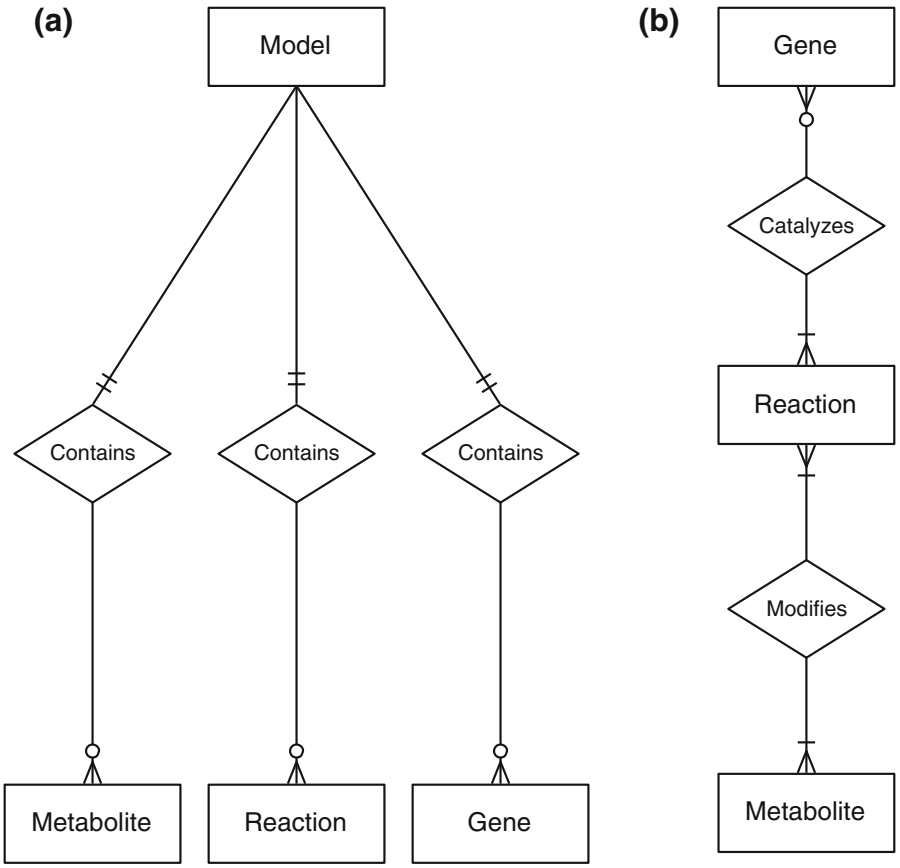


Figure 1.2: Entity relationship diagrams for core classes in COBRApy. (a) A Model contains Metabolites, Reactions, and Genes. (b) A Reaction may be catalyzed by 0 or more Genes. Reactions catalyzed by 0 Genes are spontaneous. A Reaction may be catalyzed by different sets of Genes. Reactions modify 1 or more Metabolites. A Reaction that modifies only 1 metabolite is an external boundary condition. A Metabolite may be modified by many different Reactions.

The object-based design of COBRApy provides the user with the ability to

directly access attributes for each object (Figure 1.1), whereas with the COBRA Toolbox for MATLAB biological entities and their attributes are each contained within separate lists. For example, with COBRApy, a Metabolite object provides information about its chemical Formula and associated biochemical Reactions, whereas, with the COBRA Toolbox for MATLAB, one must query multiple tables to access these values and modify multiple tables to update these values.

1.4.2 Key capabilities

COBRApy comes with variants of the published metabolic network models (M-Models) for Salmonella enterica Typhimurium LT2 [THS+11] and *Escherichia coli* K-12 MG1655 [OCN+11]. These models can be loaded with the cobra.test.create_test_model function; with *S. Typhimurium* LT2 being the default model. Additionally, COBRApy can read SBML-formatted models [BKJH08] downloaded from a variety of sources, such as the Model SEED [HDB+10] and the BioModels database [LDR+10].

A common operation performed with M-Models is to optimize for the maximum flux through a specific reaction in a defined growth medium [OTP10]. The S. Typhimurium LT2 model comes with a variety of media whose compositions are specified in the model's media_compositions attribute. Here, we initialize the Model's boundary conditions to mimic the minimal MgM medium [BBD+99] and then perform a linear optimization to calculate the maximal flux through the Reaction biomass_iRR1083_metals. Biomass_iRR1083_metals is a reaction that approximates the materials required to support S. Typhimurium LT2 growth in a minimal medium where approximately 0.3 grams dry weight S. Typhimurium LT2 are produced per hour. It is important to note that cellular composition can vary as a function of growth

rate [SMK58], therefore, for biological accuracy it may be necessary to construct a new biomass reaction if the simulated, or experimentally-observed, growth rate is substantially different [LHL+12, PK97].

Flux balance analysis of M-Models has enjoyed substantial success in qualitative analyses of gene essentiality [THS+11]. These studies used simulations to identify which genes or synthetic lethal gene-pairs are essential for biomass production in a given condition. The lists of essential genes and synthetic lethal gene-pairs may then be targeted to inhibit microbial growth or excluded from manipulation when constructing designer strains [BPM03]. COBRApy provides functions for automating single and double gene deletion studies in the cobra.flux_analysis module.

Because of the presence of equivalent alternative optima in constraint based-simulations of metabolism [MS03], many reactions may theoretically be able to carry a wide range of flux for a given simulation objective. Flux variability analysis (FVA) is often used to calculate the amount of flux a reaction can carry while still simulating the maximum flux through the objective function subject to a specified tolerance. Flux variability analyses can be used to identify problems in model structure [SLP11] or 'pinch-points' in a metabolic network. COBRApy provides automated functions for FVA in the cobra.flux_analysis.variability module.

1.4.3 Advanced capabilities

Because whole genome double deletion and FVA simulations can be time intensive with a single CPU, we have provided a function that uses Parallel Python to split the simulation across multiple CPUs for multicore machines. Additionally, there are a wide range of legacy operations that are present in the COBRA Toolbox

that can be accessed using mlabwrap. MATLAB is only necessary for accessing codes written in the COBRA Toolbox for MATLAB; it is not necessary to run the majority of COBRApy functions.

1.5 Conclusions

COBRApy is a constraint-based modeling package that is designed to accommodate the biological complexity of the next generation of COBRA models [LHL+12] and provides access to commonly used COBRA methods, such as flux balance analysis [OTP10], flux variability analysis [MS03], and gene deletion analyses [LNP12]. Through the mlabwrap module it is possible to use COBRApy to call many additional COBRA methods present in the COBRA Toolbox for MATLAB [SQF+11]. As part of The openCOBRA Project, COBRApy serves as an enabling framework for which the community can develop and contribute application specific modules.

1.6 Acknowledgements

This work was supported in part by the US National Institute of Allergy and Infectious Diseases and the US Department of Health and Human Services through interagency agreement Y1-AI-8401-01. Thanks to Palsson lab members, openCOBRA community, and the mini.cobra course participants (Spring 2012) for feedback, patches, and identifying bugs. DRH is supported in part by a Seed Award from the San Diego Center for Systems Biology funded by NIH/NIGMS (GM085764). JAL was supported by NIH U01 GM102098. This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S.

Department of Energy under Contract No. DE-AC02-05CH11231.

This chapter was copied, in whole or in part, from the following manuscript Ebrahim, A., Lerman, J.A., Palsson, B.O., and Hyduke, D.R. (2013). COBRApy: COnstraints-Based Reconstruction and Analysis for Python. BMC Syst. Biol. 7, 74.

Chapter 2

Distribution and Portability of Validated Genome-Scale Constraint-Based Models

2.1 Introduction and Problem Statement

The number of genome-scale reconstructions has massively increased over the past decade [MNP14], and the types of analyses done with these networks has rapidly increased as well [LNP12]. However, it has remained challenging to distribute these models between different research groups [RR15]. Different methods of parsing for a large number of published models can lead to disputes of whether the models solve or not [EAB+15]. Thus there is a clear need in this field for a unified and standard method of distributing these models.

Traditionally, SBML has been the most widely used standard format for distributing models in computational systems biology [HFS+03], However, it was

originally written to support dynamic models, so many models were instead distributed as ad-hoc spreadsheets. Eventually, many in the constraints-based modeling community moved to using SBML. However, support for SBML models was implemented in various incompatible ways by different software packages, thereby compromising the format's compatibility [DP14]. More insidiously, these various distribution schemes make tacit assumptions about modeling conventions that can easily lead to non-unique interpretations of the model when imported incorrectly. One particularly common pitfall has been the encoding of reaction flux bounds (constrictions of minimal and maximal reaction velocities), which evolved in several different ways. Additionally, many gene-reaction rules (Boolean expressions for the relationships between the genomic and metabolic network) are often incorrectly written and cannot be parsed. Moreover, inconsistent use of metabolite formulas and charges makes validation of model mass balances difficult. Finally, exchange, demand, and biomass reactions are often encoded in different and inconsistent ways, and there is no unified way to designate these reactions as such. These difficulties stemming from ambiguities in file encoding often result in models that appear to be unable to simulate growth to other users who may use slightly different tools. These issues have posed a large challenge to reproducibility of model computations for the constraint-based modeling community, until now.

2.2 A Potential Solution

Many of these problems can be addressed by releasing published reconstructions using newer standard formats, which were designed specifically for creation of constraint-based models. For SBML, a new version of the flux balance constraints

package has been created with a "strict" mode that specifically ensures there are no ambiguities in how a file may be parsed into a model. For web-based tools, the JSON format defined by COBRApy and Escher is based on a schema which forces unambiguous creation of models as well [KDE+15]. However, while both of these formats can be validated for syntactic compliance, further semantic validation is still necessary to ensure the models produced are usable for computations.

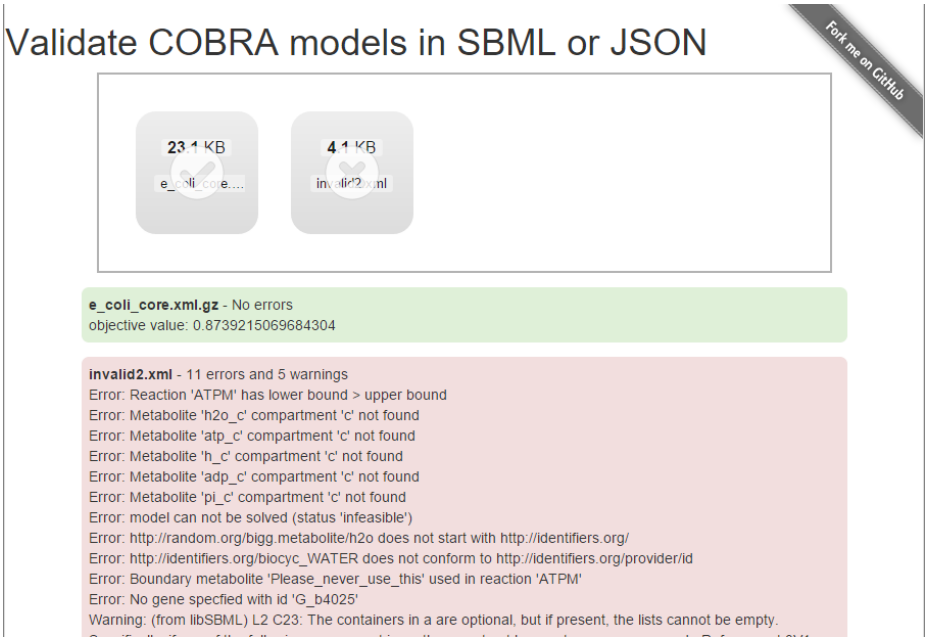


Figure 2.1: Depiction of the validator website, taken as a screenshot from http://bigg.ucsd.edu/validator. A successfully validated model has a check mark appear on it, while an unsuccessful model has an icon of an x, along with the error messages highlighted in red.

To this end, we have developed a validator (Figure 2.1) for metabolic network reconstructions distributed as SBML and JSON files, hosted at http://bigg.ucsd.edu/validator. This validator is free and open-source (under the MIT license) web-based software, making its use by researchers and peer reviewers as simple as possible. In addition to testing the file format syntax [BKJH08] and that a model can be correctly read from the file, the validator also runs semantic tests on the model itself. The

validator ensures that all reactions are either mass balanced, or are correctly annotated as a non-balanced reaction type, such as a demand, exchange, or biomass reaction using the corresponding term from the systems biology ontology (SBO) [CJK+11]). Additionally, it tests that a biomass reaction is correctly identified, and uses an exact solver to determine if this biomass reaction can sustain a non-zero flux [ACDE07]. While automated validation is not a substitute for a quality-controlled reconstruction process [TP10], the availability of this validator will help ensure published models conform to this established set of quality metrics. If peer reviewers and model authors alike ensure all published models pass validation by this tool and other ones like it, the constraint-based modeling community can overcome the current challenges it faces with model portability and computational reproducibility.

2.3 Acknowledgments

This work was funded by generous support from the Novo Nordisk Foundations through the NNF Center for Biosustainability at DTU, and by the 7th EU Framework Program for Research and Technological Development as part of a Marie Curie International Outgoing Fellowship (project AMBiCon, 332020). We acknowledge support by the German Research Foundation (DFG) and the Open Access Publishing Fund of the University of Tuebingen.

This chapter was copied, in whole or in part, from the following manuscript: Ebrahim, A., King, Z.A., Jamshidi, N., Palsson, B.O., Lewis, N.E., Drager, A. (In preparation). Distribution and Portability of Validated Genome-Scale Constraint-Based Models.

Chapter 3

Model-driven elucidation of transcriptional regulatory network in bacteria

3.1 Abstract

A model-driven approach to experimental design was applied to elucidate the transcriptional regulation by two major transcription factors (TFs), NtrC and Nac, in nitrogen metabolism of *Escherichia coli*. Genome-wide measurements with ChIP-exo and RNA-seq were performed using alternative nitrogen sources predicted by genome-scale models to activate these responses and to make differential activation of Nac. A total of 19, 249, 153, and 2171 binding sites for NtrC, Nac, RpoN and RpoD, respectively were identified, and NtrC associates preferentially with RpoN-dependent promoters, while Nac interacts with RpoD-dependent promoters. Functional analysis of the two regulons showed that the NtrC regulon primarily responds to

nitrogen limitation by attempting to increase nitrogen availability. Nac, on the other hand, re-balances flux through carbon metabolism to accommodate the change in the nitrogen source. A systems-biology computational approach was required to reconcile the behavior of these two transcription factors into a detailed and quantitative understanding of how the metabolic network responds to different nitrogen sources.

3.2 Introduction

Revealing the transcriptional regulatory network (TRN) in bacteria is important to understand metabolic flexibility and robustness in response to environmental changes [SIBG08]. A productive way to elucidate TRN at the systems level is by integrating multiple 'omics' datasets, such as ChIP (chromatin immunoprecipitation), expression profiling, and genome-scale TSS profiling. The first step of such an undertaking is to determine the relevant growth conditions where a transcription factor (TF) of interest is expected to be maximally active. Determining experimental conditions for TF activation has been frequently based on information from the literature or intuition. This approach has been used effectively, especially for specific or locally working TFs, which have one or a few activation conditions. Regulon elucidation has been performed for a number of subsets of bacterial metabolism including carbon metabolism, aerobic/anaerobic metabolism [CKP06b, FKE$^+$14, MYO$^+$13, PAA$^+$13], iron metabolism [SKL$^+$14], and other metabolism with nitrogen containing molecules, such as Lrp [CBK$^+$08], ArgR/TrpR [CFP$^+$11], PurR [CFE$^+$11], and RutR [SIBG08] establishing the use of key experimental methods. However, in the most of these genome-wide studies, it was assumed that the conditions chosen in those studies represent the optimal conditions to study the TFs of interest, and it has been often

ignored the possibility of differential activation or activity of those TFs under multiple conditions. For instance, there could be multiple conditions that activate global TFs, and choosing experimental conditions based on the literature could limit assessing which experimental condition is better than another to stimulate the activity of TFs in many cases.

The development of genome-scale network models, including recently developed models of the metabolic network [OCN+11] and of metabolism and expression [OLC+13] for *E. coli*, enables *in silico* exploration of virtually every imaginable experimental condition. Comparison of different network states computed under different candidate experimental conditions can shed light on varying cellular responses to the environmental changes and thus the regulatory requirements for the shift from one condition to another. For instance, transcriptional change predicted with the ME model can present a cellular response in the transcription level to the environmental signal. Thus, combining model-based simulations with existing, but limited, regulon information from the public database can assess the experimental conditions *in silico*, and can result in the prediction of conditions where a TF could be activated. In extension of this approach, it is possible to choose multiple conditions to render possible differential activation of TFs that the conventional and non-systems approach suggested to be activated in a similar fashion.

In order to exploit the use of this model-driven experimental design approach, two major TFs in response to nitrogen limitation, NtrC, and Nac [CPGO98, MB98, ZSL+00], were chosen to search optimal conditions to activate those TFs and to produce possible differential activation of any TF under nitrogen limitation. Nitrogen metabolism is one of key parts in *E. coli* metabolism; however the transcriptional

regulation of nitrogen metabolism has been studied in a limited scope. For instance, *in vivo* binding of NtrC and Nac has not been experimentally determined at the genome-scale. Cutting-edge ChIP-exo (chromatin immunoprecipitation with exonuclease treatment) [RP12b, SKL$^+$14] and RNA-seq were performed to reconstruct the NtrC and Nac regulons, and further model-based analysis was conducted to elucidate distinct roles of NtrC and Nac regulons on nitrogen metabolism.

3.3 Results

3.3.1 Model-driven prediction of activation conditions for TFs

E. coli has been experimentally confirmed to have the ability to utilize several nitrogen sources [Rei03]. Glutamine has been repeated used as an alternative nitrogen source for nitrogen limitation in multiple publications [CKK$^+$14, CZQ$^+$09, ZSL$^+$00]. To comprehensively explorer the possibility of other nitrogen-containing molecules being utilized by *E. coli* and being more effective to render nitrogen limitting conditions thus to activate NtrC and Nac, a genome-scale model of metabolism (M model) in *E. coli* [OCN$^+$11] was used to determine viable candidate nitrogen sources. 93 nitrogen-containing nutrients that have known transporters and support *in-silico* growth were chosen and used for the further analysis (Figure S1A, Figure S1B). Then, a model of metabolism and gene expression (ME model) in *E. coli* [OLC$^+$13] was used to simulate growth on glucose and those 93 viable nitrogen source candidates (detailed description in Experimental Procedures). From simulated results, predicted gene expression for each alternative nitrogen source was compared with predicted gene

expression for growth on ammonia to find a set of predicted differentially expressed genes (Figure S1C). Previously annotated TF binding information from EcoCyc [KMPG$^+$13] was used to calculate which TFs are enriched in the predicted set of differentially expressed genes for each alternative nitrogen source (Figure S1D, Figure 3.1). For NtrC, glutamine, cytidine, and cytosine were predicted to be nitrogen sources that would change the expression of genes in which NtrC regulons were statistically enriched. Thus, NtrC activation was predicted under those conditions. However, Nac was predicted to be active under cytosine, but not cytidine. Ammonia was used as a negative control as it is known not to activate NtrC and Nac. Glutamine was used as a positive control to activate the two TFs [ZSL$^+$00]).

Based on the simulation results, cytidine and cytosine, which are not the favored alternative nitrogen sources in the conventional experiment, were determined to be best alternative nitrogen sources to activate NtrC, and to illustrate the differential activation of Nac. Thus, *E. coli* was grown on 4 nitrogen sources, ammonia, glutamine, cytidine, and cytosine, and a series of experiments including expression profiling and ChIP measurements were performed to confirm the activation of TFs under those conditions, and to experimentally measure expression change and *in vivo* TF binding sites on the *E. coli* genome. From the experimental measurements, an expanded definition of NtrC and Nac regulons was obtained, which was subsequently used with flux simulation from the metabolic models to identify roles of NtrC and Nac regulons in responding to nitrogen limitation (Figure 3.1).

3.3.2 Experimental confirmation of predicted conditions

Activation of a TF results in a series of events including transcriptional activation of a gene encoding the TF, increasing translation from the transcript resulting in more protein, and regulation of target genes by binding onto genomic locations. First, to see if there is up-regulation of *ntrC* and *nac* transcription, RNA-seq was performed with *E. coli* K-12 MG1655 cells grown on ammonium, glutamine, cytidine, and cytosine up to mid-log phase. *amtB*, which encodes an ammonia transporter, was reported to be up-regulated when glutamine was supplemented as a sole nitrogen source [ZSL$^+$00]. Therefore, transcription levels of *amtB*, *ntrC*, *nac* and 3 subunits of RNA polymerase (RNAP), *rpoN*, *rpoD*, and *rpoB*, were compared between ammonium and three alternative nitrogen sources (Figure S2). In all alternative nitrogen sources, transcription of *amtB*, *ntrC,* and *nac* was significantly up-regulated, while no significant change in the transcription level of RNAP subunits was observed. To further the comparison, the transcription level of 4,595 annotated genes in the *E. coli* genome was analyzed to see how broad the response to nitrogen limiting condition is in terms of transcriptional change (Figure 3.2A, Figure 3.2B). Using alternative nitrogen sources, glutamine, cytidine, and cytosine resulted in 667, 390, and 690 differentially expressed genes respectively, and the expression of 1046 genes (22.8%) was changed under at least one, leaving the expression of 3548 (77.2%) genes unchanged in all conditions. Expression of 221 genes was changed under all conditions. This number of differentially expressed genes is much larger than previously reported, where about 100 genes were determined to be of the nitrogen regulated (Ntr) response [Rei03].

To ascertain that up-regulation in the transcription of *ntrC* and *nac* resulted in an increased amount of protein, western blotting was performed to investigate

the protein level of NtrC, Nac, RpoN, and RpoD (Figure 3.2C). In agreement with expression profiling with RNA-seq, the protein expression of NtrC and Nac increased in glutamine, cytidine, and cytosine, although the amount of Nac protein on cytidine was much lower than that on glutamine and cytosine. This lower level may explain why the number of differentially expressed genes on cytidine was the lowest among the three conditions (Figure 3.2B).

The final component of transcription factor activation is its binding to particular genomic locations resulting in the regulation of expression of target genes. Experimental measurement of TF binding to the genome was performed with recently developed ChIP-exo (Chromatin immunoprecipitation with exonuclease treatment) by adopting the original protocol [RP12b], but modifying it for bacterial use [SKL$^+$14] (Detailed protocol in Experimental Procedures section). In total, 19, 249, 153, and 2171 binding sites across the four growth conditions were identified for NtrC, Nac, RpoN, and RpoD respectively (Figure 3.2A, Figure S3). The number of binding sites for two σ-factors, RpoN and RpoD, did not change much, whereas binding sites of NtrC and Nac increased on 3 alternative nitrogen sources. For instance, the number of binding sites for NtrC increased from 5 to 19, and Nac bindings increased from 15 to > 240.

In summary, transcriptional expression of key components in response to nitrogen limiting conditions was significantly up-regulated, which was reflected in an increase in protein abundance of those TFs. Direct measurement of TF binding sites *in vivo* in a genome-scale manner showed increasing binding events of those two TFs to the genome. Thus, alternative nitrogen sources, that were predicted from the model-driven simulation, activated NtrC and Nac in transcriptional, translational,

and binding activity leading to the discovery of 11 and 245 new binding sites for NtrC and Nac respectively.

3.3.3 Confirmation of ChIP-exo binding sites with motif analysis and comparison to known sites

In brief, ChIP-exo applies a 5'-3' strand-specific exonuclease to a chromatin immunoprecipitated sample. Deep sequencing of an exonuclease-treated ChIP sample enables the detection of exonuclease stop sites with near 1-bp resolution [RP12b, SKL$^+$14]. The ChIP-exo method is claimed to have a better resolution and sensitivity, such that the detection of very narrow peaks and weak binding sites, which was difficult with ChIP-chip or ChIP-seq. To illustrate the resolution improvement that the ChIP-exo method presents, ChIP experiment results for RpoD from the same condition with 3 different methods, ChIP-chip [CKK$^+$14], ChIP-seq, and ChIP-exo, were compared to show binding signals upstream of *rpsU-dnaG-rpoD* and *ileX* operons (Figure S4). Binding detected with the three methods is aligned near the center of the figure. ChIP-exo clearly presented the best resolution, and ChIP-seq showed a better resolution than ChIP-chip (Text S1).

Since binding peaks detected with the ChIP-exo method were so narrow, it became of interest to see if a sequence motif would be found from those peak regions, and where the sequence motif lies inside the regions. To address these questions, MEME software [BEO94] was used to retrieve sequence motifs. The sequence motifs from NtrC, Nac, RpoN, and RpoD binding sites were GCaCcaaaAtgGtGC, tGGcacgattttTGCa, ATAagnaaaanttAT, and ttgaca-15bp-gntAtaaT (lower-case characters indicate an information content <1 bit). These motifs were identical to the known motifs [CKK$^+$14,

KMPG$^+$13, NKR$^+$13, PJB98]. Except for the RpoD motif, sequence motifs were located near the center of and inside binding regions (Figure 3.2D). For RpoD, only the -10 box, gntAtaaT, was found inside the binding regions. This observation conflicts with the knowledge of RpoD, because RpoD is known to specifically recognize -10 and -35 box sequences, which are expected to be covered and protected from exonuclease activity. RpoD binding peaks align well with RpoB binding peaks (Figure S5) and transcription start sites (TSSs) were dominantly located at the center of binding regions; thus ChIP-exo might be capturing RpoB bindings that were associated with RpoD.

From 19 and 249 total binding sites for NtrC and Nac, respectively, 16 and 247 binding sites were found upstream of genes, hence called regulatory binding sites. These binding sites were compared to 4 and 3 known binding sites for NtrC and Nac. For NtrC, all of the known sites upstream of *glnL*, *glnA*, *glnH*, and *astC* [ABB$^+$02, CMM91, NRM87, UNMRM84] were detected in the dataset of this study (Figure 3.3A). Similarly, 2 of 3 known Nac binding sites, upstream of *codB* and *ydcS* [MRB03, SHR13], were detected, however Nac binding upstream of *nac* [KMPG$^+$13] was not. The claim of Nac binding for *nac* was based on evidence from experiments on *K. aerogenes* and sequence alignment *nac* promoter regions between *E. coli* and *K. aerogenes* [MB98]. However, no binding was detected from the ChIP-exo dataset (Figure S6), and *nac* does not have an RpoD promoter, thus it may be more likely that Nac does not regulate *nac* in *E. coli*.

In summary, sequence motifs, which TFs presumably recognize and bind to, were located near the middle of the binding regions and were identical to previously reported motifs. NtrC and Nac binding sites identified in this study covered all

known binding sites, while expanding the current knowledge by about 4 and 82 fold, respectively (Figure 3.3A).

3.3.4 Reconstruction of NtrC and Nac regulons

Although post-translational regulation of glutamine synthetase (*glnA*) by GlnD, GlnK, and GlnB have been extensively studied [ABN02, BN02, GR83, JSTM04, LM95, vHWC$^+$00, vHHM$^+$96, VGD$^+$94], limited information about regulation of nitrogen metabolism at the transcriptional level has been available. To shed light on transcriptional regulation by NtrC and Nac, their regulons were reconstructed by associating TF bindings with transcription units [CKK$^+$14, CZQ$^+$09]. From 3181 TUs covering 4485 genes (97.6% of annotated genes), 19 TUs were associated with NtrC and 223 TUs were associated with Nac (Figure 3.3B).

An interesting property of the TUs found in these two regulons is that they barely overlap with each other; they only have one TU in common, *insH-3* (Figure 3.3B). In the latest definitions of TUs [CKK$^+$14, CZQ$^+$09, KMPG$^+$13], transposon-related *insH-3* was annotated to make one TU on its own. However, RNA-seq profiling with paired-end reads suggested that there might be a longer transcript starting from *insH-3*, possibly including *gltI-sroC-gltJKL* (Figure 3.3C). Confirming this possibility, 49.3% of sequence reads covering intergenic regions between *insH-3* and *gltI* overlapped both with insH-3 and *gltI*, indicating contiguous transcription of those two genes. A similar approach was applied to the 6 genes downstream, and it seems there is a longer TU with at least 6 genes from *insH-3* to *gltL*. Lack of promoter upstream of *sroC* also supports this possibility [VBT$^+$03]. Thus, two promoters for RpoD and RpoN upstream of *insH-3* are likely to contribute to the transcription

of glutamate/aspartate ABC transporter (*gltIJKL*), and binding of NtrC and Nac are associated with those σ-factors (Figure 3.3C). Transcriptomic expression of the longer TU was up-regulated under nitrogen-limiting conditions by NtrC and/or Nac, resulting in increased glutamate/aspartate transporters as a scavenging mechanism.

Transcriptomic comparison to the *K. pneumoniae* genome [KHQ$^+$12] gives more insight into TU organization and its conservation (Figure S7). In *K. pneumoniae*, *gltI-sroC-gltJKL* and two upstream coding genes, *lnt* and *ybeX*, are all conserved, and transcription starts upstream of *gltI* in both species. However, *insH-3* is missing in the *K. pneumoniae* MGH78578 genome. Thus, TU of *gltI-sroC-gltJKL* is conserved in *E. coli* and *K. pneumoniae*, and possibly mostly in enterobacteria, but somehow *insH-3* got into 5' UTR of this TU in *E. coli* K-12 MG1655.

Thus, the reconstruction of the NtrC and Nac regulons presented scant overlapping between the two, revealing distinct roles in response to nitrogen limiting conditions and in regulating nitrogen metabolism.

3.3.5 Association of TF with σ-factors

The definition of two TF regulons raised the question of which σ-factor is associated with each TU in regulons, because NtrC and Nac have been postulated to interact with different σ-factors. NtrC belongs to the RpoN-dependent activator family [SB00] and interacts with RpoN [HNY$^+$06]. Nac is postulated to serve as an adaptor between NtrC and final RpoD-dependent promoters [ZSL$^+$00], but *in vivo* genome-wide experimental evidence is still missing. Thus, NtrC and Nac binding sites were combined with RpoN and RpoD binding sites to see how these TFs are associated with σ-factors.

From the ChIP-exo datasets and a calculation of closely located binding sites from the dataset, NtrC binding was associated with RpoN binding, and Nac with RpoD (Figure 3.3D). Out of 19 NtrC binding sites, 16 were found with RpoN binding near them upstream of the same gene, 9 of which were from complicated promoters with RpoD and RpoN binding sites. For instance, *glnA* has a distal RpoD-dependent promoter (glnAp1) and a proximal RpoN-dependent promoter (glnAp2) [KMPG$^+$13], which were captured from the ChIP dataset (Figure S8). NtrC binding was found upstream of the RpoN-dependent promoter, whereas no Nac binding was observed near the RpoD-dependent promoter. The upstream regulatory region of *insH-3-gltI-sroC-gltJKL* gives another example of complicated promoters (Figure 3.3C). These results reflect the previously reported extensive overlap between the RpoD and RpoN regulons [CKK$^+$14]. The majority of Nac bindings (167, 67.1%) adjoined RpoD bindings; while 15 binding sites were found in promoters having both RpoD and RpoN binding sites (Figure 3.3C). NtrC works dominantly as a transcription activator on RpoN-dependent promoters by binding upstream of the promoter in many cases (Figure 3.3E). Nac works as a dual regulator on the RpoD-dependent promoter; it binds more upstream of a promoter when up-regulating the downstream gene, while it binds downstream of a promoter when down-regulating (Figure 3.3E).

Thus, NtrC binds in the vicinity of where RpoN binds, while Nac does so near RpoD. There are multiple promoter regions that have RpoN and RpoD-dependent promoters; however NtrC and Nac bind separately except for *insH-3-gltI-sroC-gltJKL*, implying distinct roles of two regulons.

3.3.6 Contrasting functions of NtrC and Nac regulons

In addition to the observation that NtrC and Nac regulons barely overlap, functional analysis of regulons sheds more light on the distinct functions of NtrC and Nac in response to nitrogen limitation. NtrC up-regulated 41 genes (30 genes under all alternative nitrogen sources, and 11 genes in some conditions), and down-regulated 2 genes (*yeaE* in all conditions, and *mipA* only in cytosine), leaving the expression of 3 genes not changed or changed less than 2 fold (Figure 3.3F). The NtrC regulon contains 18 transporters or their subunits, 3 TFs, 1 sRNA, and 28 other enzymes. Transporters are mostly for nitrogen sources, including ammonia (*amtB*), glutamine (*glnHPQ*), glutamate (*gltIJKL*), histidine (*hisJ*), lysine/arginine (*hisQMP*, *argT*), xanthine/uracil (*rutG*), and others are less characterized (*yhdWXYZ*). NtrC also up-regulates regulatory proteins, including 3 TFs, *ntrC* itself, *nac*, and *cbl* and 2 post-translational regulatory proteins (*glnK*, and *glnL*). While the role of *cbl* in nitrogen response is still elusive, it is clear that NtrC regulates major regulatory enzymes responding to nitrogen limitation. Metabolic enzymes that NtrC regulates catalyze reactions for nitrogen-containing molecules, including glutamine (*glnA*), pyrimidine (*rutABCDEF*), arginine (*astCADBE*), and D-alanyl-D-alanine (*ddpXABCDF*).

While NtrC regulates a smaller set of genes and mostly activates the expression of target genes, Nac regulates a larger group of genes and works as a dual regulator by up-regulating 70 genes and down-regulating 79 genes (Figure 3.3F). Another difference is that NtrC regulates mostly nitrogen-related regulatory proteins, transporters, and metabolic enzymes, while Nac covers beyond nitrogen-related enzymes. For instance, Nac binds upstream of *gltP* (glutamate/aspartate transporter) and *codB* (cytosine transporter), but it also binds upstream of carbon source transporters such as *mglBAC*

(galactose ABC transporter). The Nac regulon includes a number of mostly locally acting TFs, some of which are known to be related carbon metabolism or in both carbon/nitrogen metabolism (Text S2). Moreover, Nac interestingly regulates some key genes in glycolysis and the TCA (Tricarboxylic acid cycle): phosphofructokinase (*pfkA*, and *pfkB*), citrate synthase (*gltA*), succinate dehydrogenase (*sdhCDAB*), 2-oxoglutarate dehydrogenase (*sucAB*), and succinyl-CoA synthetase (*sucCD*). COG analysis of the NtrC and Nac regulons showed genes for amino acid metabolism and signal transduction are more enriched in the NtrC regulon, while the Nac regulon has genes functionally enriched in energy production, amino acid metabolism, and two other categories (Figure S9).

In summary, NtrC with RpoN regulates TFs, transporters, and metabolic enzymes required for responding to nitrogen-limiting conditions. However, Nac, accompanied by RpoD, is responsible for regulating a broader set of genes beyond those that are directly nitrogen-related. Thus, NtrC and Nac have different functions in response to nitrogen-limitation; however the role of the Nac regulon in the response seems less obvious than the NtrC regulon.

3.3.7 Primary response of the NtrC regulon

Glutamine is the central molecule with which *E. coli* cells sense nitrogen-limiting conditions [ISK96]. The Ntr regulatory cascade is triggered by sensing the low level of glutamine to activation of NR_I with phosphorylation, which is followed by a relay of post-translational regulation (Figure 3.4A). This Ntr regulatory cascade has been extensively studied [Rei03]. In brief, low glutamine stimulates UTase (uridylyl-transferase) activity of GlnD, which is a single peptide with UTase and UR (uridylyl-

removing) activities, by binding to a single site on the enzyme [JPN98]. UTase activity of GlnD uridylylates two functionally redundant proteins, P_{II} and GlnK. Uridylylated P_{II} and GlnK fail to interact with NR_{II}, which results in a net phosphorylation of NR_I [NA00]. Phosphorylated NR_I is an active form, activating transcription of NtrC regulon genes. Uridylylation of P_{II} and possibly GlnK stimulates the deadenylylating activity of ATase (glutamine synthetase adenylyltransferase/deadenylase, *glnE*), and GlnE deadenylates glutamine synthetase (*glnA*) making it an active form [FHSW99]. As a result of this cascade, the internal level of glutamine can increase.

Unlike post-translational regulation of the Ntr regulatory cascade, the transcriptional level of this regulation has been less studied. In this regulatory cascade consisting of 10 genes, 8 genes are regulated either by NtrC or Nac. 5 genes, *ntrC*, *nac*, *ntrB*, *glnA,* and *glnK*, are up-regulated by NtrC, 1 gene, *glnD*, was up-regulated by Nac, and 2 genes, *gltB* and *gltD*, were down-regulated by Nac, while 2 genes, *glnB* and *glnE*, are not regulated by any of them (Figure 3.4A, Figure S10). Thus, NtrC and Nac control the majority of regulatory components in this cascade, forming multiple positive-forward loops. These loops make a complicated network with well-characterized network motifs including a coherent type 1 feed-forward loop (C1-FFL) and a positive auto-regulation (PAR). C1-FFL is a frequent motif found in *E. coli* [MA03] and functions as a sign-sensitive delay element and a persistence detector [MZA03], and PAR shows a slower response time than simple regulation, and may lead to a bimodal distribution of protein levels [BSS01]. These properties may contribute to the filtering out of short signals of nitrogen limitation, rendering a response with a short delay for persistent signals, and quickly shutting off the output when a nitrogen-limiting condition is relieved.

In *E. coli*, cytoplasmic glutamine is either synthesized from glutamate by glutamate synthetase (*glnA*) or is transported by the glutamine ABC transporter (*glnHPQ*) (Figure 3.4B). Both operons were up-regulated by NtrC and its associated RpoN-dependent promoters. Other than asparagine synthetase B (*asnB*), all genes that consume glutamine was not changed (fold change < 2) or down-regulated (Figure S11), indicating that *E. coli* cells change the abundance of metabolic machinery towards increasing intracellular glutamine level. In *E. coli*, glutamate can be built up from α-ketoglutarate in two reactions. One is by glutamate dehydrogenase, which is encoded by *gdhA,* and the gene has an RpoD-dependent promoter for constitutive expression and expression of *gdhA* did not change significantly with alternative nitrogen sources. The other reaction is by glutamate synthase, which is encoded by *gltBD*, and the operon also has an RpoD-dependent promoter; however Nac negatively regulates expression of *gltBD* upon alternative nitrogen sources (Figure S11).

Thus, securing a glutamine pool under nitrogen limitation necessitates expression changes of regulatory and metabolic enzymes in the Ntr regulatory cascade, and NtrC plays a central role in this complicated regulation. In addition to transcriptional regulation in the cascade, NtrC also activates transporters for favorable nitrogen sources as a scavenging mechanism, and induces expression of the other key TF, *nac*. Moreover, it becomes of interest how *E. coli* cells manage production and consumption of α-ketoglutarate under nitrogen limitation, because production of intracellular glutamine requires the expense of α-ketoglutarate, which is one of the key molecules in carbon metabolism.

3.3.8 Carbon flux rebalancing by the Nac regulon

Since α-ketoglutarate is one of the main intermediates in the TCA cycle, glycolysis and the TCA cycle pathways were analyzed in terms of TF and σ-factor bindings and expression change (Figure 3.5A). Genes in those pathways were transcribed from RpoD-dependent promoters, and surprisingly, 11 of them were regulated by Nac. Nac repressed expression of genes in the TCA cycle, *pck*, *sucAB*, *sucCD*, and *sdhBADC*. Nac also repressed glycolysis genes, *pfkA*, *pfkB*, *fbaA*, and *ppc*, but the expression fold change was less than 2. Nac did not bind upstream of genes in the PPP (pentose phosphate pathway), except for those that work in glycolysis and the PPP at the same time.

Interestingly, among genes in the TCA cycle, those downstream of α-ketoglutarate in the pathway were regulated by Nac, and those upstream were not. It was postulated that genes encoding enzymes downstream of α-ketoglutarate would be more repressed for two reasons. First, Nac works as a repressor on enzymes in this pathway (Figure 3.5A). Second, cells are under nitrogen-limiting stress, which directly constrains growth, and α-ketoglutarate is a precursor for glutamine synthesis. As postulated, all downstream genes, *sdhBADC*, *sucCD*, *lpd*, and *sucAB*, were more down-regulated than upstream genes, *icd*, *acnAB*, and *gltA* when on cytosine (Figure 3.5B). The degree of repression on glutamine and cytidine was less than on cytosine; however two alternative nitrogen sources also showed the same regulatory pattern (Figure S12).

In summary, the response to nitrogen limitation accompanies changes in gene expression of enzymes in the TCA cycle and glycolysis to rebalance carbon flux towards facilitating glutamine pool maintenance. In this process, Nac plays an important role

by regulating genes in carbon metabolism.

3.3.9 Lower activation of Nac on cytidine

The NtrC regulon has a primary responsibility of responding to the nitrogen limiting environment, while the Nac regulon has a subsidiary role of rebalancing carbon metabolism for the nitrogen source shift. The remaining question is why protein expression of Nac on cytidine was much less than that on glutamine and cytosine (Figure 3.2C). Interestingly, the number of differentially expressed genes on cytidine was significantly less than on the other alternative nitrogen sources (Figure 3.2B). Similarly, expression change of metabolic genes in the TCA cycle on cytidine, which include genes regulated by Nac, was the least among the three nitrogen sources (Figure 3.5A, Figure S12). These observations raise the question as to the molecular basis of this lower activation of Nac on cytidine.

To explain observation, flux balance analysis (FBA) with the *E. coli* metabolic model [OCN$^+$11, OTP10] and experimentally measured glucose uptake rate was performed to calculate internal flux states under different nitrogen sources (Figure 3.5C, Figure S13). On ammonia, the glucose uptake rate was 8.86 mmol/gDW/hr, and 39% of g6p (glucose 6-phosphate) went into PPP, leaving 61% remaining in the glycolysis pathway. Flux distribution through g6p on cytosine is the same as ammonia, but with a lower glucose uptake rate of 6.54 mmol/gDW/hr. On cytidine, 7.04 mmol/gDW/hr flux into g6p was split differently, with more flux towards PPP (49%) and less flux to downstream glycolysis (51%). Interestingly, however, most fluxes through PPP increased, and as a result, flux from f5p (fructose 5 phosphate) and all downstream fluxes increased. This increased flux in glycolysis resulted in more

flux through the TCA cycle, as well (Figure 3.5C). Thus, compared to ammonia, there is lower or no need, to repress activities through the TCA cycle on cytidine, which in turn requires less repression of genes of enzymes. This means there is less need for Nac to repress genes, which potentially explains why less protein level and activity of Nac was observed on cytidine.

Further analysis gives insight into how increased flux through PPP was possible on cytidine (Figure S14). Flux from cytidine uptake went into the cytidine deaminase (*cdd*) reaction to make uridine and ammonium. Uridine breaks into uracil and ribose 1-phosphate (r1p) by uridine phosphorylase (*udp*). This r1p is converted to r5p, which can go into PPP. To accommodate more r5p, there should be more xylulose 5-phosphate (xu5p), which explains more flux into PPP from g6p on cytidine. This analysis also provides a better understanding of how differently uracil could be used when cytidine or cytosine is a sole nitrogen source (Figure S14). Uracil still has 2 nitrogen molecules, however, in order to harvest them, 2 molecules of NADH and 1 of NADPH are required. FBA showed cells could fully assimilate nitrogen from cytidine by utilizing more energy, which could be a part of more activated glycolysis and TCA cycle on cytidine. However, on cytosine, FBA predicted that cells would take one nitrogen molecule from cytosine and export uracil out of the cell, which is less efficient but requires less energy.

Estimating fluxes through a metabolic reaction using the *E. coli* model and FBA analysis under different nitrogen sources helped explain the seemingly contradictory observation of lower activation of Nac on cytidine. This lower activation was due to a ribose part of cytidine could be utilized to make more energy and result in more activated glycolysis and TCA cycle. FBA also gave insights into how cells could

determine how many nitrogen molecules can be extracted from uracil depending on the energy availability in the cell.

3.3.10 Coupling of nitrogen and carbon metabolism

Regulation by Nac on genes in glycolysis and the TCA cycle, and expression and flux changes of reactions in those pathways on different nitrogen sources showed a coupling between nitrogen and carbon metabolism under nitrogen limiting conditions. In light of this observation, it was investigated whether other nitrogen sources would have this same coupling. In order to further examine this possibility, a set of nitrogen sources were chosen that was compatible with model-based simulation and are known to support growth of *E. coli*. A viable set of nitrogen sources was collected by compiling knowledge from literature [Rei03] and a public database [KBMCV+09], and resulted in 44 nitrogen sources that have been tested for *E. coli* growth. These viable nitrogen sources were matched up with computational predictions, and 27 of them were predicted and experimentally confirmed to support *E. coli* growth, while 10 of them were predicted and experimentally verified not to support growth. 7 of the sources were predicted to support growth but *in silico* growth was disproved by experimental confirmation, resulting in an 84% prediction rate (Figure 3.6A). From 27 sources predicted and experimentally proven to support growth as a nitrogen source with glucose for a carbon source, 25 of them, excluding ammonia and cytosine, were predicted to support growth without any carbon source in simulation. This means that most of the nitrogen sources that *E. coli* expects to utilize could work as carbon sources as well. Cytosine was the only one organic nitrogen source that was predicted not to allow growth *in silico*, but a growth experiment of *E. coli* in minimal media

supplemented with cytosine confirmed growth.

To explore the effect of using nitrogen sources that can also be carbon sources, FBA was performed for 27 nitrogen sources to see if using those nitrogen sources would change fluxes through glycolysis and the TCA cycle. Most of reactions were predicted to have increased fluxes when compared to ammonia on 25 nitrogen sources that could support growth without any other carbon source (Figure 3.6B). Cytosine, however, was predicted to have slightly lower fluxes through glycolysis and the TCA cycle as postulated, while cytidine, which showed less activation of Nac activity, was to have higher fluxes through those pathways in model simulation. Thus, Nac may work in the coupling of nitrogen and carbon metabolism and become activated when *E. coli* is exposed to non-favorable nitrogen sources and the TCA cycle is not active enough to pump out enough α-ketoglutarate and energy molecules to harvest nitrogen molecules.

To put this regulatory mechanism of nitrogen metabolism by NtrC and Nac in a broader context, gene conservation analysis was performed on genes in transcriptional regulation, post-translational regulation, and metabolic enzymes in glutamine metabolism and the TCA cycle (Figure S15). Glutamine-related genes, particularly *glnA*, are conserved in all range of bacteria and even in archaea. Genes in the TCA cycle are less conserved than glutamine-related genes, but still well conserved in full or in part. Interestingly, regulatory enzymes in nitrogen metabolism are significantly less conserved. σ-factors, *rpoD,* and *rpoN*, seem ubiquitous in the bacteria kingdom. However, *ntrC* and *ntrB*, which are located distant in *E. coli*, exist only in a subset of proteobacteria, which agrees with previous reports [Fis99]. *nac* is the least conserved regulator only found in some genera including Escherichia and Shigella. *nac* conser-

vation in Salmonella is controversial because the protein identity is less than 35%, and *nac* was not detected with southern blot [MB98]. Moreover, the *nac* orthologs STM0692 in *S. typhimurium* LT2 is found right after *fur*, which is not its expected location (Figure S16).

In summary, nitrogen sources that bacteria anticipate to utilize can activate regulation not only on nitrogen metabolism, but also on carbon metabolism. This regulation is mediated by altering intracellular requirements of α-ketoglutarate and by directly using nitrogen sources as carbon sources. These in sum contribute to change in activities in glycolysis and TCA cycle. Nac can control this coupling between nitrogen and carbon metabolism, however there could be many other TFs that are involved in this coupling. In addition, *nac* exists in a limited set of genera in enterobacteria, which also suggests the existence of other mechanisms in this metabolism coupling.

3.4 Discussion

3.4.1 Selection of optimal experimental conditions by model-driven experimental design

Computation with metabolic models systematizes biochemical, genetic, and genomic knowledge into a mathematical framework that enables a mechanistic description of metabolic physiology. In this study, genome-scale models were leveraged to improve our understanding of the regulation of nitrogen metabolism and its interplay with carbon metabolism. First, an approach using the *i*OL1650-ME model was developed to predict transcription factor activity under a given condition. This approach assumes that genes which are computed by the model to change in expression will

in a cell change their expression in response to transcription factors which regulate them. Based on this assumption, experimental conditions were chosen to activate two important, yet relatively uncharacterized, transcription factors involved in responses to nitrogen limitation, greatly increasing the sizes of their known regulons. We believe this approach can be extended to biological regulators of metabolism in general.

This approach could be also exploited to investigate post-transcriptional regulation, such as regulation by sRNA. To examine this possibility, regulatory information for sRNA with experimental evidence was obtained from the public database BSRD [LHC$^+$13], and a hypergeometric test was performed with this information and differential expression from ME model simulation was predicted. Out of about 81 sRNAs annotated in the *E. coli* K-12 MG1655 genome, 30 had regulatory information in BSRD database. Among them, *sgrS*, *ryhB*, *micA*, and *gcvB* were predicted to account for some differential expression under alternative nitrogen sources (Figure S17A). In the previous study, *gcvB* was predicted to be involved in the regulation of amino acid availability from a network biology approach [MCK$^+$11], which is connected to nitrogen metabolism. Similarly, *micA* regulates the *tsx* membrane porin, that is responsible for the uptake of nucleosides and deoxynucleosides [MBSB88], which are popular alternative nitrogen sources. *sgrS* and *ryhB* are more involved in carbon metabolism, such as the TCA cycle [DM12] and glucose transporters [KKMA06], which suggests their possible role in rebalancing carbon fluxes that are induced from a nitrogen source shift. In order to gain more insight on those sRNAs in nitrogen metabolism, their expression changes on alternative nitrogen sources were investigated with RNA-seq expression profiling (Figure S17B). *gcvB* was significantly down-regulated on experimented nitrogen sources, while the other 3 sRNAs up-regulated at least one nitrogen

source. Thus, expression of predicted sRNAs changed on unfavorable nitrogen sources, which suggests their possible involvement in nitrogen response.

