COBRAme: A Computational Framework for Building and Manipulating Models of Metabolism and Gene Expression

Colton J. Lloyd*, Ali Ebrahim*, Laurence Yang, Zachary King, Edward Catoiu, Edward J. O'Brien, Joanne K. Liu, Bernhard O. Palsson

Abstract

Genome-scale models of metabolism and macromolecular expression (ME-models) explicitly compute the optimal proteome composition of a growing cell. ME-models expand upon the well-established genome-scale models of metabolism (M-models), and they enable new and exciting insights that are fundamental to understanding the basis of cellular growth. ME-models have increased predictive capabilities and accuracy due to their inclusion of the biosynthetic costs for the machinery of life, but they come with a significant increase in model size and complexity. This challenge results in models which are both difficult to compute and challenging to understand conceptually. As a result, ME-models exist for only two organisms (*Escherichia coli* and *Thermotoga maritima*) and are still used by relatively few researchers. To address these challenges, we have developed a new software framework called COBRAme for building and simulating ME-models. It is coded in Python and built on COBRApy, a popular platform for using M-models. COBRAme streamlines computation and analysis of ME-models. It provides tools to simplify the construction and manipulation of ME-models to enable ME-model reconstructions for new organisms. We used COBRAme to reconstruct a condensed *E. coli* ME-model called *i*LE1678-ME. This new model gives virtually identical solutions to previous ME-models while using ¼ the number of free variables and solving in ~10 minutes, a marked improvement over the ~6 hour solve time of previous ME-model formulations. This manuscript outlines the architecture of COBRAme and demonstrates how ME-models can be built and edited most efficiently using the software.

Introduction

Genome-scale metabolic models (M-models) have shown significant success predicting various aspects of cellular metabolism by integrating all of the experimentally determined metabolic reactions taking place in an organism of interest [1–4]. These predictions are enabled based on the stoichiometric constraints of the organism's metabolic reaction network and metabolic interactions with the environment. M-models are capable of accurately predicting the metabolic capabilities of an organism, but they require defined substrate input constraints and empirical metabolite measurements to make predictions of its growth capabilities. Therefore, a focus of

development in the field of genome-scale models has been to increase the scope and capabilities of M-models [5].

Recently, M-models have been extended to include the synthesis of the gene expression machinery and its use to compute the entire metabolic and gene expression proteome [6–9]. These ME-models integrate Metabolism and Expression on the genome scale, and they are capable of explicitly computing a large percentage (> 80% in some cases) of the proteome by mass in enterobacteria [10]. In other words, not only do ME-models compute optimal metabolic flux states, as do M-models, but they also compute the optimal proteome allocation for sustaining the metabolic phenotype. ME-models enable a wide range of new biological questions to be investigated including direct calculations of proteome allocation [11], metabolic pathway usage and the effects of membrane and volume constraints [7]. Furthermore, their ability to compute the optimal proteome abundances for a given condition make them ideal for mechanistically integrating transcriptomics and proteomics data.

So far ME-models have been constructed for only two organisms, *Thermotoga maritima* [8] and *Escherichia coli* K-12 MG1655 [6,7,9,12]. The slow pace of ME-model construction can be attributed to two basic challenges with ME-models. First, ME-models are much slower to numerically solve than M-models; it takes 5 orders of magnitude more CPU time to solve *i*OL1650-ME [6] than it does the corresponding *i*JO1366 M-model [13]. As a result, while M-models can be solved on personal computers, ME-models currently require large clusters or supercomputers. While increased computing power is generally becoming more readily available, alleviating the computational challenge, other challenges that come with ME-models are not as easily addressed. M-models can use generalized software tools [14–18], 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. Second, 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.