The model-driven experimental design approach used in this study provides a method with which to generate hypotheses from systems computation and prediction. As this approach successfully predicted experimental conditions to activate two major regulons in nitrogen metabolism, it can be extended to other TFs or post-transcriptional regulatory components in general.

3.4.2 Model-guided regulon elucidation and characterization

Genome-scale models have been previously used to analyze cellular responses to environmental stimuli [CAK$^+$13, FKE$^+$14, MGK$^+$14]. Therefore, in addition to using modeling to guide experimental design, this previous work was built on to assess the broader physiological role of NtrC and Nac regulons using the *i*JO1366 model. Quantitative changes in flux through pathways associated with carbon metabolism were determined from model-based simulation in response to alternative nitrogen source usage, and it was found that fluxes through major pathways on carbon metabolism are significantly different by which nitrogen sources were used. For example, higher flux through glycolysis, the TCA cycle, and the pentose phosphate pathway with growth on cytidine than with cytosine, which provided insight into the lower activity of Nac on cytidine.

More broadly, computations with metabolic models predict that other organic nitrogen sources which contain carbon as well can influence flux through carbon metabolism, revealing the importance of the different roles of NtrC and Nac given the interplay of carbon and nitrogen metabolism. This study with alternate nitrogen

sources demonstrates the capabilities of genome-scale models for studying broad systematic effects in general for both experimental design and data analysis, and the approach is readily applicable to other transcriptional regulators as well.

3.4.3 Application of ChIP-exo method in bacterial studies

Since the advent of ChIP-exo method used in eukaryotic TF studies [RP11], the method has been extensively deployed in eukaryotic studies [RP12b, SBCC13, VP13, YVB$^+$12], but it has been recently applied to bacterial studies on transcriptional regulation [SKL$^+$14]. The advantages of ChIP-exo methods over other long-established ChIP methods provide for the more accurate identification of binding sites of DNA-binding proteins in bacteria. Better resolution enables the detection of exact binding regions and the identification of possible multiple bindings in the same regulatory region, which was practically impossible with previous methods. In addition, the ChIP-exo method made it possible to measure TF binding events upstream of genes encoding small non-coding RNA products, such as sRNAs and tRNAs. Moreover, ChIP-chip and ChIP-seq results for NAPs (nucleoid-associated proteins) produces rather broad binding peaks [CKBP08, PKA$^+$12], which made it difficult to distinguish possible multiple bindings of NAPs onto genomic DNA. For instance, NAPs with a wrapping function are expected to interact with genomic DNA on multiple contact points; however it is not easy to study this feature of NAPs with previous methods. The application of ChIP-exo method to NAPs is expected to produce more details about the nature of NAP bindings. It could also contribute to more accurate investigations into transcription termination factors, which were studied with ChIP-chip method [MDP$^+$09]. Thus, ChIP-exo method allows more accurate measurements of *in vivo*

binding of DNA-binding proteins including TFs and NAPs in bacteria, which will contribute to a better understanding of components in broad categories including transcription initiation, transcription termination, post-transcriptional regulation, and epigenetic regulation.

3.4.4 TF binding sites in non-regulatory regions

One of the unexpected findings resulting from the ChIP-exo method is the binding of TFs and σ-factors in non-regulatory regions. Given the function of those proteins in gene regulation, they are expected to bind onto regulatory regions mostly near promoters to regulate the gene expression of target genes. However, it has been continuously reported that TFs bind on non-regulatory regions including those inside coding regions. Similarly, an unexpected large number of bindings for NtrC, Nac, RpoN, and RpoD were observed in so-called non-regulatory regions. 4 (21%), 281 (113%), 232 (152%), and 461 (21%) binding sites were identified for them, respectively (Figure S18A). Interestingly, Nac and RpoN showed more bindings on non-regulatory regions than on regulatory regions. Thus, it seems non-regulatory bindings of TFs and σ-factors are common. Non-regulatory bindings also have identical sequence motifs (Figure S18C); however they have significantly weaker binding intensities (Figure S18B, rank sum test p-value < 0.05).

The function or consequence of those bindings is still elusive. Possible explanations include that these bindings are happening because the sequence of binding motifs have been built during the course of evolution [SIBG08], some of them are involved in regulation of non-annotated genes such as anti-sense transcripts, some of TFs could work as NAPs such as *lrp* [DD10] or there are more possible binding sites for TFs but

regulatory bindings are most stable by interaction with neighboring proteins and are to be detected by experimental methods. For σ-factors, some cases were observed where their bindings in coding region were associated with weaker transcription on the opposite strand, indicating possible involvement in anti-sense transcription. It is harder to understand the high number of binding sites of Nac on non-regulatory regions. It could be simply evolutionary effect, or maybe Nac may work as NAP as well.

3.4.5 Nitrogen starvation is coupled to other responses

Coupling between nitrogen and carbon metabolism has been investigated at different levels. Change of glucose uptake and α-ketoglutarate accumulation according to availability of nitrogen is one mechanism to block glucose uptake in nitrogen limitation [DSWR11]. A quantitative physiological approach showed that the expression of carbon catabolic genes changes upon limitation of nitrogen [YOH$^+$13]. This study revealed one molecular mechanism as to how coupling between nitrogen and carbon metabolism is implemented in the mediation of Nac and α-ketoglutarate. Moreover, model-based analysis provided more insight into how Nac-mediated metabolic coupling can respond differently on different nitrogen sources.

In addition to the coupling of nitrogen and carbon metabolism, response to nitrogen limitation can induce a much larger stress response by increasing the ppGpp level in *E. coli* [BBP$^+$14]. *relA* encodes GDP pyrophosphokinase which can produce ppGpp (Figure S19A), and NtrC binds upstream of *relA* (Figure S19B). Under the experimented nitrogen sources, expression of *relA* increased (Figure S18C).

Response to nitrogen limitation requires not only a response in nitrogen

metabolism, which is essentially regulated by NtrC, but also a response in carbon metabolism to accommodate the flux shift in glycolysis and the TCA cycle on alternative nitrogen sources, which is, in part, regulated by Nac. In addition, nitrogen limitation can also induce a broader ppGpp stress response mediated with NtrC. Thus, nitrogen starvation is coupled to multiple other responses.

In conclusion, model-driven experimental design proved effective in determining activation conditions for regulatory TFs in nitrogen metabolism, and this approach can be extended to study other aspects of regulatory networks in general. The recently developed ChIP-exo method provides advantages over previous ChIP methods, and contributes to more accurate measurements of TFs and σ-factors in nitrogen metabolism. The definition of NtrC and Nac regulons with flux prediction from a bacterial metabolic network model illuminated contrasting roles of those regulons, emphasizing the unexpected role of Nac on the coupling of nitrogen and carbon metabolism. These results reveal a new dimension of systems approaches on transcriptional regulatory network such that integration of model-based simulation and genome-wide experimental measurements can facilitate a better understanding of bacterial transcription regulation, which may be limited with just one approach.

3.5 Materials and Methods

The prediction of activation conditions for transcription factors was performed with simulating ME model [OLC$^+$13] under 93 candidate nitrogen sources and by calculating hypergeometric enrichment of TF bindings from the public database [KMPG$^+$13]. *E. coli* K-12 MG1655 and its derivatives were grown on ammonia, glutamine, cytidine, and cytosine to perform RNA-seq expression profiling and ChIP-

exo experiments by modifying the original protocol [RP11, SKL+14]. Calculation of differentially expressed genes was conducted by using bowtie [LTPS09] and Cuffdiff [TWP+10]). ChIP-exo reads were processed with MACE software (https://code.google.com/p/chip-exo/). Flux analysis to calculate fluxes through metabolic reactions under different nitrogen sources was performed with *E. coli* M model [OCN+11] and COBRApy [ELPH13]. More detailed procedures are described in Supplementary Experimental Procedures section.

3.6 Author Contributions

DK, AE, and BOP concieved the study. DK performed the experiments and did performed the biological analysis. AE performed the modeling computations.

3.7 Acknowledgements

We thank Aarash Bordbar for discussion on interpretation of model-based simulation and Zak King for helping with visualization of metabolic flux states. We also thank Marc Abrams for helpful assistance in writing and editing the manuscript. This research was supported by Novo Nordisk Foundation Center for Biosustainability at the Danish Technical University and by NIH NIGMS (National Institute of General Medical Sciences) grant GM102098. This research used resources of the National Energy Research Scientic Computing Center, which is supported by the Office of Science of the US Department of Energy under Contract No. DE-AC02-05CH11231. The whole dataset of ChIP-exo and RNA-seq has been deposited to GEO with the accession number of GSE54905.

This chapter was copied, in whole or in part, from the following manuscript: Kim, D., Ebrahim, A., Seo, S., Palsson, B.O. (In preparation). Model-driven elucidation of transcriptional regulatory network in bacteria

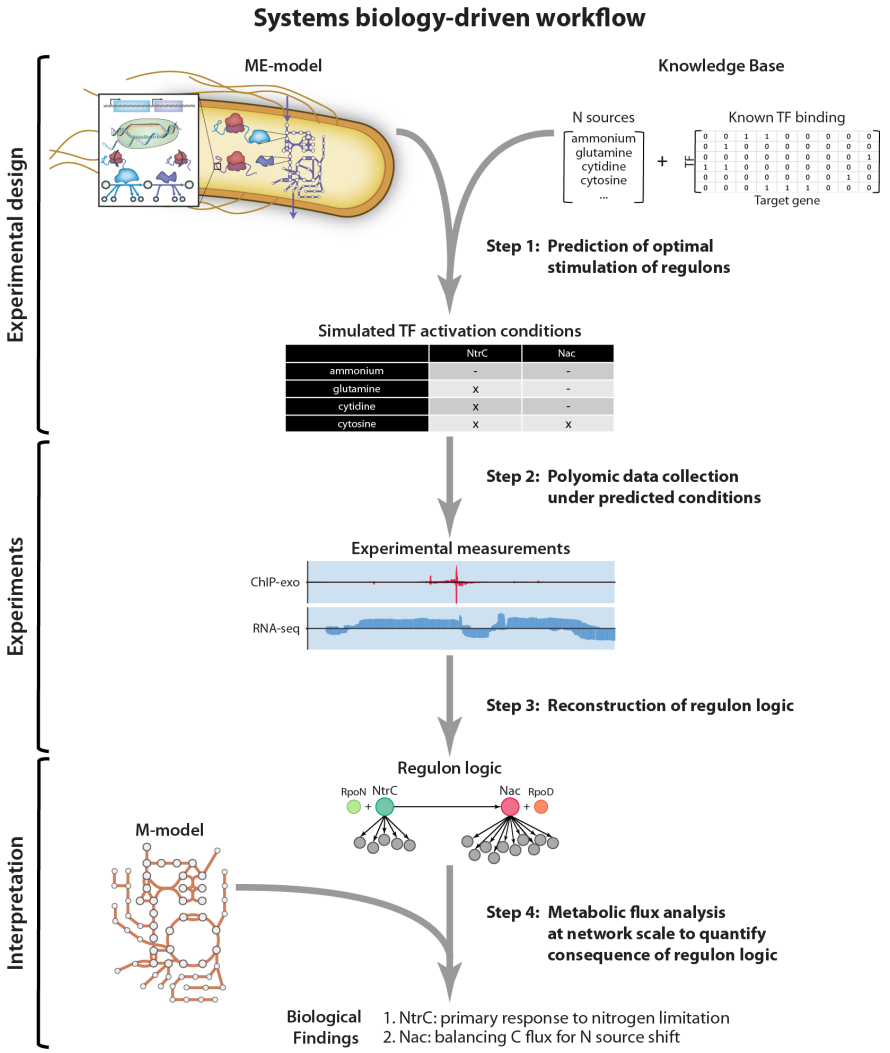


Figure 3.1: **Workflow of model-driven experimental design, regulon definition from experimental measurement, and model-based interpretation of regulon role.** Model-driven experimental design was achieved using a genome-scale model (GEM) of the metabolic network in *E. coli* K-12 MG1655, with a list of viable nitrogen sources, and known TF binding sites. Alternative nitrogen sources, glutamine, cytidine, and cytosine, were predicted to best activate NtrC and/or Nac. ChIP-exo and RNA-seq experiments were performed to obtain *in vivo* measurements of TF binding sites and gene expression. From experimental data, the definition of NtrC and Nac regulons were expanded. The expanded NtrC and Nac regulons were combined with metabolic flux map calculations using the *E. coli* GEM to elucidate the contrasting roles of these two regulons.

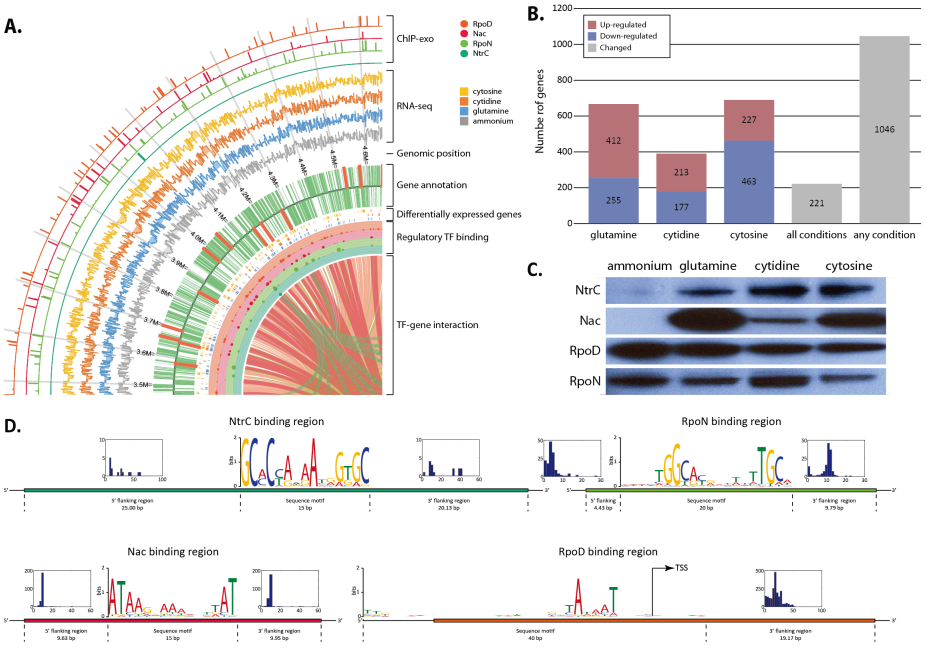


Figure 3.2: **Experimental measurement of gene expression changes and TF binding sites using ChIP-exo.** (A) Genome-wide experimental measurement of gene expression changes and TF binding sites for NtrC, Nac, RpoN, and RpoD on ammonia, glutamine, cytidine, and cytosine was achieved, and differential gene expression, TF regulons, and TF-gene interaction was calculated and interpreted from experimental measurement. (B) Alternative nitrogen sources rendered genome-wide expression changes, and expression of 1046 genes (22.8%) changed on at least one condition, leaving the majority of 3548 genes (77.2%) unchanged. The number of differentially expressed genes for glutamine, cytidine, and cytosine relative to ammonia was 667, 390, and 690, respectively. (C) Protein expression of two major nitrogen-responsive TFs, NtrC and Nac, and two σ-factors, RpoD and RpoN, was measured by western blotting. Expression of NtrC and Nac increased on alternative nitrogen sources. However, Nac expression on cytidine increased significantly less than on glutamine or cytosine. (D) Motif analysis on ChIP-exo binding sites for NtrC, RpoN, Nac, and RpoD resulted in previously known sequence motifs. Interestingly, sequence motifs were found near middle of binding sites.

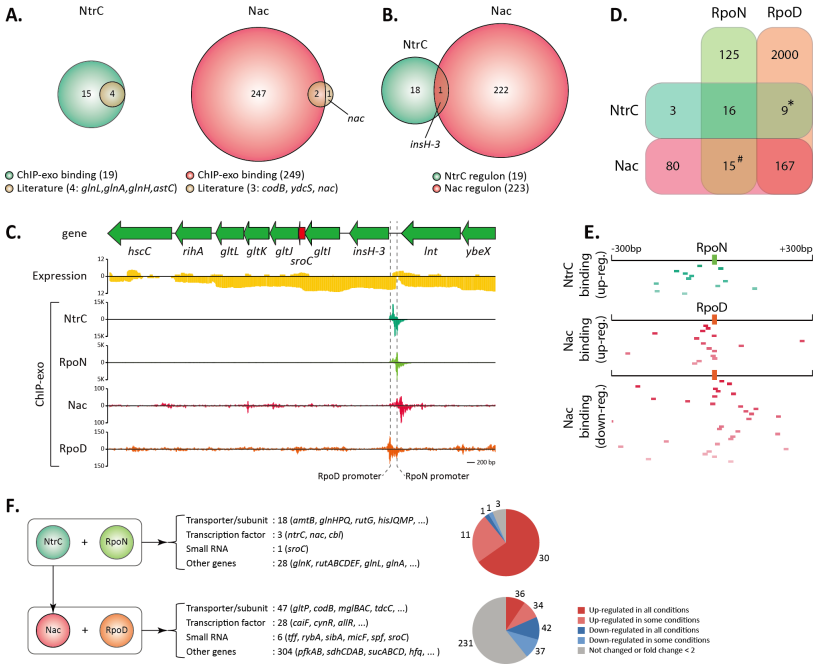


Figure 3.3: **Reconstruction of NtrC and Nac regulons.** (A) Binding sites for NtrC and Nac from ChIP-exo data were compared to previously reported binding sites. 19 NtrC binding sites included 4 known binding sites, and 249 Nac binding sites included 2 of 3 known ones, leaving only the *nac* binding site undetected. (B) TF binding sites were combined with known TU information, resulting in the definition of 19 and 223 TUs. NtrC and Nac regulons share only 1 TU, which is *insH-3*. (C) Binding sites of NtrC associated with RpoN, and Nac with RpoD were identified upstream of *insH-3*. Analysis of paired-end reads from RNA-seq data suggests contiguous transcript starting from *insH-3* ranging to *gltL*. (D) Association of TF binding sites with α-factor binding sites. 16 NtrC binding events out of 19 were accompanied with RpoN binding. 167 Nac binding events associated with RpoD binding. 3 and 80 bindings of NtrC and Nac were not associated with neither of σ-factors. Similarly, 125 and 2000 binding sites of RpoN and RpoD were neither associated with NtrC nor Nac. (*9 NtrC bindings of the 16 were also identified near RpoD binding sites. $^\#$15 Nac bindings of 167 were also detected near RpoN binding sites. These 24 cases are complicated promoters with both RpoD and RpoN binding sites.) (E) NtrC binds more upstream of target genes when it up-regulates their expression. When up-regulating expression of target genes, Nac tends to bind upstream from them, while it tends to bind downstream of target genes when it down-regulates their expression. (F) NtrC associates with RpoN-dependent promoters, and regulates 18 transporter genes, 3 TF genes, 1 sRNA, and 28 other enzymes. NtrC mostly up-regulates target genes. In contrast, Nac associates with RpoD-dependent promoters, and regulates 47 transporter genes, 28 TF genes, 6 sRNAs, and 304 other genes. Nac works as a dual-regulator.

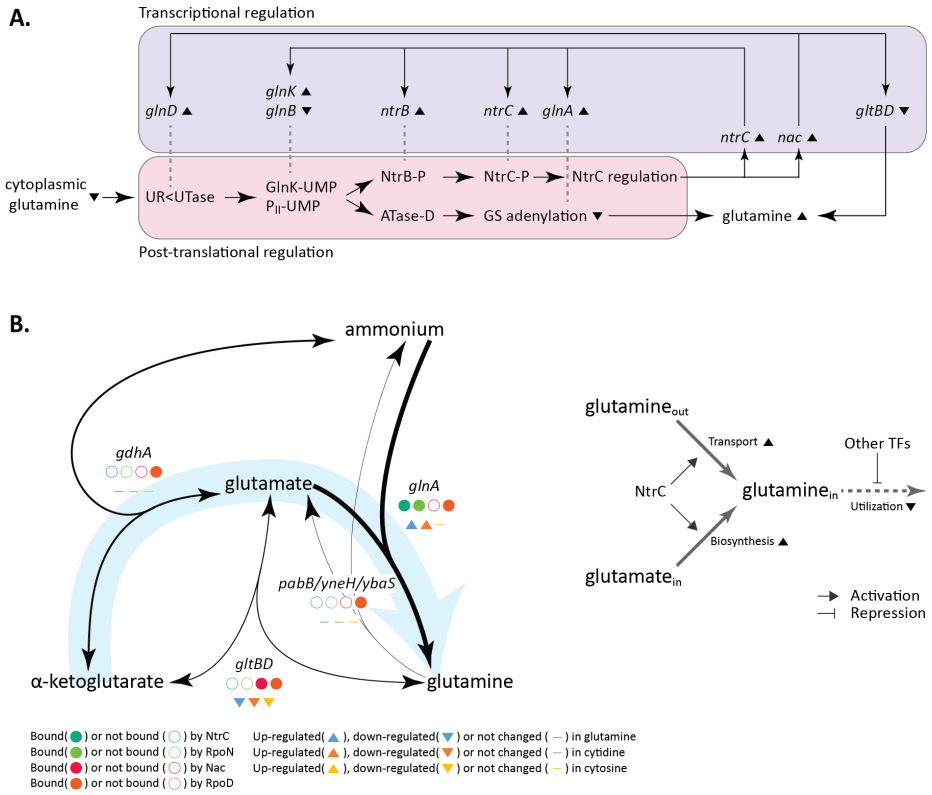


Figure 3.4: **Regulation by NtrC in Ntr regulatory cascade and glutamine-related enzymes.** (A) Ntr regulatory cascade in post-translational regulation is expanded with transcriptional regulation by NtrC and Nac. A low level of cytoplasmic glutamine results in activation of existing NtrC, and in turn NtrC activates itself and most of the regulatory components in the Ntr regulatory cascade, which increases the intracellular amount of glutamine. (B) When exposed to non-favorable nitrogen sources, *E. coli* up-regulates genes in producing glutamine, and down-regulates or does not change expression of genes in consuming glutamine. NtrC plays an important role in up-regulation. In *E. coli*, increasing glutamine level necessitates the expense of more glutamate and α-ketoglutarate. (Width of arrows between molecules correlates expression level of genes.)

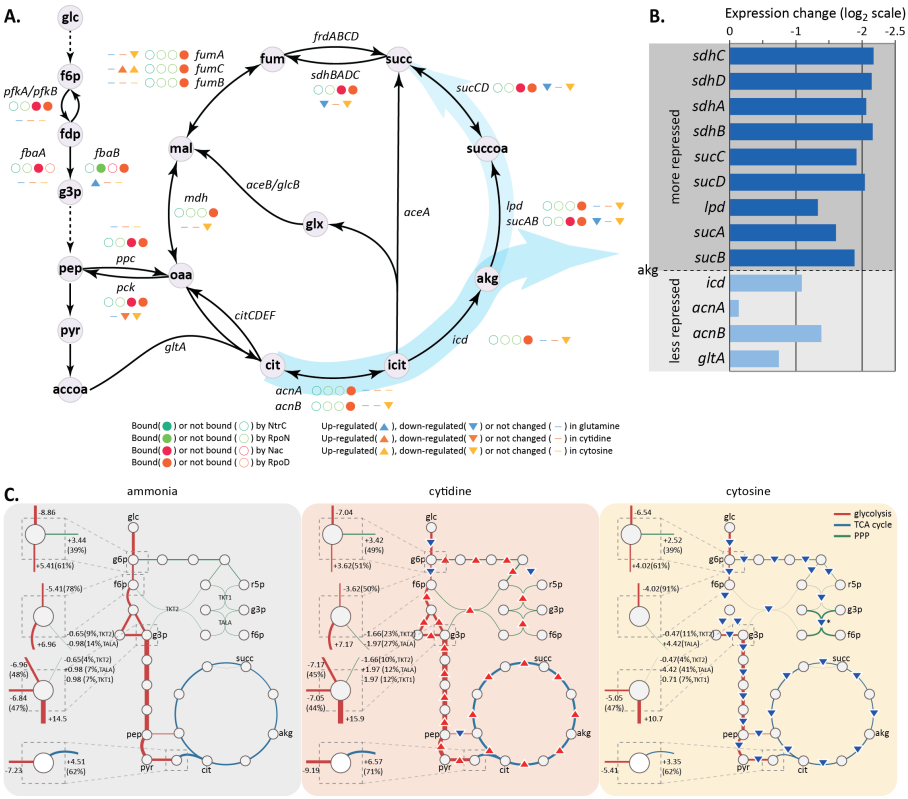


Figure 3.5: **Network-level regulation by Nac on glycolysis and TCA cycle in carbon metabolism.** (A) Nac regulates the expression of enzymes in glycolysis and the TCA cycle, including *pfkA*, *pfkB*, *fbaA*, *ppc*, *pck*, *sdhBADC*, *sucCD*, and *sucAB*. (B) Expression of genes downstream from α-ketoglutarate in the TCA cycle was more repressed than genes upstream. (C) FBA analysis was performed on ammonia, cytidine, and cytosine with experimentally measured glucose uptake rate. (There are genes and reactions with minor flux that are omitted for clarity of figure. *The absolute flux through the reaction was higher than the flux on cytidine; however the direction of the reaction was predicted to be reversed. Positive flux values mean outflux, and negative flux values mean influx.) Abbreviation: glc: glucose, g6p: glucose 6-phosphate, f6p: fructose 6-phosphate, g3p: glyceraldehyde 3-phosphate, pep: phosphoenolpyruvate, pyr: pyruvate, cit: citrate, akg: α-ketoglutarate, succ: succinate, r5p: ribose 5-phosphate, TKT1: transketolase, TKT2: transketolase, TALA: transaldolase.

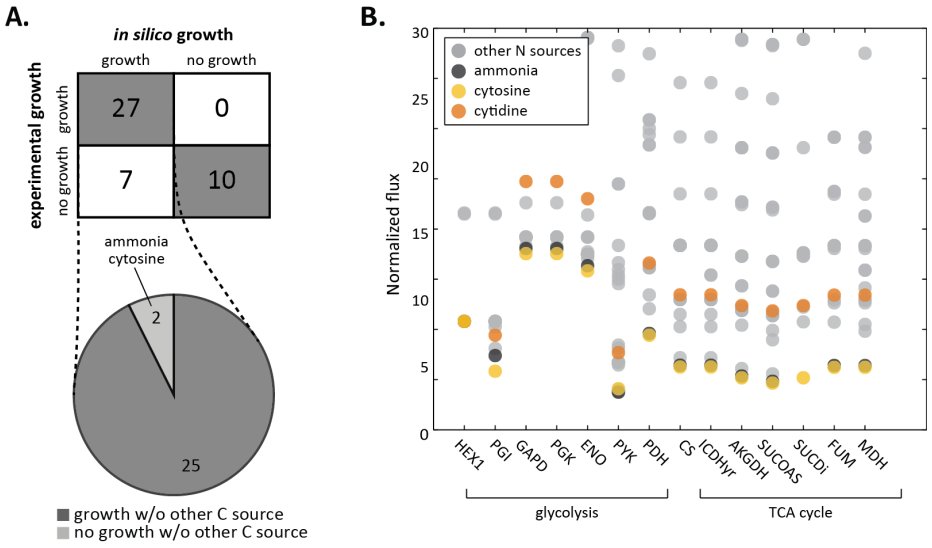


Figure 3.6: **Coupling of nitrogen and carbon metabolism and conservation of enzymes in nitrogen-limiting response.** (A) Comparing *in silico* growth and experimental growth for 44 nitrogen sources that are in the *E. coli* metabolic model and have experimental data available. 27 nitrogen sources with glucose for a carbon source support growth *in silico* and in experiments. Of 27 nitrogen sources, 25 are predicted to support growth without any other carbon source, leaving ammonia and cytosine not supporting growth. (B) Flux through glycolysis and the TCA cycle was calculated on 27 nitrogen sources with glucose for a carbon source. The flux on cytosine was slightly less than the flux on ammonia, and other nitrogen sources showed more flux on glycolysis and the TCA cycle, which explains the fluctuation of carbon metabolism by nitrogen sources.

Chapter 4

Multi-omic data integration enables discovery of hidden biological regularities

4.1 Abstract

The rapid growth in size and complexity of biological data sets has led to a grand challenge referred to as *Big Data to Knowledge*. Here we address a critical need for the development of advanced data integration methods to enable multi-level analysis of genomic, transcriptomic, ribosomal profiling, proteomic, and fluxomic data across multiple experimental conditions. First, we show that pairwise integration of primary omics data reveals biological regularities that tie certain cellular processes together in *Escherichia coli*: the number of protein molecules made per mRNA transcript and the number of ribosomes required per translated protein molecule. Second, we show that genome-scale models, which are based on genomic and bibliomic data, enable the

quantitative synchronization of disparate omics data types. Integrating omics data with models enabled the discovery of two novel regularities: condition invariant *in vivo* turnover rates of enzymes and the correlation of protein structural motifs and translational pausing. How these regularities relate to one another *mechanistically* is formally represented in a computable knowledge base, which allows for the coherent interpretation and prediction of fitness and selection underlying cellular physiology.

4.2 Results and Discusssion

Progress of the biological sciences in the era of big data will depend on how we address the following question: "How do we connect multiple disparate data types to obtain a meaningful understanding of the biological functions of an organism?" Owing to large-scale improvements in omics technologies, we can now quantitatively track changes in biological processes in unprecedented detail [BPS13, JP06]. While such measurements span a diverse range of cellular activities, developing an understanding of how these data types quantitatively relate to one another and to the phenotypic characteristics of the organism remains elusive. This issue is central to the so-called *Big Data to Knowledge* (BD2K) grand challenge, which aims to integrate multiple disparate data types into a biologically meaningful, multi-level structure [AM03, dGOC$^+$08].

Interpretation of disparate data requires understanding how the primary measurements of different omics data are *quantitatively* coupled to one another [CEL$^+$14]. We approach this task by identifying regularities (relationships between biological data types that remain relatively constant across conditions) between pairwise omics data types. While some regularities can readily be discovered through direct pairwise omics data comparisons, we find that other regularities emerge only through more intricate

analysis leveraged by mechanistically-based network reconstructions [TP10]. Such reconstructions can be used as a context for poly-omic data integration and analysis [TP10, HLP13], and, when combined with constraint-based modeling approaches [OTP10, OMP15], provide important links between omics data and phenotypic characteristics of the organism.

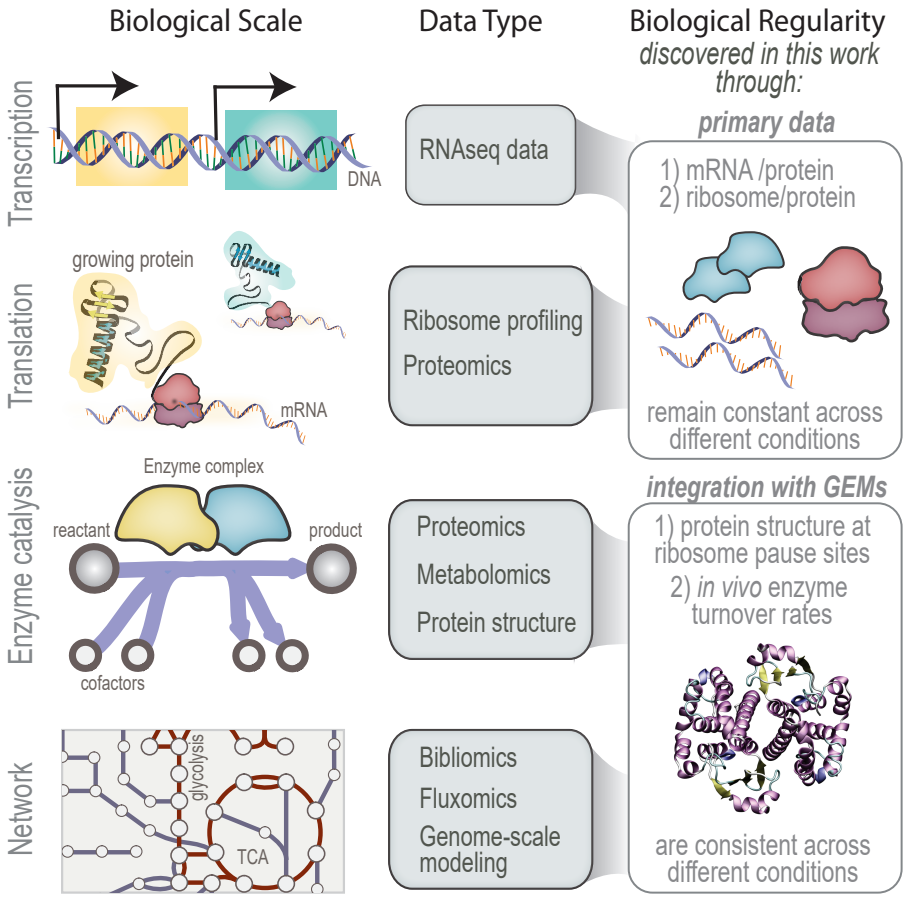


Figure 4.1: A multi-scale, multi-omics framework detects significant biological regularities in E. coli. Tracing the central dogma of biology (left column), we can link specific data types (middle column) to explain each of these biological processes. In this work, novel biological regularities that relate these processes are discovered through: (i) primary omics data (top box, right column) and (ii) integration with genome-scale models of metabolism (GEMs; bottom box, right column).

As we will show, this approach leads to a comprehensive synchronization of poly-

omic data with computed growth states. The approach directly addresses the BD2K grand challenge and is made conceptually accessible by tracing the 'information flow' through the familiar 'central dogma' to establish relationships between measurements and cell physiology (Figure 4.1).

First, we examine the information flow from transcription to translation to protein production by identifying correlations across primary omics data types, such as RNAseq [SBL$^+$11], ribosome profiling [LBGW14, IGNW09, LST$^+$15] and proteomics [SKV$^+$16], collected for batch growth on glucose, fumarate, pyruvate, and acetate (Figure 4.1, "primary data box"). We found relatively poor correlations of mRNA to protein across conditions ($r^2 < 0.4$), consistent with previous studies [LVK$^+$10, GCWG03] Stronger correlations ($r^2 > 0.8$) emerge when analyzing the ratio of protein per mRNA (ρ_{PM}) on a per-gene basis (the difference between peptide abundance and relative mRNA read counts per gene for multiple growth conditions; Figure D.1(a)). Computing the median coefficient of variation shows that changes in ρ_{PM} across conditions are relatively invariant. In addition, we find the number of ribosomes required (ribosome occupancy of mRNA) per protein translated is also relatively invariant across all four conditions ($r^2 > 0.7$; Figure D.1(b)).

Second, we examined pairwise relationships between other omics data types, such as ribosome profiling, proteomics and fluxomics, by integrating these data types into next generation genome-scale models (Figure 4.1, "integration with GEMs box"). Genome-scale models of metabolism (GEMs) are based on the annotated sequence and analysis of the bibliome for functionally annotated gene products [TP10]. The most recent generations of genome-scale models incorporate protein structural information [CAK$^+$13] and allow for the computation of the synthesis of the entire proteome of a

cell in addition to the balanced use of its metabolic network [OLC+13]. These models can integrate multiple layers of biological organization to balance the use of all cellular components to achieve a cellular state. It can thus extend our understanding of how information flows from translation to protein folding and catalysis, and its role in producing whole cell functions.

We examined how information flows during protein translation, which includes protein folding. Recent studies indicate a possible link between translation speed and proper folding [IGNW09, LOW12]. Analysis of translational pausing has typically been approached from a sequence-based viewpoint (*19*). Here, we approach this analysis from a different perspective, by correlating the occurrence of translational pausing on a transcript to the location of nearby protein secondary (2^o) structure motifs (Figure 4.2). The establishment of this correlation is based on; 1) ribosome profiling [LBGW14, IGNW09, LST+15], which provides ample information on the queuing of ribosomes along mRNA transcripts, and 2) a recent network reconstruction that contains comprehensive protein structural information linked to the translated protein at the proteome-scale [CAK+13].

Several striking regularities in translational pausing and protein structure are consistently observed across multiple growth conditions. We find that pause sites are enriched ($p_{value} < 0.01$ using a hypergeometric test) downstream of specific secondary structure motifs, such as α-helices and β-sheets (Figure 4.2(a), Figure D.2) but are not significantly enriched at the termini of domains (See Supplementary Information). On average, pausing becomes most substantial downstream of α-helices and β-sheets, which, in the majority of cases, fall either on disordered regions of the protein or on helical residues. Such instances account for more than 35–40% of pause sites in

different conditions (see Figure D.2). In addition, we find that Shine-Dalgarno (SD) like sequences equally account for 40% of pause sites (see Supplementary Methods), consistent with previous studies [LOW12]. Pause sites are particularly enriched ($p_{value} < 0.01$ using a hypergeometric test) downstream SD-like sequences.

Do SD-like sequences drive co-translational pausing to ensure proper protein folding? Of the pausing instances linked to SD-like sequences, we find that, on average, nearly half of these pausing regions also fall in the nearby vicinity (five to ten codons) of helices or sheets (Figure 4.2(b)). The link between pausing, SD-like sequence and protein secondary structure becomes clear when comparing the average occurence of SD-like sequence genome-wide (9%) with their occurrence directly downstream of α-helices (35%) and β-sheets (18%). Together, these sequence and structure motifs account for the majority of pause sites (60%) or nearly half of the total ribosome occupancy (Figure D.4). These findings provide a mechanistic understanding of translational pausing and support the potential role of SD-like codons to drive cotranslational pausing to ensure proper protein folding (Figure 4.2(c)).

How does information flow between an individual enzyme's catalytic activity and the activity of an entire network? To evaluate the effective turnover rate of enzymes, reaction flux per enzyme can be directly computed using experimental values for both flux (the rate of reactions) and enzyme abundance [AVP$^+$12] on a small scale (mainly for central carbon metabolism). To assess enzyme turnover on a genome-scale, we computed the ratio of an enzyme's abundance (measured from proteomics data) and its corresponding flux derived from network-based analyses (Figure 4.3). These ratios quantitatively couple experimentally-derived flux estimates and protein abundances to make a quantitative connection between data types.

Estimates of enzyme turnover rates (k_{eff}), which represent coupling coefficients between the fluxome and the proteome, were analyzed across four nutrient conditions to understand the effect that carbon uptake has on metabolic enzyme turnover rates. We find that these parameters show considerable regularity in relating flux to protein abundance, which suggests that *in vivo* turnover rate for most enzymes does not strongly depend on growth in diverse batch culture settings. For high-flux metabolic reactions, the estimated turnover rates were consistent across all four conditions (a total of 284 k_{eff}s; Figure 4.3b), with high correlation between any two conditions (Figure 4.3c and Figure D.5). The computed k_{eff} were averaged across experimental conditions to give the largest set of flux-per-enzyme parameters estimated computationally to date under *in vivo* conditions. It is important to note that these estimated k_{eff} do not have a direct relationship with fundamental enzyme kinetic parameters obtained *in vitro* but can be viewed as an *in vivo* data-driven estimate of the enzyme turnover rate.

While these correlations provide information about relationships between biological components and, in some cases, take on predictive value (Figure 4.4a), understanding their collective influence on cell physiology is harder to decipher. This issue can be addressed using a genome-scale model that assesses cost-benefit tradeoffs from a cell-centric perspective [OMP15, BMKP14]. Genome-scale models compute the value of cellular components relative to the function of all other cellular components. To this end, k_{eff} values provide the minimum 'capital expenditure' for protein synthesis required to achieve a unit of flux through a given reaction. Thus as a group, the k_{eff}s provide coupling between proteome allocation and achievement of a physiological state.

The knowledge of the biological regularities identified in this work enables the parameterization of coupling constraints used in a genome-scale model. A parameterized model allows for prediction of responses to environmental perturbations. We tested the predictive capacity of a model containing parameter values derived from multiple conditions described above (see Supplementary Information for parameterization method) to compute optimal cellular composition under new environmental conditions. We perturbed a reference growth state through the addition of nutrients to the medium: batch growth on glucose was supplemented with adenine, glycine, tryptophan or threonine. We collected omics data sets under these four perturbed conditions to compare gene expression changes to the computated responses.

Using the parameterized model, we predicted differential gene expression (Figure 4.4(b)). We find high predictive accuracies of significant changes in gene expression (p-values ranging 0.04 to $4e^{-6}$ using a hypergeometric test). Using the parameterized model, we are able to predict the regulation of genes that accompany changes in supplementation to a new growth environment. Such environmental changes oftentimes causes non-intuitive shifts in what precursors the cell uses to synthesize amino acid molecules (Figure 4.4(c); Supplementary Discussion).

Taken together, we demonstrate an ability to systematically integrate multi-omic data to enable discovery of multiple hidden biological regularities. These regularities take on biological meaning when put into the context of a network reconstruction that is comprised of fundamentally structured relationships between cellular components. We have shown that this contextualization leads to: (i) insights into underlying biological mechanisms during protein translation and (ii) predictive computations based on cellular-econometric cost-benefit ratios associated with the function of the

cell as a whole. Thus both multi-omic data analysis and genome-scale models will play an important role in establishing big data analysis frameworks to explain and predict cellular physiology.

4.3 Acknowledgements

This work was funded from a generous gift from the Novo Nordisk Foundation to the Center for Biosustainability. It was also supported by DE-FOA-000014 from the U.S. Department of Energy, NIH R01 GM057089 from the National Institutes of Health, and by the Swiss National Science Foundation (grant p2elp2_148961 to E.B). This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the US Department of Energy under Contract No. DE-AC02-05CH11231. The authors gratefully acknowledge Dr. Mahmoud Al-Bassam, and Dr. Jinwoo Kim for scientific discussions on ribosome profiling.

This chapter was copied, in whole or in part, from the following manuscript Ebrahim, A., Brunk, E., Tan, J., O'Brien, J., Kim, D., Szubin, R., Lerman, J.A., Lechner, Al, Sastry, A., Bordbar, A., Feist, A.M., Palsson, B.O. (under review) Multi-omic data integration enables discovery of hidden biological regularities. Nature Communications

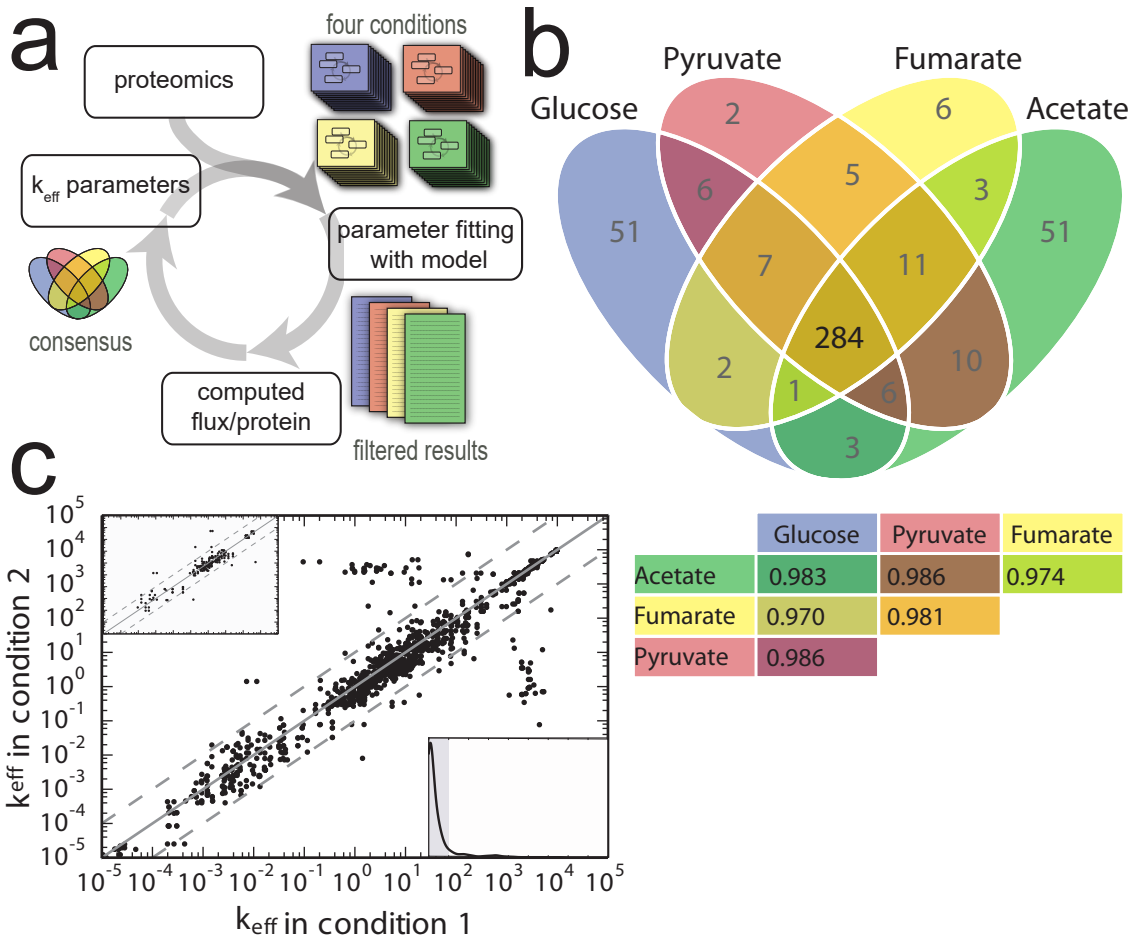


Figure 4.2: Effective enzyme turnover rates (k_{eff}) as regularities emerging from coupling quantitative in vivo proteomic data with genome scale modeling. (a) Iterative workflow for generating k_{eff} values from different nutrient conditions. This panel is a schematic of the overall workflow. A detailed version is found in the Supplement. (b) Venn diagram of k_{eff} shows all four conditions share 90% of the same estimates (Pearson correlations below). (c) Pairwise comparisons across four conditions for keff parameters demonstrate 94% are within one order of magnitude. The upper inset show the parameter estimation for the 10% most variable components of the proteome between the four conditions examined. The lower inset show a histogram of the distances of every point from the diagonal line. The gray box contains the 94% of the values that deviate from one another within an order of magnitude. A more detailed version is found in Figure D.5.

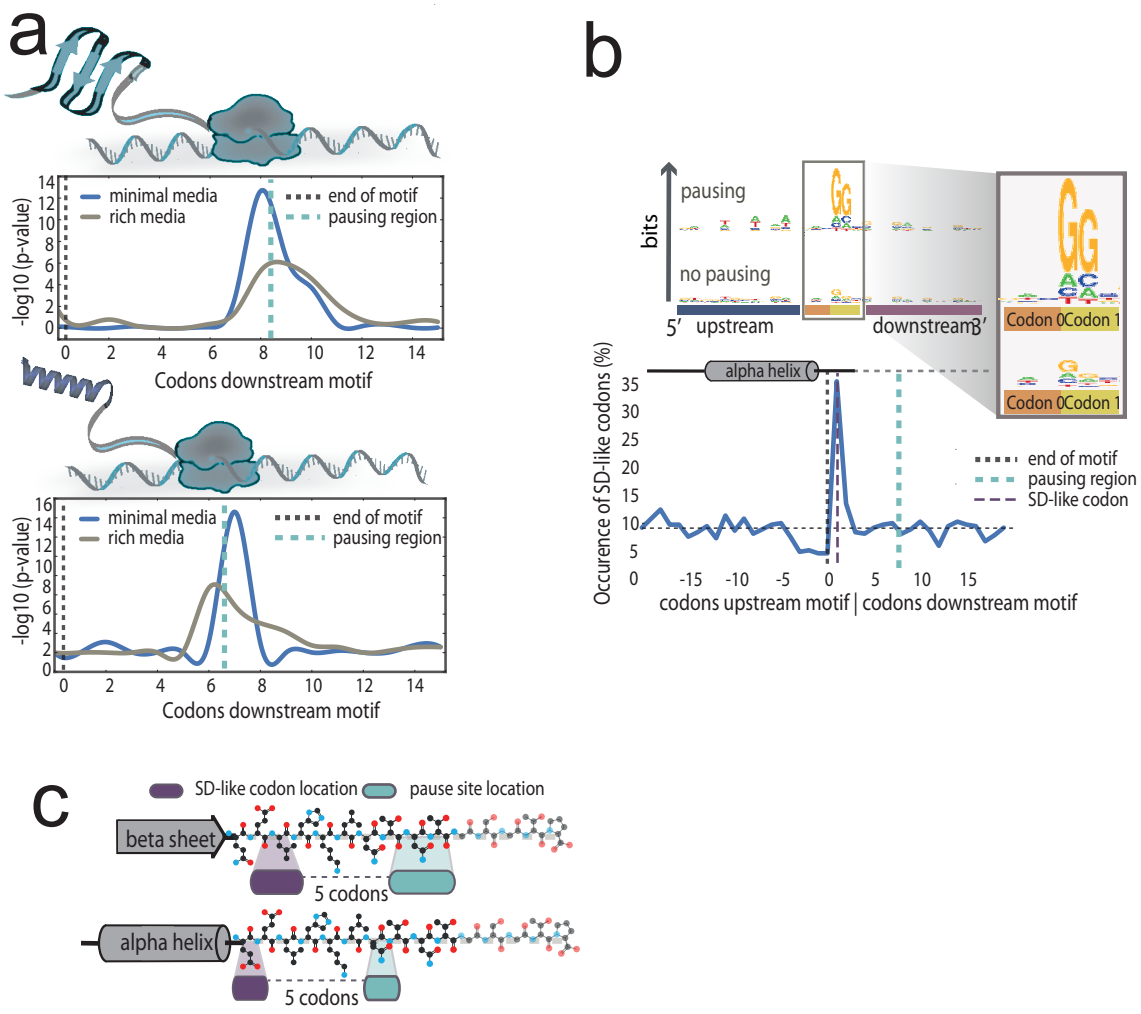


Figure 4.3: Regularities in translational pausing and structural motifs. (a) Analysis of ribosome profiling and translational pausing in conjunction with protein structure properties. Pausing is enriched at positions downstream of protein secondary structures (top: beta sheets, bottom: alpha helices). These correlations are consistent across conditions. (b) Protein structure motifs that exhibit pausing have increased propensity for SD-like sequences (c) A cartoon depiction of the relationship between structure, translation and sequence.

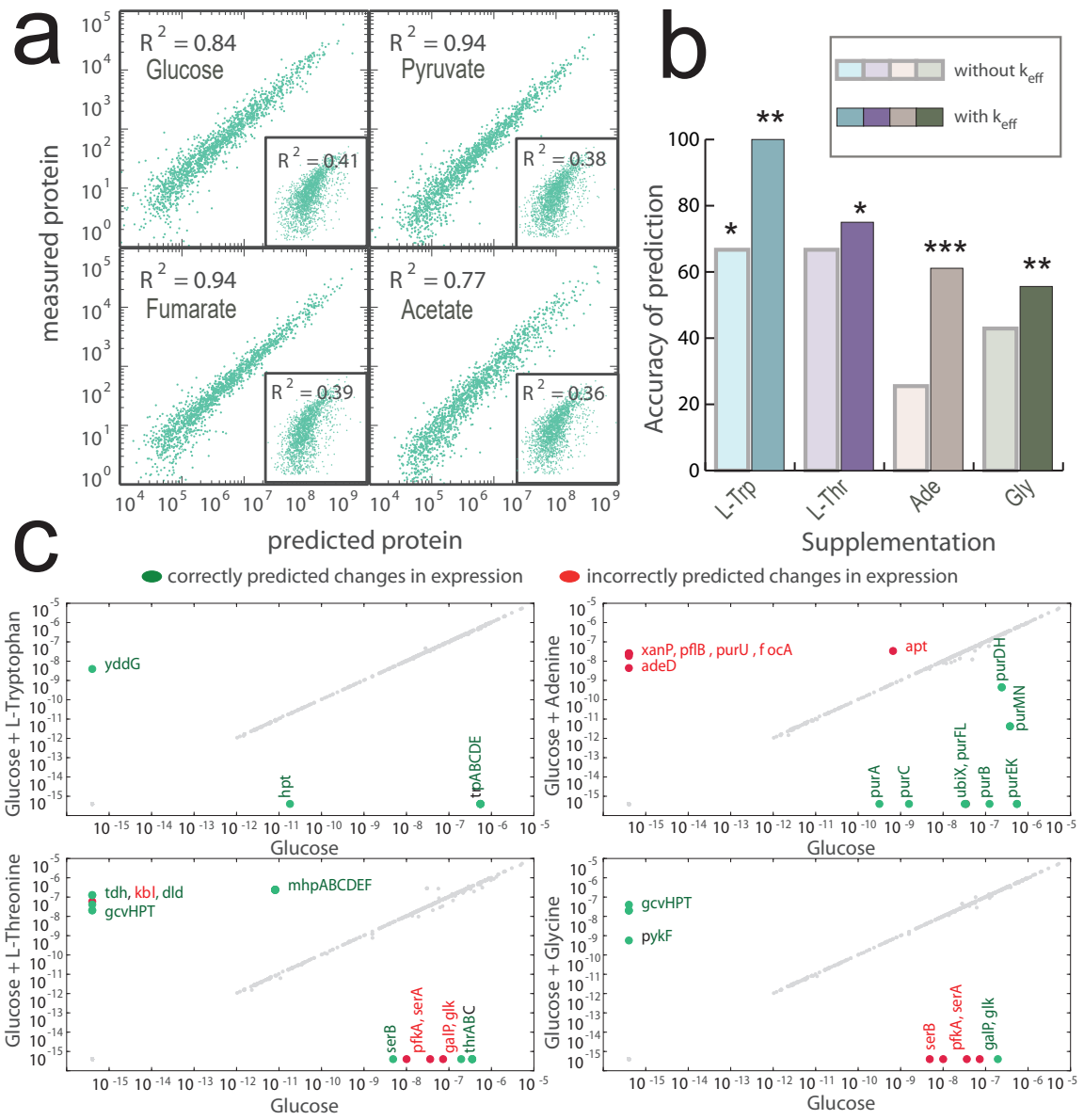


Figure 4.4: Predicting the results of perturbation from a parameterized homeostatic state. (a) Using a cross-validation approach, protein abundance is predicted by mRNA levels using information (ρ_{PM}) obtained from other conditions ($R^2 > 0.75$). Condition-specific mRNA and protein levels show little correlation (inset) (b) Accuracy of predicting differential expression is significantly enhanced using keff parameters. (c) Changes in gene expression and protein abundance predicted in different media supplementation. Accuracies range between 56–100% and specific genes are significantly enriched ($p < 0.05$ using a hypergeometric distribution).

Chapter 5

Construction of Genome-scale models of Metabolism and Gene Expression for the *Escherichia coli* family

5.1 Abstract

Although they have only existed for two organisms, genome-scale models of Metabolism and gene Expression (ME) have brought exciting new capabilities to the genome scale modeling field. We reformulated and expanded the previous ME models of *E. coli* K12 MG1655 into a new *i*LE1683-ME model, greatly improving the model and adding predictive capabilities. Furthermore, we use this *i*LE1683-ME model to create strain-specific ME models for 40 enterobacterial strains. These models predicted flux states and proteome allocation strategies which match experiments reported in

the literature, and segment the differing phenotypes of the different strain families. The availability of this set of multi-strain ME models opens up new possibilities in performing multi-strain comparative analyses.

5.2 Introduction

Genome-scale Metabolic models (M-models) have shown significant success predicting the metabolic capabilities of a cell by integrating all of the experimentally determined enzymatic reactions taking place in an organism of interest [BMKP14, OMP15, LNP12, MPF13]. These predictions are enabled based on the stoichiometric constraints of the organism and metabolic interactions with the environment. Metabolic models, though useful, have the limitation of computing metabolic flux states and directly related properties. Therefore,focus of research in this field has been to increase the scope and capabilities of M models [KSM$^+$12]. Recently, M models have been extended to include the synthesis of the gene expression machinery and its use to compute the entire metabolic proteome [OLC$^+$13, LOBL$^+$14, LHL$^+$12]. These models integrate Metabolism and Expression on the genome-scale, termed ME models, and they are capable of explicitly computing a large fraction of the proteome. ME models enables a wide range of new biological questions to be investigated including direct calculations of proteome allocation [LSO$^+$15], metabolic pathway usage and the effects of membrane and volume constraints [LOBL$^+$14]. Furthermore, their ability to compute the optimal proteome abundances for a given condition make them ideal for mechanistically integrating transcriptomics and proteomics data.

However, in spite of increased predictive capabilities ME models bring, up until now, ME models have been constructed for only two organisms (*Thermotoga maritima*

[LHL+12] and *Eshcerichia coli* K12 MG1655 [OLC+13, LOBL+14]). The lack of model proliferation can be attributed to the additional challenges which ME modeling brings. Firstly, ME models are much slower to solve than M model; it takes 5 orders of magnitude more CPU time to solve *i*OL1650-ME than it does the corresponding *i*JO1366 M model. As a result, while M models can be solved on personal computers, ME models require large clusters or supercomputers. While increased computing power is generally becoming more readily available, other challenges that come with ME models are not as easily addressed. M models can used generalized software tools [SQF+11, GDDFL13, ALS+13, ELPH13, KDE+15], but each organism's ME model has required its own dedicated codebase and database schema, which makes advances for one organism's model difficult to apply to another organism. Moreover, the large model sizes and complex structure have made analysis of the model difficult and time consuming. Therefore, each organism's ME model has required dedicated person-years of effort.

Generally, ME models have a larger parameter space than ME models. However, even M models have critical critical global parameters are to account for energy demands which are not directly modeled. These are usually divided into two separate parameters: GAM (growth-associated maintenance) and NGAM (non-growth associated maintenance). In M models, GAM forces additional ATP hydrolysis flux to account for energy which must be expended by the cell on processes necessary for growth, such as transcription, translation, and cell division. However, ME models directly account for so many of these processes, and even account for the expression cost of "non-ME functions" by accounting for unmodeled protein; therefore, the general GAM requirements in the model can be greatly reduced [OLC+13]. On the

other hand, NGAM accounts for energy demands which do not scale with growth rate. As these processes are generally also not directly accounted for in the ME model, the nature of this parameter changes less between M and ME models.

5.3 Results and Discussion

5.3.1 The "miniME" Reformulation of ME models

The starting point for the strain-specific model-building processes were the network reconstructions built in the previous ME models: the *i*OL1650-ME [OLC$^+$13] and *i*JL1678-ME [LOBL$^+$14]. Those reconstructions were updated to reflect biological discoveries since the original models were published, and some new content was added as well. However, the ME models built from these reconstructions were done in a manner which was very specific to *Eshcerichia coli* K-12 MG1655. Additionally, the formulation used by *i*OL1650-ME was unoptimized for newer generations of high-precision numeric simplex solvers, such as SoPlex. As a result, a new more generalized formulation was required which could be used as a basis for modeling other bacterial strains as well. The end result of this process is the latest version of the *Eshcerichia coli* K12 MG1655 ME model, *i*LE1683-ME.

We reformulated the *i*OL1650-ME model (Figure 5.1) to allow for rapid building and solving of models for other bacterial strains. A major goal of the reformulation was to greatly reduce the ME-model matrix size and complexity to make the new ME model more amenable to high-precision numeric solvers; therefore, the reformulation has therefore been dubbed "miniME." The biggest mathematical difference between the original ME formulation [OLC$^+$13] and miniME is the elimination of most inequality

constraints, which greatly reduces the polytope of feasible fluxes at suboptimal growth rates. At the optimum, the model will not waste resources, and as a result, all the computed values are pushed up against their inequality constraints anyways, rendering them as equalities. Therefore, reformulating the model with equalities alone will compute the same optimal flux state, but results in a much simpler problem from a numerical point of view.

As a consequence of using only equality constraints, in the miniME formulation, reactions which occur in a number of individual steps or sub-reactions (i.e., ribosome formation, translation, etc.) can be lumped into a single reaction. Lumping the formation of major macromolecules as described above makes the ME-model much more modular in nature, which greatly simplified the process of adding in new processes such as DNA replication. This process also removes the majority of "coupling constraints" and associated variables in the original formulation, which makes the models much smaller. The *i*LE1683-ME model has a matrix with 80% fewer columns than *i*OL1650-ME. This dramatically speeds up the solving procedure, and allows processes such as iterative refinement, which uses rational arithmetic and is unsuited for fast vector SIMD operations, to become feasible for fast and accurate solutions. Perhaps more importantly, this formulation and reduced size makes the models more understandable to its human users by making it amenable to visualization tools like escher [KDE+15].

Unlike the replacement of inequalities, some of the changes made in the miniME formulation purposefully changed the model in a nonequivalent way. One of the most significant differences was assigning a "dummy protein" with a representative amino acid composition as the enzymes for "orphan" reactions. These are non-spontaneous reactions which do not have a known enzymatic catalyst. The previous formulation

therefore resulted in a slight bias towards using these reactions, which did not have an associated protein expression cost, which is no longer present in the miniME formulation. Additionally, in miniME, protein carriers (i.e., acyl carrier protein) are assigned as catalysts to their transfer reactions, as that is what the carriers are effectively doing. Therefore, the miniME formulation will require translation of these carriers in order for them to participate in reactions.

5.3.2 Global Parameter Characterization

The miniME formulation allows for direct investigation to investigate the effects of many global model parameters. Many of these parameters are inferred from experimental data, which requires assumptions about consistent behavior across experimental conditions, and even in many cases with M models, across organisms [MNP14]. However, because ME models directly represent more of the constraints experienced by a cell, they can exhibit different and more nuanced responses to some of these global parameters.

In M models, increasing GAM and NGAM linearly decreases the maximum possible computed growth rate. This linear relationship is used to estimate these parameters from data. However, in proteome-limited ME models modeled on rich media, this relationship between these energy requirements and maximum possible growth rate is no longer strictly linear. For both GAM and NGAM, we observe that past a cutoff, the effect of increasing the energy requirement on growth decreases (Figure 5.2a). This demonstrates how the assumptions made for M models (which assume nutrient limitation), are not necessarily applicable to proteome-limited situations. Moreover, the NGAM parameter has a significant effect on flux distributions throughout

the cell, and is a significant determinant of the ratio of carbon flux through glycolysis and the pentose phosphate pathway.

Another critical global ME-model parameter is the unused protein fraction, which represents the fraction of the total modeled proteome by mass which is allocated to a dummy protein with a representative amino acid sequence. Because it effectively increases the cost of each protein linearly, it is unsurprising that increasing this parameter causes a nearly linear decrease in maximum modeled growth rate (Figure 5.2b). Even on rich nutrient excess media, ME models are infeasible beyond a 95% fraction unused protein. This parameter sweep gives a production envelope for the model of the cost of every extra unit of "non-ME" proteome.

5.3.3 Extending from *E. coli* K12 MG1655 to 40 related strains

A workflow was used to map genetic content and reconstructed metabolic models [MCA$^+$13] from 40 related *E. coli* and *Shigella* strains onto *i*LE1683-ME to generate 40 strain-specific ME models. These models contain a subset of the expression machinery in the original reconstruction [OLC$^+$13, LOBL$^+$14, TJFP09] as identified by gene homology, which is integrated with each organism's specific metabolic capabilities [MCA$^+$13]. This procedure relies on the assumption that the expression machinery is generally well-conserved between the other strain and *E. coli* K12 MG1655, which generally holds for these enterobacterial strains [VDTR97].

ME models represent the core set of gene functions which are required for growth of a strain [YTO$^+$15]. From this point of view, missing gene functions from this core set of required functions in these annotated genomes represent targets for

annotation improvement. The core set of required genes for the *i*LE1683-ME model was determined by identifying singly essential genes for simulated growth in rich media above 0.1 hr-1, but which are not essential in M models. Missing homologs to these essential genes represent gaps in the E reconstruction for each strain, analogous to "orphaned" metabolic functions which are known to occur in an organism but without an identified gene [BCP, OP12, GUN^{+}15, RPC^{+}06]. These missing genes were identified for each strain (Figure 5.3), and were substituted with the appropriate gene from *E. coli* K12 MG1655 for modeling purposes to require a cost for performing the respective functions.

When simulated under batch rich media, some of the strain types clustered by flux state (Figure 5.4). A principal component analysis reveals several major modes of differentiation. The primary mode accounts for 67% of the observed variation, and includes reactions representative of oxidative phosphorylation, while the secondary mode (which accounts for 14% of the variation) has reaction which correspond to fermentation pathways. The tertiary mode accounts for 11% of the observed variation, and represents a difference between ethanol and acetate fermentation. The *E. coli* pathogens (both intestinal and extraintestinal) all cluster together, near the more varied commensal strains. The *Shigella* strains, however, exhibit much more variation. Two strains in particular, *Shigella boydii* CDC 3083-94 (iSbBS512-ME) and *Shigella sonnei* Ss046 (iSSON-ME) cluster with the rest of the *E. coli* strains, separate from the other *Shigella* strains. This is consistent with classification of the strains by sequence homology, which have even suggested *Shigella* sonnei strains are a part of *Eshcerichia coli* [PLR00]. One thing all of these modeled *Shigella* strains have in common, however, is a statistically significant high acetate secretion flux through

the phosphotransacetylase (PTAr) and acetate kinase (ACKr) reactions (p ¡ 2.2E-4), which also matches in vivo experimental observations [KMC+14]. A major difference between the outlying *Shigella* strains and the rest of the strains is a lack of flux through the TCA cycle (specifically through citrate synthase (CS), α-ketoglutarate dehydrogenase (AKGDH), and isocitrate dehydrogenase (ICDHyr)), which is consistent with experimental ^{13}C fluxomics of pathogenic *Shigella flexneri* [KMC+14]. The ME models for *Shigella* predict flux states which match physiological observations and *Shigella*'s pathogenic lifestyle of intracellular infection.

5.3.4 Conclusions

The miniME formulation used in the new *i*LE1683-ME mode the 40 related strain-specific ME models greatly improves the versatility of ME models. The formulation and methods here make ME modeling much more approachable and significantly less time intensive. The process presented here is a first step, which will hopefully continue to a proliferation of ME models across the tree of life in the future to mirror the progress made with M models over the past two decades.

5.4 Materials and Methods

5.4.1 Optimization Procedure

The resulting stoichiometric matrix for each ME-model consists of both reaction coefficients for each metabolite as well as growth rate (μ) dependant coupling constraints for the macromolecules (Figure 5.1). This ME-matrix is nonlinear only in the variable μ, but remains quasi-convex [Lau], implying that for any feasible μ,

all smaller values must be similarly feasible. Therefore, these nonlinear problems can be solved for the maximal possible μ by a binary search solving successive linear programs at different values of μ to find the largest value of μ which gives a feasible flux state, as done in the *i*OL1650-ME [OLC+13]. While searching for the maximal μ, each individual linear program is maximizing for a representative dummy-protein. Because of the model reformulation, this allows for the same algorithm to be used for both batch and nutrient limited growth, which required different procedures in *i*OL1650-ME [OLC+13].

To perform the binary search, the following procedure was implemented in cobrapy [ELPH13]. First, each symbolic coefficient or reaction bound was compiled into a function by sympy [JvMG12]. Then, a linear program was created for the linear programming solver, with all of these symbolic functions evaluated at 0. While the model will always be feasible at 0, starting with a known feasible point results in a basis which can be used to speed up the next run. Afterwards, for each instance of the binary search in mu, values in the linear program were replaced by recomputed ones, and the problem was resolved using the last feasible basis.

While any linear programming solver supported by cobrapy could technically have been used, unlike M-models [EAB+15], ME models are very ill-scaled [OLC+13]. Therefore, the SoPlex solver (version 2.2) was integrated with cobrapy and used [Wun97]. This solver supports 80-bit ("long double") numerical precision with x87 instructions as well as iterative refinement in rational arithmetic to reduce numerical error. To prevent the solver from stalling when using the previous feasible basis (which corresponds to a slightly different problem), each solve with a previous basis was limited to 20,000 iterations, after which the basis was removed and a cold solve was

performed.

5.4.2 Multiple ME Model Construction

To construct these ME models, the M reconstructions [MCA+13] were obtained from BiGG [KLD+15]. Additionally, a homology table between each strain and *E. coli* K12 MG1655 was generated using the same method used in the M reconstruction process [MCA+13]. A "full" model of *i*LE1683-ME was constructed, which produced every single annotated ORF in *E. coli* K12 MG1655. The TU architecture was then removed, and each individual ORF was assigned its own transcription unit [CZQ+09]. All MG1655 genes which had no homologs in the particular strain were then removed, except for ones which were judged to be "E" essential. For the remaining homologous genes, the transcription and translation reactions were updated to use sequences from the new strain. Transcription and translation reactions were added for all remaining genes in the strain (which had no homologs in MG1655).

To add in the extra metabolic content for each strain specific model, the additional reactions between that strain's M model and the MG1655 model it was built from [OCN+11] were computed by a set difference. Exchanges. demands, and spontaneous reactions were added freely, and orphan reactions were also assigned a dummy protein. For all other reactions, the boolean gene reaction rule was converted to disjunctive normal form by sympy [JvMG12]. In this form, AND represents members of a complex, and OR represents separate complexes which can perform the same task, giving a list of pseudo-complexes. These were created, assuming a stoichiometry of each component of 1, and assigned to the newly added reactions. Manual curation was necessary to map between M-specific versions of some proteins used in the M-models.

For example acyl carrier protein, which is represented as the metabolite "acp_c" in the M reconstructions, but is a protein (b1094) which must be translated in ME models [EV68].

To cluster strains by expression type, flux states for each model were generated under batch growth for rich media, which means effectively infinite (1000 mmol/gDw/hr) bounds were set on all uptake rates. The high flux reactions for water exchange and transport were removed from the flux states, followed by principal component analysis performed using scikit-learn [PVG+11].

5.5 Acknowledgements

We would like to thank Zachary King, Aarash Bordbar, Justin Tan, Bin Du, and Joshua Lerman for informative discussions. This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the US Department of Energy under Contract No. DE-AC02-05CH11231.

This chapter was copied, in whole or in part, from the following manuscript: Ebrahim, A., Lloyd, C., Monk, J., Yang, L., O'Brien, E.J., Liu, J.K., Palsson, B.O. (In preparation). Construction of Genome-scale models of Metabolism and Gene Expression for the *Escherichia coli* family.

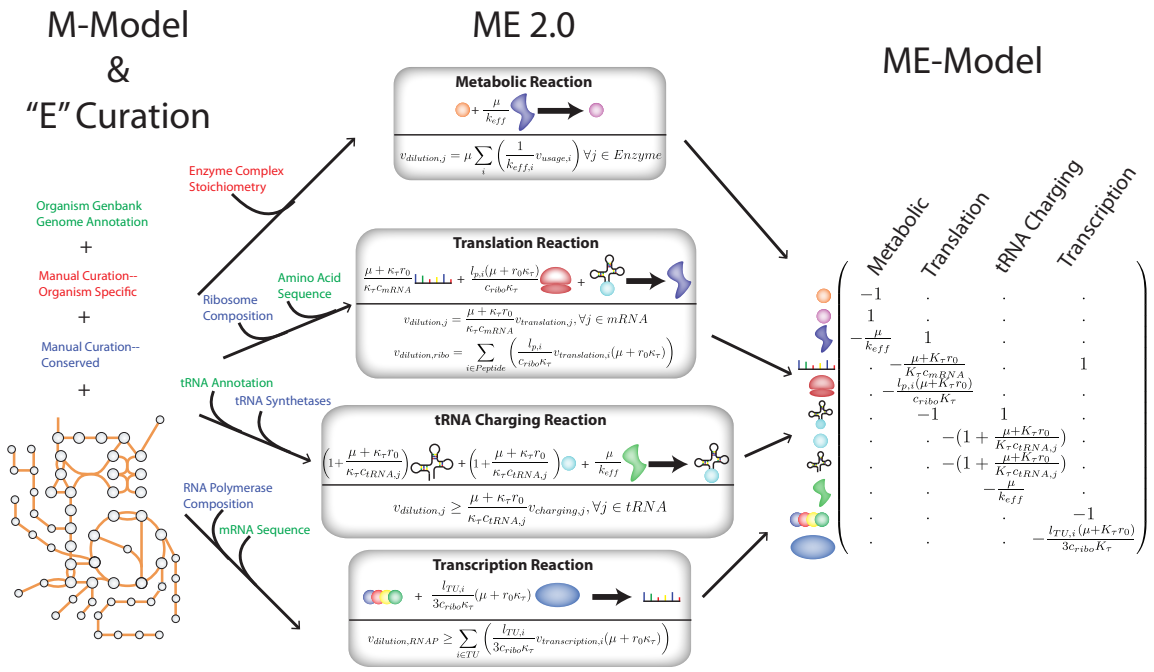


Figure 5.1: An overview of the "miniME" formulation for ME models. The original ME formulation created a large matrix by introducing coupling constraints. In miniME, instead of coupling constraints, dilution reactions to the daughter cell for the required macromolecules are embedded directly in the reaction which requires them. For example, the first column shows that for a metabolic reaction, a small amount $(\frac{\mu}{k_{\text{eff}}})$ of the catalyzing enzyme is also used in the process.

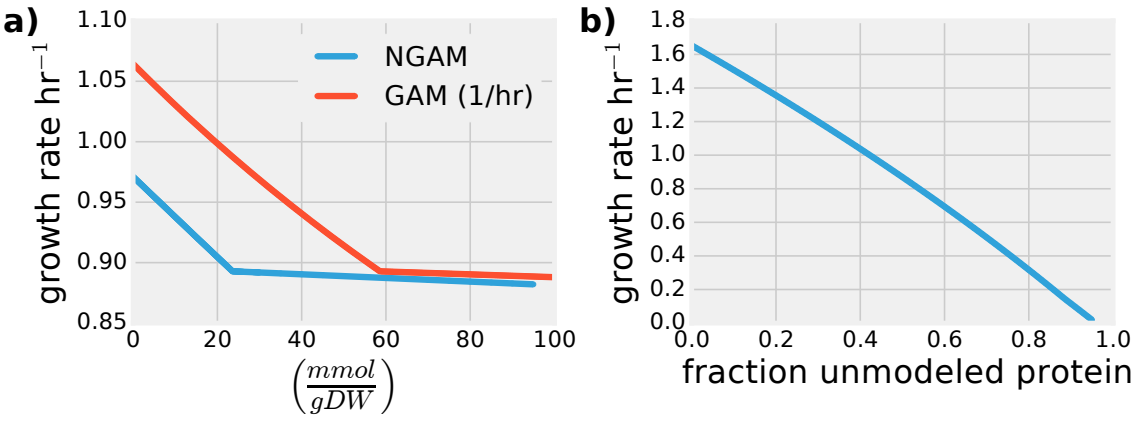


Figure 5.2: (a) Predicted *i*LE1683-ME growth rates with nutrient excess in rich media while varying the growth associated (GAM) and non-growth associated (NGAM) requirements. (b). Predicted iLE1683-ME growth rates with nutrient excess in rich media with varying unmodeled protein loads.

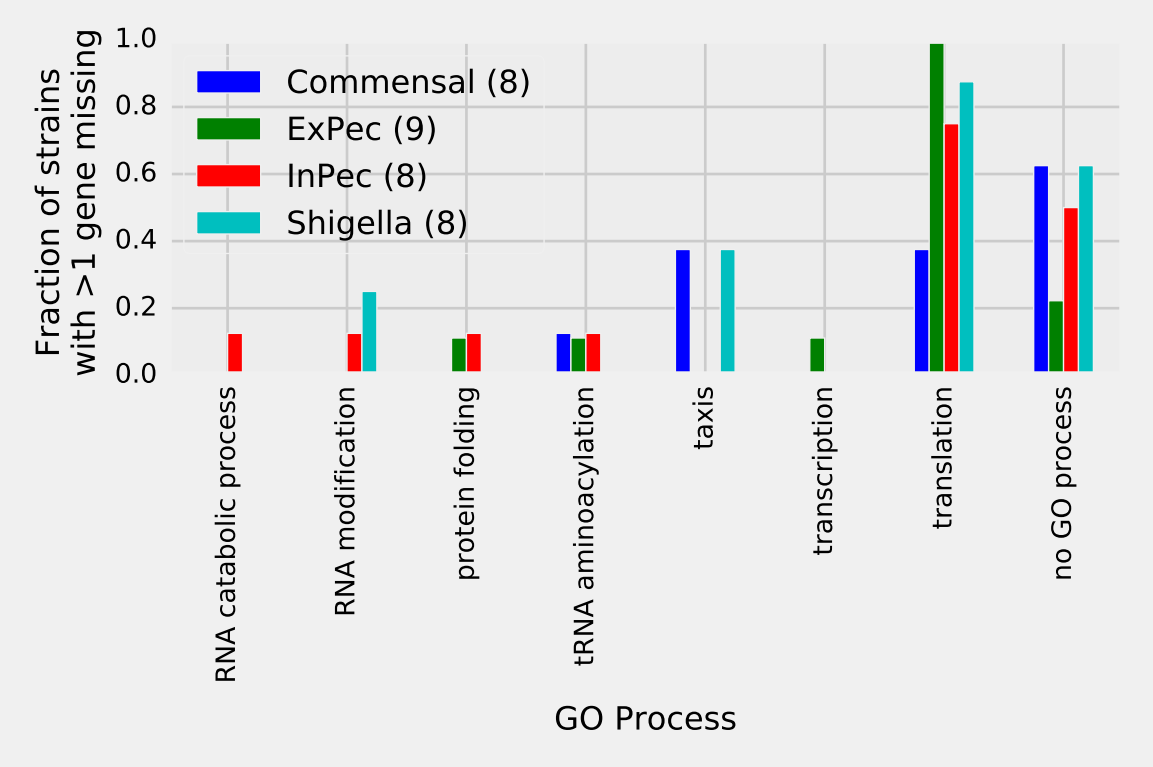


Figure 5.3: Missing annotated genes in strain families. The fraction of the strains in each category without gene homology (determined via bidirectional blast) for at least one gene involved in the listed GO process. The numbers in parentheses denotes the total number of E. coli strains that fall into that category.

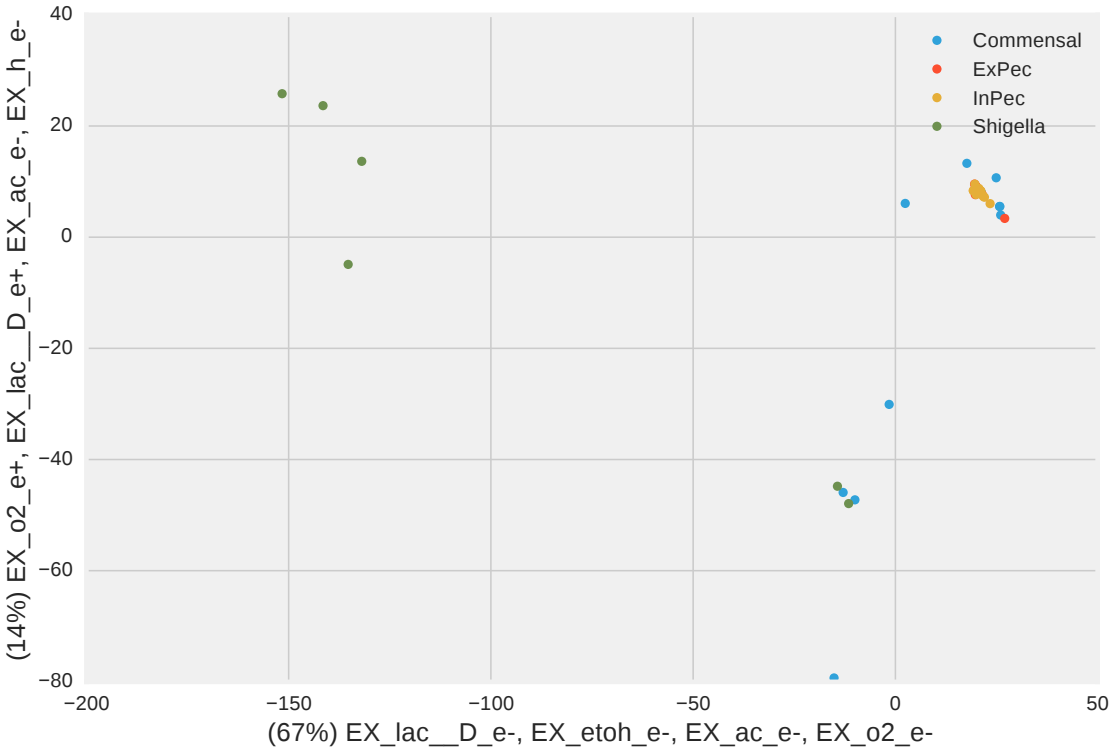


Figure 5.4: Principle component analysis of ME computed flux states for 40 different strains. The flux distribution for each model on batch growth in rich media was computed, and a principal component analysis was generated. The 5 reactions with most significant coefficients for each mode are listed along its axis, with a sign to indicate the direction. The first dimension (along the x-axis) accounts for 67% of the variation, while the second (along the y-axis) accounts for 15%.

Appendix A

Documentation for COBRApy

The source code for this documentation can be found at https://github.com/opencobra/cobrapy.

A.1 Getting Started

To begin with, cobrapy comes with bundled models for Salmonella and E. coli, as well as a "textbook" model of E. coli core metabolism. To load a test model, type

```
In [1]: from __future__ import print_function
        import cobra.test

        # "ecoli" and "salmonella" are also valid arguments
        model = cobra.test.create_test_model("textbook")
```

The reactions, metabolites, and genes attributes of the cobrapy model are a special type of list called a DictList, and each one is made up of Reaction, Metabolite and Gene objects respectively.

```
In [2]: print(len(model.reactions))
        print(len(model.metabolites))
        print(len(model.genes))
```

```
95
72
137
```

Just like a regular list, objects in the DictList can be retrived by index. For example, to get the 30th reaction in the model (at index 29 because of 0-indexing):

```
In [3]: model.reactions[29]

Out[3]: <Reaction EX_glu__L_e at 0x7f56b0ea3198>
```

Addictionally, items can be retrived by their id using the get_by_id() function. For example, to get the cytosolic atp metabolite object (the id is "atp_c"), we can do the following:

```
In [4]: model.metabolites.get_by_id("atp_c")

Out[4]: <Metabolite atp_c at 0x7f56b0ed7cc0>
```

As an added bonus, users with an interactive shell such as IPython will be able to tab-complete to list elements inside a list. While this is not recommended behavior for most code because of the possibility for characters like "-" inside ids, this is very useful while in an interactive prompt:

```
In [5]: model.reactions.EX_glc__D_e.lower_bound

Out[5]: -10.0
```

A.1.1 Reactions

We will consider the reaction glucose 6-phosphate isomerase, which interconverts glucose 6-phosphate and fructose 6-phosphate. The reaction id for this reaction in our test model is PGI.

```
In [6]: pgi = model.reactions.get_by_id("PGI")
        pgi
```

```
Out[6]: <Reaction PGI at 0x7f56b0e396d8>
```

We can view the full name and reaction catalyzed as strings

```
In [7]: print(pgi.name)
        print(pgi.reaction)

glucose-6-phosphate isomerase
g6p_c <=> f6p_c
```

We can also view reaction upper and lower bounds. Because the pgi.lower_bound < 0, and pgi.upper_bound > 0, pgi is reversible

```
In [8]: print(pgi.lower_bound, "< pgi <", pgi.upper_bound)
        print(pgi.reversibility)

-1000.0 < pgi < 1000.0
True
```

We can also ensure the reaction is mass balanced. This function will return elements which violate mass balance. If it comes back empty, then the reaction is mass balanced.

```
In [9]: pgi.check_mass_balance()

Out[9]: {}
```

In order to add a metabolite, we pass in a dict with the metabolite object and its coefficient

```
In [10]: pgi.add_metabolites({model.metabolites.get_by_id("h_c"): -1})
         pgi.reaction

Out[10]: 'g6p_c + h_c <=> f6p_c'
```

The reaction is no longer mass balanced

```
In [11]: pgi.check_mass_balance()

Out[11]: {'H': -1.0, 'charge': -1.0}
```

We can remove the metabolite, and the reaction will be balanced once again.

```
In [12]: pgi.pop(model.metabolites.get_by_id("h_c"))
         print(pgi.reaction)
         print(pgi.check_mass_balance())

g6p_c <=> f6p_c
{}
```

It is also possible to build the reaction from a string. However, care must be taken when doing this to ensure reaction id's match those in the model. The direction of the arrow is also used to update the upper and lower bounds.

```
In [13]: pgi.reaction = "g6p_c --> f6p_c + h_c + green_eggs + ham"

unknown metabolite 'green_eggs' created
unknown metabolite 'ham' created

In [14]: pgi.reaction

Out[14]: 'g6p_c --> f6p_c + green_eggs + h_c + ham'

In [15]: pgi.reaction = "g6p_c <=> f6p_c"
         pgi.reaction

Out[15]: 'g6p_c <=> f6p_c'
```

A.1.2 Metabolites

We will consider cytosolic atp as our metabolite, which has the id atp_c in our test model.

```
In [16]: atp = model.metabolites.get_by_id("atp_c")
         atp

Out[16]: <Metabolite atp_c at 0x7f56b0ed7cc0>
```

We can print out the metabolite name and compartment (cytosol in this case).

```
In [17]: print(atp.name)
         print(atp.compartment)
```

```
ATP
c
```

We can see that ATP is a charged molecule in our model.

```
In [18]: atp.charge
```

```
Out[18]: -4
```

We can see the chemical formula for the metabolite as well.

```
In [19]: print(atp.formula)
```

```
C10H12N5O13P3
```

The reactions attribute gives a frozenset of all reactions using the given metabolite. We can use this to count the number of reactions which use atp.

```
In [20]: len(atp.reactions)
```

```
Out[20]: 13
```

A metabolite like glucose 6-phosphate will participate in fewer reactions.

```
In [21]: model.metabolites.get_by_id("g6p_c").reactions
```

```
Out[21]: frozenset({<Reaction GLCpts at 0x7f56b0eabcc0>,
                    <Reaction Biomass_Ecoli_core at 0x7f56b0e9e358>,
                    <Reaction G6PDH2r at 0x7f56b0eab9e8>,
                    <Reaction PGI at 0x7f56b0e396d8>})
```

A.1.3 Genes

The gene_reaction_rule is a boolean representation of the gene requirements for this reaction to be active as described in Schellenberger et al 2011 Nature Protocols 6(9):1290-307.

The GPR is stored as the gene_reaction_rule for a Reaction object as a string.

```
In [22]: gpr = pgi.gene_reaction_rule
         gpr

Out[22]: 'b4025'
```

Corresponding gene objects also exist. These objects are tracked by the reactions itself, as well as by the model

```
In [23]: pgi.genes

Out[23]: frozenset({<Gene b4025 at 0x7f56b0e8fac8>})

In [24]: pgi_gene = model.genes.get_by_id("b4025")
         pgi_gene

Out[24]: <Gene b4025 at 0x7f56b0e8fac8>
```

Each gene keeps track of the reactions it catalyzes

```
In [25]: pgi_gene.reactions

Out[25]: frozenset({<Reaction PGI at 0x7f56b0e396d8>})
```

Altering the gene_reaction_rule will create new gene objects if necessary and update all relationships.

```
In [26]: pgi.gene_reaction_rule = "(spam or eggs)"
         pgi.genes

Out[26]: frozenset({<Gene eggs at 0x7f56b0e35ba8>, <Gene spam at 0x7f56b0e390f0>})

In [27]: pgi_gene.reactions

Out[27]: frozenset()
```

Newly created genes are also added to the model

```
In [28]: model.genes.get_by_id("spam")

Out[28]: <Gene spam at 0x7f56b0e390f0>
```

The delete_model_genes function will evaluate the gpr and set the upper and lower bounds to 0 if the reaction is knocked out. This function can preserve existing deletions or reset them using the cumulative_deletions flag.

```
In [29]: cobra.manipulation.delete_model_genes(model, ["spam"],
                                               cumulative_deletions=True)
         print("after 1 KO: %4d < flux_PGI < %4d" %
               (pgi.lower_bound, pgi.upper_bound))

         cobra.manipulation.delete_model_genes(model, ["eggs"],
                                               cumulative_deletions=True)
         print("after 2 KO:  %4d < flux_PGI < %4d" %
               (pgi.lower_bound, pgi.upper_bound))
```

```
after 1 KO: -1000 < flux_PGI < 1000
after 2 KO:     0 < flux_PGI <    0
```

The undelete_model_genes can be used to reset a gene deletion

```
In [30]: cobra.manipulation.undelete_model_genes(model)
         print(pgi.lower_bound, "< pgi <", pgi.upper_bound)
```

```
-1000 < pgi < 1000
```

A.2 Building a Model

This simple example demonstrates how to create a model, create a reaction, and then add the reaction to the model.

We'll use the '3OAS140' reaction from the STM_1.0 model:

1.0 malACP[c] + 1.0 h[c] + 1.0 ddcaACP[c] → 1.0 co2[c] + 1.0 ACP[c] + 1.0 3omrsACP[c]

First, create the model and reaction.

```
In [1]: from cobra import Model, Reaction, Metabolite
        # Best practise: SBML compliant IDs
        cobra_model = Model('example_cobra_model')
```

```
reaction = Reaction('3OAS140')
reaction.name = '3 oxoacyl acyl carrier protein synthase n C140 '
reaction.subsystem = 'Cell Envelope Biosynthesis'
reaction.lower_bound = 0.  # This is the default
reaction.upper_bound = 1000.  # This is the default
reaction.objective_coefficient = 0. # this is the default
```

We need to create metabolites as well. If we were using an existing model, we could use get_by_id to get the apporpriate Metabolite objects instead.

```
In [2]: ACP_c = Metabolite(
            'ACP_c',
            formula='C11H21N2O7PRS',
            name='acyl-carrier-protein',
            compartment='c')
        omrsACP_c = Metabolite(
            '3omrsACP_c',
            formula='C25H45N2O9PRS',
            name='3-Oxotetradecanoyl-acyl-carrier-protein',
            compartment='c')
        co2_c = Metabolite(
            'co2_c',
            formula='CO2',
            name='CO2',
            compartment='c')
        malACP_c = Metabolite(
            'malACP_c',
            formula='C14H22N2O10PRS',
            name='Malonyl-acyl-carrier-protein',
            compartment='c')
        h_c = Metabolite(
            'h_c',
            formula='H',
            name='H',
            compartment='c')
        ddcaACP_c = Metabolite(
            'ddcaACP_c',
            formula='C23H43N2O8PRS',
            name='Dodecanoyl-ACP-n-C120ACP',
            compartment='c')
```

Adding metabolites to a reaction requires using a dictionary of the metabolites and their stoichiometric coefficients. A group of metabolites can be added all at once,

or they can be added one at a time.

```
In [3]: reaction.add_metabolites({malACP_c: -1.0,
                                  h_c: -1.0,
                                  ddcaACP_c: -1.0,
                                  co2_c: 1.0,
                                  ACP_c: 1.0,
                                  omrsACP_c: 1.0})


        reaction.reaction  # This gives a string representation of the reaction

Out[3]: 'ddcaACP_c + h_c + malACP_c --> 3omrsACP_c + ACP_c + co2_c'
```

The gene_reaction_rule is a boolean representation of the gene requirements for this reaction to be active as described in Schellenberger et al 2011 Nature Protocols 6(9):1290-307. We will assign the gene reaction rule string, which will automatically create the corresponding gene objects.

```
In [4]: reaction.gene_reaction_rule = '( STM2378 or STM1197 )'
        reaction.genes

Out[4]: frozenset({<Gene STM2378 at 0x7fada4592908>, <Gene STM1197 at
        0x7fada45927f0>})
```

At this point in time, the model is still empty

```
In [5]: print('%i reactions initially' % len(cobra_model.reactions))
        print('%i metabolites initially' % len(cobra_model.metabolites))
        print('%i genes initially' % len(cobra_model.genes))

0 reactions initially
0 metabolites initially
0 genes initially
```

We will add the reaction to the model, which will also add all associated metabolites and genes

```
In [6]: cobra_model.add_reaction(reaction)


        # Now there are things in the model
        print('%i reaction' % len(cobra_model.reactions))
        print('%i metabolites' % len(cobra_model.metabolites))
        print('%i genes' % len(cobra_model.genes))
```

```
1 reaction
6 metabolites
2 genes
```

We can iterate through the model objects to observe the contents

```
In [7]: # Iterate through the the objects in the model
        print("Reactions")
        print("---------")
        for x in cobra_model.reactions:
            print("%s : %s" % (x.id, x.reaction))

        print("")
        print("Metabolites")
        print("-----------")
        for x in cobra_model.metabolites:
            print('%9s : %s' % (x.id, x.formula))

        print("")
        print("Genes")
        print("-----")
        for x in cobra_model.genes:
            associated_ids = (i.id for i in x.reactions)
            print("%s is associated with reactions: %s" %
                  (x.id, "{" + ", ".join(associated_ids) + "}"))
```

```
Reactions
---------
3OAS140 : ddcaACP_c + h_c + malACP_c --> 3omrsACP_c + ACP_c + co2_c

Metabolites
-----------
3omrsACP_c : C25H45N2O9PRS
ddcaACP_c : C23H43N2O8PRS
    ACP_c : C11H21N2O7PRS
    co2_c : CO2
 malACP_c : C14H22N2O10PRS
      h_c : H

Genes
-----
STM2378 is associated with reactions: {3OAS140}
STM1197 is associated with reactions: {3OAS140}
```

A.3 Reading and Writing Models

Cobrapy supports reading and writing models in SBML (with and without FBC), JSON, MAT, and pickle formats. Generally, SBML with FBC version 2 is the preferred format for general use. The JSON format may be more useful for cobrapy-specific functionality.

The package also ships with test models in various formats for testing purposes.

```
In [1]: import cobra.test
        import os
        from os.path import join

        data_dir = cobra.test.data_directory

        print("mini test files: ")
        print(", ".join(i for i in os.listdir(data_dir)
                        if i.startswith("mini")))



        textbook_model = cobra.test.create_test_model("textbook")
        ecoli_model = cobra.test.create_test_model("ecoli")
        salmonella_model = cobra.test.create_test_model("salmonella")
```

```
mini test files:
mini.mat, mini_cobra.xml, mini.json, mini_fbc2.xml.gz, mini_fbc2.xml.bz2,
mini_fbc2.xml, mini_fbc1.xml, mini.pickle
```

A.3.1 SBML

The Systems Biology Markup Language is an XML-based standard format for distributing models which has support for COBRA models through the FBC extension version 2.

Cobrapy has native support for reading and writing SBML with FBCv2. Please note that all id's in the model must conform to the SBML SID requirements in order to generate a valid SBML file.

```
In [2]: cobra.io.read_sbml_model(join(data_dir, "mini_fbc2.xml"))

Out[2]: <Model mini_textbook at 0x7fa5e44d1a58>

In [3]: cobra.io.write_sbml_model(textbook_model, "test_fbc2.xml")
```

There are other dialects of SBML prior to FBC 2 which have previously been use to encode COBRA models. The primary ones is the "COBRA" dialect which used the "notes" fields in SBML files.

Cobrapy can use libsbml, which must be installed separately (see installation instructions) to read and write these files. When reading in a model, it will automatically detect whether fbc was used or not. When writing a model, the use_fbc_package flag can be used can be used to write files in this legacy "cobra" format.

```
In [4]: cobra.io.read_sbml_model(join(data_dir, "mini_cobra.xml"))

Out[4]: <Model mini_textbook at 0x7fa5ba4d12e8>

In [5]: cobra.io.write_sbml_model(textbook_model, "test_cobra.xml",
                                  use_fbc_package=False)
```

A.3.2 JSON

cobrapy models have a JSON (JavaScript Object Notation) representation. This format was crated for interoperability with escher.

```
In [6]: cobra.io.load_json_model(join(data_dir, "mini.json"))

Out[6]: <Model mini_textbook at 0x7fa5ba4a3128>

In [7]: cobra.io.save_json_model(textbook_model, "test.json")
```

A.3.3 MATLAB

Often, models may be imported and exported soley for the purposes of working with the same models in cobrapy and the MATLAB cobra toolbox. MATLAB has its own ".mat" format for storing variables. Reading and writing to these mat files from python requires scipy.

A mat file can contain multiple MATLAB variables. Therefore, the variable name of the model in the MATLAB file can be passed into the reading function:

```
In [8]: cobra.io.load_matlab_model(join(data_dir, "mini.mat"),
                                   variable_name="mini_textbook")
```

```
Out[8]: <Model mini_textbook at 0x7fa5ba483198>
```

If the mat file contains only a single model, cobra can figure out which variable to read from, and the variable_name paramter is unnecessary.

```
In [9]: cobra.io.load_matlab_model(join(data_dir, "mini.mat"))
```

```
Out[9]: <Model mini_textbook at 0x7fa5ba4a3f28>
```

Saving models to mat files is also relatively straightforward

```
In [10]: cobra.io.save_matlab_model(textbook_model, "test.mat")
```

A.3.4 Pickle

Cobra models can be serialized using the python serialization format, pickle.

Please note that use of the pickle format is generally not recommended for most use cases. JSON, SBML, and MAT are generally the preferred formats.