We addressed the above challenges by developing a computational framework – analogous to the widely used software for M-models, COBRApy [17] for building, manipulating, simulating and interpreting ME-model results, called COBRAme. COBRAme is designed to: 1) be generally applicable to any organism with an existing metabolic reconstruction (M-model) 2) use protocols and commands familiar to current users of COBRApy 3) construct ME-models with a reaction structure that is easily interpreted by the user 4) construct models that solve orders of magnitude faster than previous ME-models [6]. As a result of the above considerations, we hope that COBRAme and its associated tools, presented here, will accelerate the development and use of models of metabolism and expression.

Design and Implementation

Python

The COBRAme software is written entirely in Python 2.7 and requires the COBRApy [17] software package to enable full COBRA model functionality. Additionally, COBRAme requires the SymPy Python module [19] in order to handle μ, the symbolic variable representing cellular growth rate, which participates as a component of many stoichiometric coefficients in the ME-matrix. The BioPython package [20] is used by COBRAme to construct transcription, translation and tRNA charging reactions for each gene product in the organism's genbank genome annotation. The ME-model is solved using the SoPlex solver [21,22] via an API written in Python and included as part of this project. Further, the ECOLIme python package is included in this work and contains information pertaining to *E. coli* gene expression and scripts to build *i*LE1678-ME starting with the *E. coli* metabolic model, *i*JO1366 [13].

COBRAme ME-model Structure

Constructing an ME-model requires the incorporation of information pertaining to many different cellular processes. For instance, in order to construct a translation reaction for the ME-model, the sequence of the gene, the codon table for the organism, the tRNAs for each codon, ribosome translation rates, elongation factor usage, etc. must be incorporated. Further, several processes in the ME-model recur for many genes that are transcribed or translated in a template-like fashion [12]. To address these challenges, the COBRAme ME-model was structured to compartmentalize information for individual cellular processes. A key component of this approach was the separation of the ME-model into two major python class types: the information storage vessels called ProcessData and the functional model reactions called MEReactions (**Figure 1**).

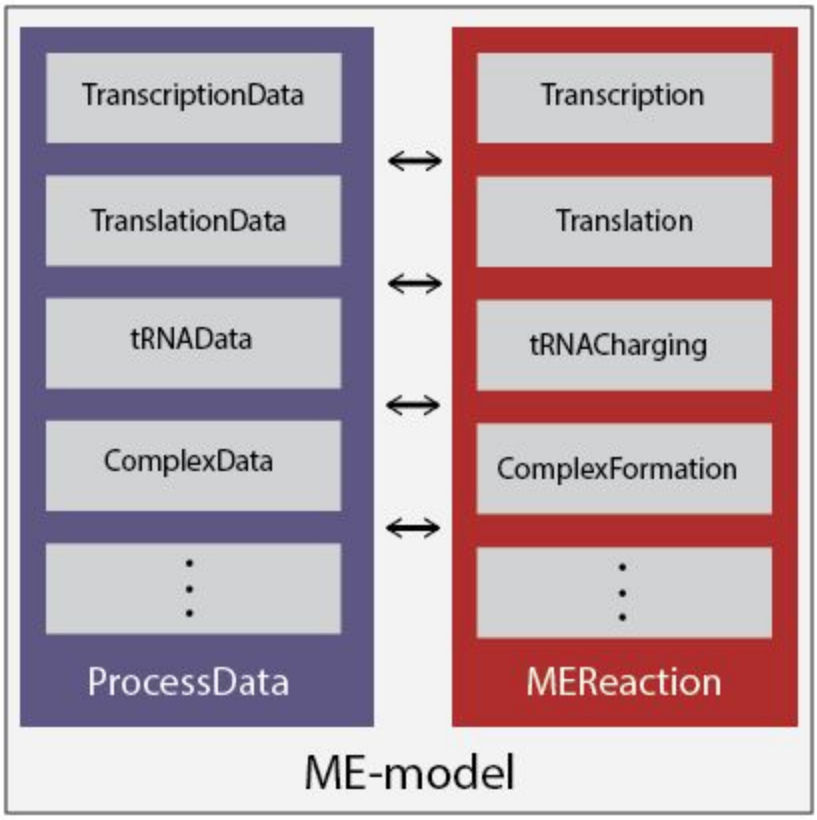

Figure 1: Compartmentalization of ME-model components. Building an ME-model requires the union of numerous data types. To facilitate and enhance this process, the ME-model is broken into two main python classes. The first of these is the ProcessData class which solely contains information associated with each of the processes listed in the purple box. The second is the MEReaction class which compiles the information contained in the appropriate ProcessData to create the model reactions. (further descriptions of each subclass can be found in the **COBRAme Documentation**).