A.4 Simulating with FBA

Simulations using flux balance analysis can be solved using Model.optimize(). This will maximize or minimize (maximizing is the default) flux through the objective reactions.

```
In [1]: import pandas
        pandas.options.display.max_rows = 100

        import cobra.test
        model = cobra.test.create_test_model("textbook")
```

A.4.1 Running FBA

```
In [2]: model.optimize()
Out[2]: <Solution 0.87 at 0x10ddd0080>
```

The Model.optimize() function will return a Solution object, which will also be stored at model.solution. A solution object has several attributes:

- f: the objective value
- status: the status from the linear programming solver
- x_dict: a dictionary of {reaction_id: flux_value} (also called "primal")
- x: a list for x_dict
- y_dict: a dictionary of {metabolite_id: dual_value}.
- y: a list for y_dict

For example, after the last call to model.optimize(), the status should be 'optimal' if the solver returned no errors, and f should be the objective value

```
In [3]: model.solution.status
Out[3]: 'optimal'
In [4]: model.solution.f
Out[4]: 0.8739215069684305
```

Analyzing FBA solutions

Models solved using FBA can be further analyzed by using summary methods, which output printed text to give a quick representation of model behavior. Calling the summary method on the entire model displays information on the input and output behavior of the model, along with the optimized objective.

```
In [5]: model.summary()
```

```
IN FLUXES                      OUT FLUXES                    OBJECTIVES
o2_e        -21.80             h2o_e     29.18
Biomass_Ecoli_core    0.874
glc__D_e    -10.00             co2_e     22.81
nh4_e        -4.77             h_e       17.53
pi_e         -3.21
```

In addition, the input-output behavior of individual metabolites can also be inspected using summary methods. For instance, the following commands can be used to examine the overall redox balance of the model

```
In [6]: model.metabolites.nadh_c.summary()
```

```
PRODUCING REACTIONS -- Nicotinamide adenine dinucleotide - reduced
---------------------------------------------------------------------
  %      FLUX   RXN ID                          REACTION
 41.6%     16     GAPD        g3p_c + nad_c + pi_c <=> 13dpg_c + h_c +
nadh_c
 24.1%    9.3      PDH     coa_c + nad_c + pyr_c --> accoa_c + co2_c +
nadh_c
 13.1%    5.1    AKGDH    akg_c + coa_c + nad_c --> co2_c + nadh_c +
succoa_c
 13.1%    5.1      MDH              mal__L_c + nad_c <=> h_c + nadh_c +
oaa_c
  8.0%    3.1 Bioma...   1.496 3pg_c + 3.7478 accoa_c + 59.81 atp_c +
0.36...

CONSUMING REACTIONS -- Nicotinamide adenine dinucleotide - reduced
---------------------------------------------------------------------
  %      FLUX   RXN ID                          REACTION
100.0%    -39   NADH16   4.0 h_c + nadh_c + q8_c --> 3.0 h_e + nad_c +
q8h2_c
```

Or to get a sense of the main energy production and consumption reactions

```
In [7]: model.metabolites.atp_c.summary()

PRODUCING REACTIONS -- ATP
--------------------------
  %      FLUX   RXN ID                          REACTION
 66.6%     46   ATPS4r     adp_c + 4.0 h_e + pi_c <=> atp_c + h2o_c + 3.0
h_c
 23.4%     16      PGK                       3pg_c + atp_c <=> 13dpg_c +
adp_c
  7.4%    5.1   SUCOAS     atp_c + coa_c + succ_c <=> adp_c + pi_c +
succoa_c
  2.6%    1.8      PYK                    adp_c + h_c + pep_c --> atp_c +
pyr_c

CONSUMING REACTIONS -- ATP
--------------------------
  %      FLUX   RXN ID                          REACTION
 76.5%    -52 Bioma...   1.496 3pg_c + 3.7478 accoa_c + 59.81 atp_c +
0.36...
 12.3%   -8.4     ATPM                     atp_c + h2o_c --> adp_c + h_c +
pi_c
 10.9%   -7.5      PFK                    atp_c + f6p_c --> adp_c + fdp_c +
h_c
```

A.4.2 Changing the Objectives

The objective function is determined from the objective_coefficient attribute of the objective reaction(s). Generally, a "biomass" function which describes the composition of metabolites which make up a cell is used.

```
In [8]: biomass_rxn = model.reactions.get_by_id("Biomass_Ecoli_core")
```

Currently in the model, there is only one objective reaction (the biomass reaction), with an objective coefficient of 1.

```
In [9]: model.objective

Out[9]: {<Reaction Biomass_Ecoli_core at 0x116510828>: 1.0}
```

The objective function can be changed by assigning Model.objective, which can be a reaction object (or just it's name), or a dict of {Reaction: objective_coefficient}.

```
In [10]: # change the objective to ATPM
         model.objective = "ATPM"

         # The upper bound should be 1000, so that we get
         # the actual optimal value
         model.reactions.get_by_id("ATPM").upper_bound = 1000.
         model.objective
```

```
Out[10]: {<Reaction ATPM at 0x1165107b8>: 1}
```

```
In [11]: model.optimize().f
```

```
Out[11]: 174.99999999999997
```

The objective function can also be changed by setting Reaction.objective_coefficient directly.

```
In [12]: model.reactions.get_by_id("ATPM").objective_coefficient = 0.
         biomass_rxn.objective_coefficient = 1.

         model.objective
```

```
Out[12]: {<Reaction Biomass_Ecoli_core at 0x116510828>: 1.0}
```

A.4.3 Running FVA

FBA will not give always give unique solution, because multiple flux states can achieve the same optimum. FVA (or flux variability analysis) finds the ranges of each metabolic flux at the optimum.

```
In [13]: fva_result = cobra.flux_analysis.flux_variability_analysis(
             model, model.reactions[:20])
         pandas.DataFrame.from_dict(fva_result).T.round(5)
```

```
Out[13]:                maximum   minimum
         ACALD          0.00000   0.00000
         ACALDt        -0.00000   0.00000
         ACKr          -0.00000   0.00000
```

```
ACONTa                6.00725   6.00725
ACONTb                6.00725   6.00725
ACt2r                 0.00000   0.00000
ADK1                 -0.00000   0.00000
AKGDH                 5.06438   5.06438
AKGt2r                0.00000   0.00000
ALCD2x                0.00000   0.00000
ATPM                  8.39000   8.39000
ATPS4r               45.51401  45.51401
Biomass_Ecoli_core    0.87392   0.87392
CO2t                -22.80983 -22.80983
CS                    6.00725   6.00725
CYTBD                43.59899  43.59899
D_LACt2               0.00000   0.00000
ENO                  14.71614  14.71614
ETOHt2r               0.00000   0.00000
EX_ac_e              -0.00000   0.00000
```

Setting parameter fraction_of_optimium=0.90 would give the flux ranges for reactions at 90% optimality.

```
In [14]: fva_result = cobra.flux_analysis.flux_variability_analysis(
             model, model.reactions[:20], fraction_of_optimum=0.9)
         pandas.DataFrame.from_dict(fva_result).T.round(5)
```

Out[14]:

```
                      maximum   minimum
ACALD                 0.00000  -2.54237
ACALDt               -0.00000  -2.54237
ACKr                 -0.00000  -3.81356
ACONTa                8.89452   0.84859
ACONTb                8.89452   0.84859
ACt2r                 0.00000  -3.81356
ADK1                 17.16100   0.00000
AKGDH                 8.04593   0.00000
AKGt2r                0.00000  -1.43008
ALCD2x                0.00000  -2.21432
ATPM                 25.55100   8.39000
ATPS4r               59.38106  34.82562
Biomass_Ecoli_core    0.87392   0.78653
CO2t                -15.20653 -26.52885
CS                    8.89452   0.84859
CYTBD                51.23909  35.98486
D_LACt2               0.00000  -2.14512
ENO                  16.73252   8.68659
ETOHt2r               0.00000  -2.21432
EX_ac_e               3.81356   0.00000
```

Running FVA in summary methods

Flux variability analysis can also be embedded in calls to summary methods. For instance, the expected variability in substrate consumption and product formation can be quickly found by

```
In [15]: model.optimize()
         model.summary(fva=0.95)
```

```
IN FLUXES                      OUT FLUXES                      OBJECTIVES
o2_e          -21.80 ± 1.91    h2o_e         27.86 ± 2.86
Biomass_Ecoli_core    0.874
glc__D_e       -9.76 ± 0.24    co2_e         21.81 ± 2.86
nh4_e          -4.84 ± 0.32    h_e           19.51 ± 2.86
pi_e           -3.13 ± 0.08    for_e          2.86 ± 2.86
                               ac_e           0.95 ± 0.95
                               acald_e        0.64 ± 0.64
                               pyr_e          0.64 ± 0.64
                               etoh_e         0.55 ± 0.55
                               lac__D_e       0.54 ± 0.54
                               succ_e         0.42 ± 0.42
                               akg_e          0.36 ± 0.36
                               glu__L_e       0.32 ± 0.32
```

Similarly, variability in metabolite mass balances can also be checked with flux variability analysis

```
In [16]: model.metabolites.pyr_c.summary(fva=0.95)
```

```
PRODUCING REACTIONS -- Pyruvate
-------------------------------
  %             FLUX   RXN ID                          REACTION
 85.0%   9.76 ± 0.24   GLCpts                      glc__D_e + pep_c -->
g6p_c + pyr_c
 15.0%   6.13 ± 6.13      PYK                   adp_c + h_c + pep_c -->
atp_c + pyr_c

CONSUMING REACTIONS -- Pyruvate
-------------------------------
  %             FLUX   RXN ID                          REACTION
 78.9%  11.34 ± 7.43      PDH     coa_c + nad_c + pyr_c --> accoa_c + co2_c
+ nadh_c
 21.1%   0.85 ± 0.02 Bioma...   1.496 3pg_c + 3.7478 accoa_c + 59.81 atp_c
+ 0.36...
```

In these summary methods, the values are reported as a the center point +/- the range of the FVA solution, calculated from the maximum and minimum values.

A.4.4 Running pFBA

Parsimonious FBA (often written pFBA) finds a flux distribution which gives the optimal growth rate, but minimizes the total sum of flux. This involves solving two sequential linear programs, but is handled transparently by cobrapy. For more details on pFBA, please see Lewis et al. (2010).

```
In [17]: FBA_sol = model.optimize()
         pFBA_sol = cobra.flux_analysis.optimize_minimal_flux(model)
```

These functions should give approximately the same objective value

```
In [18]: abs(FBA_sol.f - pFBA_sol.f)
```

```
Out[18]: 1.1102230246251565e-16
```

A.5 Simulating Deletions

```
In [1]: import pandas
        from time import time

        import cobra.test
        from cobra.flux_analysis import \
            single_gene_deletion, single_reaction_deletion, \
            double_gene_deletion, double_reaction_deletion

        cobra_model = cobra.test.create_test_model("textbook")
        ecoli_model = cobra.test.create_test_model("ecoli")
```

A.5.1 Single Deletions

Perform all single gene deletions on a model

```
In [2]: growth_rates, statuses = single_gene_deletion(cobra_model)
```

These can also be done for only a subset of genes

```
In [3]: gr, st = single_gene_deletion(cobra_model,
                                      cobra_model.genes[:20])
        pandas.DataFrame.from_dict({"growth_rates": gr,
                                    "status": st})
```

Out[3]:

	growth_rates	status
b0116	0.782351	optimal
b0118	0.873922	optimal
b0351	0.873922	optimal
b0356	0.873922	optimal
b0474	0.873922	optimal
b0726	0.858307	optimal
b0727	0.858307	optimal
b1241	0.873922	optimal
b1276	0.873922	optimal
b1478	0.873922	optimal
b1849	0.873922	optimal
b2296	0.873922	optimal
b2587	0.873922	optimal
b3115	0.873922	optimal
b3732	0.374230	optimal
b3733	0.374230	optimal
b3734	0.374230	optimal
b3735	0.374230	optimal
b3736	0.374230	optimal
s0001	0.211141	optimal

This can also be done for reactions

```
In [4]: gr, st = single_reaction_deletion(cobra_model,
                                         cobra_model.reactions[:20])
        pandas.DataFrame.from_dict({"growth_rates": gr,
                                    "status": st}).round(4)
```

Out[4]:

	growth_rates	status
ACALD	0.8739	optimal
ACALDt	0.8739	optimal
ACKr	0.8739	optimal
ACONTa	0.0000	optimal
ACONTb	0.0000	optimal
ACt2r	0.8739	optimal

```
ADK1                      0.8739  optimal
AKGDH                     0.8583  optimal
AKGt2r                    0.8739  optimal
ALCD2x                    0.8739  optimal
ATPM                      0.9166  optimal
ATPS4r                    0.3742  optimal
Biomass_Ecoli_core        0.0000  optimal
CO2t                      0.4617  optimal
CS                       -0.0000  optimal
CYTBD                     0.2117  optimal
D_LACt2                   0.8739  optimal
ENO                      -0.0000  optimal
ETOHt2r                   0.8739  optimal
EX_ac_e                   0.8739  optimal
```

A.5.2 Double Deletions

Double deletions run in a similar way. Passing in return_frame=True will cause them to format the results as a pandas Dataframe

```
In [5]: double_gene_deletion(cobra_model, cobra_model.genes[-5:],
                             return_frame=True).round(4)
```

Out[5]:

	b2464	b0008	b2935	b2465	b3919
b2464	0.8739	0.8648	0.8739	0.8739	0.704
b0008	0.8648	0.8739	0.8739	0.8739	0.704
b2935	0.8739	0.8739	0.8739	0.0000	0.704
b2465	0.8739	0.8739	0.0000	0.8739	0.704
b3919	0.7040	0.7040	0.7040	0.7040	0.704

By default, the double deletion function will automatically use multiprocessing, splitting the task over up to 4 cores if they are available. The number of cores can be manually sepcified as well. Setting use of a single core will disable use of the multiprocessing library, which often aids debuggging.

```
In [6]: start = time()  # start timer()
        double_gene_deletion(ecoli_model, ecoli_model.genes[:300],
                             number_of_processes=2)
        t1 = time() - start
        print("Double gene deletions for 200 genes completed in "
```

```
              "%.2f sec with 2 cores" % t1)

        start = time()  # start timer()
        double_gene_deletion(ecoli_model, ecoli_model.genes[:300],
                             number_of_processes=1)
        t2 = time() - start
        print("Double gene deletions for 200 genes completed in "
              "%.2f sec with 1 core" % t2)

        print("Speedup of %.2fx" % (t2 / t1))
```

```
Double gene deletions for 200 genes completed in 27.03 sec with 2 cores
Double gene deletions for 200 genes completed in 40.73 sec with 1 core
Speedup of 1.51x
```

Double deletions can also be run for reactions

```
In [7]: double_reaction_deletion(cobra_model,
                                 cobra_model.reactions[2:7],
                                 return_frame=True).round(4)
```

Out[7]:

	ACKr	ACONTa	ACONTb	ACt2r	ADK1
ACKr	0.8739	0.0	0.0	0.8739	0.8739
ACONTa	0.0000	0.0	0.0	0.0000	0.0000
ACONTb	0.0000	0.0	0.0	0.0000	0.0000
ACt2r	0.8739	0.0	0.0	0.8739	0.8739
ADK1	0.8739	0.0	0.0	0.8739	0.8739

A.6 Phenotype Phase Plane

Phenotype phase planes will show distinct phases of optimal growth with different use of two different substrates. For more information, see Edwards et al.

Cobrapy supports calculating and plotting (using matplotlib) these phenotype phase planes. Here, we will make one for the "textbook" E. coli core model.

```
In [1]: %matplotlib inline
        from IPython.display import set_matplotlib_formats
        set_matplotlib_formats('png', 'pdf')
```

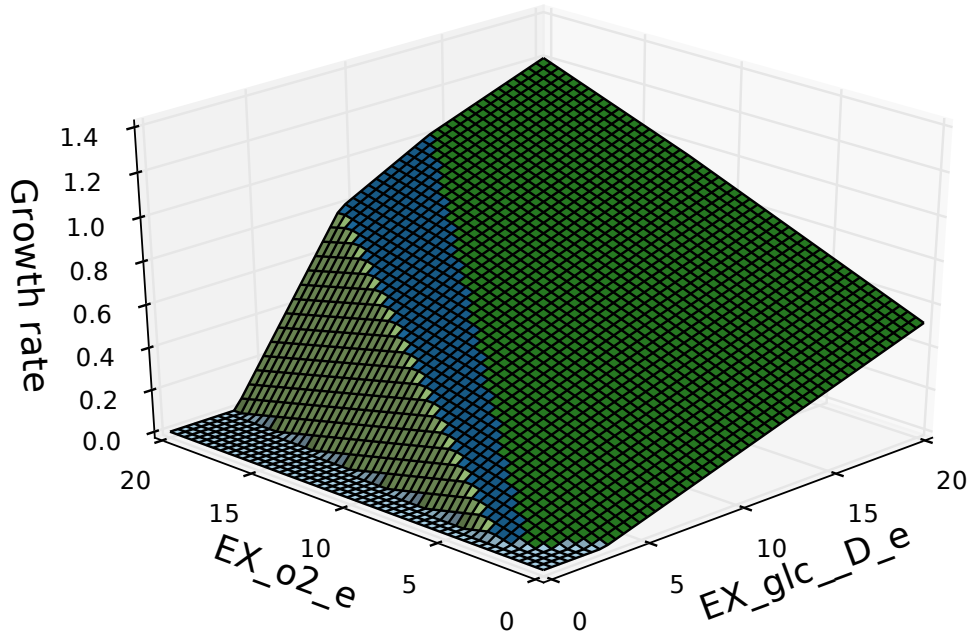
```
from time import time

import cobra.test
from cobra.flux_analysis import calculate_phenotype_phase_plane

model = cobra.test.create_test_model("textbook")
```

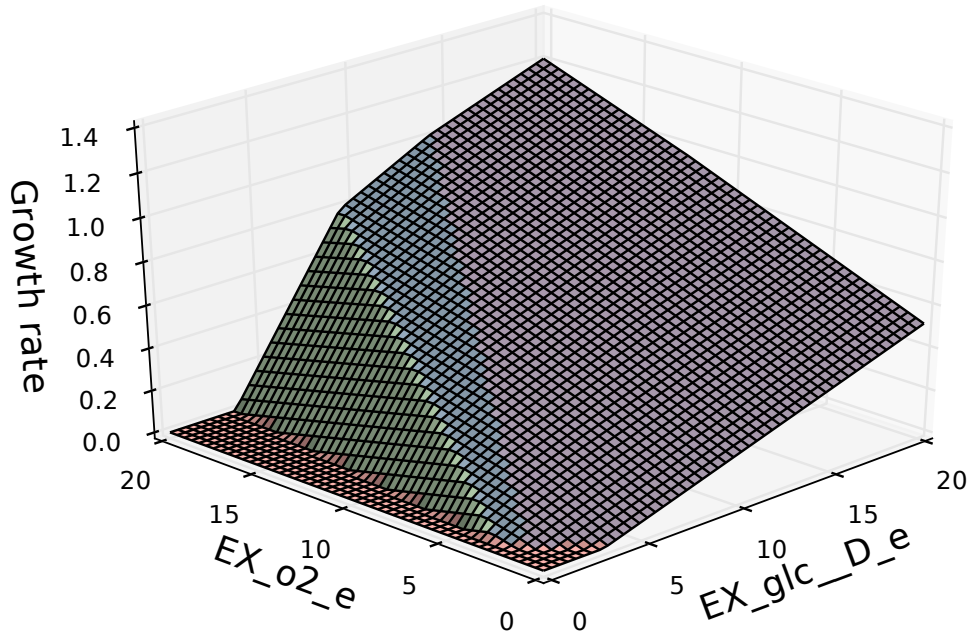
We want to make a phenotype phase plane to evaluate uptakes of Glucose and Oxygen.

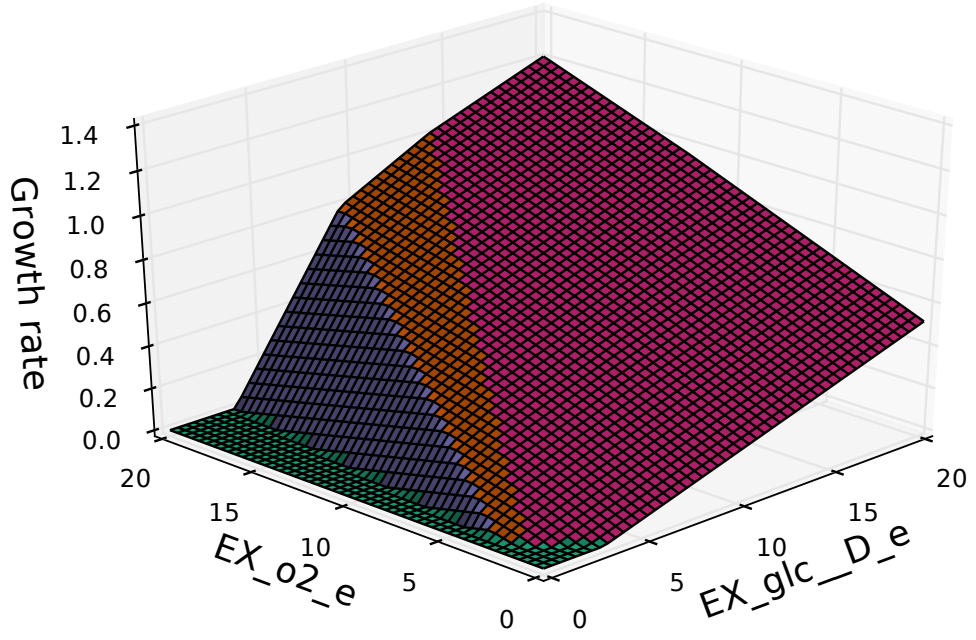
```
In [2]: data = calculate_phenotype_phase_plane(
            model, "EX_glc__D_e", "EX_o2_e")
        data.plot_matplotlib();
```



If palettable is installed, other color schemes can be used as well

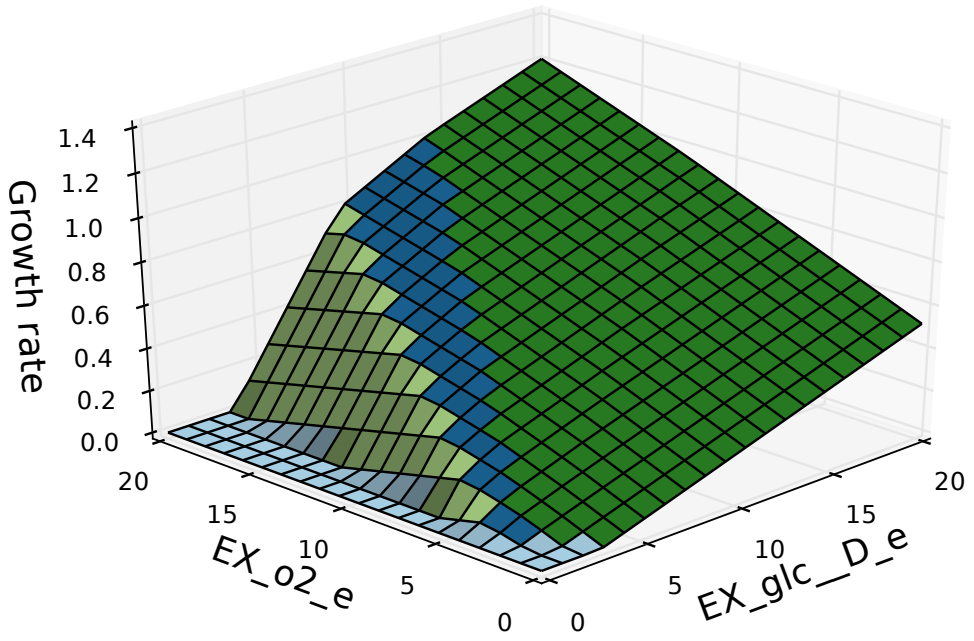
```
In [3]: data.plot_matplotlib("Pastel1")
        data.plot_matplotlib("Dark2");
```


Growth rate
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
20
15
10
5
0
EX_o2_e
0
5
10
15
20
EX_glc__D_e


Growth rate
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
20
15
10
5
0
EX_o2_e
0
5
10
15
20
EX_glc__D_e

The number of points which are plotted in each dimension can also be changed

```
In [4]: data = calculate_phenotype_phase_plane(
            model, "EX_glc__D_e", "EX_o2_e",
            reaction1_npoints=20, reaction2_npoints=20)
        data.plot_matplotlib();
```



The code can also use multiple processes to speed up calculations

```
In [5]: start_time = time()
        calculate_phenotype_phase_plane(
            model, "EX_glc__D_e", "EX_o2_e",
            reaction1_npoints=100, reaction2_npoints=100,
            n_processes=1)
        print("took %.2f seconds with 1 process" % (time() - start_time))

        start_time = time()
        calculate_phenotype_phase_plane(
            model, "EX_glc__D_e", "EX_o2_e",
            reaction1_npoints=100, reaction2_npoints=100,
            n_processes=4)
        print("took %.2f seconds with 4 process" % (time() - start_time))
```

```
took 0.44 seconds with 1 process
took 0.25 seconds with 4 process
```

A.7 Mixed-Integer Linear Programming

A.7.1 Ice Cream

This example was originally contributed by Joshua Lerman.

An ice cream stand sells cones and popsicles. It wants to maximize its profit, but is subject to a budget.

We can write this problem as a linear program:

max cone · cone_margin + popsicle · popsicle margin

subject to

cone · cone_cost + popsicle · popsicle_cost ≤ budget

```
In [1]: cone_selling_price = 7.
        cone_production_cost = 3.
        popsicle_selling_price = 2.
        popsicle_production_cost = 1.
        starting_budget = 100.
```

This problem can be written as a cobra.Model

```
In [2]: from cobra import Model, Metabolite, Reaction

        cone = Reaction("cone")
        popsicle = Reaction("popsicle")

        # constrainted to a budget
        budget = Metabolite("budget")
        budget._constraint_sense = "L"
        budget._bound = starting_budget
        cone.add_metabolites({budget: cone_production_cost})
        popsicle.add_metabolites({budget: popsicle_production_cost})

        # objective coefficient is the profit to be made from each unit
        cone.objective_coefficient = \
            cone_selling_price - cone_production_cost
        popsicle.objective_coefficient = \
            popsicle_selling_price - popsicle_production_cost
```

```
m = Model("lerman_ice_cream_co")
m.add_reactions((cone, popsicle))

m.optimize().x_dict
```

```
Out[2]: {'cone': 33.333333333333336, 'popsicle': 0.0}
```

In reality, cones and popsicles can only be sold in integer amounts. We can use the variable kind attribute of a cobra.Reaction to enforce this.

```
In [3]: cone.variable_kind = "integer"
        popsicle.variable_kind = "integer"
        m.optimize().x_dict
```

```
Out[3]: {'cone': 33.0, 'popsicle': 1.0}
```

Now the model makes both popsicles and cones.

A.7.2 Restaurant Order

To tackle the less immediately obvious problem from the following XKCD comic:

```
In [4]: from IPython.display import Image
        Image(url=r"http://imgs.xkcd.com/comics/np_complete.png")
```

```
Out[4]: <IPython.core.display.Image object>
```

We want a solution satisfying the following constraints:

$$\begin{pmatrix} 2.15 & 2.75 & 3.35 & 3.55 & 4.20 & 5.80 \end{pmatrix} \cdot \vec{v} = 15.05$$

$$\vec{v}_i \geq 0$$

$$\vec{v}_i \in \mathbb{Z}$$

This problem can be written as a COBRA model as well.

```
In [5]: total_cost = Metabolite("constraint")
        total_cost._bound = 15.05

        costs = {"mixed_fruit": 2.15, "french_fries": 2.75,
                 "side_salad": 3.35, "hot_wings": 3.55,
                 "mozarella_sticks": 4.20, "sampler_plate": 5.80}

        m = Model("appetizers")

        for item, cost in costs.items():
            r = Reaction(item)
            r.add_metabolites({total_cost: cost})
            r.variable_kind = "integer"
            m.add_reaction(r)

        # To add to the problem, suppose we want to
        # eat as little mixed fruit as possible.
        m.reactions.mixed_fruit.objective_coefficient = 1

        m.optimize(objective_sense="minimize").x_dict
```

```
Out[5]: {'french_fries': 0.0,
         'hot_wings': 2.0,
         'mixed_fruit': 1.0,
         'mozarella_sticks': 0.0,
         'sampler_plate': 1.0,
         'side_salad': 0.0}
```

There is another solution to this problem, which would have been obtained if we had maximized for mixed fruit instead of minimizing.

```
In [6]: m.optimize(objective_sense="maximize").x_dict
```

```
Out[6]: {'french_fries': 0.0,
         'hot_wings': 0.0,
         'mixed_fruit': 7.0,
         'mozarella_sticks': 0.0,
         'sampler_plate': 0.0,
         'side_salad': 0.0}
```

A.7.3 Boolean Indicators

To give a COBRA-related example, we can create boolean variables as integers, which can serve as indicators for a reaction being active in a model. For a reaction

flux v with lower bound -1000 and upper bound 1000, we can create a binary variable b with the following constraints:

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

$$-1000 \cdot b \leq v \leq 1000 \cdot b$$

To introduce the above constraints into a cobra model, we can rewrite them as follows

$$v \leq b \cdot 1000 \Rightarrow v - 1000 \cdot b \leq 0$$

$$-1000 \cdot b \leq v \Rightarrow v + 1000 \cdot b \geq 0$$

```
In [7]: import cobra.test
        model = cobra.test.create_test_model("textbook")

        # an indicator for pgi
        pgi = model.reactions.get_by_id("PGI")
        # make a boolean variable
        pgi_indicator = Reaction("indicator_PGI")
        pgi_indicator.lower_bound = 0
        pgi_indicator.upper_bound = 1
        pgi_indicator.variable_kind = "integer"
        # create constraint for v - 1000 b <= 0
        pgi_plus = Metabolite("PGI_plus")
        pgi_plus._constraint_sense = "L"
        # create constraint for v + 1000 b >= 0
        pgi_minus = Metabolite("PGI_minus")
        pgi_minus._constraint_sense = "G"

        pgi_indicator.add_metabolites({pgi_plus: -1000,
                                       pgi_minus: 1000})
        pgi.add_metabolites({pgi_plus: 1, pgi_minus: 1})
        model.add_reaction(pgi_indicator)


        # an indicator for zwf
        zwf = model.reactions.get_by_id("G6PDH2r")
        zwf_indicator = Reaction("indicator_ZWF")
        zwf_indicator.lower_bound = 0
        zwf_indicator.upper_bound = 1
        zwf_indicator.variable_kind = "integer"
        # create constraint for v - 1000 b <= 0
        zwf_plus = Metabolite("ZWF_plus")
```

```
        zwf_plus._constraint_sense = "L"
        # create constraint for v + 1000 b >= 0
        zwf_minus = Metabolite("ZWF_minus")
        zwf_minus._constraint_sense = "G"

        zwf_indicator.add_metabolites({zwf_plus: -1000,
                                       zwf_minus: 1000})
        zwf.add_metabolites({zwf_plus: 1, zwf_minus: 1})

        # add the indicator reactions to the model
        model.add_reaction(zwf_indicator)
```

In a model with both these reactions active, the indicators will also be active

```
In [8]: solution = model.optimize()
        print("PGI indicator = %d" % solution.x_dict["indicator_PGI"])
        print("ZWF indicator = %d" % solution.x_dict["indicator_ZWF"])
        print("PGI flux = %.2f" % solution.x_dict["PGI"])
        print("ZWF flux = %.2f" % solution.x_dict["G6PDH2r"])
```

```
PGI indicator = 1
ZWF indicator = 1
PGI flux = 4.86
ZWF flux = 4.96
```

Because these boolean indicators are in the model, additional constraints can be applied on them. For example, we can prevent both reactions from being active at the same time by adding the following constraint:

$$b_{\text{pgi}} + b_{\text{zwf}} = 1$$

```
In [9]: or_constraint = Metabolite("or")
        or_constraint._bound = 1
        zwf_indicator.add_metabolites({or_constraint: 1})
        pgi_indicator.add_metabolites({or_constraint: 1})

        solution = model.optimize()
        print("PGI indicator = %d" % solution.x_dict["indicator_PGI"])
        print("ZWF indicator = %d" % solution.x_dict["indicator_ZWF"])
        print("PGI flux = %.2f" % solution.x_dict["PGI"])
        print("ZWF flux = %.2f" % solution.x_dict["G6PDH2r"])
```

```
PGI indicator = 1
ZWF indicator = 0
PGI flux = 9.82
ZWF flux = 0.00
```

A.8 Quadratic Programming

Suppose we want to minimize the Euclidean distance of the solution to the origin while subject to linear constraints. This will require a quadratic objective function. Consider this example problem:

$$\textbf{min } \tfrac{1}{2}\left(x^2 + y^2\right)$$

subject to

$$x + y = 2$$

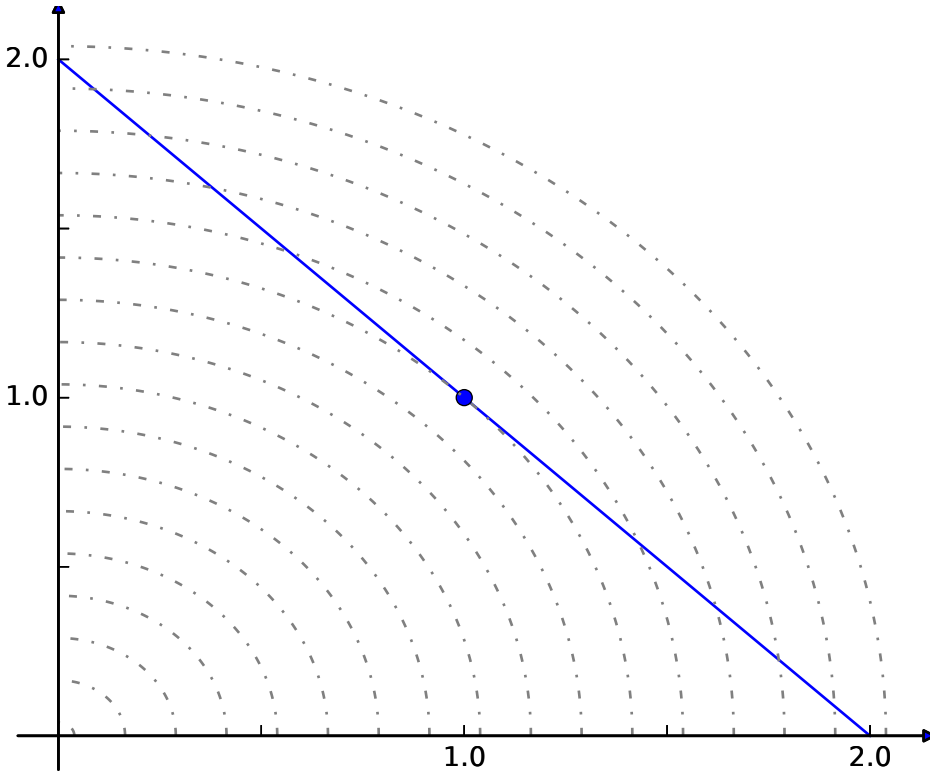
$$x \geq 0$$

$$y \geq 0$$

This problem can be visualized graphically:

```
In [1]: %matplotlib inline
        import plot_helper

        plot_helper.plot_qp1()
```



The objective can be rewritten as $\frac{1}{2}v^T \cdot \mathbf{Q} \cdot v$, where $v = \begin{pmatrix} x \\ y \end{pmatrix}$ and $\mathbf{Q} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

The matrix $\mathbf{Q}$ can be passed into a cobra model as the quadratic objective.

```
In [2]: import scipy

        from cobra import Reaction, Metabolite, Model, solvers
```

The quadratic objective $\mathbf{Q}$ should be formatted as a scipy sparse matrix.

```
In [3]: Q = scipy.sparse.eye(2).todok()
        Q
```

```
Out[3]: <2x2 sparse matrix of type '<class 'numpy.float64'>'
                with 2 stored elements in Dictionary Of Keys format>
```

In this case, the quadratic objective is simply the identity matrix

```
In [4]: Q.todense()
```

```
Out[4]: matrix([[ 1.,  0.],
                [ 0.,  1.]])
```

We need to use a solver that supports quadratic programming, such as gurobi or cplex. If a solver which supports quadratic programming is installed, this function will return its name.

```
In [5]: print(solvers.get_solver_name(qp=True))
```

```
cplex
```

```
In [6]: c = Metabolite("c")
        c._bound = 2
        x = Reaction("x")
        y = Reaction("y")
        x.add_metabolites({c: 1})
        y.add_metabolites({c: 1})
        m = Model()
        m.add_reactions([x, y])
        sol = m.optimize(quadratic_component=Q, objective_sense="minimize")
        sol.x_dict
```

```
Out[6]: {'x': 1.0, 'y': 1.0}
```

Suppose we change the problem to have a mixed linear and quadratic objective.

$$\textbf{min } \tfrac{1}{2}\left(x^2 + y^2\right) - y$$

subject to

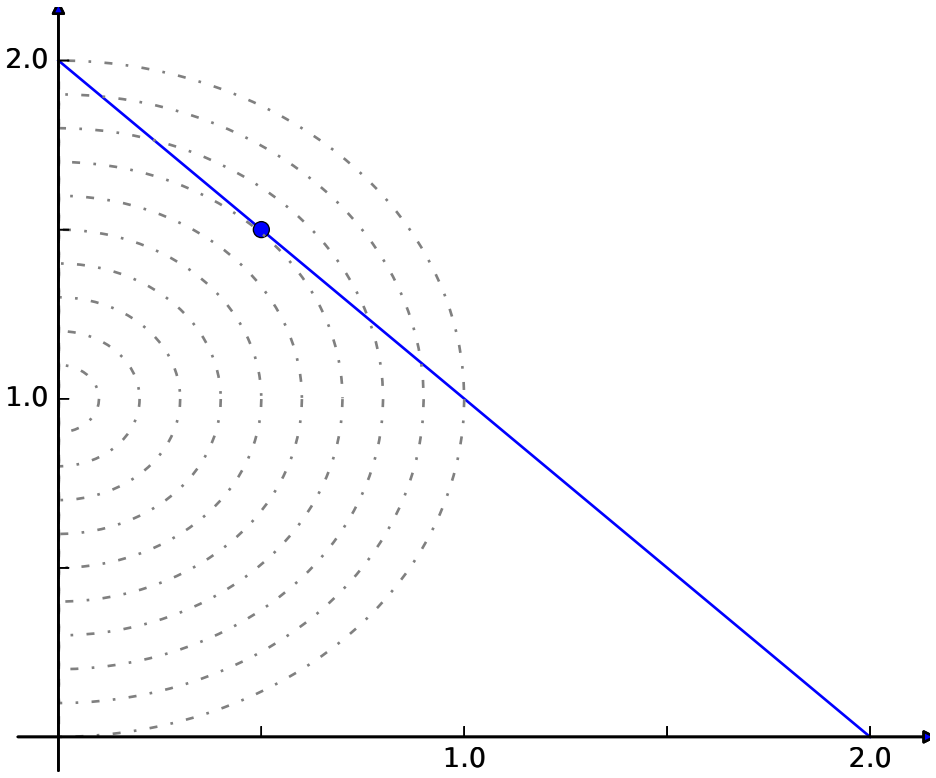
$$x + y = 2$$

$$x \geq 0$$

$$y \geq 0$$

Graphically, this would be

```
In [7]: plot_helper.plot_qp2()
```



QP solvers in cobrapy will combine linear and quadratic coefficients. The linear portion will be obtained from the same objective_coefficient attribute used with LP's.

```
In [8]: y.objective_coefficient = -1
        sol = m.optimize(quadratic_component=Q, objective_sense="minimize")
        sol.x_dict

Out[8]: {'x': 0.5, 'y': 1.5}
```

A.9 Loopless FBA

The goal of this procedure is identification of a thermodynamically consistent flux state without loops, as implied by the name.

Usually, the model has the following constraints.

$$S \cdot v = 0$$

$$lb \leq v \leq ub$$

However, this will allow for thermodynamically infeasible loops (referred to as type 3 loops) to occur, where flux flows around a cycle without any net change of metabolites. For most cases, this is not a major issue, as solutions with these loops can usually be converted to equivalent solutions without them. However, if a flux state is desired which does not exhibit any of these loops, loopless FBA can be used. The formulation used here is modified from Schellenberger et al.

We can make the model irreversible, so that all reactions will satisfy

$$0 \leq lb \leq v \leq ub \leq \max(ub)$$

We will add in boolean indicators as well, such that

$$\max(ub) \cdot i \geq v$$

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

We also want to ensure that an entry in the row space of S also exists with negative values wherever v is nonzero. In this expression, $1 - i$ acts as a not to indicate inactivity of a reaction.

$$S^{\mathsf{T}} x - (1 - i)(\max(ub) + 1) \leq -1$$

We will construct an LP integrating both constraints.

$$\begin{pmatrix} S & 0 & 0 \\ -I & \max(ub)I & 0 \\ 0 & (\max(ub) + 1)I & S^{\mathsf{T}} \end{pmatrix} \cdot \begin{pmatrix} v \\ i \\ x \end{pmatrix} \begin{matrix} = \\ \geq \\ \leq \end{matrix} \begin{matrix} 0 \\ 0 \\ \max(ub) \end{matrix}$$

Note that these extra constraints are not applied to boundary reactions which bring metabolites in and out of the system.

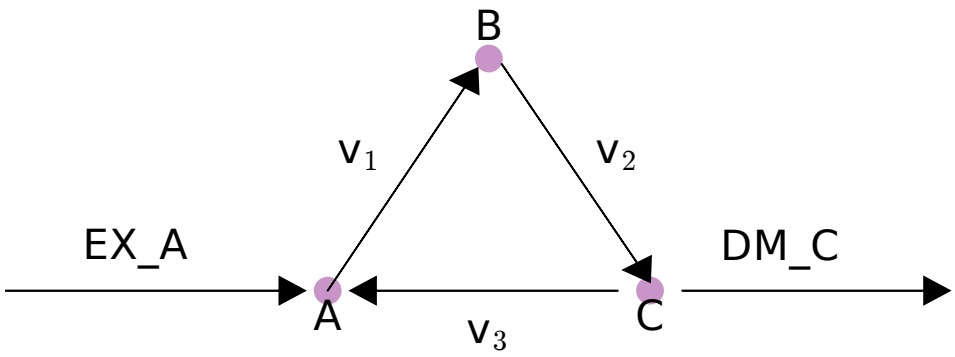
```
In [1]: %matplotlib inline
        import plot_helper

        import cobra.test
        from cobra import Reaction, Metabolite, Model
        from cobra.flux_analysis.loopless import construct_loopless_model
        from cobra.flux_analysis import optimize_minimal_flux
        from cobra.solvers import get_solver_name
```

We will demonstrate with a toy model which has a simple loop cycling A -> B -> C -> A, with A allowed to enter the system and C allowed to leave. A graphical

view of the system is drawn below:

```
In [2]: plot_helper.plot_loop()
```




```
In [3]: test_model = Model()
        test_model.add_metabolites([Metabolite(i) for i in "ABC"])
        test_model.add_reactions([Reaction(i) for i in
                                  ["EX_A", "DM_C", "v1", "v2", "v3"]])

        test_model.reactions.EX_A.add_metabolites({"A": 1})
        test_model.reactions.DM_C.add_metabolites({"C": -1})
        test_model.reactions.DM_C.objective_coefficient = 1

        test_model.reactions.v1.add_metabolites({"A": -1, "B": 1})
        test_model.reactions.v2.add_metabolites({"B": -1, "C": 1})
        test_model.reactions.v3.add_metabolites({"C": -1, "A": 1})
```

While this model contains a loop, a flux state exists which has no flux through reaction v3, and is identified by loopless FBA.

```
In [4]: solution = construct_loopless_model(test_model).optimize()
        print("loopless solution: status = " + solution.status)
        print("loopless solution: v3 = %.1f" % solution.x_dict["v3"])

loopless solution: status = optimal
loopless solution: v3 = 0.0
```

If there there is no forced flux through a loopless reaction, parsimonious FBA will also have no flux through the loop.

```
In [5]: solution = optimize_minimal_flux(test_model)
        print("parsimonious solution: status = " + solution.status)
        print("parsimonious solution: v3 = %.1f" % solution.x_dict["v3"])
```

```
parsimonious solution: status = optimal
parsimonious solution: v3 = 0.0
```

However, if flux is forced through v3, then there is no longer a feasible loopless solution, but the parsimonious solution will still exist.

```
In [6]: test_model.reactions.v3.lower_bound = 1
        solution = construct_loopless_model(test_model).optimize()
        print("loopless solution: status = " + solution.status)
```

```
loopless solution: status = infeasible
```

```
In [7]: solution = optimize_minimal_flux(test_model)
        print("parsimonious solution: status = " + solution.status)
        print("parsimonious solution: v3 = %.1f" % solution.x_dict["v3"])
```

```
parsimonious solution: status = optimal
parsimonious solution: v3 = 1.0
```

Loopless FBA is also possible on genome scale models, but it requires a capable MILP solver. If one is installed, cobrapy can detect it automatically using the get_solver_name function

```
In [8]: mip_solver = get_solver_name(mip=True)
        print(mip_solver)
```

```
cplex
```

```
In [9]: salmonella = cobra.test.create_test_model("salmonella")
        construct_loopless_model(salmonella).optimize(solver=mip_solver)
```

```
Out[9]: <Solution 0.38 at 0x7f9285d7ffd0>
```

```
In [10]: ecoli = cobra.test.create_test_model("ecoli")
         construct_loopless_model(ecoli).optimize(solver=mip_solver)
```

```
Out[10]: <Solution 0.98 at 0x7f9285c89470>
```

A.10 Gapfillling

GrowMatch and SMILEY are gap-filling algorithms, which try to to make the minimal number of changes to a model and allow it to simulate growth. For more information, see Kumar et al.. Please note that these algorithms are Mixed-Integer Linear Programs, which need solvers such as gurobi or cplex to function correctly.

```
In [1]: import cobra.test

        model = cobra.test.create_test_model("salmonella")
```

In this model D-Fructose-6-phosphate is an essential metabolite. We will remove all the reactions using it, and at them to a separate model.

```
In [2]: # remove some reactions and add them to the universal reactions
        Universal = cobra.Model("Universal_Reactions")
        for i in [i.id for i in model.metabolites.f6p_c.reactions]:
            reaction = model.reactions.get_by_id(i)
            Universal.add_reaction(reaction.copy())
            reaction.remove_from_model()
```

Now, because of these gaps, the model won't grow.

```
In [3]: model.optimize().f

Out[3]: 2.821531499799383e-12
```

A.10.1 GrowMatch

We will use GrowMatch to add back the minimal number of reactions from this set of "universal" reactions (in this case just the ones we removed) to allow it to grow.

```
In [4]: r = cobra.flux_analysis.growMatch(model, Universal)
        for e in r[0]:
            print(e.id)
```

```
GF6PTA
FBP
MAN6PI_reverse
TKT2_reverse
PGI_reverse
```

We can obtain multiple possible reaction sets by having the algorithm go through multiple iterations.

```
In [5]: result = cobra.flux_analysis.growMatch(model, Universal,
                                                iterations=4)
        for i, entries in enumerate(result):
            print("---- Run %d ----" % (i + 1))
            for e in entries:
                print(e.id)
```

```
---- Run 1 ----
GF6PTA
FBP
MAN6PI_reverse
TKT2_reverse
PGI_reverse
---- Run 2 ----
F6PP
GF6PTA
TALA
MAN6PI_reverse
F6PA_reverse
---- Run 3 ----
GF6PTA
MAN6PI_reverse
TKT2_reverse
F6PA_reverse
PGI_reverse
---- Run 4 ----
F6PP
GF6PTA
FBP
TALA
MAN6PI_reverse
```

A.10.2 SMILEY

SMILEY is very similar to growMatch, only instead of setting growth as the objective, it sets production of a specific metabolite

```
In [6]: r = cobra.flux_analysis.gapfilling.SMILEY(model, "ac_e",
                                                  Universal)
        for e in r[0]:
            print(e.id)
```

```
GF6PTA
MAN6PI_reverse
TKT2_reverse
F6PA_reverse
PGI_reverse
```

A.11 Solver Interface

Each cobrapy solver must expose the following API. The solvers all will have their own distinct LP object types, but each can be manipulated by these functions. This API can be used directly when implementing algorithms efficiently on linear programs because it has 2 primary benefits:

1. Avoid the overhead of creating and destroying LP's for each operation
2. Many solver objects preserve the basis between subsequent LP's, making each subsequent LP solve faster

We will walk though the API with the cglpk solver, which links the cobrapy solver API with GLPK's C API.

```
In [1]: import cobra.test

        model = cobra.test.create_test_model("textbook")
        solver = cobra.solvers.cglpk
```

A.11.1 Attributes and functions

Each solver has some attributes:

solver_name

The name of the solver. This is the name which will be used to select the solver in cobrapy functions.

```
In [2]: solver.solver_name

Out[2]: 'cglpk'

In [3]: model.optimize(solver="cglpk")

Out[3]: <Solution 0.87 at 0x7fd42ad90c18>
```

_SUPPORTS_MILP

The presence of this attribute tells cobrapy that the solver supports mixed-integer linear programming

```
In [4]: solver._SUPPORTS_MILP

Out[4]: True
```

solve

Model.optimize is a wrapper for each solver's solve function. It takes in a cobra model and returns a solution

```
In [5]: solver.solve(model)

Out[5]: <Solution 0.87 at 0x7fd42ad90908>
```

create_problem

This creates the LP object for the solver.

```
In [6]: lp = solver.create_problem(model, objective_sense="maximize")
        lp

Out[6]: <cobra.solvers.cglpk.GLP at 0x3e846e8>
```

solve_problem

Solve the LP object and return the solution status

```
In [7]: solver.solve_problem(lp)

Out[7]: 'optimal'
```

format_solution

Extract a cobra.Solution object from a solved LP object

```
In [8]: solver.format_solution(lp, model)

Out[8]: <Solution 0.87 at 0x7fd42ad90668>
```

get_objective_value

Extract the objective value from a solved LP object

```
In [9]: solver.get_objective_value(lp)

Out[9]: 0.8739215069684909
```

get_status

Get the solution status of a solved LP object

```
In [10]: solver.get_status(lp)

Out[10]: 'optimal'
```

change_variable_objective

change the objective coefficient a reaction at a particular index. This does not change any of the other objectives which have already been set. This example will double and then revert the biomass coefficient.

```
In [11]: model.reactions.index("Biomass_Ecoli_core")

Out[11]: 12

In [12]: solver.change_variable_objective(lp, 12, 2)
         solver.solve_problem(lp)
         solver.get_objective_value(lp)

Out[12]: 1.7478430139369818

In [13]: solver.change_variable_objective(lp, 12, 1)
         solver.solve_problem(lp)
         solver.get_objective_value(lp)

Out[13]: 0.8739215069684909
```

change variable_bounds

change the lower and upper bounds of a reaction at a particular index. This example will set the lower bound of the biomass to an infeasible value, then revert it.

```
In [14]: solver.change_variable_bounds(lp, 12, 1000, 1000)
         solver.solve_problem(lp)

Out[14]: 'infeasible'

In [15]: solver.change_variable_bounds(lp, 12, 0, 1000)
         solver.solve_problem(lp)

Out[15]: 'optimal'
```

change_coefficient

Change a coefficient in the stoichiometric matrix. In this example, we will set the entry for ADP in the ATMP reaction to in infeasible value, then reset it.

```
In [16]: model.metabolites.index("atp_c")

Out[16]: 16

In [17]: model.reactions.index("ATPM")

Out[17]: 10

In [18]: solver.change_coefficient(lp, 16, 10, -10)
         solver.solve_problem(lp)

Out[18]: 'infeasible'

In [19]: solver.change_coefficient(lp, 16, 10, -1)
         solver.solve_problem(lp)

Out[19]: 'optimal'
```

set_parameter

Set a solver parameter. Each solver will have its own particular set of unique paramters. However, some have unified names. For example, all solvers should accept "tolerance_feasibility."

```
In [20]: solver.set_parameter(lp, "tolerance_feasibility", 1e-9)

In [21]: solver.set_parameter(lp, "objective_sense", "minimize")
         solver.solve_problem(lp)
         solver.get_objective_value(lp)

Out[21]: 0.0

In [22]: solver.set_parameter(lp, "objective_sense", "maximize")
         solver.solve_problem(lp)
         solver.get_objective_value(lp)

Out[22]: 0.8739215069684912
```

A.11.2 Example with FVA

Consider flux variability analysis (FVA), which requires maximizing and minimizing every reaction with the original biomass value fixed at its optimal value. If we used the cobra Model API in a naive implementation, we would do the following:

```
In [23]: %%time
         # work on a copy of the model so the original is not changed
         m = model.copy()

         # set the lower bound on the objective to be the optimal value
         f = m.optimize().f
         for objective_reaction, coefficient in m.objective.items():
             objective_reaction.lower_bound = coefficient * f

         # now maximize and minimze every reaction to find its bounds
         fva_result = {}
         for r in m.reactions:
             m.change_objective(r)
             fva_result[r.id] = {
                 "maximum": m.optimize(objective_sense="maximize").f,
                 "minimum": m.optimize(objective_sense="minimize").f
             }

CPU times: user 171 ms, sys: 0 ns, total: 171 ms
Wall time: 171 ms
```

Instead, we could use the solver API to do this more efficiently. This is roughly how cobrapy implementes FVA. It keeps uses the same LP object and repeatedly maximizes and minimizes it. This allows the solver to preserve the basis, and is much faster. The speed increase is even more noticeable the larger the model gets.

```
In [24]: %%time
         # create the LP object
         lp = solver.create_problem(model)

         # set the lower bound on the objective to be the optimal value
         solver.solve_problem(lp)
         f = solver.get_objective_value(lp)
         for objective_reaction, coefficient in model.objective.items():
             objective_index = model.reactions.index(objective_reaction)
```

```
            # old objective is no longer the objective
            solver.change_variable_objective(lp, objective_index, 0.)
            solver.change_variable_bounds(
                lp, objective_index, f * coefficient,
                objective_reaction.upper_bound)

        # now maximize and minimze every reaction to find its bounds
        fva_result = {}
        for index, r in enumerate(model.reactions):
            solver.change_variable_objective(lp, index, 1.)
            result = {}
            solver.solve_problem(lp, objective_sense="maximize")
            result["maximum"] = solver.get_objective_value(lp)
            solver.solve_problem(lp, objective_sense="minimize")
            result["minimum"] = solver.get_objective_value(lp)
            solver.change_variable_objective(lp, index, 0.)
            fva_result[r.id] = result
```

```
CPU times: user 8.28 ms, sys: 25 µs, total: 8.31 ms
Wall time: 8.14 ms
```

A.12 Using the COBRA toolbox with cobrapy

This example demonstrates using COBRA toolbox commands in MATLAB from python through pymatbridge.

```
In [1]: %load_ext pymatbridge
```

```
Starting MATLAB on ZMQ socket ipc:///tmp/pymatbridge-57ff5429-02d9-4e1a-
8ed0-44e391fb0df7
Send 'exit' command to kill the server
...MATLAB started and connected!
```

```
In [2]: import cobra.test
        m = cobra.test.create_test_model("textbook")
```

The model_to_pymatbridge function will send the model to the workspace with the given variable name.

```
In [3]: from cobra.io.mat import model_to_pymatbridge
        model_to_pymatbridge(m, variable_name="model")
```

Now in the MATLAB workspace, the variable name 'model' holds a COBRA toolbox struct encoding the model.

```
In [4]: %%matlab
        model
```

```
model =

            rev: [95x1 double]
       metNames: {72x1 cell}
              b: [72x1 double]
      metCharge: [72x1 double]
              c: [95x1 double]
         csense: [72x1 char]
          genes: {137x1 cell}
    metFormulas: {72x1 cell}
           rxns: {95x1 cell}
        grRules: {95x1 cell}
       rxnNames: {95x1 cell}
    description: [11x1 char]
              S: [72x95 double]
             ub: [95x1 double]
             lb: [95x1 double]
           mets: {72x1 cell}
     subSystems: {95x1 cell}
```

First, we have to initialize the COBRA toolbox in MATLAB.

```
In [5]: %%matlab --silent
        warning('off'); % this works around a pymatbridge bug
        addpath(genpath('~/cobratoolbox/'));
        initCobraToolbox();
```

Commands from the COBRA toolbox can now be run on the model

```
In [6]: %%matlab
        optimizeCbModel(model)
```

```
ans =

           x: [95x1 double]
           f: 0.8739
           y: [71x1 double]
           w: [95x1 double]
        stat: 1
    origStat: 5
      solver: 'glpk'
        time: 3.2911
```

FBA in the COBRA toolbox should give the same result as cobrapy (but maybe just a little bit slower :))

```
In [7]: %time
        m.optimize().f

CPU times: user 0 ns, sys: 0 ns, total: 0 ns
Wall time: 5.48 µs

Out[7]: 0.8739215069684909
```

Appendix B

Distributing Validated Constraint-Based Models in SBML

This document gives a short overview of the most relevant SBML constructs and related concepts to ensure unambiguous representations of constraint-based models. Several source-code snippets of the recent FBC v2 standard are contrasted to the various earlier non-standard approaches to point out specifically what must be changed. It also describes appropriate use of the systems biology ontology (SBO) terms which are relevant to COBRA models, and suggests a naming convention for metabolites, genes, and reactions in these models.

B.1 Systems Biology Ontology Terms and COBRA models

The Systems Biology Ontology (often abbreviated as SBO) allows any major element in an SBML model can be annotated with a controlled vocabulary [CJK+11,

HFS+03]. SBO terms generally are written as 7 digit numbers with a prefix (for example, "SBO:0000632"). However, these terms are only useful as a standard controlled vocabulary if they are used in a consistent manner. Most of the metabolic models created up until now, even for different organisms and by different labs across the world, share many common structural elements. Assigning and using SBO terms to demarcate these similarities should therefore help with model interoperability.

In the context of COBRA models, the following SBO terms have been created to be applied to certain classes of reactions. Most importantly, SBO terms have been assigned to various pseudoreactions, which are modeling constructs and do not represent *in vivo* chemical transformations. Use of these terms are important especially when ensuring models are consistent and mass balanced, because they signal that these particular reactions are purposefully imbalanced. Thus far, the following terms have been assigned: sink reactions (SBO:0000632), exchange reactions (SBO:0000627), demand reactions (SBO:0000628), and biomass production (SBO:0000629). Similarly, an SBO term has been applied to the ATP maintenance (SBO:0000630), which is a special reaction in that represents the non-growth associated maintenance energy costs.

Finally, to distinguish the parameters used by COBRA models from other types of models (such as Michaelis constants for kinetic modeling, for example), SBO terms have been assigned to indicate parameters are used as upper and lower flux bounds. Often, bounds are set to a large positive value, a large negative value, or 0 to indicate reaction directionality without a particular numeric constraint. For these reactions, the SBO term default flux bound (SBO:0000626) is appropriate. Other bounded fluxes should simply use flux bound (SBO:0000625).

B.2 Strict Models

The fbc standard defines a "strict" mode, which is applicable to the common formulation used by M models with flux balance analysis. This mode should always be used when distributing genome-scale models. Most constraint-based modeling tools (such as COBRApy) will assume that models passed into them fulfill the extra requirements imposed as a consequence of using the strict flag. This flag is placed on the model element.

```
<sbml xmlns= ... >
  <model fbc:strict="true" id="demo_model">
    ...
  </model>
</sbml>
```

B.3 Defining Flux Bounds

In FBC version 2, flux bounds are listed as parameters, and reactions

```
<listOfParameters>
  <parameter id="lb" constant="true" value="-1000"
      ... sboTerm="SBO:0000626"/>
  <parameter id="ub" constant="true" value="1000"
      ... sboTerm="SBO:0000626"/>
</listOfParameters>
<listOfReactions>
  <reaction id="R_ACKr" ... fbc:lowerFluxBound="lb" fbc:upperFluxBound="ub">
    ...
  </reaction>
```

```
</listOfReactions>
```

By contrast, previous versions of SBML (specifically L2V4) defined flux bounds as kinetic laws, with the convention that the specific names be used for their identifiers. However, these bounds were not always present, and did not always have the exact correct identifier.

```
<!-- OLD WAY DON'T DO THIS -->
<reaction id="R_DM_5DRIB" ...>
  <listOfReactants>
    ...
  </listOfReactants>
  <kineticLaw>
    <math xmlns="http://www.w3.org/1998/Math/MathML">
      <ci> FLUX_VALUE </ci>
    </math>
    <listOfParameters>
      <parameter id="LOWER_BOUND" value="0" units= ... />
      <parameter id="UPPER_BOUND" value="999" units= ... />
      <parameter id="OBJECTIVE_COEFFICIENT" value="0"/>
      <!-- for some reason the solution was stored too -->
      <parameter id="FLUX_VALUE" value= ... />
    </listOfParameters>
  </kineticLaw>
</reaction>
```

The short lived fbc1 specification also had its own way of defining flux bounds, which is also deprecated and should no longer be used.

```
<!-- OLD WAY DON'T DO THIS -->
<fbc:listOfFluxBounds>
```

```
  <fbc:fluxBound fbc:reaction="R_sink" fbc:operation="greaterEqual"
                 fbc:value="0"/>
  <fbc:fluxBound fbc:reaction="R_sink" fbc:operation="lessEqual"
                 fbc:value="999999"/>
</fbc:listOfFluxBounds>
<listOfReactions>
  <reaction id="R_sink" ... >
    ...
  </reaction>
</listOfReactions>
```

B.4 Reaction Reversibility

It turns out that SBML and COBRA models have slightly different definitions of reversibility, with a subtle difference. SBML defines an irreversible reaction as one with strictly positive flux, while COBRA modelers tend to think of irreversible reactions as only proceeding in one direction. The difference becomes apparent when considering a reaction which proceeds only in reverse. In SBML, this would be considered a reversible reaction. To handle this, the reversibility flag which is written should use the SBML convention, and define reactions as reversible if they can carry any negative flux. However, when reading in a model, the flux bounds should be used to determine reversibility using the COBRA definitions, as using the reversibility flag alone will result in an incorrect model [EAB+15]. Because strict fbc requires the flux bound parameters (which take precedence) to be specified, the reversibility flag can always be ignored.

```
<listOfParameters> ...
```

```
  <parameter id="lb" constant="true" value="-1000" ... />
  <parameter id="ub" constant="true" value="0" ... />
</listOfParameters>
<listOfReactions>
```

B.5 Gene Reaction Rules

The correct way to specify the relationship between genes and reactions in fbc v2 is with a tree which expresses the boolean relationship. Generally, AND relationships are thought of as complexes, and relationships represent alternate complexes. Therefore, expressing the boolean expression in disjunctive normal form is the most biologically relevant, as it lists all the individual complexes which can catalyze the reaction. However, this conversion is by no means required, and any boolean expression of AND and OR operations with genes are allowed.

The method used to express this in SBML is analogous to how reactions refer through metabolites through SBML species. Genes are listed in one part of the document, and the gene reaction rule refers to them by id.

```
<fbc:listOfGeneProducts>
  <fbc:geneProduct fbc:id="G_b2925"
      fbc:label="b2925" fbc:name="fbaA"/>
  <fbc:geneProduct fbc:id="G_b1773" ... />
  <fbc:geneProduct fbc:id="G_b2097" ... />
</fbc:listOfGeneProducts>
<listOfReactions>
  <reaction id="R_FBA"... >
    <fbc:geneProductAssociation>
```

```
      <fbc:or>
        <fbc:geneProductRef fbc:geneProduct="G_b1773"/>
        <fbc:geneProductRef fbc:geneProduct="G_b2097"/>
        <fbc:geneProductRef fbc:geneProduct="G_b2925"/>
      </fbc:or>
    </fbc:geneProductAssociation>
    ...
  </reaction>
</listOfReactions>
```

The old way of doing this was simply to specify a text string encoding a boolean expression in the notes section. There were several issues with this structure. Most importantly, there was no validation of these boolean strings, which were frequently malformed.

```
<!-- OLD WAY DON'T DO THIS -->
<reaction id="R_13PPDHr" ...>
  <notes>
    <html:p>GENE_ASSOCIATION: ( Gmet_1046  or  Gmet_1053 )</html:p>
    <html:p>PROTEIN_ASSOCIATION: ( Adh1 ) or ( Adh2 )</html:p>
  </notes>
  ...
</reaction>
```

B.6 Biomass Objective Reactions

While fbc v2 allows specification of multiple objectives, many software tools do not in fact support this, so this feature should not be relied on. Additionally, the format allows specification of multiple reactions with an objective. However, a

well-formed model will only have a single active biomass reaction at any one time. By comparison, the older format encoded the objective coefficient in the same place it encoded flux bounds.

```
<fbc:listOfObjectives fbc:activeObjective="obj_max">
  <fbc:objective fbc:type="maximize" fbc:id="obj_max">
    <fbc:listOfFluxObjectives>
      <fbc:fluxObjective fbc:coefficient="1"
                         fbc:reaction="Gm_biomass_GS15_WT_79p20M"/>
    </fbc:listOfFluxObjectives>
  </fbc:objective>
</fbc:listOfObjectives>
```

B.7 Conventions for Identifiers

Defining meaningful identifiers can be difficult because of the limited selection of characters that can be used in SBML. Because identifiers cannot start with numbers, all SBML id's should have prefixes. The following convention is recommended: "G_" for gene products, "M_" for species (metabolites), and "R_" for reactions. For example, instead of "3PG" (which would be an invalid id in SBML), "M_3PG" should be used for 3-phosphoglycerate. Metabolite and reaction id standardization remains a general challenge for the field. However, in the case of genes, the problem already been partially solved by the NCBI gene system. Therefore, in the case of genes, more defined conventions can be defined. Gene identifiers should use the "G_" prefix followed by the NCBI gene id. The NCBI gene id can contain "." (which is invalid in SBML), which should be escaped by "_SBML_DOT_".

Appendix C

Model-driven elucidation of transcriptional regulatory network in bacteria - Supplementary Information

C.1 Advantages of ChIP-exo over previous methods

Binding peaks detected with ChIP-exo for NtrC, Nac, RpoN, and RpoD had average widths of 57.2, 34.6, 33.7, and 50.1 bp, respectively, with low variations in peak width (Figure S21). Another advantage of the ChIP-exo method is that strong binding is reflected in an increased peak height (number of reads) while not broadening the peak, while ChIP-chip and ChIP-seq methods have a tendency to show broader

peaks for stronger binding sites. This property helps locate the genomic sequences that a TF recognizes and binds to with improved resolution (Figure S4).

Notably, the sharpness of ChIP-exo binding peaks makes it possible, for the first time, to detect binding sites upstream of sRNAs. In *E. coli* K-12 MG1655, there are about 81 annotated sRNAs. Their sizes range from 53 to 436 bp with the average of 137.1 bp. Binding peaks from two previous methods are much wider than the sRNA length, and, in many cases, sRNAs are located close to neighboring genes. Thus, it has been technically difficult to distinguish TF binding sites for sRNAs. With an improved resolution, ChIP-exo overcomes this issue, enabling direct measurement of TF binding upstream of sRNAs. For instance, *spf* is 109 bp long, and binding of Nac and RpoD was identified by ChIP-exo method (Figure S22). The binding peak of RpoD was found near the 5' end of *spf* gene, and was clearly separated from Nac binding peak. These observations were further supported with sequence motif analysis (Figure S22).

Thus, ChIP-exo shows advantages over two established ChIP methods. Improved resolution and sensitivity of this method contributed to more accurate annotation of binding regions of TFs and σ-factors in this study, further distinguishing spatial binding patterns between TF and associated σ-factors, which is discussed in detail below.

C.2 Locally acting TFs regulated by Nac.

The Nac regulon includes a number of mostly locally acting TFs, some of which are known to be related carbon metabolism or in both carbon/nitrogen metabolism, such as *cynR* (cyanate binding transcriptional activator), *csiR* (carbon starvation

induced regulator), *sfsB* (maltose metabolism related regulator), *gutM* (glucitol regulator), *ebgR* (evolved β-galactosidase repressor), *tdcA* (threonine and serine transcriptional regulator), *deoR* (deoxyribose regulator), *allR* (allantoin repressor), *caiF* (carnitine regulator), *lrp* (leucine-responsive regulatory protein), *lysR* (lysine regulator), *feaR* (phenylethylamine regulator), *xapR* (xanthosine/deoxyinosine transcriptional regulator), *asnC* (asparagine regulator), and *metR* (methionine biosynthesis related regulator).

C.3 Supplementary Experimental Procedures

C.3.1 Prediction of activation conditions for transcription factor

In order to simulate ME model [OLC$^+$13] for *in silico E. coli* growth under various conditions, candidate nitrogen sources were chosen from simulation with M model [OCN$^+$11] which shares the same metabolites with ME model. There are 1039 cytoplasmic metabolites in *iJO*1366 M model, and among them, 620 cytoplasmic metabolites have more than one nitrogen molecules. From those metabolites, 178 nitrogen containing cytoplasmic metabolites with exchange reaction in the model were preserved. With glucose as sole carbon source with -10 mmol/gDW/hr uptake rate, and each nitrogen-containing metabolite with exchange reaction with -10 mmol/gDW/hr uptake rate, M model was used in simulation to decide if the nitrogen containing molecule supports *in silico* growth (Figure S1A). From the set of *in silico* growth with 178 nitrogen containing metabolites, threshold 0.5 mmol/gDW/hr was used to decide growth and non-growth (Figure S1B), which resulted in 93 candidate nitrogen sources,

which support *in silico* growth with glucose.

The ME model was used to simulate expression of genes under 93 candidate nitrogen sources with -10 mmol/gDW/hr and glucose for carbon source with -10 mmol/gDW/hr. For each simulation under an alternate nitrogen source, expression was compared to simulated growth on ammonium, with a cutoff of a factor of 2 set to identify predicted sets of genes. The hypergeometric enrichment of each known TF regulon taken from Ecocyc [KMPG+13] in each of these differential gene sets was used to predict the likelihood of transcription factor activity under each condition (Figure S1C, Figure S1D). For NtrC and Nac, 24 conditions and 2 conditions respectively resulted in predictions of TF activity. (Figure S1E).

C.3.2 Bacterial strains, media, and growth conditions

All strains used in this study are *E. coli* K-12 MG1655 and its derivatives for tandem epitopes. The *E. coli* strains harboring *ntrC*-8myc and *nac*-8myc were generated as described previously [CKP06a]. M9 minimal media [KHQ+12] was used for ammonia and W2 minimal media [PCI+95] was used for nitrogen-limiting conditions. For nitrogen-limiting conditions, 0.2% (w/v) glutamine, cytidine, and cytosine were added for alternative nitrogen sources. M9 or W2 minimal media was supplemented with 0.2% glucose (w/v) and 1 ml trace element solution (100X) containing 1 g EDTA, 29 mg $ZnSO_4.7H_2O$, 198 mg $MnCl_2.4H_2O$, 254 mg $CoCl_2.6H_2O$, 13.4 mg $CuCl_2$, and 147 mg $CaCl_2$.

Glycerol stocks of *E. coli* strains were inoculated into M9 or W2 minimal media and cultured overnight at 37°C with constant agitation. Cultures were then diluted 1:100 into 50 mL of fresh minimal media and cultured at 37 °C to mid-log

phase ($OD_{600} \approx 0.5$ for ammonia, glutamine, and cytidine, and $OD_{600} \approx 0.25$) before harvest.

C.3.3 ChIP-exo experiment

To identify TF and σ-factor binding maps *in vivo*, we isolated the DNA bound to NtrC, Nac, and RpoN and RpoD from formaldehyde cross-linked *E. coli* cells by chromatin immunoprecipitation (ChIP) with the specific antibodies that specifically recognizes myc tag (9E10, Santa Cruz Biotechnology) and RpoD (2G10, Neoclone) and RpoN (6RN3, Neoclone) subunits of RNA polymerase complex, respectively, and Dynabeads Pan Mouse IgG magnetic beads (Invitrogen) followed by stringent washings as described previously [CKK$^+$14]. ChIP materials (chromatin-beads) were used to perform on-bead enzymatic reactions of the ChIP-exo method [RP12a] with following modifications. Briefly, the sheared DNA of chromatin-beads was repaired by the NEBNext End Repair Module (New England Biolabs) followed by the addition of a single dA overhang and ligation of the first adaptor (5'-phosphorylated) using dA-Tailing Module (New England Biolabs) and NEBNext Quick Ligation Module (New England Biolabs), respectively. Nick repair was performed by using PreCR Repair Mix (New England Biolabs). Lambda exonuclease- and $RecJ_f$ exonuclease-treated chromatin was eluted from the beads and the protein-DNA cross-link was reversed by overnight incubation at 65°C. RNAs- and Proteins-removed DNA samples were used to perform primer extension and second adaptor ligation with following modifications. The DNA samples incubated for primer extension as described previously [RP12a] were treated with dA-Tailing Module (New England Biolabs) and NEBNext Quick Ligation Module (New England Biolabs) for second adaptor ligation. The DNA

sample purified by GeneRead Size Selection Kit (Qiagen) was enriched by polymerase chain reaction (PCR) using Phusion High-Fidelity DNA Polymerase (New England Biolabs). The amplified DNA samples were purified again by GeneRead Size Selection Kit (Qiagen) and quantified using Qubit dsDNA HS Assay Kit (Life Technologies). Quality of the DNA sample was checked by running Agilent High Sensitivity DNA Kit using Agilent 2100 Bioanalyzer (Agilent) before sequenced using MiSeq (Illumina) in accordance with the manufacturer's instructions. Each modified step was also performed in accordance with the manufacturer's instructions. ChIP-exo experiments were performed in biological duplicate.

C.3.4 RNA-seq expression profiling

Three milliliters of cells from mid-log phase culture were mixed with 6 ml RNAprotect Bacteria Reagent (Qiagen). Samples were mixed immediately by vortexing for 5 seconds, incubated for 5 minutes at room temperature, and then centrifuged at 5000$\times g$for 10 minutes. The supernatant was decanted and any residual supernatant was removed by inverting the tube once onto a paper towel. Total RNA samples were then isolated using RNeasy Plus Mini kit (Qiagen) in accordance with the manufacturer's instruction. Samples were then quantified using a NanoDrop 1000 spectrophotometer (Thermo Scientific) and quality of the isolated RNA was checked by running RNA 6000 Pico Kit using Agilent 2100 Bioanalyzer (Agilent).

Paired-end, strand-specific RNA-seq was performed using the dUTP method [LYA$^+$10] with the following modifications. The ribosomal RNAs were removed from 2 μg of isolated total RNA with Ribo-Zero rRNA Removal Kit (Epicentre) in accordance with the manufacturer's instruction. Subtracted RNA was fragmented for 2.5 min at

70 °C with RNA Fragmentation Reagents (Ambion), and then fragmented RNA was recovered with ethanol precipitation. Random primer (3 μg) and fragmented RNA in 4 μl was incubated in 5 μl total volume at 70 °C for 10 minutes, and cDNA or the first strand was synthesized using SuperScript III first-strand synthesis protocol (Invitrogen). The cDNA was recovered by phenol-chloroform extraction followed by ethanol precipitation. The second strand was synthesized from this cDNA with 20 μl of fragmented cDNA:RNA, 4 μl of 5× first strand buffer, 30 μl of 5× second strand buffer, 4 μl of 10 mM dNTP with dUTP instead of dTTP, 2 μl of 100 mM DTT, 4 μl of *E. coli* DNA polymerase (Invitrogen), 1 μl of *E. coli* DNA ligase (Invitrogen), 1 μl of *E. coli* RNase H (Invitrogen) in 150 μl of total volume. This reaction mixture was incubated at 16 °C for 2 hours, and fragmented DNA was recovered with PCR clean-up kit (QIAGEN) and eluted in 30 μl of nuclease-free water. The fragmented DNA was end-repaired with End Repair Kit (New England Biolabs), and dA-tailed with dA-Tailing Kit (New England Biolabs), and then ligated with 7.5 μg of DNA adaptor mixture with Quick Ligation Kit (New England Biolabs). The adaptor-ligated DNA was size-selected to removed un-ligated adaptors with GeneRead Size Selection Kit (QIAGEN), and treated with 1 U of USER enzyme (New England Biolabs) in 30 μl of total volume, and incubated at 37 °C for 15 minutes followed by 5 minutes at 95 °C. The USER-treated DNA was amplified by PCR to generate sequencing library for Illumina sequencing. The samples were sequenced using MiSeq (Illumina) in accordance with the manufacturer's instructions. All RNA-seq experiments were performed in biological duplicate.

C.3.5 Peak calling for ChIP-exo dataset

Sequence reads generated from ChIP-exo were mapped onto the reference genome (NC_000913.2) using bowtie [LTPS09] with default options to generate SAM output files. MACE program (https://code.google.com/p/chip-exo/) was used to define peak candidates from biological duplicates for each experimental condition with sequence depth normalization. To reduce false-positive peaks, peaks with signal-to-noise (S/N) ratio less than 1.5 were removed. The noise level was set to the top 5% of signals at genomic positions because top 5% makes a background level in plateau (Figure S20A) and top 5% intensities from each ChIP-exo replicates across conditions correlate well with the total number of reads (Figure S20B). The calculation of S/N ratio resembles the way to calculate ChIP-chip peak intensity where IP signal was divided by Mock signal. Then, each peak was assigned to the nearest gene. Genome-scale data were visualized using MetaScope (http://systemsbiology.ucsd.edu/Downloads/MetaScope).

C.3.6 Classification of regulatory and non-regulatory binding sites

For binding sites of NtrC, Nac, RpoN and RpoD, a binding site was categorized as regulatory if it is located with 300 bp upstream of a target gene, and as non-regulatory if else. Non-regulatory regions cover intragenic region without downstream genes and intergenic region without promoter nearby.

C.3.7 Motif search from ChIP-exo peaks

The sequence motif analysis for TFs and σ-factors was performed using the MEME software suite [BBB+09]. For NtrC, Nac, and RpoN, sequences in binding regions were extracted from the reference sequence (NC_000913.2). For RpoD, sequences were extended by 20 bp away from target genes, because only -10 box was found without that extension. MEME was run for regulatory bindings for all conditions, regulatory bindings for at least one condition, and non-regulatory bindings of NtrC, Nac, RpoN and RpoD.

C.3.8 Calculation of differentially expressed gene

Sequence reads generated from RNA-seq were mapped onto the reference genome (NC_000913.2) using bowtie [LTPS09] with the maximum insert size of 1000 bp, and 2 maximum mismatches after trimming 3 bp at 3' ends. SAM files generated from bowtie, then, were then used for Cufflinks (http://cufflinks.cbcb.umd.edu/) [TWP+10] to calculate fragments per kilobase of exon per million fragments (FPKM) and Cuffdiff to calculate differential expression. Cufflinks and Cuffdiff were run with default options with library type of dUTP RNA-seq with default normalization method (classic-fpkm). From Cuffdiff output, genes with differential expression with log2 fold change > 1.0 and q-value < 0.01 were considered as differentially expressed genes. Genome-scale data were visualized using MetaScope (http://systemsbiology.ucsd.edu/Downloads/MetaScope).

C.3.9 COG functional enrichment

NtrC and Nac regulons were categorized according to their annotated clusters of orthologous groups (COG) category. Functional enrichment of COG categories in NtrC and Nac target genes was determined by performing hypergeometric test, and *p-value* < 0.05 was considered significant.

C.3.10 Measuring growth rate and glucose uptake rates on different nitrogen sources

M9 minimal media [KHQ$^+$12] was used for ammonia and W2 minimal media [PCI$^+$95]) was used for glutamine, cytidine, and cytosine as nitrogen-limiting conditions. For nitrogen-limiting conditions, 0.2% (w/v) glutamine, cytidine, and cytosine were added. M9 or W2 minimal media was supplemented with 0.2% glucose (w/v) and 1 ml trace element solution (100X) containing 1 g EDTA, 29 mg $ZnSO_4.7H_2O$, 198 mg $MnCl_2.4H_2O$, 254 mg $CoCl_2.6H_2O$, 13.4 mg $CuCl_2$, and 147 mg $CaCl_2$.

A glycerol stock of *E. coli* K-12 MG1655strain was inoculated into fresh M9 or W2 minimal media and cultured overnight at 37°C with constant agitation. Cultures were then diluted 1:200 into 100 mL of fresh minimal media and cultured at 37 °C to late-log phase, and were sampled 6 or 7 time during early to mid-log phase. Optical density at $OD_{600}$was measured to get growth rates under different nitrogen sources, and glucose uptake was measured by HPLC for each time point. Growth and glucose uptake measurement was performed by biological triplicates. Measured growth rates for ammonia, glutamine, cytidine, and cytosine were 0.860, 0.839, 0.828, and 0.650 mmol/gDW/hr, and measured glucose uptake rates were -8.86, -7.68, -7.04, and -6.55 mmol/gDW/hr.

C.3.11 FBA analysis with glucose uptake rates

FBA analysis was performed with *i*JO1366 *E. coli* metabolic model [OCN+11] and COBRApy [ELPH13]. For experimental nitrogen sources, the simulation was executed with measured glucose uptake rates. For a parameter of nitrogen uptake rate, an unspecified uptake rate with a lower bound of -10 mmol/gDW/hr was used ammonia and cytosine, because they were all used less after optimization and this parameter calculated close growth rates to measured rates. For cytidine, the uptake rate was set to -2.982 mmol/gDW/hr to match *in vivo* growth rate to the measured one, otherwise the model chose to uptake as much as cytidine available and to generate much higher and unrealistic growth rate. This is because cytidine could be utilized as a carbon and energy source. Simulated flux values were then normalized by the growth rate for each condition.

For 27 nitrogen sources that have evidences of supporting growth in *in silico* simulation and experiments, there was no measured growth rate or glucose uptake rates for those nitrogen sources. Thus the uptake rate of -10 mmol/gDW/hr was used for glucose uptake rate, and the uptake rates of -10 mmol/gDW/hr divided by the number of nitrogen molecules for nitrogen sources were used for alternative nitrogen sources. Similarly, all simulated flux values were then normalized by the growth rates for each simulated condition.

C.3.12 Comparison of *in silico* growth with experimental evidence

In order to compare *in silico* growth prediction with experimental evidences, evidences were compiled from the literature and the public database [Rei03]. There

were 23 nitrogen sources that were confirmed to support *E. coli* growth from the literature. From the biology data from Ecocyc, there are growth test results for 95 nitrogen sources, 18 out of which were removed because of inconsistency in replicates. The criteria was giving 1 for growth call, 0.5 for low growth call, and -1 for no growth, summing scores from 4 biological replicates, and filtering out any case with a score from summation for 4 replicates for each nitrogen source between, but not including, -2 and 2. There were 77 nitrogen sources with consistent results from biology, and 33 of them were removed because they are not in *i*JO1633 model. The remaining 44 tested nitrogen sources included 23 nitrogen sources from the literature. Out of 44 nitrogen sources, 27 of them agreed in supporting growth *in vivo* and in experiments, 10 of them agreed in not supporting growth in both, and 7 of them were predicted to support growth in simulation, but experimental proved otherwise.

C.3.13 Conservation analysis of nitrogen-related genes

Gene annotation of 286 species and strains ranging from Escherichia to archaea, were obtained from the SEED server (http://theseed.org) and ortholog calculation to *E. coli* K-12 MG1655 was performed on RAST (Rapid Annotation using Subsystem Technology) server [ABB+08]. From RAST output, orthologous genes with bi-directional hits were only retained. Conservation level of 6 genes in transcriptional regulation, 3 genes in post-translational regulation, and 7 genes in metabolic enzymes in TCA cycle were calculated from orthologs retained from RAST output.

C.4 Supplementary Figures

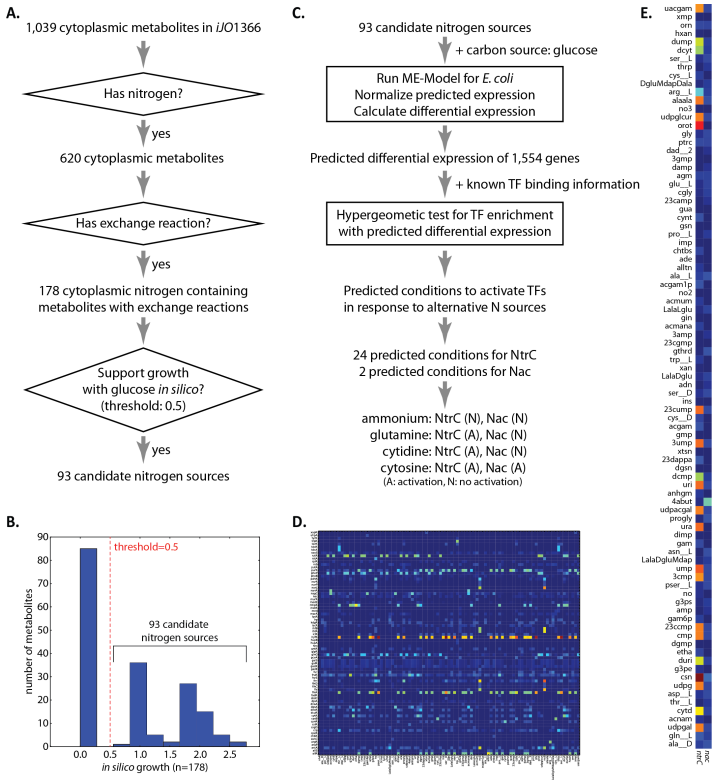


Figure C.1: Flowchart of model-driven prediction for activation conditions for nitrogen-responsive transcription factors. (A) 93 candidate nitrogen sources were selected from 1039 cytoplasmic metabolites from iJO1366 model with simulation of in silico growth. (B) Histogram of *in silico* growth rates for 178 nitrogen-containing metabolites. (C) Calculation of activation conditions for TFs under different conditions with 93 candidate nitrogen sources with hypergeometric test on differentially expressed gene sets and known TF binding information. (D) Heatmap of p-values of hypergeometric tests for 93 nitrogen sources and TFs. (E) Heatmap of p-values of hypergeometric tests for NtrC and Nac under different nitrogen sources. NtrC was predicted to be active under 24 conditions, and Nac was predicted to be active under 2 conditions. Abbreviations for metabolites are described in Table S1.

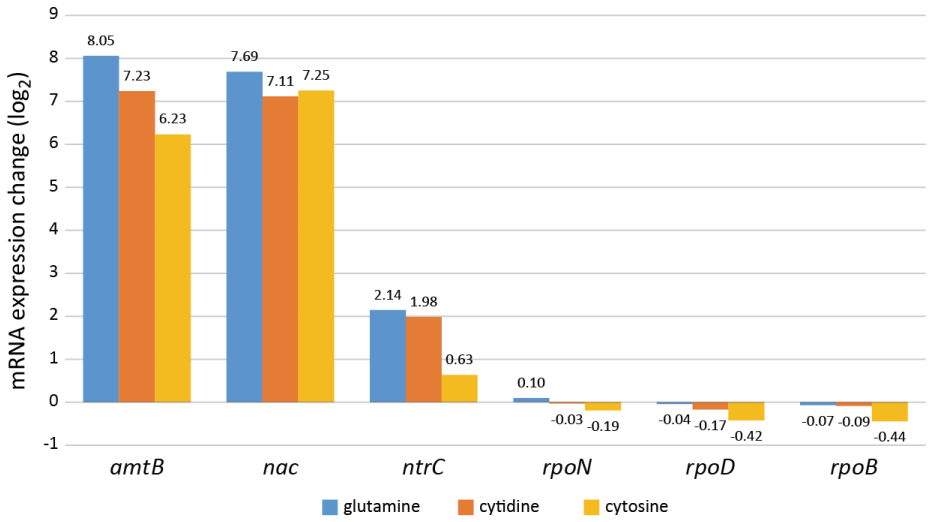


Figure C.2: **Expression changes of *amtB*,*nac*, *ntrC* *rpoN*, *rpoD*, and *rpoB* under glutamine, cytidine, and cytosine, compared to ammonia.**

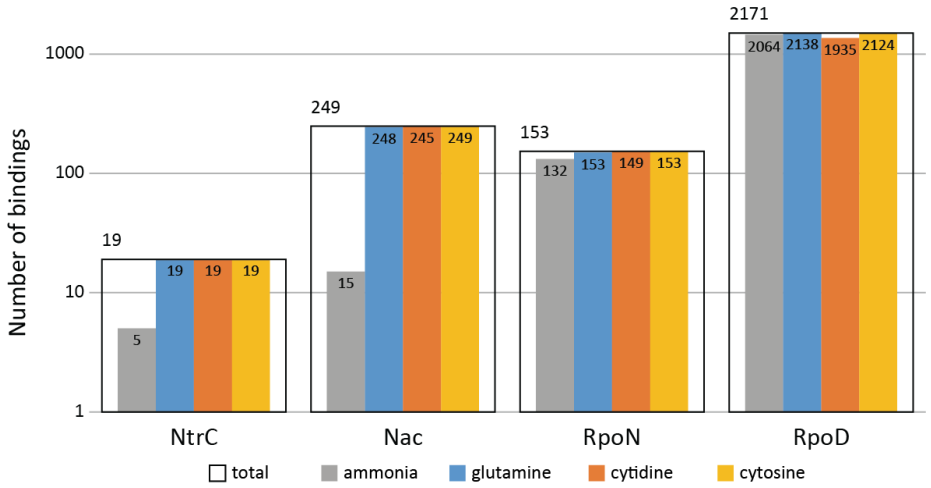


Figure C.3: Number of binding sites (19, 249, 153, and 2171) for NtrC, Nac, RpoN, and RpoD, respectively, identified under the four experimented conditions. Binding sites for NtrC and Nac increased, indicating activation of those TFs.

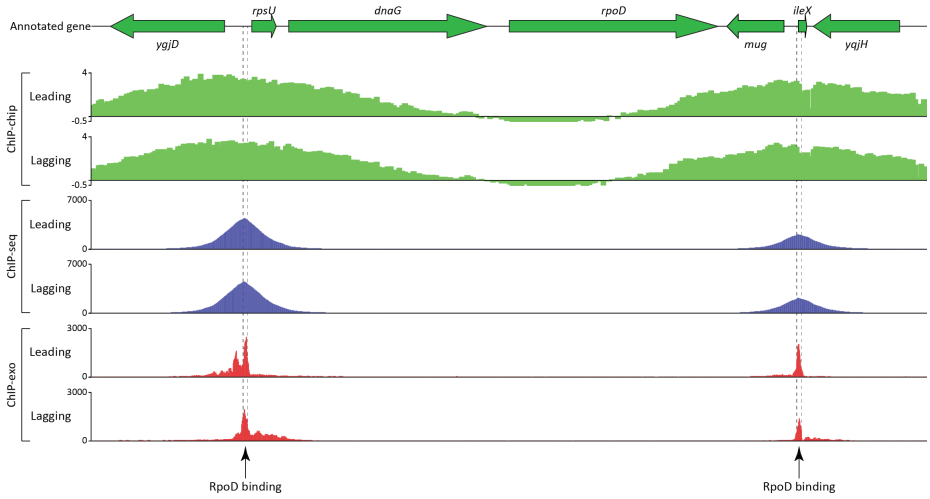


Figure C.4: **Comparison of ChIP methods: ChIP-chip, ChIP-seq, and ChIP-exo.** Regions upstream of *rpsU-dnaG-rpoD* and upstream of *ileX/mug* are shown with binding peaks detected with ChIP-chip (two upper lanes), with ChIP-seq (two middle lanes), and with ChIP-exo (two bottom lanes).

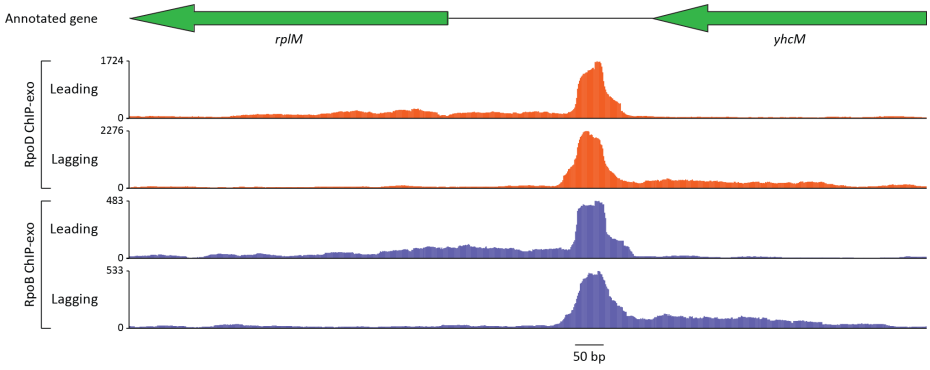


Figure C.5: **ChIP-exo bindings of RpoD and RpoB upstream of *rplM*.**

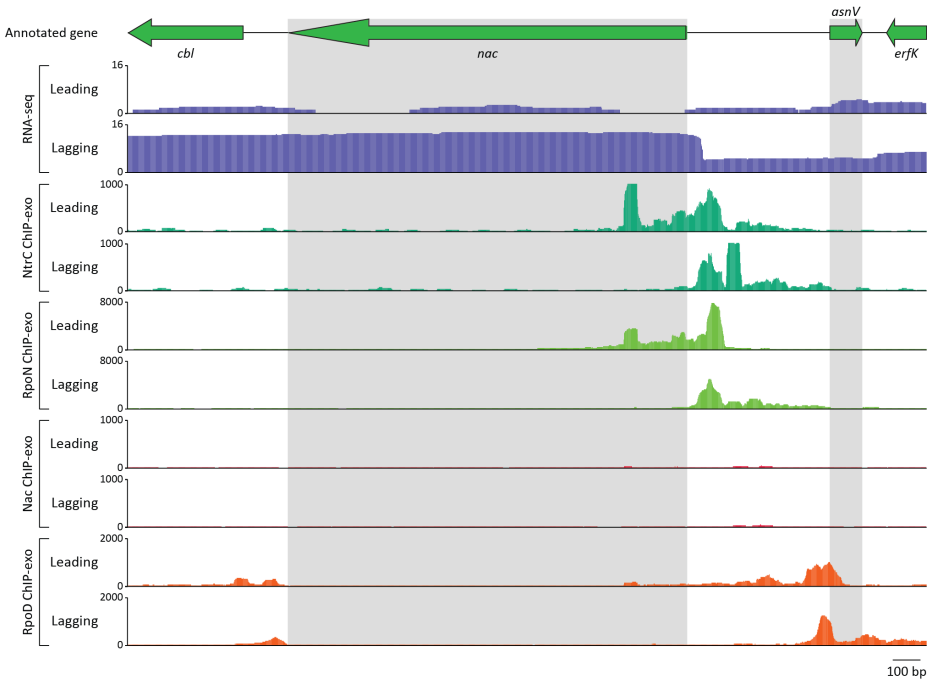


Figure C.6: **ChIP-exo bindings of NtrC and RpoN upstream of *nac*, but no binding of Nac.** Bindings of NtrC and associated RpoN were detected where transcription of *nac* started. However, no Nac binding on *nac* promoter region was observed. The RpoD binding shown seems associated with transcription of *asnV*.

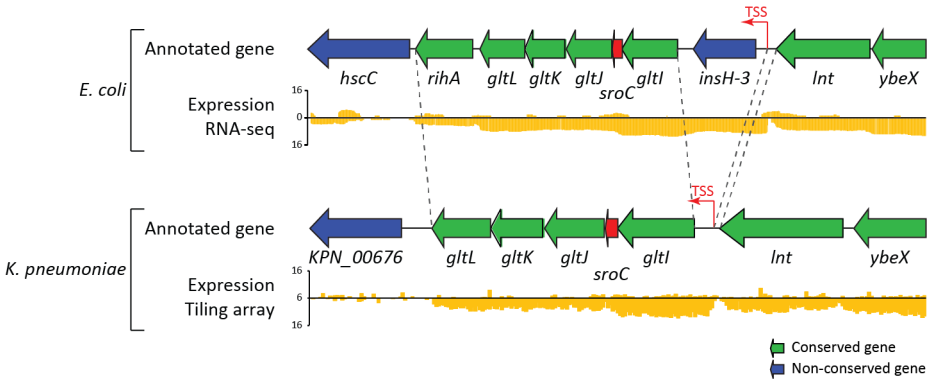


Figure C.7: **Conservation comparison of *gltIJKL* between** *E. coli* and *K. pneumoniae*. Genomic regions near gltIJKL are well conserved between *E. coli* and *K. pneumoniae*. Upstream regions, including *lnt* and *ybeX*, are fairly conserved, except that one insertion element, *insH-3*, is located between gltI and lnt only in *E. coli*. TSSs upstream of *insH-3* in *E. coli* and upstream of *gltI* in *K. pneumoniae* were observed [KHQ+12]. Based on analysis of paired-end sequence reads in *E. coli*, it seems there is a longer transcript starting from *insH-3* at least to *gltL*, suggesting a new TU, *insH-3-gltI-sroC-gltJKL*.

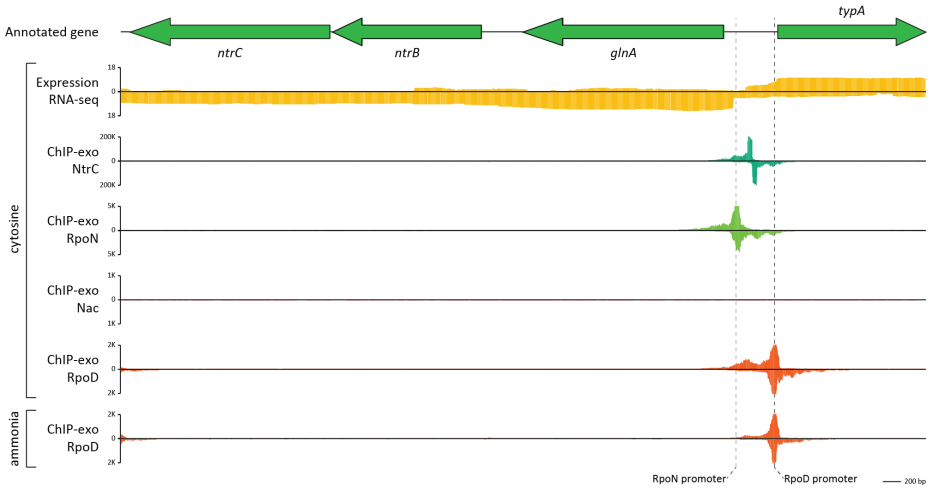


Figure C.8: **ChIP-exo bindings of NtrC, RpoN, and RpoD upstream of *glnA*.** Binding of RpoN accompanied with NtrC binding was observed upstream TUs starting from *glnA*. Similarly, RpoD binding was identified as well, however no Nac binding was found.