ProcessData

COBRAme constructs ME-models that contain two major Python class types. The first of these is the ProcessData type, which is used to store information associated with a cellular process. The type of information contained in each ProcessData type is summarized in the **COBRAme Documentation**. This method of information storage has several advantages over alternatives such as establishing a database to query information as it is needed, which was the approach used to build previous ME-model versions. For example, this method simplifies the dissemination of the information used to construct an ME-model given that the information can now be included as part of a published ME-model without requiring the user to install and populate a database. Further, this gives the ability to compartmentalize the information based on which cellular processes it is necessary for and, given that this information is contained in python objects, methods can be implemented to further allow data contained in each ProcessData instance to be manipulated. This method also reduces error by enabling many features to be computed using defined inputs in a consistent way. For example, the amino acid

sequence for a protein can be dynamically computed using a gene's nucleotide sequence and codon table.

MEReaction

The second of the major Python class types contained within a ME-model is the MEReaction. This class contains the functional ME-model reaction that inherits from the COBRApy reaction and are thus ultimately incorporated into the ME-matrix. In addition to the functionality of COBRApy reactions, MEReactions contain functions to read and process the information contained in ProcessData objects and to update this information into a complete, functional reaction. Part of compiling a functional reaction also includes imposing the appropriate coupling constraints in many cases (coupling constraints detailed in **Supplement Text** and **COBRAme Documentation**). These coupling constraints are imposed directly as part of the MEReaction's update method and varies depending on the reaction type. Since MEReactions are associated with the information used to construct them through ProcessData, this codebase has the ability to easily query, edit and update the information used to construct the reaction into the MEReaction and therefore model.

ME-model Construction Workflow

ME-models of *E. coli* are constructed using the two Python packages presented here, COBRAme and ECOLIme. COBRAme contains the class definitions and necessary functions required to build and manipulate a working ME-model. COBRAme is written to be organism agnostic such that it can be used to build and manipulate an ME-model for any organism. ECOLIme contains *E. coli* specific information (e.g. the *E. coli* ribosome composition) as well as functions to process files containing E. coli reaction information (e.g. the text file containing transcription unit definitions) and associate them with the ME-model being constructed. Therefore, ECOLIme is required to assemble the reaction and gene expression information to construct *i*LE1678-ME, and COBRAme, on the other hand, is needed to effectively manipulate a constructed ME-model. The components and further demonstrations of the utility of each of these packages is outlined in the **COBRAme Documentation**.