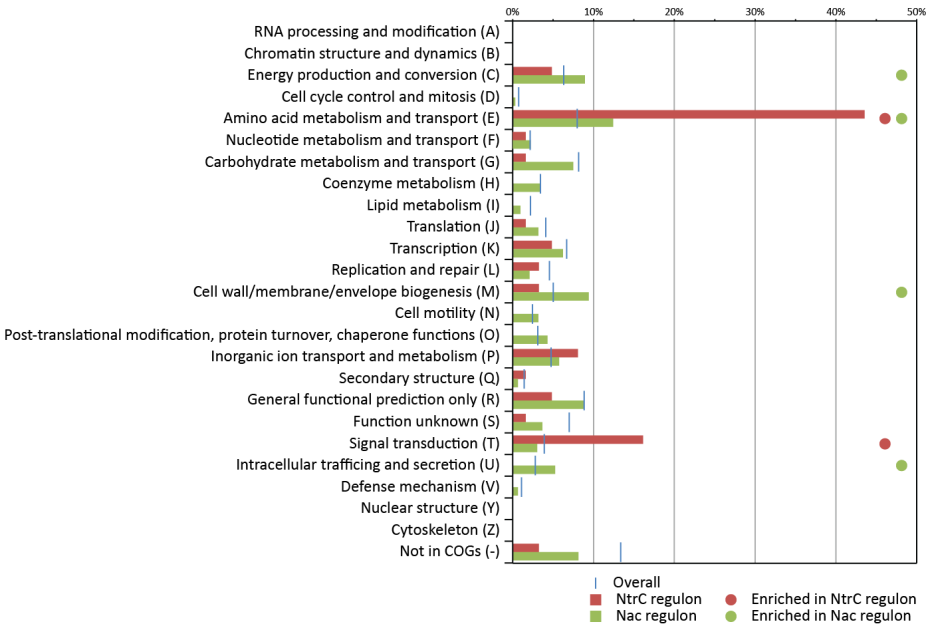


Figure C.9: **COG analysis on NtrC and Nac regulons.**NtrC regulon has functions enriched in "Amino acid metabolism and transport" and "Signal transduction" categories (Hypergeometric test p-value < 0.05). Nac regulon has enriched functions in "Energy production and conversion", "Amino acid metabolism and transport", "Cell wall/membrane/envelope biogenesis", and "Intracellular trafficking and secretion" categories (p-value < 0.05).

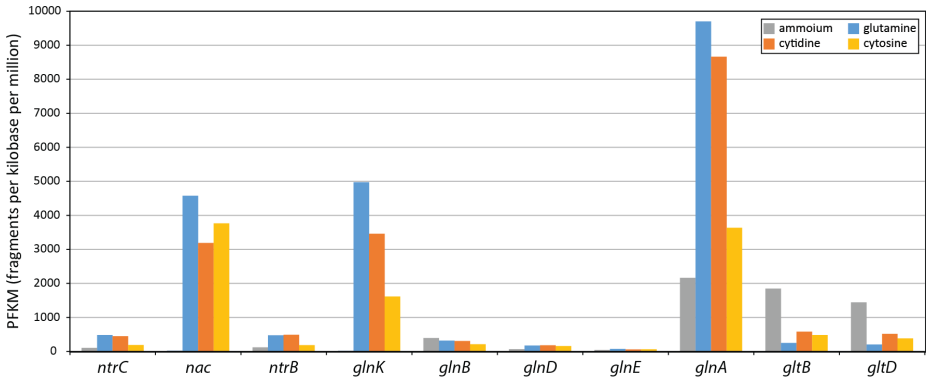


Figure C.10: **Gene expression in mRNA level (PFKM) of genes in Ntr regulatory cascade under ammonia, glutamine, cytidine, and cytosine.** Expression of *ntrC*, *nac*, *ntrB*, *glnK*, *glnD*, *glnE*, and *glnA* was up-regulated on alternative nitrogen sources, while expression of *gltB*, and *gltD* was down-regulated.

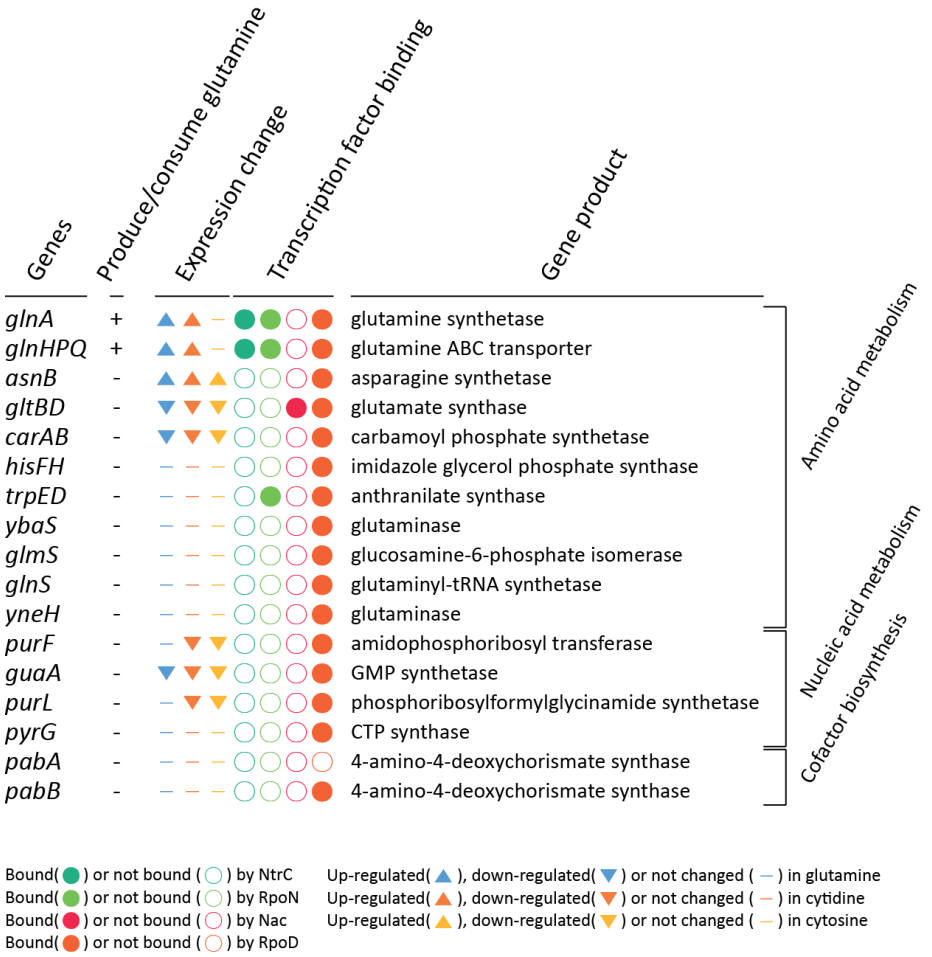


Figure C.11: **Gene expression changes of glutamine-related enzymatic genes and TF binding upstream of those genes.** Genes that produce glutamine were up-regulated, whereas genes that consume glutamine were not changed or down-regulated.

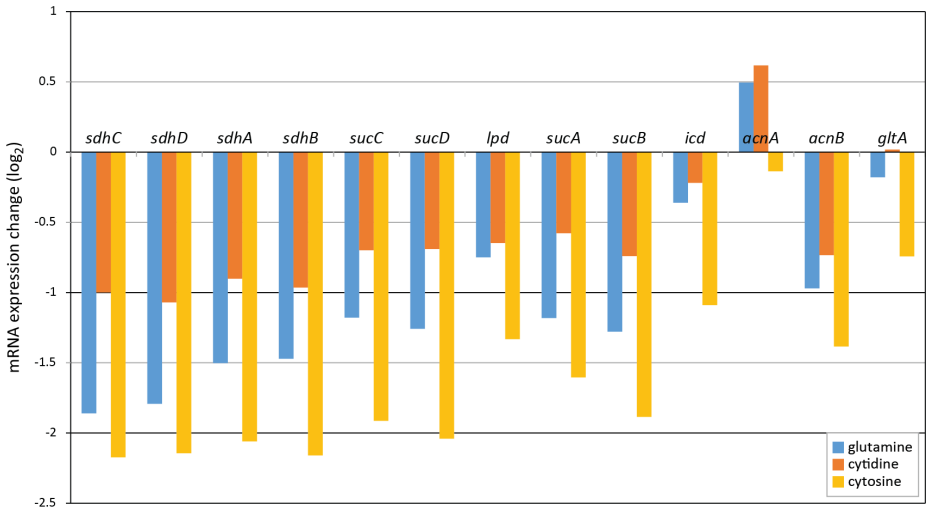


Figure C.12: **Gene expression changes in mRNA level of genes in TCA cycle on glutamine, cytidine, and cytosine when compared to gene expression on ammonia.** Genes upstream of α-ketoglutarate in TCA cycle, *sdhDC*, *sdhAB*, *sucCD*, *lpd*, and *sucAB*, were more repressed than genes downstream of α-ketoglutarate, *icd*, *acnA*, *acnB*, and *gltA*.

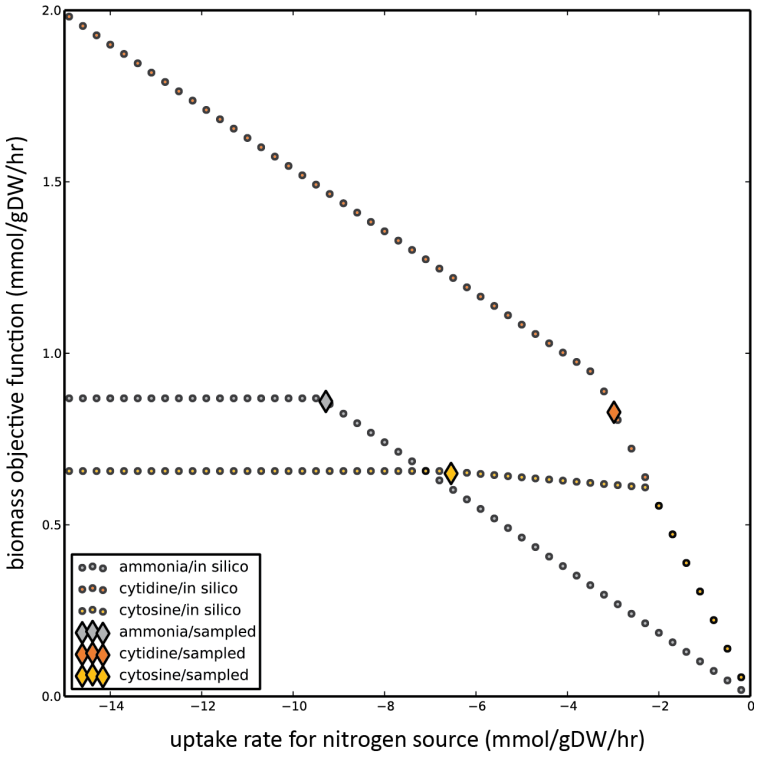


Figure C.13: **Biomass objective function of different nitrogen source uptake rate.** Simulation with experimentally measured glucose uptake rate and varying nitrogen source uptake rate was run (shown as dots). Sample points for flux analysis were also denoted (shown as diamonds). How these sample points were determined is described in more detail in Supplementary Experimental Procedures.

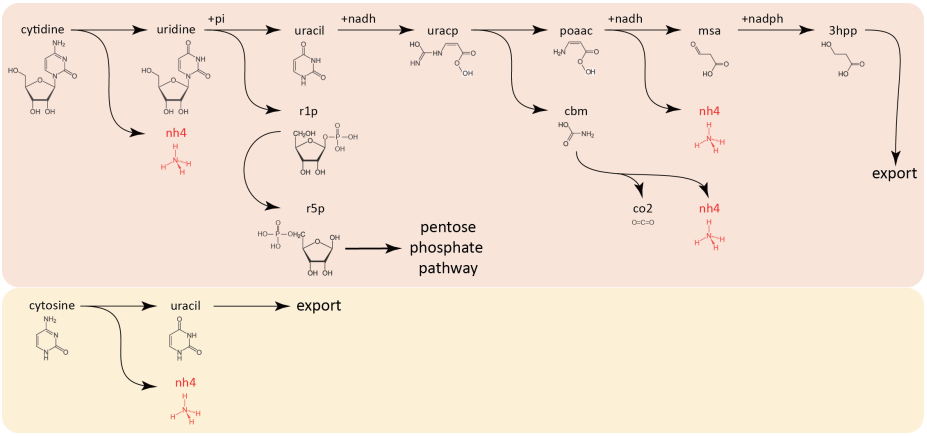


Figure C.14: **Pathways to assimilate nitrogen molecules from cytidine and cytosine when they are used as sole nitrogen sources.** A pathway to assimilate nitrogen from cytidine and cytosine was predicted from FBA analysis. Simulation predicted that cytidine would result in the assimilation of 3 ammonia molecules and ribose fed to pentose phosphate pathway. Conversely, the model predicted that the cell would harvest 1 ammonium from cytosine and export the remaining part when grown on cytosine.

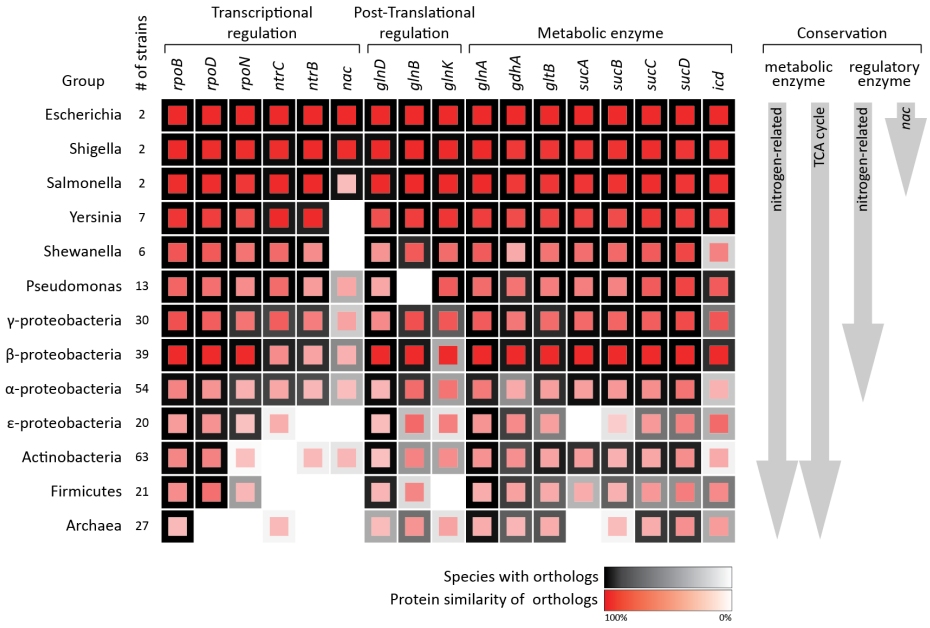


Figure C.15: **Conservation of enzymes in nitrogen response was calculated in various groups of species ranging from Escherichia genus to Archaea.** Metabolic enzymes seem more conserved than transcription factors.

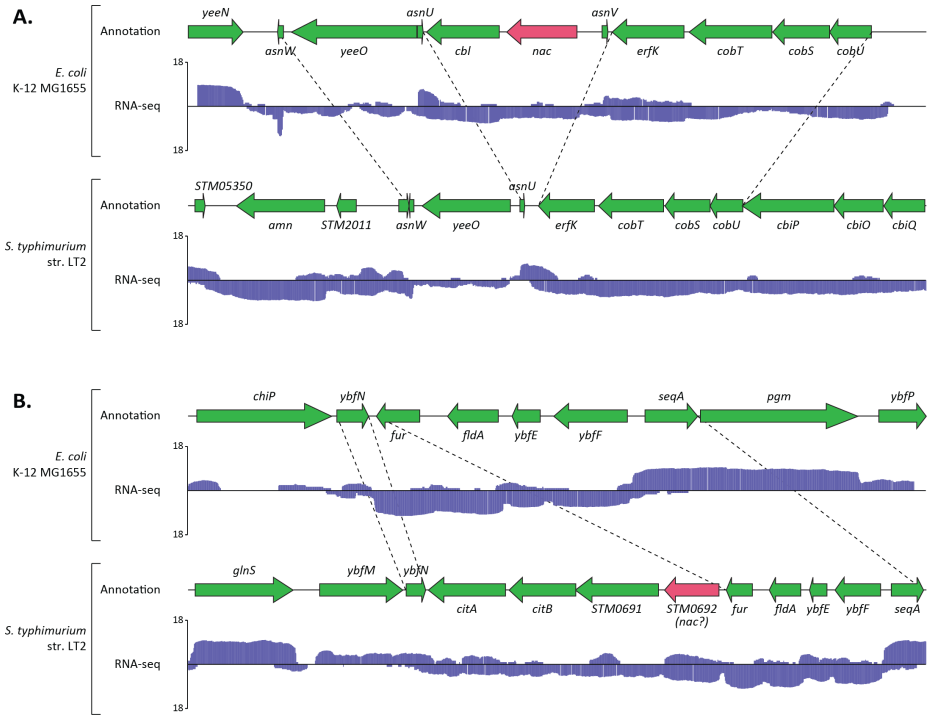


Figure C.16: **Comparison of genomic regions surrounding *E. coli nac* and *S. typhimurium nac* candidate (STM0692).** *S. typhimurium nac* ortholog was found at unexpected region. In *E. coli*, *nac* is located in *nac-cbl* TU, which are surrounded by *asnW*, *yeeO*, and *asnU* on the left flanking and *erfK*, *cobT*, *cobS*, and *cobU* on the right one. However, in *S. typhimurium* str. LT2, genomic regions containing *asnW*, *yeeO*, and *asnU*, and *erfK*, *cobT*, *cobS*, and *cobU* are conserved, but there is no gene between *asnU* and *erfK*. Similarly, in *E. coli* K-12 MG1655, there is no gene between *ybfN* and *fur*. However, in *S. tythimurium*, a genomic region containing *citA*, *citB*, *STM0691*, and *nac* candidate (*STM0692*) is located in between.

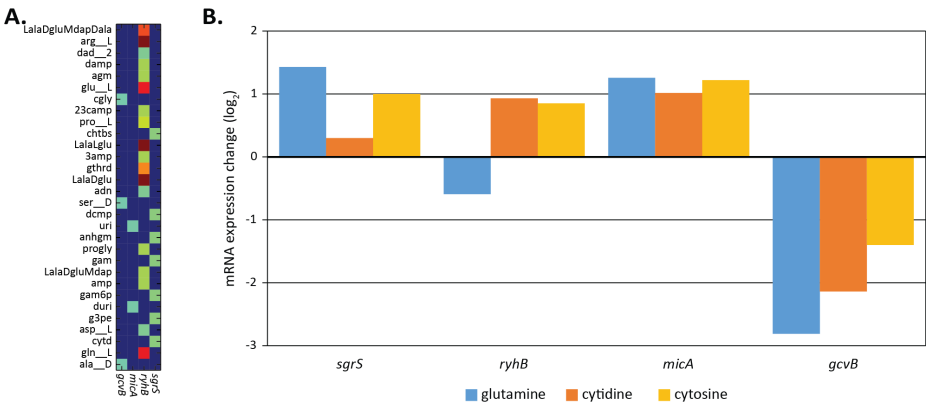


Figure C.17: **Prediction of sRNA involvement in nitrogen-response.** (A) sRNA regulation information from the public database BSRD and simulation of different 93 nitrogen sources with ME model was used to perform hypergeometric test to identify possible sRNAs acting on alternative nitrogen sources. 4 sRNAs, *gcvB*, *micA*, *ryhB*, and *sgrS*, were predicted to be involved. Heatmap shows p-values of hypergeometric test. (B) Transcription expression changes of those sRNAs on alternative nitrogen sources were calculated.

A.

	Regulatory all-bound	Regulatory	Non-regulatory
NtrC	5	19	**4**
Nac	15	249	**281**
RpoN	132	153	**232**
RpoD	1898	2171	**461**

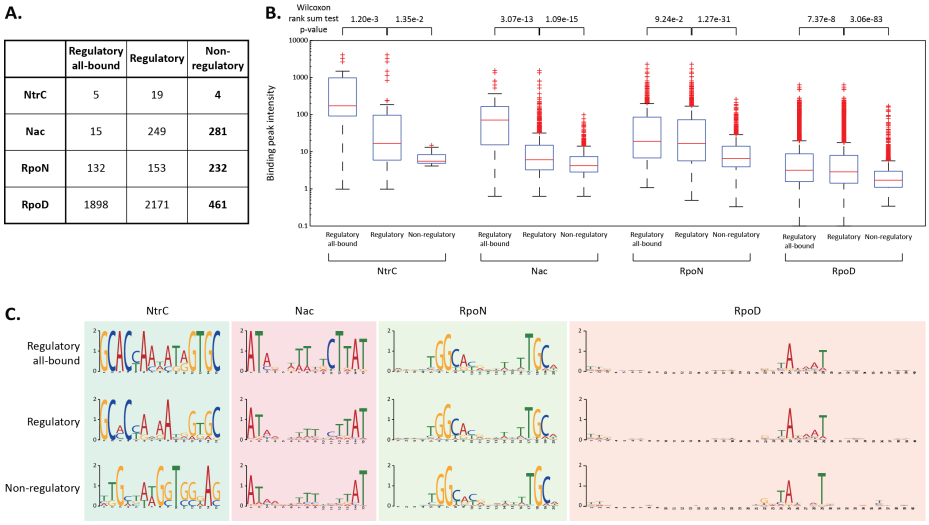


Figure S18. Non-regulatory bindings of TFs and σ-factors. (A) Number of bindings of NtrC, Nac, RpoN and RpoD in regulatory regions with occurrences in all nitrogen sources, in regulatory regions at least in one condition, and in non-regulatory regions. (B) Binding intensities of regulatory regions and non-regulatory regions were calculated. Non-regulatory bindings have weaker intensities. (C) Motif analysis was performed to each group of bindings, resulting in identical sequence motifs.

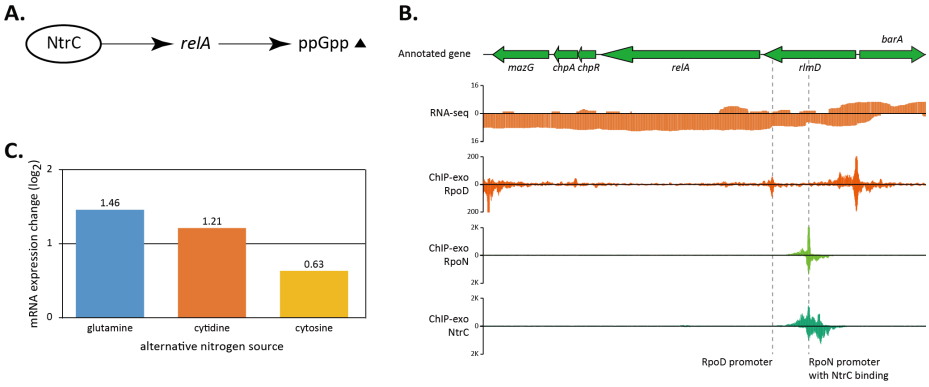


Figure C.18: **Stringent response mediated by NtrC.** (A) NtrC binds upstream of *relA*, activating its target gene. Up-regulation of *relA* can result in increase of ppGpp. (B) Based on ChIP-exo results for RpoD and RpoN, *relA* has at least 2 promoters in the coding region of *rlmD*. The proximal promoter is RpoD-dependent, while the distal one is RpoN-dependent. The distal RpoN-dependent promoter is associated with NtrC binding. (C) As a consequence of NtrC binding on RpoN-dependent promoter, transcription of *relA* was moderately up-regulated on alternative nitrogen sources, compared to ammonia.

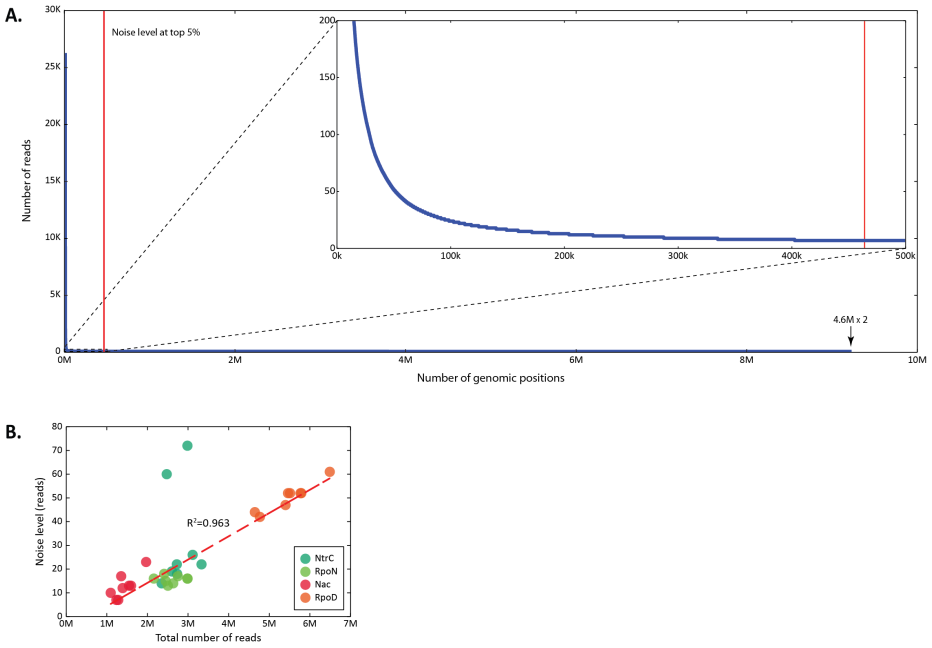


Figure C.19: **Determination of noise level in ChIP-exo sequencing reads.** (A) Noise level was determined to be top 5% signal intensity from every genomic position on forward and reverse strands. (B) Noise level correlated well with the total number of reads.

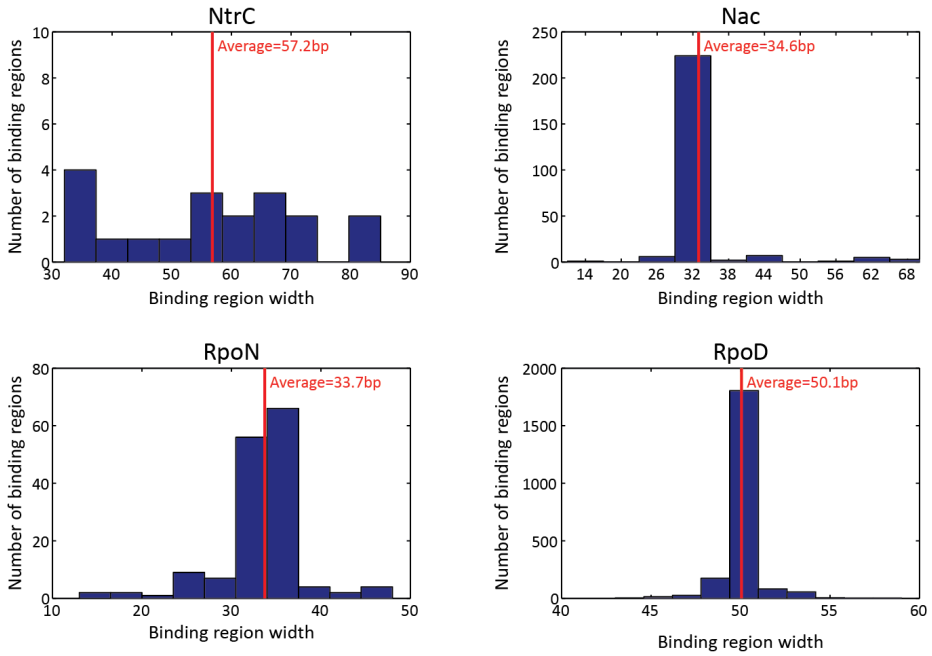


Figure C.20: **Width distribution of ChIP-exo binding sites for NtrC, Nac, RpoN, and RpoD.** The average binding widths for NtrC, Nac, RpoN, and RpoD are 57.2, 34.6, 33.7 and 50.1 bp.

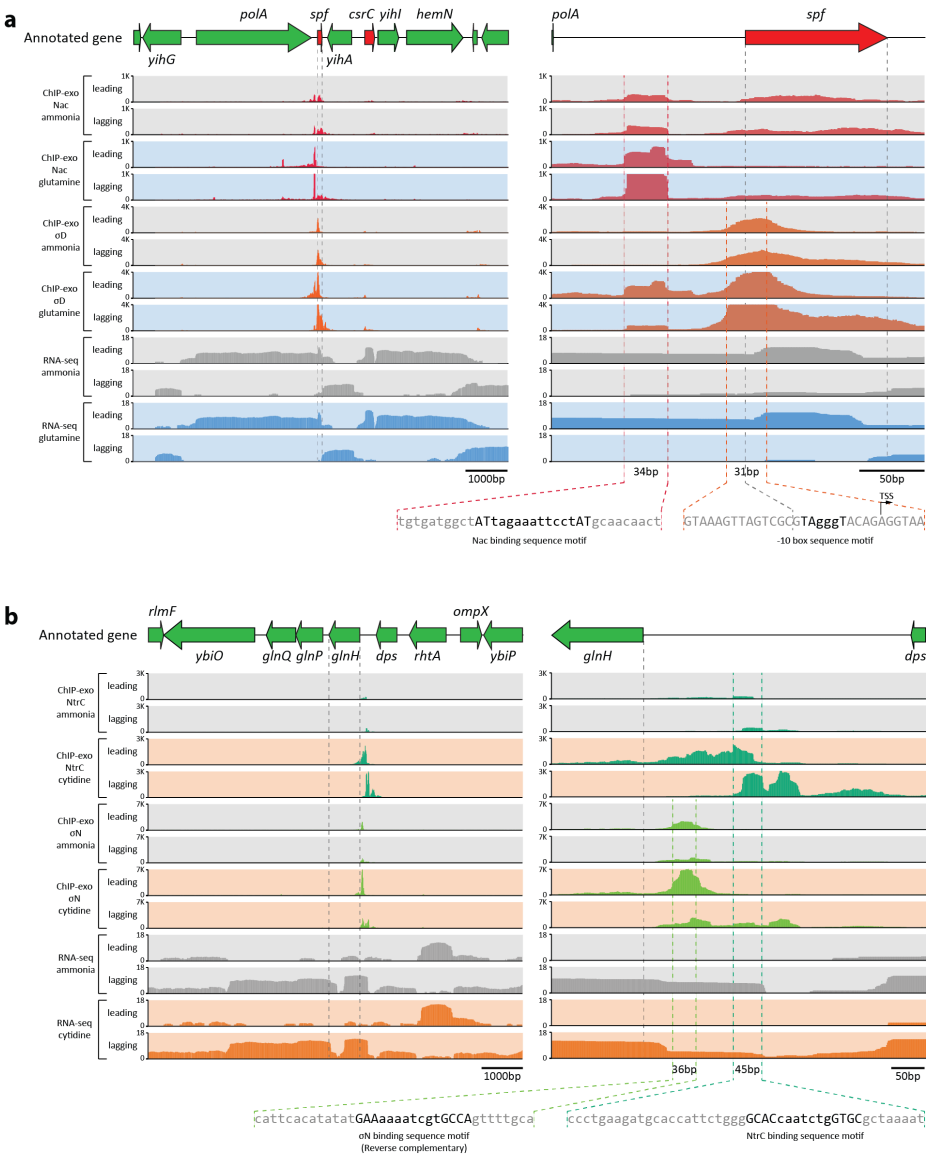


Figure C.21: **Examples of Nac/RpoD and NtrC/RpoN bindings and associated gene expression.** (A) ChIP-exo bindings for Nac and RpoD in the genomic region near *spf* small RNA under ammonia and glutamine conditions. Improved resolution of ChIP-exo method shows binding of Nac and RpoD upstream of small RNA *spf* whose length is 109 bp. (B) ChIP-exo bindings for NtrC and RpoN in the genomic region near glnHPQ, which encodes glutamine ABC transporter subunits, and expression of those genes under ammonia and cytidine conditions.

Appendix D

Multi-omic data integration enables discovery of hidden biological regularities - Supplementary Information

D.1 Methods

D.1.1 mRNA seq methods

Cells were harvested at mid-log (OD600 $\approx$ 0.3) in biological duplicates for each condition. From each sample, 3mL of culture were mixed with 6mL RNAprotect Bacteria Reagent (Qiagen), incubated for 5 minutes, and then centrifuged at 5000g for 10 minutes at room temperature. Total RNA samples were then isolated from the pellet using the RNeasy Plus Mini kit (Qiagen). Samples were quantified using

a NanoDrop 1000 (Thermo Scientific) and an Agilent RNA 6000 Nano Kit with an Agilent 2100 Bioanalyzer. Strand-specific mRNA libraries were created using the dUTP method, with ribosomal rRNA subtraction with the Ribo-Zero rRNA Removal Kit (Epicentre). Libraries were run on a Miseq (Illumina, CA) multiplexed with 2 duplicates per run as per the manufacturer's instructions. Expression values were computed using the bowtie [LS12] and cufflinks [TWP+10] packages. The processed data were uploaded to GEO under accession numbers GSE59759 and GSE59760.

D.1.2 Ribosome profiling

In order to compute the ribosome per protein ratios, *E. coli* MG1655 cells were grown in glucose (5g/L), pyruate (sodium pyruvate 3.3g/L), fumarate (disodium fumarate 2.8 g/L) and acetate (sodium acetate, 3.5 g/L). Ribosome profiling datasets were generated and analyzed according to the procedure detailed in by Latif [LST+15]. 100ug/mL chloramphenicol was added 2 min prior to harvest, cells were harvested at mid-log (OD600 $\approx$ 0.4) by centrifugation at 5000xg for 3 minutes. Ice cold lysis buffer (25 mM Tris pH 8.0, 25 mM NH4Cl, 10 mM MgOAc, 0.8 % Triton X-100, 100 U/mL RNase-free DNase I, 0.3 U/μL Superase-In, 1.55 mM Chloramphenicol, and 17 μM 5-guanylyl imidodiphosphate (GMPPNP)) was added and the cells were resuspended quickly followed by flash freezing in liquid nitrogen. Repeated freeze-thaw cycles were used to lyse cells followed by addition of sodium deoxycholate to a concentration of 0.3%. Lysate was then clarified by centrifugation. 25AU of RNA was digested using 6000U of MNase for 2 hours at 25C, and quenched by adding 2.5uL of 500mM EGTA. Monosomes were recovered using Sephacryl S400 Microspin columns, followed by removal of ribosomal RNA using a Ribo-Zero-rRNA Removal Kit (Epicentre). 3' ends

of the RNA fragments were desphosphorylated using T4 Polynucleotide Kinase for 30 minutes at 37C. The NEBNext Small RNA Library Prep Set for Illumina protocol was carried out till the 5' adapter ligation step. tRNA's were removed using hybridization to custom DNA oligo probes followed by RNAse H treatment as described by Latif [LST+15]. The NEBNext protocol was then completed and sequencing was carried out on a Illumina MiSeq.

As the MiSeq did not provide sufficient read depth for the in house ribosome profiling datasets to confidently determine pause locations, ribosome profiling data was obtained from Li [LBGW14] for MOPS Rich and glucose MOPS minimal media (GEO Accession: GSE53767). Ribosome densities were derived using a similar protocol to Li [LBGW14]. Adapters were trimmed using cutadapt version 1.8 [Mar11]. Reads were mapped using bowtie version 1.0.0 [LS12] to *E. coli* MG1655 (NC000913), allowing for a maximum of 1 mismatch. Reads mapping to tRNA, rRNA and other non-coding RNA locations were discarded. There has been considerable discourse about the location of the A and P site relative to the ends of the reads, and in order not to bias our analysis based on the location, we chose to assign reads to the 3' end, which has been shown to be better conserved and aligned in prokaryotic ribosome profiling datasets [MTH15, WGGB15]. Ribosome density across each gene was then dropoff corrected by fitting to an exponential function as was done by Li [LBGW14].

D.1.3 Genome-wide Secondary Structure Annotations

The GEM-PRO reconstruction for *E. coli* (*i*JO1366 [OCN+11]) {Brunk, Mih *et. al.* 2016 *BMC Systems Biology*} was used to provide structure-based annotations for the most representative protein structures found in the publicly available PDB

database [BWF$^+$00]. PDB files were parsed using STRIDE [FA95] and Biopython [CAC$^+$09] to determine the location of secondary and tertiary structural elements on a codon-specific basis. This resulted in high confidence secondary and tertiary structure annotations for 623 non-transport (or membrane-bound) genes in *E. coli*.

Data Retrieval and Manipulation

Incorporating protein-related information into a GEM involves four stages of semi-automated curation: (i) map the genes of the organism to available experimental protein structures, found in publicly available databases, such as the Protein Data Bank (PDB); (ii) determine genes with and without available protein structures and perform homology modeling using the I-TASSER suite of programs [Zha09] to fill in gaps where crystallographic or NMR structures are not available; (iii) perform ranking and filtering of PDB structures for each gene based on a set selection criteria (e.g., resolution, number of mutations, completeness); (iv) map GEM genes to other databases (e.g., BRENDA [CSG$^+$09, SCE$^+$04], SwissProt [BBA$^+$03], Pfam [FMT$^+$10], SCOP [MBHC95]) for complementary protein-structure derived data. The quality of the reconstruction expansion process to include high confidence protein structures is considered by carrying out a series of QC/QA verification steps during the ranking and filtering stage. The GEM annotation of the organism of interest is stored in SBML and Matlab formats and many organisms can be found in the BiGG database [SPCP10, KLD$^+$15]. Amino acid sequence of the proteins of interest are stored in FASTA format. To map protein structural data to a GEM, we make use of Python modules, ProDy [BMB11, McK12] and Biopython [CAC$^+$09] to parse information in the PDB files. The molecular visualization software VMD [HDS96] was used for

viewing the 3D structure of the modeled protein and the predicted functional sites and the creation of images. Installation of PfamScan and HMMER3 algorithms are required for generating protein fold families for certain proteins [FMT+10, FBC+14]. Open source software for protein structural predictions are available and are used in conjunction with the IPython framework.

D.1.4 Tertiary Structure/ Protein Domain annotations

Starting from the protein structures linked to metabolic genes in the GEM-PRO model, we annotated tertiary domains for each protein using the SCOP knowledgebase [MBHC95] and FATCAT [YG03] alignment tools. As a result of this analysis, the fraction of the protein aligning to an annotated tertiary domain was recorded and stored as an additional datatype in the GEM-PRO reconstruction. The starting and ending amino acid of every domain within a protein were quality controlled and checked by aligning the PDB sequence with the amino acid sequence (FASTA) from *E. coli* MG1655 to fix offsets between the PDB residue numbering scheme and the actual amino acid sequence numbers. Hypergeometric enrichment testing was performed to determine codons upstream and/or downstream from the start and end of any tertiary domain annotation that is enriched for pause sites.

D.1.5 Identification of Shine-Dalgarno-like codons

Similar to those defined in Li [LOW12], we considered the following Shine-Dalgarno-like codons: 'AGG','GGA','GAG','GGG','GGT','GTG'. Nucleotide sequences from *E. coli* MG1655 (Genbank accession: NC000913 [BCC+13]), were read in frame to identify SD-like codon positions. Hypergeometric enrichment testing

was used to determine downstream codons enriched for pause sites.

D.1.6 Ribosome density/pause site accounting (Figures D.2-D.3)

The ribosome density at each codon was summed across all three nucleotides, and divided by the mean of the gene to get the normalized density. In an effort to increase the signal in an inherently noisy datatype, pause sites were defined as codons which had a normalized density of over 5, instead of on a per nucleotide basis as was done in Li [LOW12].

Hypergeometric enrichment testing was used to determine enrichment for pause sites at codon positions downstream from the ends of pause sites. p-values were calculated based on the formula for the survival function (1 - cumulative distribution function) shown below:

$$p(x) = 1 - \sum_{i=0}^{x-1} \frac{\binom{m}{i}\binom{N-m}{k-i}}{\binom{N}{k}}$$

where N refers to the total number of codons in the genes tested, m refers to the number of secondary structures, k refers to the total number of pause sites, and x refers to the number of pause sites which fell on a specific codon position downstream of the secondary structure we are testing.

Codons downstream from αhelices, βsheets, and turn secondary structural elements, as well as Shine-Dalgarno-like sequences were considered to be significantly enriched for pausing if the hypergeometric enrichment tests indicated that the p-values <0.01. To calculate the number of pause sites accounted for by sequence and/or

structural elements, pause sites which did not align with secondary structure or SD-like sequences were labeled "unaccounted". The same procedure was used for determining the proportion of pause sites accounted for by structural features and SD-like sequence features.

D.1.7 Computational Method for Predicting k_{eff} parameters

As noted above, ME simulations require several parameters, one of which is the effective catalytic rate of enzymes k_{eff}, which in turn affects the proteomic and ribosomal cost of running each reaction. While using SASA as a first approximation results in a correct overall prediction of 80% of the cell proteome by mass, improving these parameters can greatly affect the predictive power of the model for specific genes. We make use of the most extensive quantitative proteomic data in *E. coli* to date which accounts for 55% of all ORF's and 95% of the proteome [SKV+16]. Because of the difficulty in simultaneously predicting effective catalytic rates and reaction flux values, we developed an iterative workflow for updating the model parameters. This workflow is described below, and each section corresponds to a panel in Figure D.6.

Part A: Iterative Simulation Procedure

These ME simulations had the overall goal of minimizing the difference between the simulated proteome and the measured proteome for each experimental condition. The growth rate μ was set to the experimentally determined values. To improve solution times with the SoPlex linear programming solver [Wun96] with our formulation, we collapsed linear pathways into single reactions using COBRApy [ELPH13], which were detected by identifying metabolites present in exactly two reactions.

In order to reconcile experimentally measured protein concentrations with the simulations, we want to solve a linear program which will minimize the Manhattan distance between the expressed and measured protein production at the fixed growth rate. The Manhattan distance was used instead of the Euclidian distance because it can be computed in the context of a linear program, while a Euclidian distance minimization requires a quadratic objective, which the SoPlex solver can not handle. To construct this problem, we added the following additional constraints to the ME linear program in terms of the corresponding measured protein amounts y_i for each predicted gene translation flux variable x_i (unmeasured proteins had no applied constraints).

$$x_i \geq 0$$

$$s_{+i} \geq 0$$

$$s_{-i} \geq 0$$

$$x_i + s_{+i} \geq y_i$$

$$x_i - s_{-i} \leq y_i$$

This allowed us to minimize the error term $\sum_i (s_{+i} + s_{-i})$, which is equivalent to minimizing $\sum_i |x_i - y_i|$ and gives the closest flux state to the experimental data while satisfying the ME constraints when solving the linear program. If these parameters resulted in an infeasible model that could not simulate growth at μ, the simulation was halted. Otherwise, using the predicted fluxes predicted by this simulation, and the experimentally measured proteomics data, we calculated the k_{eff} for each reaction

enzyme which we used in the next iteration of this workflow. This simulation loop was run a total of 3 times to allow the loops to converge to a set of k_{eff} values.

Part B: Sampling and Simulation

The iterative simulation procedure described above might give a flux state that is dependent on the original set of k_{eff} parameters used in the first round. Therefore, the k_{eff}'s were randomly initialized within two orders of magnitude of the value computed from SASA. The iterative sampling procedure was repeated repeated 300 times for each experimental condition, and each time a new random k_{eff} parameter set was generated in the manner described.

Part C: Result Aggregation and Filtering

For each experimental condition, the loop was started 300 times. However, because some parameter sets were unable to simulate growth at μ, some subset of those simulations failed before reaching 3 iterations. It was run successfully through 3 loops 148 times for glucose, 186 times for pyruvate, 97 times for fumarate, and 83 times for acetate. Between these successful runs, there was a slight variation in reactions used because of the different starting k_{eff} parameters. Therefore, only reactions that were active for 90% of the successful runs were considered. These parameters were averaged to give a consensus set of k_{eff} parameters for each condition.

Part D: Cross-condition parameter comparison

The intersection of these k_{eff} parameters under each condition was determined between all 4 conditions (Figure 4.3b). The 284 parameters for reaction/catalyst pairs that were in all conditions were averaged to get a consensus set of k_{eff} parameters

which were used for ME computations. Additionally, the pairs in common between each of the conditions were compared to give Pearson correlations, shown in the table in Figure 4.3b.

D.1.8 Predictions of mRNA expression under identical conditions to proteomic data

We sought to evaluate the effect of the estimated parameter values on predictions of differential gene expression under identical experimental conditions as the proteomics used to parameterize the model. We simulated a switch of the primary carbon substrate from fumarate to acetate, and obtained mRNA sequencing data (GSE59759) for *E. coli* K-12 BW25113 (obtained from the Coli Genetic Stock Center) cultivated under identical experimental conditions to those of the proteomic data. Bach growth simulations were then performed using 3 different sets of k_{eff} parameters: the initial solvent-accessible-surface area parameters, k_{eff} parameters derived from fluxomics (set A), and k_{eff} parameters derived from fluxomics and our proteomics-based algorithm (set A + B). Predicted differential expression was compared to the set of genes identified as differentially expressed by cufflinks [TWP+10] at a false discovery rate of 0.05. In order to account for accuracy on a level field with respect to sensitivity, we varied the computational cutoff so that we would have a similar number of correct predictions (as close to 20 as possible), allowing us to directly compare the number of incorrect predictions. Using the the original *i*OL1650-ME model k_{eff} values obtained from protein size based estimation, we observed a significant number of false positive predictions due to incorrect pathway usage, with 17 false positives found out of 37 total predictions. Using a consensus k_{eff} parameter set derived from both experimentally

measured flux values [NSS06] (set A) and model-predicted values (set B) yielded only 6 false positives with the same number of correct predictions. Additionally, we were able to improve predictions greatly using the 28 k_{eff} parameter values from set A alone (Figure D.7), which only constrained parameters in central carbon metabolism. This result suggests that the accuracy of a ME model is most sensitive to k_{eff} parameters for reactions which lie along its high flux backbone.

D.1.9 Predicting Differential Gene Expression with *i*OL1650-ME

Simulations were performed using the same procedure as in the *i*OL1650-ME manuscript [OLC+13] for batch growth on D-Glucose with both k_{eff} parameter sets A and A+B. For each supplementation simulation, the uptake reaction for that particular metabolite was set to be unbounded. In the case of the Adenine supplementation, the reactions HXAND, XAND, and URIC were blocked (Figure D.8). Genes which changed by more than a factor of 16 (a $\log_2$ change of more than 4) were predicted to be differentially expressed. This gives a stringent criterion which identifies genes which are predicted to change by a significant enough magnitude to manifest experimentally (by comparison, a $\log_2$ change of 2 is often used with microrray gene expression data to filter out changes in expression, which, while statistically significant, are not of a high enough magnitude to really be considered relevant). Predictions of gene differential expression were considered correct if cufflinks obtained a false discovery rate of less than 0.05 for that gene in the mRNA sequencing data and the gene expression changed in the same direction (either both increase or both decrease) in both the predictions and mRNA sequencing data. Hypergeometric enrichment p-values were calculated

using the scipy statistics package using the survival function + $\frac{1}{2}$ * probability mass function of the distribution.

D.1.10 Sampling of M-model flux states in *i*JO1366

The optGpSampler [MHM14] software was used to sample flux states in the *i*JO1366 M model using its python API. First, the model was reduced by removing blocked reactions as identified by flux variability analysis in COBRApy [ELPH13]. For each simulation, the lower bound on the biomass reaction was set to 90% of its optimal value. Additionally, the reactions HXAND, XAND, and URIC were blocked (Figure D.8). The sampling algorithm was then run for 100 steps and generated 10000 points for each simulation. Afterwards, the fluxes were linearly scaled such that the mean flux of the biomass reaction was 1. Sampling was run on the model with only D-Glucose uptake, and also with 10 mmol/gDw/hr uptake allowed of each of the supplements. Reactions that were unbounded (as determined by an FVA maximum greater than 500 or an FVA minimum of less than -500) were excluded from the subsequent analysis. For each of the supplements, predicted changed reaction fluxes between the supplemented and unsupplemented samples were determined by finding reaction fluxes where (1) the mean changed by more than a factor of 2 and (2) the mean changed by more than the sum of the standard deviations for the supplemented and unsupplemented fluxes. These reactions were converted to gene predictions by assuming all genes in the gene reaction rule for the changing reaction were up or down regulated. These gene predictions were validated against mRNA sequencing data in the same manner as the ME gene differential expression predictions. This method was used instead of the more traditional method comparing pairs of samples as done

in some other studies [BNL+14] because it resulted in higher accuracy than those methods with this data.

D.2 Structure of ME models

Here we briefly introduce key formulations of the ME model, and refer the interested reader to a more complete supplementary information in O'Brien et. al. 2013 [OLC+13]. The ME model is a steady state growth model which accounts for metabolism (M) and gene expression (E). Based on an input of available nutrients to the cell such as carbon and nitrogen sources, it predicts: a) the cell's maximum growth rate in a specific condition, and at the maximum growth rate, b) metabolite uptake and excretion rates, c) metabolic reaction fluxes, and d) gene expression fluxes such as translation and transcription rates. This is achieved through formulating transcription, translation, transport and metabolic fluxes into a quasi-linear problem and solving for maximum growth rate, taking into account compartmentalization of proteins and metabolites into the cytoplasm, periplasm, and extracellular region. This is done as follows:

1. The metabolic network is described by a stoichiometric matrix, similar to those used in M-models [OCN+11] where rows represent metabolites and columns represent reactions. Coefficients represent the metabolites consumed (negative value) or produced (positive value) in each reaction. At steady state, there is no change in metabolite concentrations, hence we get:

$$S \cdot \vec{v} = 0$$

where S is the stoichiometric matrix, and $\vec{v}$ is the flux vector, allowing us to solve for

$\vec{v}$.

2. In the ME model, translation is accounted for by enforcing that proteins have to be produced for all enzyme-catalyzed reactions, proportional to the flux for that particular reaction. Protein synthesis is balanced by protein dilution (due to cell division), which is proportional to the amount of protein required to hold the predicted flux. Note that in this case we disregard direct protein degradation as it has been shown to negligible in a growing *E. coli* cell for most metabolic genes [Mau92, NK71]. While active degradation is important for many signaling proteins such as transcription factors, these proteins are a very small proportion of the whole proteome in rapidly growing cell. The depletion of most of the modeled proteins is therefore mostly through dilution caused by growth, and the depletion rate is simply the growth rate. Mathematically, this is represented below:

$$v_{\text{translation},i} = v_{\text{diluation},i} = \mu[E_i]$$

$$v_{\text{reaction},i} = k_{\text{eff},i}[E_i]$$

$$v_{\text{translation},i} = \frac{\mu}{k_{\text{eff},i}} v_{\text{reaction},i}$$

The above relationships link the required translation rate $v_{\text{translation},i}$ to the flux through the reaction $v_{\text{reaction},i}$ as well as the predicted growth rate μ and the proteins effective catalytic rate $k_{\text{eff},i}$. Unlike traditional M models where biomass has to be explicitly modeled, this inherently takes protein and macromolecule dilution into account during growth, and allows prediction of optimal protein production, and hence gene expression.

It is important to note that, in previous ME models [OLC+13], the effective

catalytic rate was set to be proportional to the respective enzyme's solvent accessible surface area (SASA). Here, we make use of condition-specific proteomics data, which vastly improves this parameter and the predictive scope of the model itself. More details of this parameterization procedure is found in the following section.

Finally, translation of proteins are catalyzed by ribosomes, requiring tRNA and its synthethases and the cost of production of these molecules are also explicitly accounted for in the ME model. Analogous to metabolic enzymes, ribosome efficiency k_{ribo} is a necessary parameter for coupling the ribosome production rate to the overall proteome translation rates, and is shown below:

$$v_{\text{synthesis, ribosome}} = v_{\text{dilution, ribosome}} = \sum_i \text{length}\left(\text{peptide}_i\right) \cdot \frac{\mu}{k_{\text{ribo}}} \cdot v_{\text{translation},i}$$

Because ribosomes are themselves partially made up of peptides which need to be synthesized, this coupling creates an asymptotic relationship between ribosome production and growth rate. Placing a limitation on cell size replicates the natural phenomenon where growth rate is constrained by the balance between enzyme and ribosome production even in the overabundance of nutrients.

3. In the ME model presented in this contribution, transcription is now achieved through the production of mRNA from nucleotides, catalyzed by RNA polymerase. This is also handled in a similar way to ribosomes and metabolic enzymes through dilution of RNA polymerase, proportional to the total flux through all transcription reactions.

D.3 ME model coupling parameters

In addition to metabolic reactions, the ME model describes various biological processes as biochemical reactions. For example, translation of a protein is described by a reaction assembling the component amino acids into peptides. An enzyme complex is then formed by a reactions which assemble together the various peptides and cofactors, and more reactions which apply the necessary post-translational modifications. Dependent reactions are linked through what are termed "coupling constraints" because they force a certain amount of flux through one of the reactions based on the level of flux through the other, effectively coupling the processes. For example, flux through a metabolic flux is coupled to production of the catalytic enzyme. A translation reaction is coupled to both reactions which produce ribosomes and reactions which transcribe the mRNA. In *i*OL1650-ME, these coupling constraints are expressed as inequalities as functions of growth rate μ to reflect how the nature of many of these constraints are nonlinear in growth. However, the constants in these functions are set for each individual coupling constraint. A detailed description of all the processes and coupling parameters in the *i*OL1650-ME model is available in the supplementary information of that manuscript [OLC+13].

One of the most critical coupling constraints to the function of the ME model is the k_{eff}, which describes the amount of enzyme required on average to sustain a unit of flux under in vivo conditions. An example coupling constraint for a metabolic reaction has the form:

$$v_{\text{metabolic}} \leq \frac{k_{\text{eff}}}{\mu} v_{\text{enzyme}}$$

This coupling constraint expresses both how as growth rate increases, more enzyme must be made to pass on to each daughter cell, and how the amount of enzyme production relates to its efficiency through the k_{eff} parameter. While the k_{eff} parameter has units of 1/time like a k_{cat}, it is strictly less than the k_{cat}, as it is describing the amount of enzyme that must be made to catalyze a unit of flux instead of the maximal flux a particular enzyme can sustain. The reason this parameter is critical to the function of the ME model is because it describes the relative cost of each enzyme. An appropriate analogy is an econometric model. Where the reactions themselves express operational costs, and the cost of the enzyme is the capital cost. A good example of this the difference between the "expensive" but efficient pyruvate dehydrogenase complex compared to pyruvate formate lyase (Figure D.11). Pyruvate dehydrogenase (PDH) and pyruvate formate lyase (PFL) both convert pyruvate to acetyl coA to allow carbon to flow through the TCA cycle. Flux through PFL will result in secretion of formate, whereas flux through PDH will result in production of CO_2, which is the *in vivo* behavior. In *E. coli*, PDH is one of the most expensive metabolic enzymes because it consists of 60 subunits. However, an optimally efficient *E. coli* cell might still produce this enzyme over PFL because its catalytic rate is more than commensurately higher, as it exploits phenomena such as substrate tunneling and multiple catalytic sites, and will therefore have a lower protein cost per unit of flux catalyzed. Therefore, in order for an ME model to correctly predict flux through PDH, it must have k_{eff} parameters for PDH and PFL which represents this tradeoff accurately. In a genome-scale ME model, these parameters affect the cost between all the various alternate pathways and isozymes. Therefore, the accuracy of the model predictions of protein expression and the physiological state of the cell will depend on

reasonable relative values of these parameters.

D.4 Simulation of Batch Growth with ME

One of the advantages of the ME model over traditional M models is its ability to account for proteome limitation. This gives the model the ability to correctly simulate batch growth, where a cell has a surplus of nutrients around it, but is limited by its ability to produce the proteins which are required to process these nutrients. Unlike M models, which can predict growth rates given a specific substrate uptake rate, a ME model can predict the maximal growth rate given an unbounded substrate uptake (allowing the model to take up as much substrate as it wants). O'Brien et al. investigated how proteome limitation begins to take effect as the substrate uptake rate approaches the optimal value, eventually causing a maximal growth rate [OLC+13]. Beyond this optimal growth rate, the model is infeasible with any substrate uptake rate because of the proteome limitation constraints. This optimal growth rate can be computationally identified by doing a binary search. Because the experimental data used in this study were all generated under batch growth conditions, the simulations of growth used this batch growth procedure, which computes a proteome-limited state.

D.5 Simulating ME with estimated parameters

In the absence of in vivo experimental measurements, the original *i*OL1650-ME model estimated k_{eff} values based on the solvent-accessible surface area (SASA), which is a function of protein size [OLC+13, MLJC87]. We sought to evaluate the effect of our new set of estimated parameters values on predictions of differential gene

expression. We simulated a switch of the primary carbon substrate from fumarate to acetate, which could be validated by the previously generated mRNA sequencing data sets (GSE59759). Using the original k_{eff} values obtained from protein size based estimation we observed a significant number of false positive predictions due to incorrect pathway usage, with 17 false positives found out of 37 total predictions. Using the consensus k_{eff} parameters derived from sets A + B yielded only 6 false positives with the same number of correct predictions. Interestingly, almost all of the improvement in prediction came from using parameter set A alone (Figure D.7), which constrained parameters in central carbon metabolism. This result suggests that the accuracy of a ME model is most sensitive to these key 28 k_{eff} parameters which lie in its high flux backbone [AKV$^+$04].

After showing that the set of estimated k_{eff} values gives better predictions for previously examined nutrient shift, we generated new experimental data for growth on dual substrates. We computed predicted differential gene expression after media supplementation with four key nutrients: Adenine, Glycine, L-Threonine, or L-Tryptophan, using 1) k_{eff} found in set A, and 2) k_{eff} found in both sets A and B (Table D.1). Genes that were computed to change in expression by more than a factor of 16 after supplementation were considered to be predictions of differential expression (Figure 4.4). Prediction validation was performed using mRNA sequencing data, some of which was taken from a previous study [BNL$^+$14], to experimentally determine differentially expressed genes. Predictions were made with a slightly modified *i*OL1650-ME (Figure D.8). For all four supplementations, the accuracy of ME predictions (Table D.1) was higher than those resulting from sampling M model flux states (Figure D.9). Moreover, the accuracy increased when comparing parameter set A + B to parameter

set A alone. Using parameter set A + B, the accuracy of predictions of ranged from 55 to 100%, and all predicted sets were significantly enriched for differentially expressed genes ($p < 0.05$ using a hypergeometric distribution).

These results suggest that a parameterized genome-scale ME model, due to its incorporation of enzyme biosynthetic costs, gives a significant improvement in prediction of gene expression over existing methods using M models alone. For example, the ME model correctly predicts the often non-intuitive shift of amino acid precursors. Specifically, when supplementing with L-Threonine, the ME model will produce L-Serine from L-Threonine directly, as observed experimentally. On the other hand, the ME model predicts no significant up regulation of glyA to produce L-Serine from Glycine supplementation, as seen in the expression profiling data. The *i*JO1366 M-model will incorrectly predict this change (Figure D.9), demonstrating how the ability to account for protein expression costs improves the accuracy of the predicted flux state. Moreover, as a result of its quantitative numerical predictions of gene expression, ME models can also predict partial up and down regulation of genes, in addition to binary responses when a gene goes from active to inactive. For example, after supplementation with Adenine, the model correctly predicts downregulation of purHD and purMN (Figure D.3).

D.6 Ribosome profiling pause site analysis

It has been established that translation rate is not constant along a gene. This has been demonstrated in-vitro using proteins such as firefly luciferase [YDZ$^+$15, SDA$^+$10] and epoxide hydrolases [HSLI15]. Various phenomena such as codon usage [YDZ$^+$15, GYY$^+$14], mRNA secondary structure [SKP89], anti-Shine-Dalgarno-like

sequences [LOW12], poly-proline stretches [WGGB15] as well as protein domains [HSLI15, ZHI09] have been implicated in determining the locations of these pause sites. Slower translation rate have been shown anecdotally to improve solubility of heterologously expressed proteins [YDZ$^+$15, HSLI15] as well as improve yield and function [SDA$^+$10, ZGC$^+$13], and has been implied to be necessary for proper co-translational folding [SD10, PBS$^+$87, KLR99, SZO14], yet at the same time impose a fitness cost on the cell [HG14]. Here we show a link between secondary structure motifs, anti-Shine-Dalgarno-like sequences, and pausing locations along the length of a transcript.

Hypergeometric enrichment testing of the pause sites downstream from the end of each structure points towards the enrichment of pausing at certain codon locations (Figure D.2A, C, E). At the same time, sequence analysis near the ends of the secondary structures that have pause sites at these locations show increased occurrences of anti-Shine-Dalgarno-like sequence, approximately 6 codons upstream from the enriched pause site for each secondary structure motif (Figure D.2B, D, F). This indicates that secondary structures tend to have pause sites downstream from their ends, and anti-SD-like sequences might be used to induce the pausing. It is noted here that in contrast to other studies which determine the pausing propensity with relation to the assumed A site, assigning ribosome density to the 3' end of the read results in a pause site which is around 2-3 codons further downstream. For example, Li *et. al.* [LOW12] found that anti-SD-like sequences were linked to pausing 8-11 nucleotides ($\approx$ 3rd–4th codon) downstream, which in our analysis occurs on the 6th codon downstream instead because of the different positional assignment of the read.

Overall, across both MOPS minimal and MOPS rich growth conditions, around

50% of ribosomal density and 60% of pause sites can be attributed to either anti-SD-like sequences or secondary structure motifs (Figure D.4). While this indicates that these two factors play a major role in ribosome pausing locations, the overlap between them is only between around 15% of all pause sites. This suggests that there are other factors unaccounted for in this study, which either require, or induce pausing, and should be the subject of further investigation.

D.7 Supplementary Figures

Table D.1: Predicted expression changes confirmed experimentally.
The ME model predicted differential expression for the following nutrient supplementations to growth of E. coli on M9 Minimal Media with D-Glucose. Two separate sets of k_{eff} were used, one using only the 28 parameters in set A, and the other also adding in the 284 modeling-derived parameters in set B. The predictions were evaluated using the set of differentially expressed genes determined from mRNA sequencing for each nutrient supplementation. Two small modifications were made to the model for Adenine, which initially had an accuracy of only 26.2% due to the genes used for Adenine degradation in the model which are not expressed and may not be functional (for details see Figure D.8). The numbers provided in the correct column are the number of genes which are predicted to be differentially expressed and also are differentially expressed in the same direction in the RNA-seq data. The number in parenthesis refers to the number of correctly-predicted differentially expressed genes which are still expressed under both conditions but varied in their predicted quantitative values. The incorrect column contains the number of genes predicted to be differentially expressed which were not in the model. These numbers are used to predict the percent accuracy, and the p-value for a hypergeometric enrichment of differentially expressed genes in the predicted set.

	k_{eff} parameters from set A				**k_{eff} parameters from set A+B**			
	Correct	Incorrect	Accuracy	p	Correct	Incorrect	Accuracy	p
L-Tryptophan	6 (0)	3	66.7	0.031	7 (0)	0	100	0.011
L-Threonine	30 (5)	15	66.7	0.119	15 (5)	5	75	0.044
Adenine	12 (4)	35	25.5	0.663	11 (4)	7	61.1	4.00E-06
Glycine	15 (0)	20	42.9	0.391	5 (0)	4	55.6	0.024

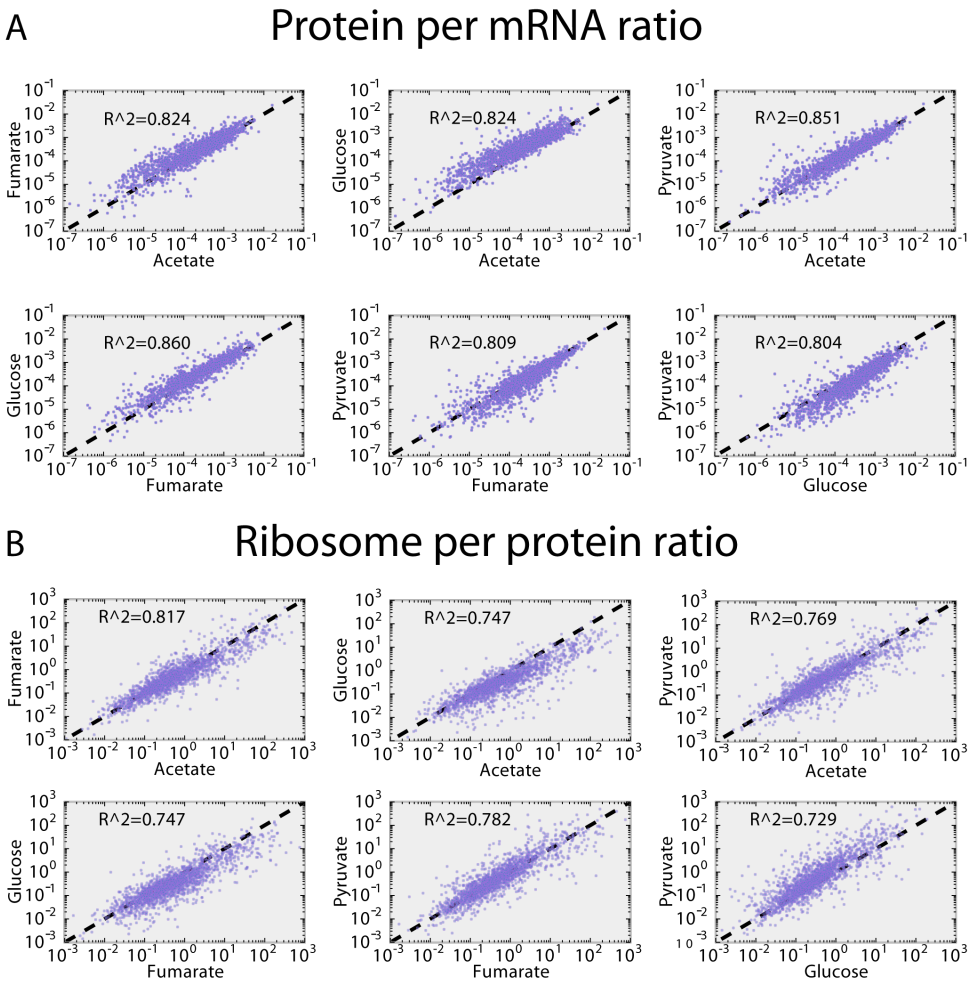


Figure D.1: Protein per mRNA ratios (A) and ribosome per protein (B) ratios across environments are highly conserved.
Ratios span several orders of magnitude across genes, but are highly conserved across different experimental conditions

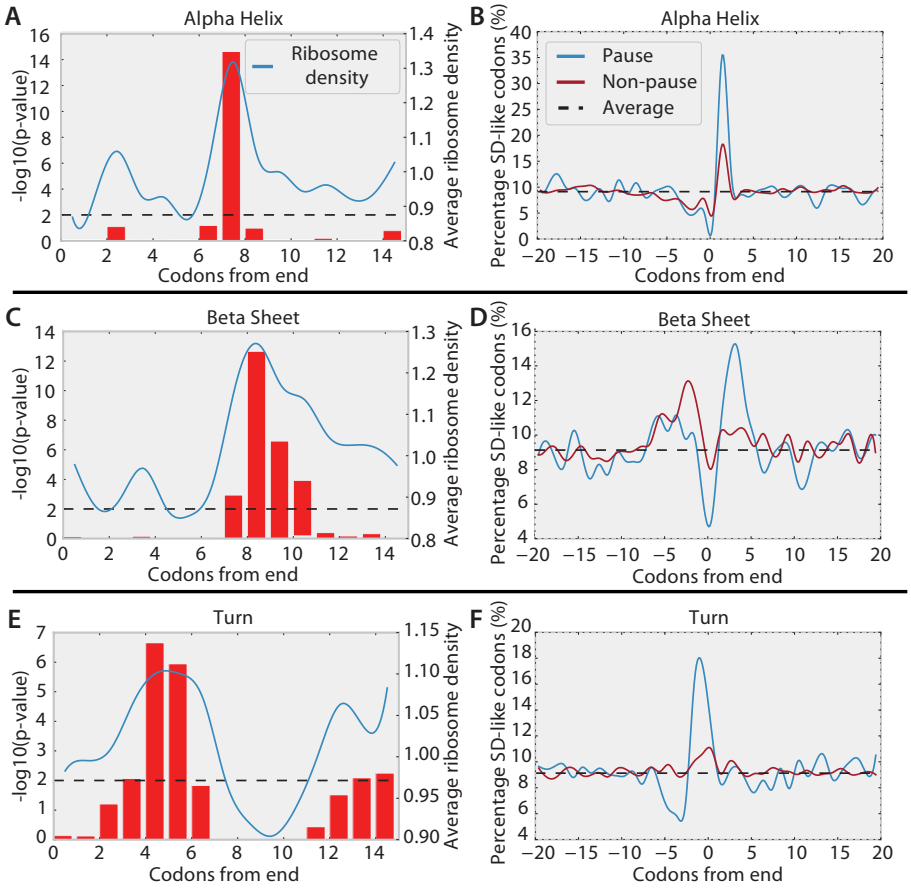


Figure D.2: Pause site enrichment and percent SD-like sequence near ends of secondary structures.

Hypergeometric enrichment testing was used to determine locations downstream of secondary structures which were enriched for pausing (A, C, E). Red bars represent p-value of enrichment. Blue line indicates the per gene normalized ribosome density at each codon position averaged across all occurrences of the secondary structure, showing a matching increase in average ribosome density. Based on the codon locations indicated by the enrichment test, all occurrences of each secondary structure were divided into those with corresponding pause sites (Pause) and those without (Non-Pause). The percentage occurrence of Shine-Dalgarno-like sequences appearing near the ends of these secondary structures is shown (B, D, E). The global average occurrence of an SD-like sequence is 9.13%, but this increases greatly at certain codon locations near the ends of secondary structures with corresponding pause sites (Blue line).

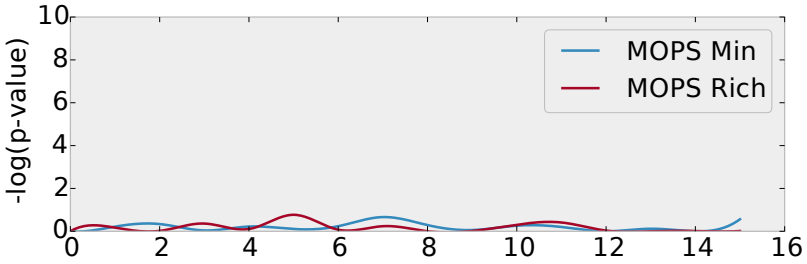


Figure D.3: Hypergeometric enrichment test of codons downstream from annotated SCOP domains.
No enrichment was found when codons downstream from the ends of domains were tested for pause site enrichment. Previous studies have found evidence that show slow translating regions between domains might be important for proper folding. However the location and even presence of pause sites might only occur on a case by case basis specific to particular domains, and fail to show enrichment when tested on the genome scale.

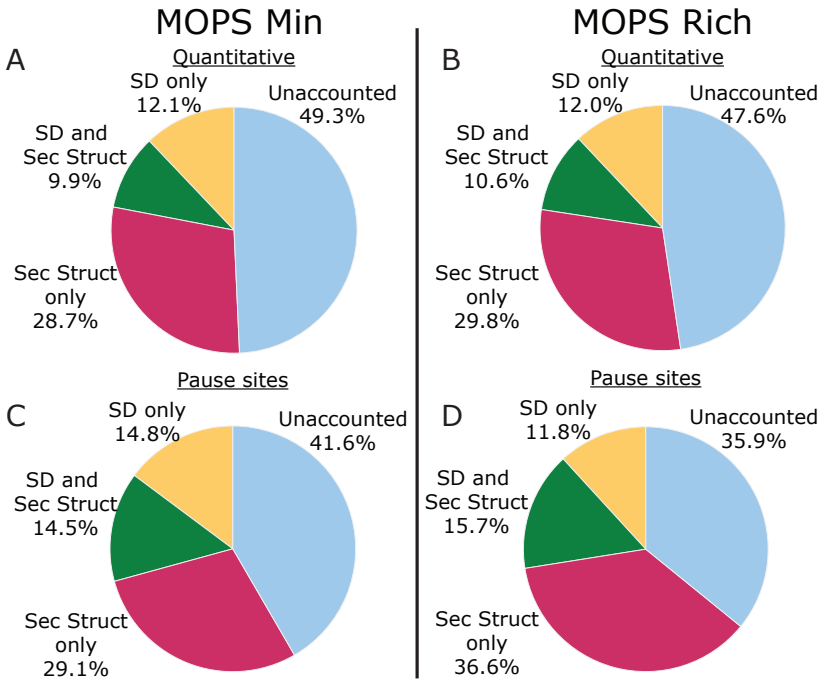


Figure D.4: Percentage of ribosome density and pause sites linked to SD-like sequences and/or Secondary Structure
Pie charts showing the distribution of pause sites linked to secondary structures and/or Shine-Dalgarno-like codons. In both MOPS minimal and MOPS Rich media, codons indicated to be pause-enriched for SD-like sequences accounted for around 20% of ribosomal density (A, B) and 30% of pause sites (C, D), while secondary structures accounted for 40% and 35% respectively. Of these, around 20–25% of these codons are indicated by both SD-like sequences and secondary structures. Around 50% of ribosomal density and 40% of all pause sites were still unaccounted for, indicating that there are other factors linked to pausing which were not included.

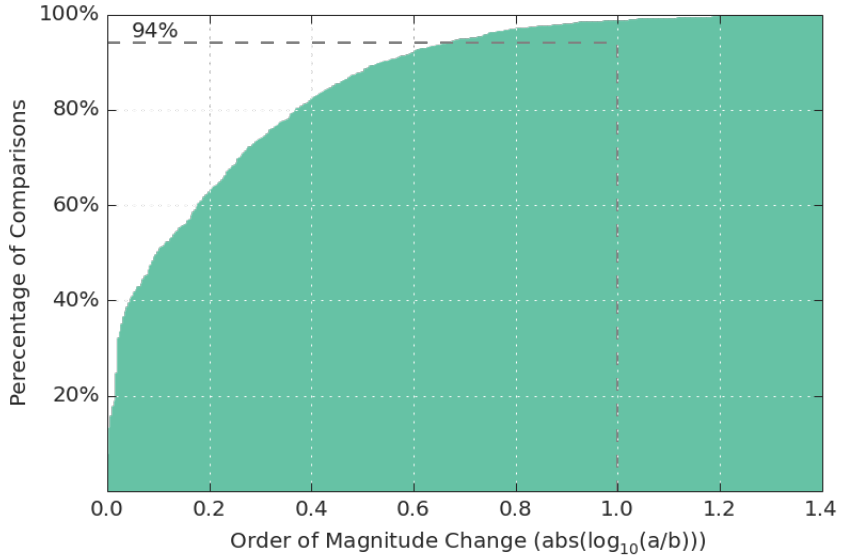


Figure D.5: Distribution of pairwise comparisons between computed k_{eff}. Each of the 284 k_{eff} was compared between conditions (6 comparisons between each of the 4 conditions). Plotted is the cumulative distribution of all these pairwise comparisons in terms of the change in order of magnitude. We observe that 94% of these comparisons remain within an order of magnitude. For the comparisons which were not within an order of magnitude, proteins associated with those complexes were more likely to catalyze multiple reactions (42.5% catalyzing more than one reaction v.s. 27.8% catalyzing more than one reaction).

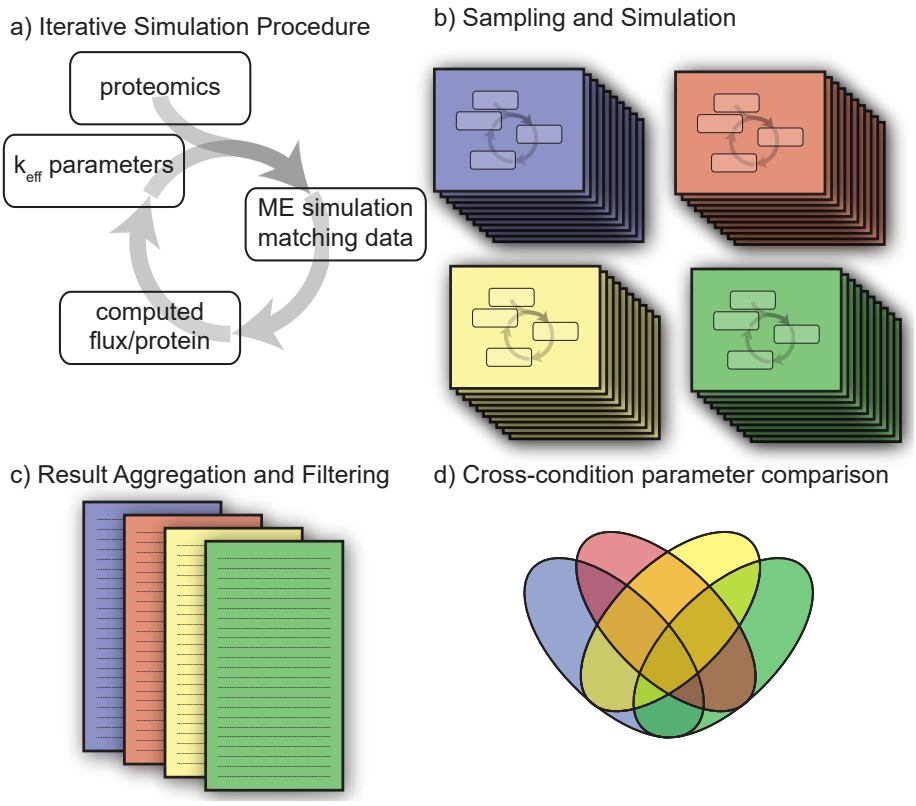


Figure D.6: Overview of the k_{eff} fitting algorithm. This procedure was used to estimate a set of k_{eff} parameters from four proteomics data sets, as described in the methods section.

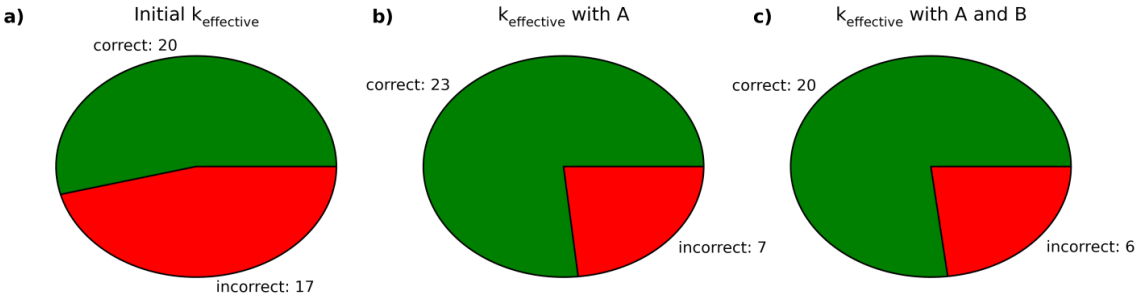


Figure D.7: Predicted Differential Expression between Fumarate and Acetate The ME model was used to predict differential expression between growth on fumarate and growth on acetate as the main carbon substrates. These predictions were run with three different sets of k_{eff} and then validated using mRNA sequencing. This is described in more detail in the "Predictions of mRNA expression under identical conditions to proteomic data" section.

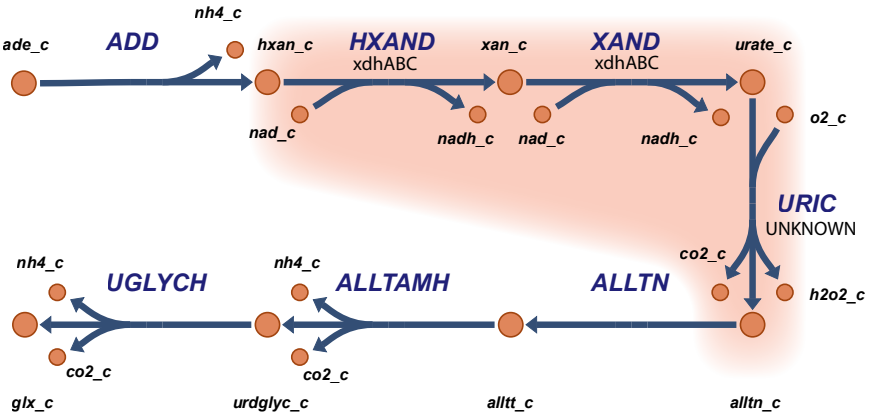


Figure D.8: Updates to the Adenine Degradation Pathway in the *E. coli* Metabolic Reconstruction.
As reconstructed in *i*JO1366 and *i*OL1650-ME, this pathway breaks down intracellular adenine to glyoxylate. However, our expression-profiling data with adenine supplementation suggests that the highlighted reactions do not occur to break down adenine, even though the upstream reaction adenine deaminase does occur. While these reactions were included in the model based on their homology to adenine degradation pathways in other organisms, it is likely that the pathway is latent or inactive in E. coli K-12. Therefore, the reactions HXAND and XAND were disabled during simulations on Adenine. Additionally, use of the unexpressed gene focB was also penalized.

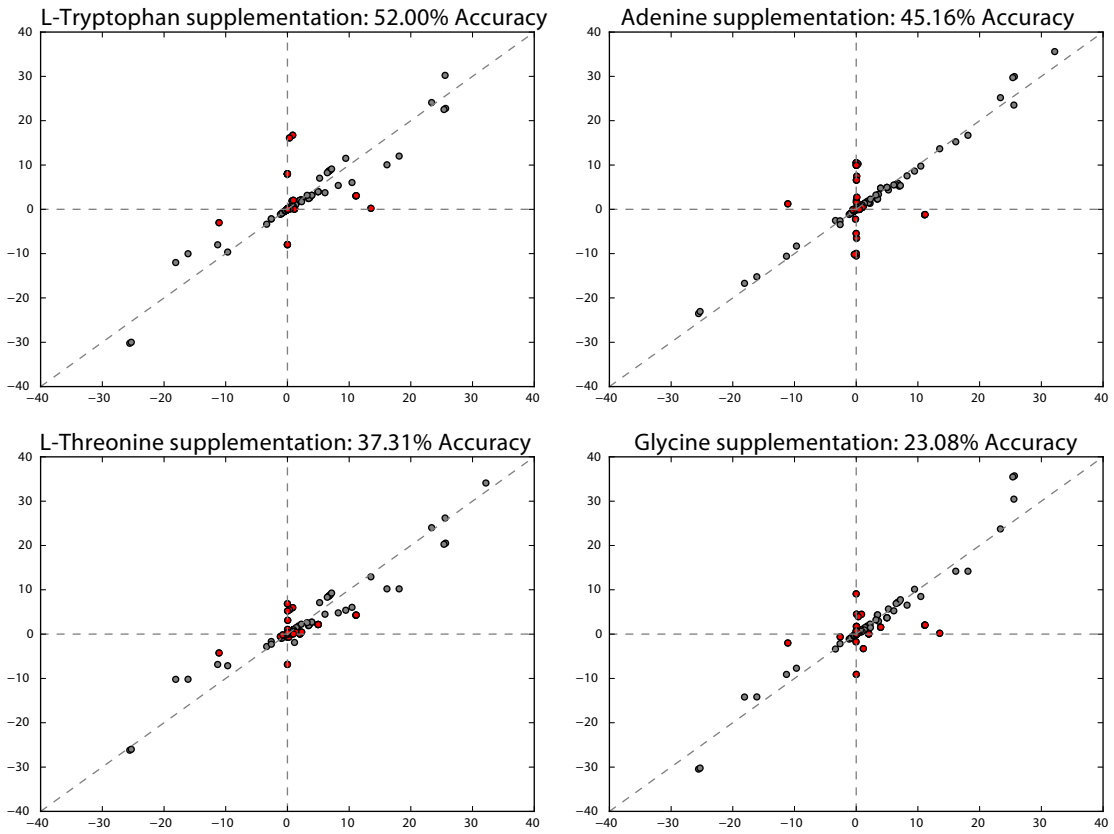


Figure D.9: Sampling *i*JO1366 Flux States Following Nutrient Supplementation.
To compare ME predictions of gene differential expression to that of the M model, sampling was run on *i*JO1366 to identify reactions which changed significantly in flux (colored red) after supplementation. For each of these four conditions, the mean flux from sampling on glucose is plotted on the x axis, and the mean flux from sampling on glucose with the nutrient supplementation is plotted on the y axis. The model gene reaction rules were used to convert these to gene expression predictions, which were compared to mRNA sequencing to give the reported accuracies.

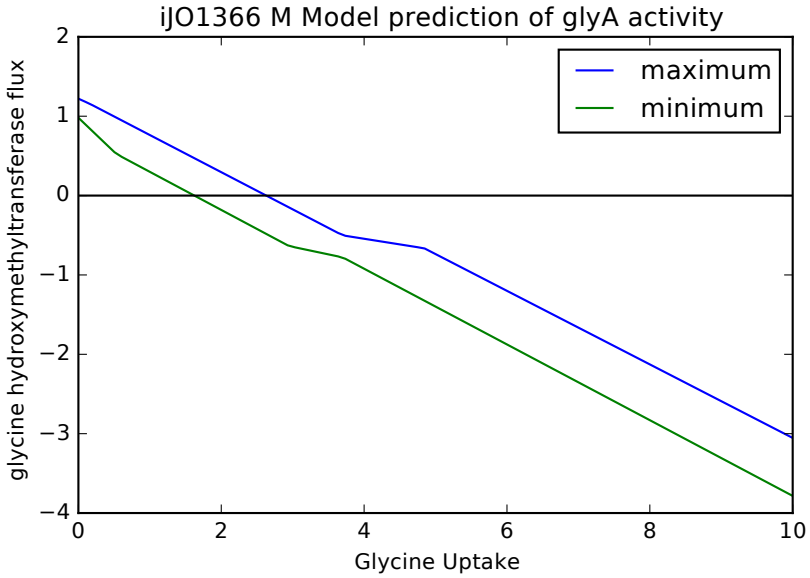


Figure D.10: Opportunistic use of glyA by an M model.
In the *i*JO1366 metabolic model, glyA will catalyze the reversible conversion of glycine to serine using tetrahydrofolate as a methyl carrier. (L-serine + tetrahydrofolate ⇔ methyltetrahydrofolate + h_2o + glycine). When allowing the model to uptake glycine, the optimal solution will (as determined by flux variability analysis at 99.9% of the optimal biomass production) use this reaction to convert glycine to serine, which is not observed in the ME model nor in RNA-sequencing data.

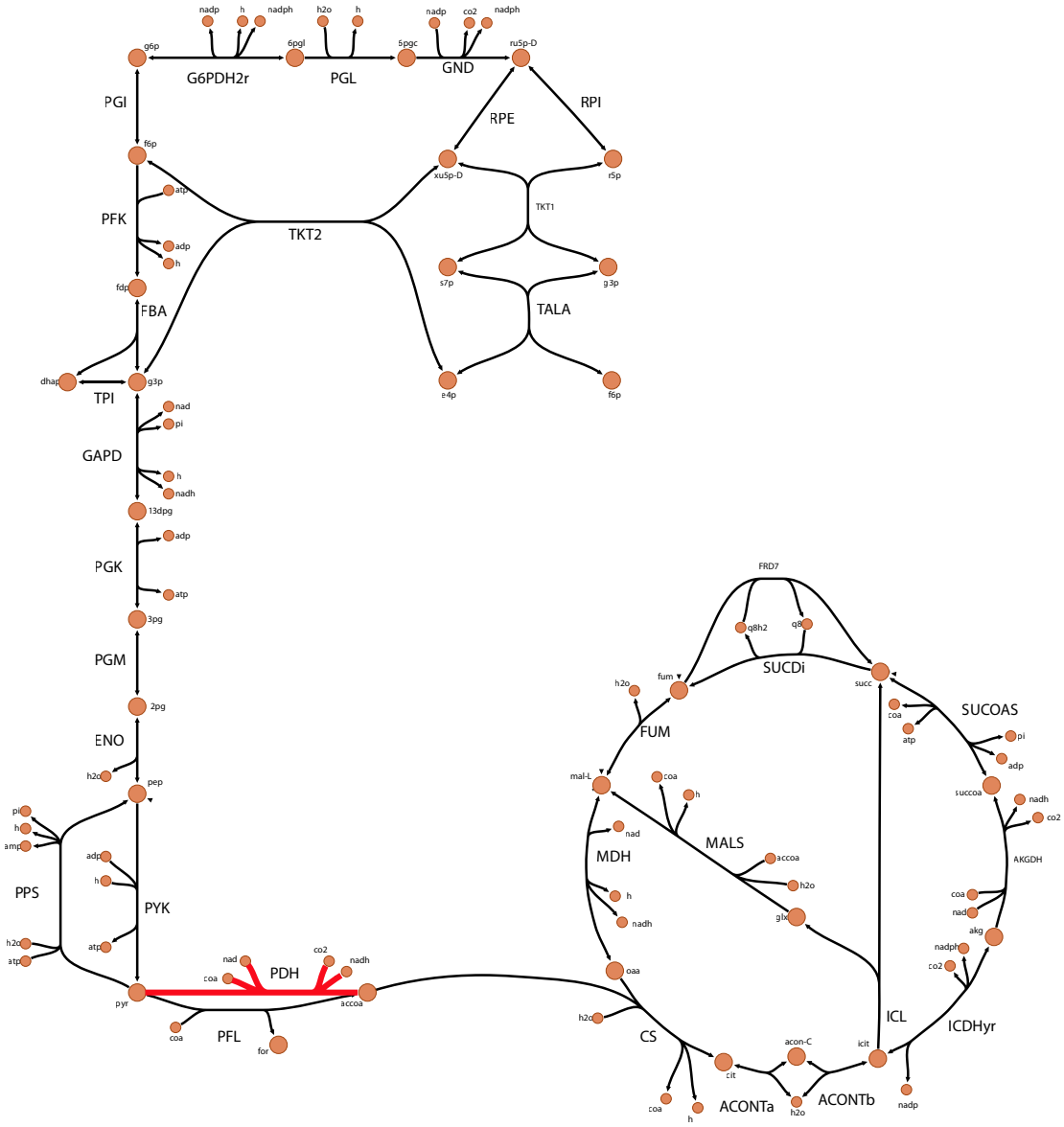


Figure D.11: An illustration of k_{eff} parameterization for a network model. The enzymes pyruvate dehydrogenase (PDH) and pyruvate formate lyase (PFL) both convert pyruvate to acetyl co-A to allow carbon to flow through the TCA cycle. However, flux through PFL will result in secretion of formate, whereas flux through PDH will result in production of CO_2, which is the in vivo behavior. In *E. coli*, PDH is one of the most expensive metabolic enzymes because it consists of 60 subunits. However, an optimally efficient *E. coli* cell might still produce this enzyme over PFL because its catalytic rate is more than commensurately higher, as it exploits phenomena such as substrate tunneling and multiple catalytic sites, and will therefore have a lower protein cost per unit of flux catalyzed. Therefore, in order for an ME model to correctly predict flux through PDH, it must have k_{eff} parameters for PDH and PFL which represents this tradeoff accurately.

Bibliography

[ABB+02] Mariette R Atkinson, Timothy A Blauwkamp, Vladamir Bondarenko, Vasily Studitsky, and Alexander J Ninfa. Activation of the glna, glnk, and nac promoters as escherichia coli undergoes the transition from nitrogen excess growth to nitrogen starvation. J. Bacteriol., 184(19):5358–5363, October 2002.

[ABB+08] Ramy K Aziz, Daniela Bartels, Aaron A Best, Matthew DeJongh, Terrence Disz, Robert A Edwards, Kevin Formsma, Svetlana Gerdes, Elizabeth M Glass, Michael Kubal, Folker Meyer, Gary J Olsen, Robert Olson, Andrei L Osterman, Ross A Overbeek, Leslie K McNeil, Daniel Paarmann, Tobias Paczian, Bruce Parrello, Gordon D Pusch, Claudia Reich, Rick Stevens, Olga Vassieva, Veronika Vonstein, Andreas Wilke, and Olga Zagnitko. The RAST server: rapid annotations using subsystems technology. BMC Genomics, 9:75, 8 February 2008.

[ABN02] Mariette R Atkinson, Timothy A Blauwkamp, and Alexander J Ninfa. Context-dependent functions of the PII and GlnK signal transduction proteins in escherichia coli. J. Bacteriol., 184(19):5364–5375, October 2002.

[ACDE07] David L Applegate, William Cook, Sanjeeb Dash, and Daniel G Espinoza. Exact solutions to linear programming problems. Oper. Res. Lett., 35(6):693–699, November 2007.

[AKV+04] E Almaas, B Kovács, T Vicsek, Z N Oltvai, and A-L Barabási. Global organization of metabolic fluxes in the bacterium escherichia coli. Nature, 427(6977):839–843, February 2004.

[ALS+13] Rasmus Agren, Liming Liu, Saeed Shoaie, Wanwipa Vongsangnak, Intawat Nookaew, and Jens Nielsen. The RAVEN toolbox and its use for generating a genome-scale metabolic model for penicillium chrysogenum. PLoS Comput. Biol., 9(3):e1002980, March 2013.

[AM03] Ruedi Aebersold and Matthias Mann. Mass spectrometry-based proteomics. Nature, 422(6928):198–207, 13 March 2003.

[AVP+12] L Arike, K Valgepea, L Peil, R Nahku, K Adamberg, and R Vilu. Comparison and applications of label-free absolute proteome quantification methods on escherichia coli. J. Proteomics, 75(17):5437–5448, September 2012.

[BBA+03] Brigitte Boeckmann, Amos Bairoch, Rolf Apweiler, Marie-Claude Blatter, Anne Estreicher, Elisabeth Gasteiger, Maria J Martin, Karine Michoud, Claire O'Donovan, Isabelle Phan, Sandrine Pilbout, and Michel Schneider. The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003. Nucleic Acids Res., 31(1):365–370, 1 January 2003.

[BBB+09] Timothy L Bailey, Mikael Boden, Fabian A Buske, Martin Frith, Charles E Grant, Luca Clementi, Jingyuan Ren, Wilfred W Li, and William S Noble. MEME SUITE: tools for motif discovery and searching. Nucleic Acids Res., 37(Web Server issue):W202–8, July 2009.

[BBD+99] C R Beuzón, G Banks, J Deiwick, M Hensel, and D W Holden. ph-dependent secretion of SseB, a product of the SPI-2 type III secretion system of salmonella typhimurium. Mol. Microbiol., 33(4):806–816, August 1999.

[BBP+14] Daniel R Brown, Geraint Barton, Zhensheng Pan, Martin Buck, and Sivaramesh Wigneshweraraj. Nitrogen stress response and stringent response are coupled in escherichia coli. Nat. Commun., 5:4115, 20 June 2014.

[BCC+13] Dennis A Benson, Mark Cavanaugh, Karen Clark, Ilene Karsch-Mizrachi, David J Lipman, James Ostell, and Eric W Sayers. GenBank. Nucleic Acids Res., 41(Database issue):D36–42, January 2013.

[BCP] Edik M Blais, Arvind K Chavali, and Jason A Papin. Linking Genome-Scale metabolic modeling and genome annotation. In Hal S Alper, editor, Systems Metabolic Engineering, Methods in Molecular Biology, pages 61–83. Humana Press.

[BEO94] Timothy L Bailey, Charles Elkan, and Others. Fitting a mixture model by expectation maximization to discover motifs in bipolymers. Proc Int Conf Intell Syst Mol Biol, 1994.

[BKJH08] Benjamin J Bornstein, Sarah M Keating, Akiya Jouraku, and Michael Hucka. LibSBML: an API library for SBML. Bioinformatics, 24(6):880–881, March 2008.

[BMB11] Ahmet Bakan, Lidio M Meireles, and Ivet Bahar. ProDy: protein dynamics inferred from theory and experiments. Bioinformatics, 27(11):1575–1577, 1 June 2011.

[BMKP14] Aarash Bordbar, Jonathan M Monk, Zachary A King, and Bernhard O Palsson. Constraint-based models predict metabolic and associated cellular functions. Nat. Rev. Genet., 15(2):107–120, February 2014.

[BMN+12] Aarash Bordbar, Monica L Mo, Ernesto S Nakayasu, Alexandra C Schrimpe-Rutledge, Young-Mo Kim, Thomas O Metz, Marcus B Jones, Bryan C Frank, Richard D Smith, Scott N Peterson, Daniel R Hyduke, Joshua N Adkins, and Bernhard O Palsson. Model-driven multi-omic data analysis elucidates metabolic immunomodulators of macrophage activation. Mol. Syst. Biol., 8(1), January 2012.

[BN02] Timothy A Blauwkamp and Alexander J Ninfa. Physiological role of the GlnK signal transduction protein of escherichia coli: survival of nitrogen starvation. Mol. Microbiol., 46(1):203–214, October 2002.

[BNL+14] Aarash Bordbar, Harish Nagarajan, Nathan E Lewis, Haythem Latif, Ali Ebrahim, Stephen Federowicz, Jan Schellenberger, and Bernhard O Palsson. Minimal metabolic pathway structure is consistent with associated biomolecular interactions. Mol. Syst. Biol., 10(7):737, 1 July 2014.

[BPM03] Anthony P Burgard, Priti Pharkya, and Costas D Maranas. Optknock: A bilevel programming framework for identifying gene knockout strategies for microbial strain optimization. Biotechnol. Bioeng., 84(6):647–657, October 2003.

[BPS13] Bonnie Berger, Jian Peng, and Mona Singh. Computational solutions for omics data. Nat. Rev. Genet., 14(5):333–346, May 2013.

[BSS01] A Becskei, B Séraphin, and L Serrano. Positive feedback in eukaryotic gene networks: cell differentiation by graded to binary response conversion. EMBO J., 20(10):2528–2535, 15 May 2001.

[BWF+00] H M Berman, J Westbrook, Z Feng, G Gilliland, T N Bhat, H Weissig, I N Shindyalov, and P E Bourne. The protein data bank. Nucleic Acids Res., 28(1):235–242, 1 January 2000.

[CAC+09] Peter J A Cock, Tiago Antao, Jeffrey T Chang, Brad A Chapman, Cymon J Cox, Andrew Dalke, Iddo Friedberg, Thomas Hamelryck, Frank Kauff, Bartek Wilczynski, and Michiel J L de Hoon. Biopython: freely available python tools for computational molecular biology and bioinformatics. Bioinformatics, 25(11):1422–1423, 1 June 2009.