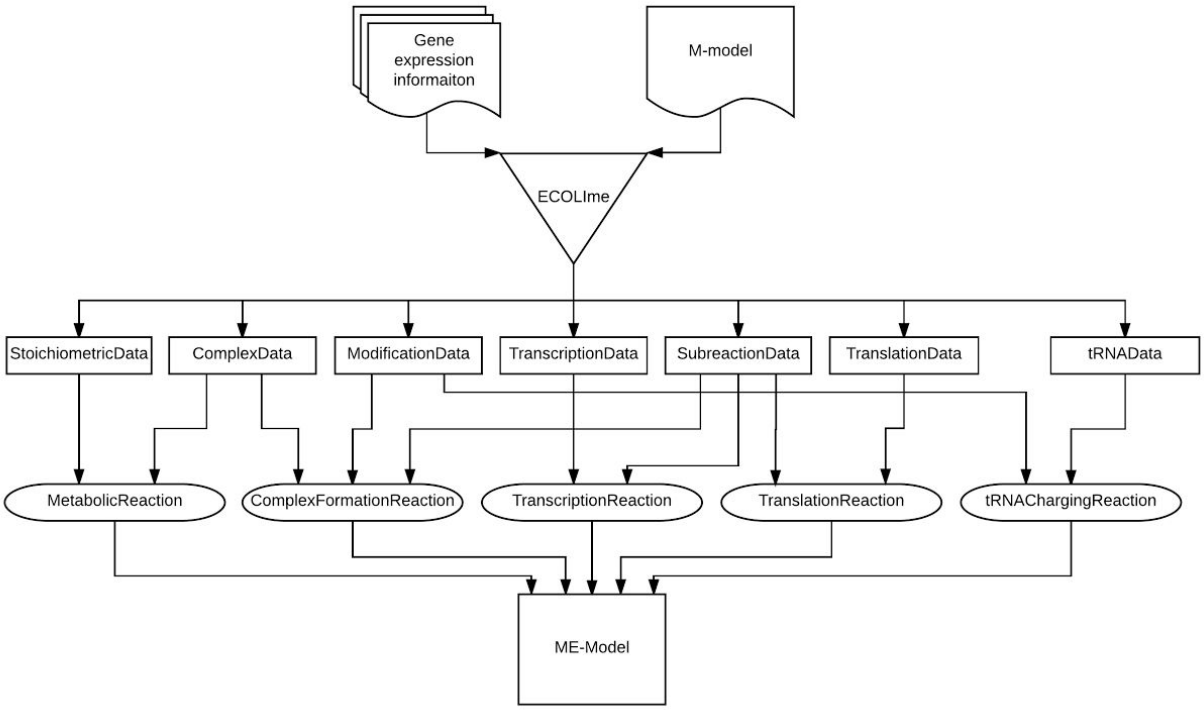

Figure 2: The flow of information from input files to ME-model, as facilitated using the '**build_ME_model**' script. The '**build_ME_model**' workflow uses the ECOLIme package to load and process the *E. coli* M-model and all of the files containing gene expression information. This information is then used to populate the different ProcessData class instances and linked to the appropriate MEReaction class instances, all of which are defined in the COBRAme package. The entirety of the MEReactions make up a working ME-model.

The process of building *i*LE1678-ME using COBRAme and ECOLIme is laid out in the building script, '**build_ME_model**' (**Figure 2**). This script goes through each of the major gene expression processes modeled in *i*LE1678-ME and uses the functionality contained within ECOLIme to read and process all relevant information. Once the information is loaded, it is used to create and populate the instances of the ProcessData associated with the information. Each of ProcessData objects are then linked to the appropriate MEReaction instance and updated to form a functioning ME-Model. A complete description of the '**build_ME_model**' script can be found in the **COBRAme Documentation**.