[CAK+13] Roger L Chang, Kathleen Andrews, Donghyuk Kim, Zhanwen Li, Adam Godzik, and Bernhard O Palsson. Structural systems biology evaluation of metabolic thermotolerance in escherichia coli. Science, 340(6137):1220–1223, 7 June 2013.

[CBK+08] Byung-Kwan Cho, Christian L Barrett, Eric M Knight, Young Seoub Park, and Bernhard ØPalsson. Genome-scale reconstruction of the lrp regulatory network in escherichia coli. Proceedings of the National Academy of Sciences, 105(49):19462–19467, December 2008.

[CBZ+09] Caroline Colijn, Aaron Brandes, Jeremy Zucker, Desmond S Lun, Brian Weiner, Maha R Farhat, Tan-Yun Cheng, D Branch Moody, Megan Murray, and James E Galagan. Interpreting expression data with metabolic flux models: predicting mycobacterium tuberculosis mycolic acid production. PLoS Comput. Biol., 5(8):e1000489, August 2009.

[CEL+14] Javier Carrera, Raissa Estrela, Jing Luo, Navneet Rai, Athanasios Tsoukalas, and Ilias Tagkopoulos. An integrative, multi-scale, genome-wide model reveals the phenotypic landscape of escherichia coli. Mol. Syst. Biol., 10(7):735, 1 July 2014.

[CFE+11] Byung-Kwan Cho, Stephen A Federowicz, Mallory Embree, Young-Seoub Park, Donghyuk Kim, and Bernhard ØPalsson. The PurR regulon in escherichia coli K-12 MG1655. Nucleic Acids Res., 39(15):6456–6464, August 2011.

[CFP+11] Byung-Kwan Cho, Stephen Federowicz, Young-Seoub Park, Karsten Zengler, and Bernhard ØPalsson. Deciphering the transcriptional regulatory logic of amino acid metabolism. Nat. Chem. Biol., 8(1):65–71, November 2011.

[CJK+11] Mélanie Courtot, Nick Juty, Christian Knüpfer, Dagmar Waltemath, Anna Zhukova, Andreas Dräger, Michel Dumontier, Andrew Finney, Martin Golebiewski, Janna Hastings, Stefan Hoops, Sarah Keating, Douglas B Kell, Samuel Kerrien, James Lawson, Allyson Lister, James Lu, Rainer Machne, Pedro Mendes, Matthew Pocock, Nicolas Rodriguez, Alice Villeger, Darren J Wilkinson, Sarala Wimalaratne, Camille Laibe, Michael Hucka, and Nicolas Le Novère. Controlled vocabularies and semantics in systems biology. Mol. Syst. Biol., 7:543, 25 October 2011.

[CKBP08] Byung-Kwan Cho, Eric M Knight, Christian L Barrett, and Bernhard ØPalsson. Genome-wide analysis of fis binding in escherichia coli indicates a causative role for A-/AT-tracts. Genome Res., 18(6):900–910, June 2008.

[CKK+14] Byung-Kwan Cho, Donghyuk Kim, Eric M Knight, Karsten Zengler, and Bernhard O Palsson. Genome-scale reconstruction of the sigma factor network in escherichia coli: topology and functional states. BMC Biol., 12:4, 24 January 2014.

[CKP06a] Byung-Kwan Cho, Eric M Knight, and Bernhard O Palsson. PCR-based tandem epitope tagging system for escherichia coli genome engineering. Biotechniques, 40(1):67–72, January 2006.

[CKP06b] Byung-Kwan Cho, Eric M Knight, and Bernhard ØPalsson. Transcriptional regulation of the fad regulon genes of escherichia coli by ArcA. Microbiology, 152(Pt 8):2207–2219, August 2006.

[CMM91] F Claverie-Martin and B Magasanik. Role of integration host factor in the regulation of the glnhp2 promoter of escherichia coli. Proc. Natl. Acad. Sci. U. S. A., 88(5):1631–1635, 1 March 1991.

[CP02] Markus W Covert and Bernhard ØPalsson. Transcriptional regulation in Constraints-Based metabolic models of escherichia coli ,. J. Biol. Chem., 277(31):28058–28064, August 2002.

[CPGO98] L Camarena, S Poggio, N García, and A Osorio. Transcriptional repression of gdha in escherichia coli is mediated by the nac protein. FEMS Microbiol. Lett., 167(1):51–56, 1 October 1998.

[CSG+09] Antje Chang, Maurice Scheer, Andreas Grote, Ida Schomburg, and Dietmar Schomburg. BRENDA, AMENDA and FRENDA the enzyme information system: new content and tools in 2009. Nucleic Acids Res., 37(Database issue):D588–92, January 2009.

[CZQ+09] Byung-Kwan Cho, Karsten Zengler, Yu Qiu, Young Seoub Park, Eric M Knight, Christian L Barrett, Yuan Gao, and Bernhard ØPalsson. The transcription unit architecture of the escherichia coli genome. Nat. Biotechnol., 27(11):1043–1049, November 2009.

[DD10] Shane C Dillon and Charles J Dorman. Bacterial nucleoid-associated proteins, nucleoid structure and gene expression. Nat. Rev. Microbiol., 8(3):185–195, March 2010.

[dGOC+08] Lyris M F de Godoy, Jesper V Olsen, Jürgen Cox, Michael L Nielsen, Nina C Hubner, Florian Fröhlich, Tobias C Walther, and Matthias Mann. Comprehensive mass-spectrometry-based proteome quantification of haploid versus diploid yeast. Nature, 455(7217):1251–1254, 30 October 2008.

[DM12] Guillaume Desnoyers and Eric Massé. Noncanonical repression of translation initiation through small RNA recruitment of the RNA chaperone hfq. Genes Dev., 26(7):726–739, 1 April 2012.

[DP14] Andreas Dräger and Bernhard ØPalsson. Improving collaboration by standardization efforts in systems biology. Front Bioeng Biotechnol, 2:61, 8 December 2014.

[DSWR11] Christopher D Doucette, David J Schwab, Ned S Wingreen, and Joshua D Rabinowitz. α-ketoglutarate coordinates carbon and nitrogen utilization via enzyme I inhibition. Nat. Chem. Biol., 7(12):894–901, December 2011.

[EAB+15] Ali Ebrahim, Eivind Almaas, Eugen Bauer, Aarash Bordbar, Anthony P Burgard, Roger L Chang, Andreas Dräger, Iman Famili, Adam M Feist, Ronan Mt Fleming, Stephen S Fong, Vassily Hatzimanikatis, Markus J Herrgård, Allen Holder, Michael Hucka, Daniel Hyduke, Neema Jamshidi, Sang Yup Lee, Nicolas Le Novère, Joshua A Lerman, Nathan E Lewis, Ding Ma, Radhakrishnan Mahadevan, Costas Maranas, Harish Nagarajan, Ali Navid, Jens Nielsen, Lars K Nielsen, Juan Nogales, Alberto Noronha, Csaba Pal, Bernhard O Palsson, Jason A Papin, Kiran R Patil, Nathan D Price, Jennifer L Reed, Michael Saunders, Ryan S Senger, Nikolaus Sonnenschein, Yuekai Sun, and Ines Thiele. Do genome-scale models need exact solvers or clearer standards? Mol. Syst. Biol., 11(10):831, 14 October 2015.

[ELPH13] Ali Ebrahim, Joshua A Lerman, Bernhard O Palsson, and Daniel R Hyduke. COBRApy: COnstraints-Based reconstruction and analysis for python. BMC Syst. Biol., 7:74, 8 August 2013.

[EV68] J Elovson and P R Vagelos. Acyl carrier protein. x. acyl carrier protein synthetase. J. Biol. Chem., 243(13):3603–3611, 10 July 1968.

[FA95] D Frishman and P Argos. Knowledge-based protein secondary structure assignment. Proteins, 23(4):566–579, December 1995.

[FBC+14] Robert D Finn, Alex Bateman, Jody Clements, Penelope Coggill, Ruth Y Eberhardt, Sean R Eddy, Andreas Heger, Kirstie Hetherington, Liisa Holm, Jaina Mistry, Erik L L Sonnhammer, John Tate, and Marco Punta. Pfam: the protein families database. Nucleic Acids Res., 42(Database issue):D222–30, January 2014.

[FHSW99] K Forchhammer, A Hedler, H Strobel, and V Weiss. Heterotrimerization of PII-like signalling proteins: implications for PII-mediated signal transduction systems. Mol. Microbiol., 33(2):338–349, July 1999.

[Fis99] Susan H Fisher. Regulation of nitrogen metabolism in bacillus subtilis: vive la difference! Mol. Microbiol., 32(2):223–232, 1999.

[FKE+14] Stephen Federowicz, Donghyuk Kim, Ali Ebrahim, Joshua Lerman, Harish Nagarajan, Byung-Kwan Cho, Karsten Zengler, and Bernhard Palsson. Determining the control circuitry of redox metabolism at the genome-scale. PLoS Genet., 10(4):e1004264, April 2014.

[FMT+10] Robert D Finn, Jaina Mistry, John Tate, Penny Coggill, Andreas Heger, Joanne E Pollington, O Luke Gavin, Prasad Gunasekaran, Goran Ceric, Kristoffer Forslund, Liisa Holm, Erik L L Sonnhammer, Sean R Eddy, and Alex Bateman. The pfam protein families database. Nucleic Acids Res., 38(Database issue):D211–22, January 2010.

[FP08] Adam M Feist and Bernhard O Palsson. The growing scope of applications of genome-scale metabolic reconstructions using escherichia coli. Nat. Biotechnol., 26(6):659–667, June 2008.

[FZF+11] Christian Frezza, Liang Zheng, Ori Folger, Kartik N Rajagopalan, Elaine D MacKenzie, Livnat Jerby, Massimo Micaroni, Barbara Chaneton, Julie Adam, Ann Hedley, Gabriela Kalna, Ian P M Tomlinson, Patrick J Pollard, Dave G Watson, Ralph J Deberardinis, Tomer Shlomi, Eytan Ruppin, and Eyal Gottlieb. Haem oxygenase is synthetically lethal with the tumour suppressor fumarate hydratase. Nature, 477(7363):225–228, 8 September 2011.

[GCJJPG+08] Socorro Gama-Castro, Verónica Jiménez-Jacinto, Martín Peralta-Gil, Alberto Santos-Zavaleta, Mónica I Peñaloza Spinola, Bruno Contreras-Moreira, Juan Segura-Salazar, Luis Muñiz Rascado, Irma Martínez-Flores, Heladia Salgado, César Bonavides-Martínez, Cei Abreu-Goodger, Carlos Rodríguez-Penagos, Juan Miranda-Ríos, Enrique Morett, Enrique Merino, Araceli M Huerta, Luis Treviño Quintanilla, and Julio Collado-Vides. RegulonDB (version 6.0): gene regulation model of escherichia coli K-12 beyond transcription, active (experimental) annotated promoters and textpresso navigation. Nucleic Acids Res., 36(Database issue):D120–4, January 2008.

[GCWG03] Dov Greenbaum, Christopher Colangelo, Kenneth Williams, and Mark Gerstein. Comparing protein abundance and mRNA expression levels on a genomic scale. Genome Biol., 4(9):117, August 2003.

[GDDFL13] Gabriel Gelius-Dietrich, Abdelmoneim Amer Desouki, Claus Jonathan Fritzemeier, and Martin J Lercher. sybil – efficient constraint-based modelling in R. BMC Syst. Biol., 7(1):1–8, 2013.

[GJPP09] Erwin P Gianchandani, Andrew R Joyce, Bernhard ØPalsson, and Jason A Papin. Functional states of the genome-scale escherichia coli transcriptional regulatory system. PLoS Comput. Biol., 5(6):e1000403, June 2009.

[GR83] E Garcia and S G Rhee. Cascade control of escherichia coli glutamine synthetase. purification and properties of PII uridylyltransferase and uridylyl-removing enzyme. J. Biol. Chem., 258(4):2246–2253, 25 February 1983.

[GUN+15] Gabriela I Guzmán, José Utrilla, Sergey Nurk, Elizabeth Brunk, Jonathan M Monk, Ali Ebrahim, Bernhard O Palsson, and Adam M Feist. Model-driven discovery of underground metabolic functions in escherichia coli. Proc. Natl. Acad. Sci. U. S. A., 112(3):929–934, 20 January 2015.

[GYY+14] Justin Gardin, Rukhsana Yeasmin, Alisa Yurovsky, Ying Cai, Steve Skiena, and Bruce Futcher. Measurement of average decoding rates of the 61 sense codons in vivo. Elife, 3, 27 October 2014.

[HBH07] Christopher S Henry, Linda J Broadbelt, and Vassily Hatzimanikatis. Thermodynamics-Based metabolic flux analysis. Biophys. J., 92(5):1792–1805, March 2007.

[HDB+10] Christopher S Henry, Matthew DeJongh, Aaron A Best, Paul M Frybarger, Ben Linsay, and Rick L Stevens. High-throughput generation, optimization and analysis of genome-scale metabolic models. Nat. Biotechnol., 28(9):977–982, September 2010.

[HDS96] W Humphrey, A Dalke, and K Schulten. VMD: visual molecular dynamics. J. Mol. Graph., 14(1):33–8, 27–8, February 1996.

[HFB+04] M Hucka, A Finney, B J Bornstein, S M Keating, B E Shapiro, J Matthews, B L Kovitz, M J Schilstra, A Funahashi, J C Doyle, and H Kitano. Evolving a lingua franca and associated software infrastructure for computational systems biology: the systems biology markup language (SBML) project. Syst. Biol., 1(1):41–53, June 2004.

[HFS+03] M Hucka, A Finney, H M Sauro, H Bolouri, J C Doyle, H Kitano, A P Arkin, B J Bornstein, D Bray, A Cornish-Bowden, A A Cuellar, S Dronov, E D Gilles, M Ginkel, V Gor, I I Goryanin, W J Hedley, T C Hodgman, J-H Hofmeyr, P J Hunter, N S Juty, J L Kasberger, A Kremling, U Kummer, N Le Novère, L M Loew, D Lucio, P Mendes, E Minch, E D Mjolsness, Y Nakayama, M R Nelson, P F Nielsen, T Sakurada, J C Schaff, B E Shapiro, T S Shimizu, H D Spence, J Stelling, K Takahashi, M Tomita, J Wagner, and J Wang. The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. Bioinformatics, 19(4):524–531, March 2003.

[HG14] Karan S Hingorani and Lila M Gierasch. Comparing protein folding in vitro and in vivo: foldability meets the fitness challenge. Curr. Opin. Struct. Biol., 24:81–90, February 2014.

[HHG+11] Andreas Hoppe, Sabrina Hoffmann, Andreas Gerasch, Christoph Gille, and Hermann-Georg Holzhütter. FASIMU: flexible software

for flux-balance computation series in large metabolic networks. BMC Bioinformatics, 12:28, 22 January 2011.

[HJT+07] Daniel R Hyduke, Laura R Jarboe, Linh M Tran, Katherine J Y Chou, and James C Liao. Integrated network analysis identifies nitric oxide response networks and dihydroxyacid dehydratase as a crucial target in escherichia coli. Proc. Natl. Acad. Sci. U. S. A., 104(20):8484–8489, 15 May 2007.

[HLP13] Daniel R Hyduke, Nathan E Lewis, and Bernhard ØPalsson. Analysis of omics data with genome-scale models of metabolism. Mol. Biosyst., 9(2):167–174, 2 February 2013.

[HNY+06] Yi-Xin Huo, Bei-Yan Nan, Cong-Hui You, Zhe-Xian Tian, Annie Kolb, and Yi-Ping Wang. FIS activates glnap2 in escherichia coli: role of a DNA bend centered at -55, upstream of the transcription start site. FEMS Microbiol. Lett., 257(1):99–105, April 2006.

[HP10] Daniel R Hyduke and Bernhard ØPalsson. Towards genome-scale signalling network reconstructions. Nat. Rev. Genet., 11(4):297–307, April 2010.

[HSLI15] Anne-Katrin Hess, Paul Saffert, Klaus Liebeton, and Zoya Ignatova. Optimization of translation profiles enhances protein expression and solubility. PLoS One, 10(5):e0127039, 12 May 2015.

[IGNW09] Nicholas T Ingolia, Sina Ghaemmaghami, John R S Newman, and Jonathan S Weissman. Genome-Wide analysis in vivo of translation with nucleotide resolution using ribosome profiling. Science, 324(5924):218–223, 10 April 2009.

[ISK96] T P Ikeda, A E Shauger, and S Kustu. Salmonella typhimurium apparently perceives external nitrogen limitation as internal glutamine limitation. J. Mol. Biol., 259(4):589–607, 21 June 1996.

[JP06] Andrew R Joyce and Bernhard ØPalsson. The model organism as a system: integrating 'omics' data sets. Nat. Rev. Mol. Cell Biol., 7(3):198–210, March 2006.

[JPN98] P Jiang, J A Peliska, and A J Ninfa. Enzymological characterization of the signal-transducing uridylyltransferase/uridylyl-removing enzyme (EC 2.7.7.59) of escherichia coli and its interaction with the PII protein. Biochemistry, 37(37):12782–12794, 15 September 1998.

[JSTM04] Arnaud Javelle, Emmanuele Severi, Jeremy Thornton, and Mike Merrick. Ammonium sensing in escherichia coli. role of the ammonium transporter AmtB and AmtB-GlnK complex formation. J. Biol. Chem., 279(10):8530–8538, 5 March 2004.

[JvMG12] David Joyner, Ondřej Čertík, Aaron Meurer, and Brian E Granger. Open source computer algebra systems: SymPy. ACM Commun. Comput. Algebra, 45(3/4):225–234, 23 January 2012.

[KBMCV+09] Ingrid M Keseler, César Bonavides-Martínez, Julio Collado-Vides, Socorro Gama-Castro, Robert P Gunsalus, D Aaron Johnson, Markus Krummenacker, Laura M Nolan, Suzanne Paley, Ian T Paulsen, Martin Peralta-Gil, Alberto Santos-Zavaleta, Alexander Glennon Shearer, and Peter D Karp. EcoCyc: a comprehensive view of escherichia coli biology. Nucleic Acids Res., 37(Database issue):D464–70, January 2009.

[KDE+15] Zachary A King, Andreas Dräger, Ali Ebrahim, Nikolaus Sonnenschein, Nathan E Lewis, and Bernhard O Palsson. Escher: A web application for building, sharing, and embedding Data-Rich visualizations of biological pathways. PLoS Comput. Biol., 11(8):e1004321, August 2015.

[KHQ+12] Donghyuk Kim, Jay Sung-Joong Hong, Yu Qiu, Harish Nagarajan, Joo-Hyun Seo, Byung-Kwan Cho, Shih-Feng Tsai, and Bernhard ØPalsson. Comparative analysis of regulatory elements between escherichia coli and klebsiella pneumoniae by genome-wide transcription start site profiling. PLoS Genet., 8(8):e1002867, 9 August 2012.

[KKMA06] Hiroshi Kawamoto, Yukari Koide, Teppei Morita, and Hiroji Aiba. Base-pairing requirement for RNA silencing by a bacterial small RNA and acceleration of duplex formation by hfq. Mol. Microbiol., 61(4):1013–1022, August 2006.

[KLD+15] Zachary A King, Justin Lu, Andreas Dräger, Philip Miller, Stephen Federowicz, Joshua A Lerman, Ali Ebrahim, Bernhard O Palsson, and Nathan E Lewis. BiGG models: A platform for integrating, standardizing and sharing genome-scale models. Nucleic Acids Res., 17 October 2015.

[KLR99] A A Komar, T Lesnik, and C Reiss. Synonymous codon substitutions affect ribosome traffic and protein folding during in vitro translation. FEBS Lett., 462(3):387–391, 3 December 1999.

[KMC+14] David Kentner, Giuseppe Martano, Morgane Callon, Petra Chiquet, Maj Brodmann, Olga Burton, Asa Wahlander, Paolo Nanni, Nathanaël Delmotte, Jonas Grossmann, Julien Limenitakis, Ralph Schlapbach, Patrick Kiefer, Julia A Vorholt, Sebastian Hiller, and Dirk Bumann. Shigella reroutes host cell central metabolism to obtain high-flux nutrient supply for vigorous intracellular growth. Proc. Natl. Acad. Sci. U. S. A., 111(27):9929–9934, 8 July 2014.

[KMPG+13] Ingrid M Keseler, Amanda Mackie, Martin Peralta-Gil, Alberto Santos-Zavaleta, Socorro Gama-Castro, César Bonavides-Martínez, Carol Fulcher, Araceli M Huerta, Anamika Kothari, Markus Krummenacker, Mario Latendresse, Luis Muñiz Rascado, Quang Ong, Suzanne Paley, Imke Schröder, Alexander G Shearer, Pallavi Subhraveti, Mike Travers, Deepika Weerasinghe, Verena Weiss, Julio Collado-Vides, Robert P Gunsalus, Ian Paulsen, and Peter D Karp. EcoCyc: fusing model organism databases with systems biology. Nucleic Acids Res., 41(Database issue):D605–12, January 2013.

[KRaSN12] Il-Kwon Kim, António Roldão, Verena Siewers, and Jens Nielsen. A systems-level approach for metabolic engineering of yeast cell factories. FEMS Yeast Res., 12(2):228–248, March 2012.

[KSM+12] Jonathan R Karr, Jayodita C Sanghvi, Derek N Macklin, Miriam V Gutschow, Jared M Jacobs, Benjamin Bolival, Nacyra Assad-Garcia, John I Glass, and Markus W Covert. A Whole-Cell computational model predicts phenotype from genotype. Cell, 150(2):389–401, July 2012.

[KvK11] Steffen Klamt and Axel von Kamp. An application programming interface for CellNetAnalyzer. Biosystems., 105(2):162–168, August 2011.

[Lau] Laurence Yang, Ding Ma, Ali Ebrahim,Colton J. Lloyd, Michael A. Saunders, Bernhard O. Palsson. solveME: fast and reliable solution of nonlinear ME models. Under Review.

[LBGW14] Gene-Wei Li, David Burkhardt, Carol Gross, and Jonathan S Weissman. Quantifying absolute protein synthesis rates reveals principles underlying allocation of cellular resources. Cell, 157(3):624–635, April 2014.

[LBY+03] James C Liao, Riccardo Boscolo, Young-Lyeol Yang, Linh My Tran, Chiara Sabatti, and Vwani P Roychowdhury. Network component analysis: reconstruction of regulatory signals in biological systems. Proc. Natl. Acad. Sci. U. S. A., 100(26):15522–15527, 23 December 2003.

[LDR+10] Chen Li, Marco Donizelli, Nicolas Rodriguez, Harish Dharuri, Lukas Endler, Vijayalakshmi Chelliah, Lu Li, Enuo He, Arnaud Henry, Melanie I Stefan, Jacky L Snoep, Michael Hucka, Nicolas Le Novère, and Camille Laibe. BioModels database: An enhanced, curated and annotated resource for published quantitative kinetic models. BMC Syst. Biol., 4:92, 29 June 2010.

[LHC+13] Lei Li, Dandan Huang, Man Kit Cheung, Wenyan Nong, Qianli Huang, and Hoi Shan Kwan. BSRD: a repository for bacterial small regulatory RNA. Nucleic Acids Res., 41(Database issue):D233–8, January 2013.

[LHL+12] Joshua A Lerman, Daniel R Hyduke, Haythem Latif, Vasiliy A Portnoy, Nathan E Lewis, Jeffrey D Orth, Alexandra C Schrimpe-Rutledge, Richard D Smith, Joshua N Adkins, Karsten Zengler, and Bernhard O Palsson. In silico method for modelling metabolism and gene product expression at genome scale. Nat. Commun., 3:929, July 2012.

[LKCL14] Meiyappan Lakshmanan, Geoffrey Koh, Bevan K S Chung, and Dong-Yup Lee. Software applications for flux balance analysis. Brief. Bioinform., 15(1):108–122, January 2014.

[LM95] J Liu and B Magasanik. Activation of the dephosphorylation of nitrogen regulator i-phosphate of escherichia coli. J. Bacteriol., 177(4):926–931, February 1995.

[LNP12] Nathan E Lewis, Harish Nagarajan, and Bernhard O Palsson. Constraining the metabolic genotype–phenotype relationship using a phylogeny of in silico methods. Nat. Rev. Microbiol., 10(4):291–305, April 2012.

[LOBL+14] Joanne K Liu, Edward J O Brien, Joshua A Lerman, Karsten Zengler, Bernhard O Palsson, and Adam M Feist. Reconstruction and modeling protein translocation and compartmentalization in escherichia coli at the genome-scale. BMC Syst. Biol., 8(1):110, 18 September 2014.

[LOW12] Gene-Wei Li, Eugene Oh, and Jonathan S Weissman. The anti-Shine-Dalgarno sequence drives translational pausing and codon choice in bacteria. Nature, March 2012.

[LS12] Ben Langmead and Steven L Salzberg. Fast gapped-read alignment with bowtie 2. Nat. Methods, 9(4):357–359, April 2012.

[LSO+15] Ryan A LaCroix, Troy E Sandberg, Edward J O'Brien, Jose Utrilla, Ali Ebrahim, Gabriela I Guzman, Richard Szubin, Bernhard O Palsson, and Adam M Feist. Use of adaptive laboratory evolution to discover key mutations enabling rapid growth of escherichia coli K-12 MG1655 on glucose minimal medium. Appl. Environ. Microbiol., 81(1):17–30, January 2015.

[LST+15] Haythem Latif, Richard Szubin, Justin Tan, Elizabeth Brunk, Anna Lechner, Karsten Zengler, and Bernhard O Palsson. A streamlined ribosome profiling protocol for the characterization of microorganisms. Biotechniques, 58(6):329–332, June 2015.

[LTPS09] Ben Langmead, Cole Trapnell, Mihai Pop, and Steven L Salzberg. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biol., 10(3):R25, 2009.

[LVK+10] Jon M Laurent, Christine Vogel, Taejoon Kwon, Stephanie A Craig, Daniel R Boutz, Holly K Huse, Kazunari Nozue, Harkamal Walia, Marvin Whiteley, Pamela C Ronald, and Edward M Marcotte. Protein abundances are more conserved than mRNA abundances across diverse taxa. Proteomics, 10(23):4209–4212, December 2010.

[LYA+10] Joshua Z Levin, Moran Yassour, Xian Adiconis, Chad Nusbaum, Dawn Anne Thompson, Nir Friedman, Andreas Gnirke, and Aviv Regev. Comprehensive comparative analysis of strand-specific RNA sequencing methods. Nat. Methods, 7(9):709–715, September 2010.

[MA03] S Mangan and U Alon. Structure and function of the feed-forward loop network motif. Proc. Natl. Acad. Sci. U. S. A., 100(21):11980–11985, 14 October 2003.

[Mar11] Marcel Martin. Cutadapt removes adapter sequences from high-throughput sequencing reads. EMBnet.journal, 17(1):10–12, 2 May 2011.

[Mau92] M R Maurizi. Proteases and protein degradation in escherichia coli. Experientia, 48(2):178–201, 15 February 1992.

[MB98] W B Muse and R A Bender. The nac (nitrogen assimilation control) gene from escherichia coli. J. Bacteriol., 180(5):1166–1173, March 1998.

[MBHC95] A G Murzin, S E Brenner, T Hubbard, and C Chothia. SCOP: a structural classification of proteins database for the investigation of sequences and structures. J. Mol. Biol., 247(4):536–540, 7 April 1995.

[MBSB88] C Maier, E Bremer, A Schmid, and R Benz. Pore-forming activity of the tsx protein from the outer membrane of escherichia coli. demonstration of a nucleoside-specific binding site. J. Biol. Chem., 263(5):2493–2499, 15 February 1988.

[MCA+13] Jonathan M Monk, Pep Charusanti, Ramy K Aziz, Joshua A Lerman, Ned Premyodhin, Jeffrey D Orth, Adam M Feist, and Bernhard ØPalsson. Genome-scale metabolic reconstructions of multiple escherichia coli strains highlight strain-specific adaptations to nutritional environments. Proceedings of the National Academy of Sciences, page 201307797, November 2013.

[MCK+11] Sheetal R Modi, Diogo M Camacho, Michael A Kohanski, Graham C Walker, and James J Collins. Functional characterization of bacterial sRNAs using a network biology approach. Proc. Natl. Acad. Sci. U. S. A., 108(37):15522–15527, 13 September 2011.

[McK12] Wes McKinney. Python for data analysis: Data wrangling with Pandas, NumPy, and IPython. " O'Reilly Media, Inc.", 2012.

[MDP+09] Rachel A Mooney, Sarah E Davis, Jason M Peters, Jennifer L Rowland, Aseem Z Ansari, and Robert Landick. Regulator trafficking on bacterial transcription units in vivo. Mol. Cell, 33(1):97–108, 16 January 2009.

[MGK+14] Douglas McCloskey, Jon A Gangoiti, Zachary A King, Robert K Naviaux, Bruce A Barshop, Bernhard O Palsson, and Adam 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. Biotechnol. Bioeng., 111(4):803–815, 2014.

[MHM14] Wout Megchelenbrink, Martijn Huynen, and Elena Marchiori. optGpSampler: an improved tool for uniformly sampling the solution-space of genome-scale metabolic networks. PLoS One, 9(2):e86587, 14 February 2014.

[MLJC87] Susan Miller, Arthur M Lesk, Joël Janin, and Cyrus Chothia. The accessible surface area and stability of oligomeric proteins. Nature, 328(6133):834–836, August 1987.

[MNP14] Jonathan Monk, Juan Nogales, and Bernhard O Palsson. Optimizing genome-scale network reconstructions. Nat. Biotechnol., 32(5):447–452, May 2014.

[MPF13] Douglas McCloskey, Bernhard ØPalsson, and Adam M Feist. Basic and applied uses of genome-scale metabolic network reconstructions of escherichia coli. Mol. Syst. Biol., 9:661, 2013.

[MRB03] Wilson B Muse, Christopher J Rosario, and Robert A Bender. Nitrogen regulation of the codBA (cytosine deaminase) operon from escherichia coli by the nitrogen assimilation control protein, NAC. J. Bacteriol., 185(9):2920–2926, May 2003.

[MS03] R Mahadevan and C H Schilling. The effects of alternate optimal solutions in constraint-based genome-scale metabolic models. Metab. Eng., 5(4):264–276, October 2003.

[MTH15] Andrew T Martens, James Taylor, and Vincent J Hilser. Ribosome a and P sites revealed by length analysis of ribosome profiling data. Nucleic Acids Res., 43(7):3680–3687, 20 April 2015.

[MvRTB12] Marnix H Medema, Renske van Raaphorst, Eriko Takano, and Rainer Breitling. Computational tools for the synthetic design of biochemical pathways. Nat. Rev. Microbiol., 10(3):191–202, March 2012.

[MYO+13] Kevin S Myers, Huihuang Yan, Irene M Ong, Dongjun Chung, Kun Liang, Frances Tran, Sündüz Keleş, Robert Landick, and Patricia J Kiley. Genome-scale analysis of escherichia coli FNR reveals complex features of transcription factor binding. PLoS Genet., 9(6):e1003565, June 2013.

[MZA03] S Mangan, A Zaslaver, and U Alon. The coherent feedforward loop serves as a sign-sensitive delay element in transcription networks. J. Mol. Biol., 334(2):197–204, 21 November 2003.

[NA00] A J Ninfa and M R Atkinson. PII signal transduction proteins. Trends Microbiol., 8(4):172–179, April 2000.

[NK71] K Nath and A L Koch. Protein degradation in escherichia coli. II. strain differences in the degradation of protein and nucleic acid resulting from starvation. J. Biol. Chem., 246(22):6956–6967, 25 November 1971.

[NKR+13] Pavel S Novichkov, Alexey E Kazakov, Dmitry A Ravcheev, Semen A Leyn, Galina Y Kovaleva, Roman A Sutormin, Marat D Kazanov, William Riehl, Adam P Arkin, Inna Dubchak, and Dmitry A Rodionov. RegPrecise 3.0–a resource for genome-scale exploration of transcriptional regulation in bacteria. BMC Genomics, 14:745, 1 November 2013.

[NRM87] A J Ninfa, L J Reitzer, and B Magasanik. Initiation of transcription at the bacterial glnap2 promoter by purified e. coli components is facilitated by enhancers. Cell, 50(7):1039–1046, 25 September 1987.

[NSS06] Annik Nanchen, Alexander Schicker, and Uwe Sauer. Nonlinear dependency of intracellular fluxes on growth rate in miniaturized continuous cultures of escherichia coli. Appl. Environ. Microbiol., 72(2):1164–1172, February 2006.

[OCN+11] Jeffrey D Orth, Tom M Conrad, Jessica Na, Joshua A Lerman, Hojung Nam, Adam M Feist, and Bernhard ØPalsson. A comprehensive genome-scale reconstruction of escherichia coli metabolism–2011. Mol. Syst. Biol., 7:535, 11 October 2011.

[OLC+13] Edward J O'Brien, Joshua A Lerman, Roger L Chang, Daniel R Hyduke, and Bernhard ØPalsson. Genome-scale models of metabolism and gene expression extend and refine growth phenotype prediction. Mol. Syst. Biol., 9:693, 2013.

[OMP15] Edward J O'Brien, Jonathan M Monk, and Bernhard O Palsson. Using genome-scale models to predict biological capabilities. Cell, 161(5):971–987, 21 May 2015.

[OP12] Jeffrey D Orth and BernhardøPalsson. Gap-filling analysis of the iJO1366 escherichia coli metabolic network reconstruction for discovery of metabolic functions. BMC Syst. Biol., 6:30, 1 May 2012.

[ORH05] Brett G Olivier, Johann M Rohwer, and Jan-Hendrik S Hofmeyr. Modelling cellular systems with PySCeS. Bioinformatics, 21(4):560–561, February 2005.

[OTP10] Jeffrey D Orth, Ines Thiele, and Bernhard ØPalsson. What is flux balance analysis? Nat. Biotechnol., 28(3):245–248, March 2010.

[PAA+13] Dan M Park, Md Sohail Akhtar, Aseem Z Ansari, Robert Landick, and Patricia J Kiley. The bacterial response regulator ArcA uses a diverse binding site architecture to regulate carbon oxidation globally. PLoS Genet., 9(10):e1003839, 17 October 2013.

[PBS+87] I J Purvis, A J Bettany, T C Santiago, J R Coggins, K Duncan, R Eason, and A J Brown. The efficiency of folding of some proteins is increased by controlled rates of translation in vivo. a hypothesis. J. Mol. Biol., 193(2):413–417, 20 January 1987.

[PCI+95] B S Powell, D L Court, T Inada, Y Nakamura, V Michotey, X Cui, A Reizer, M H Saier, Jr, and J Reizer. Novel proteins of the phosphotransferase system encoded within the rpon operon of escherichia coli. enzyme IIANtr affects growth on organic nitrogen and the conditional lethality of an erats mutant. J. Biol. Chem., 270(9):4822–4839, 3 March 1995.

[PJB98] Pablo J Pomposiello, Brian K Janes, and Robert A Bender. Two roles for the DNA recognition site of theklebsiella aerogenes nitrogen assimilation control protein. J. Bacteriol., 180(3):578–585, 1 February 1998.

[PK97] J Pramanik and J D Keasling. Stoichiometric model of escherichia coli metabolism: incorporation of growth-rate dependent biomass composition and mechanistic energy requirements. Biotechnol. Bioeng., 56(4):398–421, 1997.

[PKA+12] Ana I Prieto, Christina Kahramanoglou, Ruhi M Ali, Gillian M Fraser, Aswin S N Seshasayee, and Nicholas M Luscombe. Genomic analysis of DNA binding and gene regulation by homologous nucleoid-associated proteins IHF and HU in escherichia coli K12. Nucleic Acids Res., 40(8):3524–3537, April 2012.

[PLR00] G M Pupo, R Lan, and P R Reeves. Multiple independent origins of shigella clones of escherichia coli and convergent evolution of many of their characteristics. Proc. Natl. Acad. Sci. U. S. A., 97(19):10567–10572, 12 September 2000.

[PN05] Kiran Raosaheb Patil and Jens Nielsen. Uncovering transcriptional regulation of metabolism by using metabolic network topology. Proc. Natl. Acad. Sci. U. S. A., 102(8):2685–2689, 22 February 2005.

[PVG+11] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. J. Mach. Learn. Res., 12:2825–2830, November 2011.

[Rei03] Larry Reitzer. Nitrogen assimilation and global regulation in escherichia coli. Annu. Rev. Microbiol., 57:155–176, 1 May 2003.

[RP11] Ho Sung Rhee and B Franklin Pugh. Comprehensive genome-wide Protein-DNA interactions detected at Single-Nucleotide resolution. Cell, 147(6):1408–1419, December 2011.

[RP12a] Ho Sung Rhee and B Franklin Pugh. ChIP-exo method for identifying genomic location of DNA-binding proteins with near-single-nucleotide accuracy. Curr. Protoc. Mol. Biol., Chapter 21:Unit 21.24, October 2012.

[RP12b] Ho Sung Rhee and B Franklin Pugh. Genome-wide structure and organization of eukaryotic pre-initiation complexes. Nature, 483(7389):295–301, March 2012.

[RPC+06] Jennifer L Reed, Trina R Patel, Keri H Chen, Andrew R Joyce, Margaret K Applebee, Christopher D Herring, Olivia T Bui, Eric M Knight, Stephen S Fong, and Bernhard O Palsson. Systems approach to refining genome annotation. Proceedings of the National Academy of Sciences, 103(46):17480–17484, November 2006.

[RR15] Aarthi Ravikrishnan and Karthik Raman. Critical assessment of genome-scale metabolic networks: the need for a unified standard. Brief. Bioinform., 17(1), 28 February 2015.

[SB00] D J Studholme and M Buck. The biology of enhancer-dependent transcriptional regulation in bacteria: insights from genome sequences. FEMS Microbiol. Lett., 186(1):1–9, 1 May 2000.

[SBCC13] Aurelien A Serandour, Gordon D Brown, Joshua D Cohen, and Jason S Carroll. Development of an illumina-based ChIP-exonuclease method provides insight into FoxA1-DNA binding properties. Genome Biol., 14(12):R147, 27 December 2013.

[SBL+11] Björn Schwanhäusser, Dorothea Busse, Na Li, Gunnar Dittmar, Johannes Schuchhardt, Jana Wolf, Wei Chen, and Matthias Selbach. Global quantification of mammalian gene expression control. Nature, 473(7347):337–342, May 2011.

[SCE+04] Ida Schomburg, Antje Chang, Christian Ebeling, Marion Gremse, Christian Heldt, Gregor Huhn, and Dietmar Schomburg. BRENDA, the enzyme database: updates and major new developments. Nucleic Acids Res., 32(Database issue):D431–3, 1 January 2004.

[SD10] Rhodri Saunders and Charlotte M Deane. Synonymous codon usage influences the local protein structure observed. Nucleic Acids Res., 38(19):6719–6728, October 2010.

[SDA+10] Efraín Siller, Diane C DeZwaan, John F Anderson, Brian C Freeman, and José M Barral. Slowing bacterial translation speed enhances eukaryotic protein folding efficiency. J. Mol. Biol., 396(5):1310–1318, 12 March 2010.

[SHR13] Barbara L Schneider, V James Hernandez, and Larry Reitzer. Putrescine catabolism is a metabolic response to several stresses in escherichia coli. Mol. Microbiol., 88(3):537–550, May 2013.

[SIBG08] Tomohiro Shimada, Akira Ishihama, Stephen J W Busby, and David C Grainger. The escherichia coli RutR transcription factor binds at targets within genes as well as intergenic regions. Nucleic Acids Res., 36(12):3950–3955, July 2008.

[SKL+14] Sang Woo Seo, Donghyuk Kim, Haythem Latif, Edward J O'Brien, Richard Szubin, and Bernhard O Palsson. Deciphering fur transcriptional regulatory network highlights its complex role beyond iron metabolism in escherichia coli. Nat. Commun., 5:4910, 15 September 2014.

[SKP89] M A Sørensen, C G Kurland, and S Pedersen. Codon usage determines translation rate in escherichia coli. J. Mol. Biol., 207(2):365–377, 20 May 1989.

[SKV+16] Alexander Schmidt, Karl Kochanowski, Silke Vedelaar, Erik Ahrné, Benjamin Volkmer, Luciano Callipo, Kèvin Knoops, Manuel Bauer, Ruedi Aebersold, and Matthias Heinemann. The quantitative and

condition-dependent escherichia coli proteome. Nat. Biotechnol., 34(1):104–110, January 2016.

[SLP11] Jan Schellenberger, Nathan E Lewis, and Bernhard ØPalsson. Elimination of thermodynamically infeasible loops in steady-state metabolic models. Biophys. J., 100(3):544–553, 2 February 2011.

[SMK58] M Schaechter, O Maaloe, and N O Kjeldgaard. Dependency on medium and temperature of cell size and chemical composition during balanced grown of salmonella typhimurium. J. Gen. Microbiol., 19(3):592–606, December 1958.

[SPCP10] Jan Schellenberger, Junyoung O Park, Tom M Conrad, and Bernhard ØPalsson. BiGG: a biochemical genetic and genomic knowledgebase of large scale metabolic reconstructions. BMC Bioinformatics, 11(1):213, 29 April 2010.

[SQF+11] Jan Schellenberger, Richard Que, Ronan M T Fleming, Ines Thiele, Jeffrey D Orth, Adam M Feist, Daniel C Zielinski, Aarash Bordbar, Nathan E Lewis, Sorena Rahmanian, Joseph Kang, Daniel R Hyduke, and Bernhard ØPalsson. Quantitative prediction of cellular metabolism with constraint-based models: the COBRA toolbox v2.0. Nat. Protoc., 6(9):1290–1307, September 2011.

[SZO14] Arvind R Subramaniam, Brian M Zid, and Erin K O'Shea. An integrated approach reveals regulatory controls on bacterial translation elongation. Cell, 159(5):1200–1211, 20 November 2014.

[THL10] Linh M Tran, Daniel R Hyduke, and James C Liao. Trimming of mammalian transcriptional networks using network component analysis. BMC Bioinformatics, 11:511, 13 October 2010.

[THS+11] Ines Thiele, Daniel R Hyduke, Benjamin Steeb, Guy Fankam, Douglas K Allen, Susanna Bazzani, Pep Charusanti, Feng-Chi Chen, Ronan M T Fleming, Chao A Hsiung, Sigrid C J De Keersmaecker, Yu-Chieh Liao, Kathleen Marchal, Monica L Mo, Emre Özdemir, Anu Raghunathan, Jennifer L Reed, Sook-Il Shin, Sara Sigurbjörnsdóttir, Jonas Steinmann, Suresh Sudarsan, Neil Swainston, Inge M Thijs, Karsten Zengler, Bernhard O Palsson, Joshua N Adkins, and Dirk Bumann. A community effort towards a knowledge-base and mathematical model of the human pathogen salmonella typhimurium LT2. BMC Syst. Biol., 5:8, 18 January 2011.

[TJFP09] Ines Thiele, Neema Jamshidi, Ronan M T Fleming, and Bernhard ØPalsson. Genome-scale reconstruction of escherichia coli's transcriptional and translational machinery: a knowledge base, its

mathematical formulation, and its functional characterization. PLoS Comput. Biol., 5(3):e1000312, March 2009.

[TP10] Ines Thiele and Bernhard ØPalsson. A protocol for generating a high-quality genome-scale metabolic reconstruction. Nat. Protoc., 5(1):93–121, January 2010.

[TWP+10] Cole Trapnell, Brian A Williams, Geo Pertea, Ali Mortazavi, Gordon Kwan, Marijke J van Baren, Steven L Salzberg, Barbara J Wold, and Lior Pachter. Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. Nat. Biotechnol., 28(5):511–515, May 2010.

[UNMRM84] S Ueno-Nishio, S Mango, L J Reitzer, and B Magasanik. Identification and regulation of the glnl operator-promoter of the complex glnALG operon of escherichia coli. J. Bacteriol., 160(1):379–384, October 1984.

[VBD+08] Alexei Vazquez, Qasim K Beg, Marcio A Demenezes, Jason Ernst, Ziv Bar-Joseph, Albert-László Barabási, László G Boros, and Zoltán N Oltvai. Impact of the solvent capacity constraint on e. coli metabolism. BMC Syst. Biol., 2(1):7, 23 January 2008.

[VBT+03] Jörg Vogel, Verena Bartels, Thean Hock Tang, Gennady Churakov, Jacoba G Slagter-Jäger, Alexander Hüttenhofer, and E Gerhart H Wagner. RNomics in escherichia coli detects new sRNA species and indicates parallel transcriptional output in bacteria. Nucleic Acids Res., 31(22):6435–6443, 15 November 2003.

[VDTR97] M Vulić, F Dionisio, F Taddei, and M Radman. Molecular keys to speciation: DNA polymorphism and the control of genetic exchange in enterobacteria. Proc. Natl. Acad. Sci. U. S. A., 94(18):9763–9767, 2 September 1997.

[VGD+94] S G Vasudevan, C Gedye, N E Dixon, E Cheah, P D Carr, P M Suffolk, P D Jeffrey, and D L Ollis. Escherichia coli PII protein: purification, crystallization and oligomeric structure. FEBS Lett., 337(3):255–258, 17 January 1994.

[vHHM+96] W C van Heeswijk, S Hoving, D Molenaar, B Stegeman, D Kahn, and H V Westerhoff. An alternative PII protein in the regulation of glutamine synthetase in escherichia coli. Mol. Microbiol., 21(1):133–146, July 1996.

[vHWC+00] W C van Heeswijk, D Wen, P Clancy, R Jaggi, D L Ollis, H V Westerhoff, and S G Vasudevan. The escherichia coli signal transducers PII (GlnB) and GlnK form heterotrimers in vivo: fine tuning the

nitrogen signal cascade. Proc. Natl. Acad. Sci. U. S. A., 97(8):3942–3947, 11 April 2000.

[VP13] Bryan J Venters and B Franklin Pugh. Genomic organization of human transcription initiation complexes. Nature, 502(7469):53–58, 3 October 2013.

[WGGB15] Christopher J Woolstenhulme, Nicholas R Guydosh, Rachel Green, and Allen R Buskirk. High-precision analysis of translational pausing by ribosome profiling in bacteria lacking EFP. Cell Rep., 11(1):13–21, 7 April 2015.

[Wun96] Roland Wunderling. Paralleler und objektorientierter Simplex-Algorithmus. PhD thesis, Technische Universität Berlin, 1996.

[Wun97] R Wunderling. SOPLEX: the sequential object-oriented simplex class library, 1997.

[WW08] Jeremiah Wright and Andreas Wagner. The systems biology research tool: evolvable open-source software. BMC Syst. Biol., 2:55, 29 June 2008.

[YDZ+15] Chien-Hung Yu, Yunkun Dang, Zhipeng Zhou, Cheng Wu, Fangzhou Zhao, Matthew S Sachs, and Yi Liu. Codon usage influences the local rate of translation elongation to regulate co-translational protein folding. Mol. Cell, 59(5):744–754, 3 September 2015.

[YG03] Yuzhen Ye and Adam Godzik. Flexible structure alignment by chaining aligned fragment pairs allowing twists. Bioinformatics, 19 Suppl 2:ii246–55, October 2003.

[YOH+13] Conghui You, Hiroyuki Okano, Sheng Hui, Zhongge Zhang, Minsu Kim, Carl W Gunderson, Yi-Ping Wang, Peter Lenz, Dalai Yan, and Terence Hwa. Coordination of bacterial proteome with metabolism by cyclic AMP signalling. Nature, 500(7462):301–306, 15 August 2013.

[YTO+15] Laurence Yang, Justin Tan, Edward J O'Brien, Jonathan M Monk, Donghyuk Kim, Howard J Li, Pep Charusanti, Ali Ebrahim, Colton J Lloyd, James T Yurkovich, Bin Du, Andreas Dräger, Alex Thomas, Yuekai Sun, Michael A Saunders, and Bernhard O Palsson. Systems biology definition of the core proteome of metabolism and expression is consistent with high-throughput data. Proc. Natl. Acad. Sci. U. S. A., 112(34):10810–10815, 25 August 2015.

[YVB+12] Kuangyu Yen, Vinesh Vinayachandran, Kiran Batta, R Thomas Koerber, and B Franklin Pugh. Genome-wide nucleosome specificity and

directionality of chromatin remodelers. Cell, 149(7):1461–1473, 22 June 2012.

[ZGC+13] Mian Zhou, Jinhu Guo, Joonseok Cha, Michael Chae, She Chen, Jose M Barral, Matthew S Sachs, and Yi Liu. Non-optimal codon usage affects expression, structure and function of clock protein FRQ. Nature, 495(7439):111–115, 7 March 2013.

[Zha09] Yang Zhang. I-TASSER: fully automated protein structure prediction in CASP8. Proteins, 77 Suppl 9:100–113, 2009.

[ZHI09] Gong Zhang, Magdalena Hubalewska, and Zoya Ignatova. Transient ribosomal attenuation coordinates protein synthesis and co-translational folding. Nat. Struct. Mol. Biol., 16(3):274–280, March 2009.

[ZSL+00] D P Zimmer, E Soupene, H L Lee, V F Wendisch, A B Khodursky, B J Peter, R A Bender, and S Kustu. Nitrogen regulatory protein c-controlled genes of escherichia coli: scavenging as a defense against nitrogen limitation. Proc. Natl. Acad. Sci. U. S. A., 97(26):14674–14679, 19 December 2000.