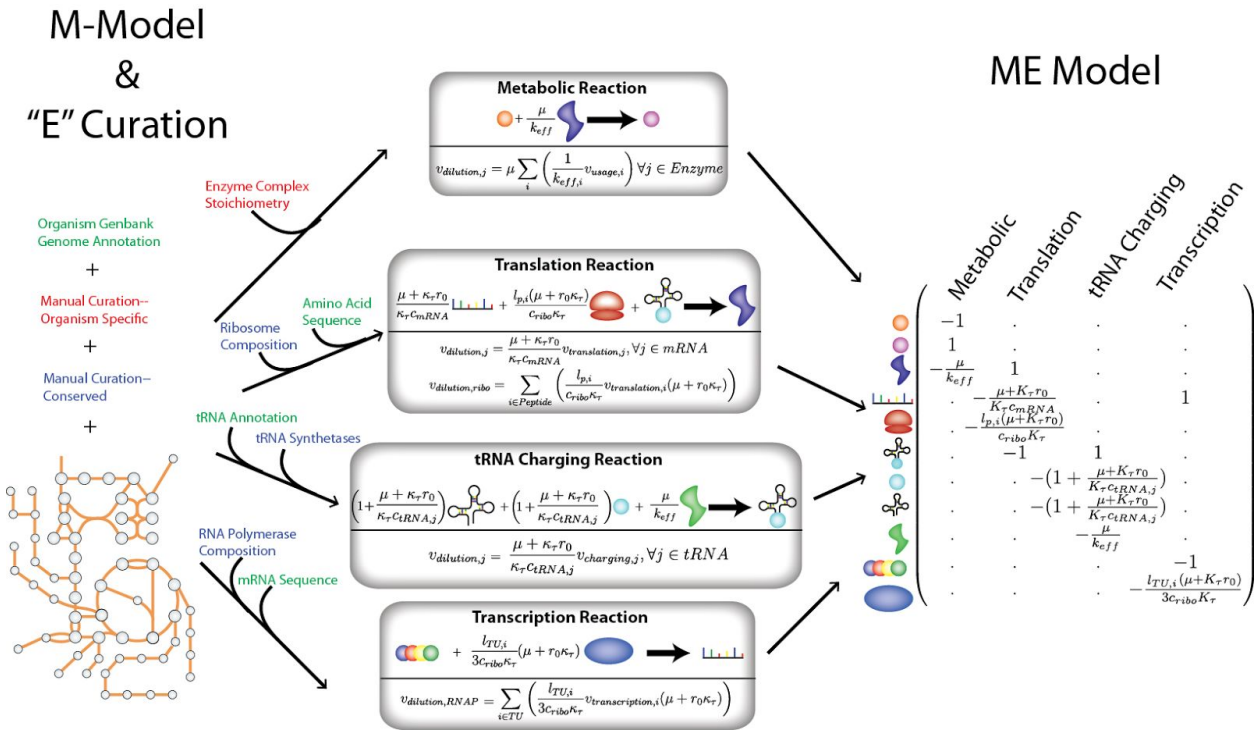

Figure 3: An overview of the COBRAme formulation for ME-models. The original ME formulation created a large matrix by introducing coupling constraints. With COBRAme, instead of coupling constraints, dilution reactions to the daughter cell for the required macromolecules are embedded directly in the reaction which requires them. For example, the first row shows that for a metabolic reaction, a small amount (μ/k_{eff}) of the catalyzing enzyme is also used in the process.

Reformulating the *E. coli* ME-model

Significant efforts were made to simplify the ME-model implementation while also optimizing the model size and time required to solve. These included: **1)** reformulating the implementation of "coupling constraints" into ME-model reactions and **2)** lumping major cellular processes such as transcription and translation into single ME-model reactions. Further, a number of updates, changes and corrections have been made to the *E. coli* ME-model reconstruction and are detailed in the **Supplement Text**.

Coupling Constraints

The biggest mathematical difference between the original ME-model formulation [6] and models constructed using COBRAme are found in the reformulation of coupling constraints. These constraints dictate the amount of any macromolecule synthesis flux that is required in order for the catalyzed reaction to carry flux. These constraints are essential to produce a working ME-model, but they inflated the number of metabolites and reactions contained in the ME-matrix, resulting in longer solve times. COBRAme improves coupling constraint

implementation by directly embedding macromolecule dilution "coupling" into its catalytic reaction (**Figure 3**).

To accomplish this, the coupling constraints and coefficients were derived as in O'Brien et. al. 2013, but were implemented in the current work as equality constraints (**Figure 3**) as opposed to inequality constraints in previous ME-model implementations[6,7]. Effectively, this means that each ME-model solution will synthesize the exact amount of each macromolecule as dictated by each coupling coefficient, thus giving the computed optimal proteome allocation for the *in silico* conditions. Previous ME-model formulations, as stated above, have applied the constraints as inequalities thus allowing the simulation to overproduce macromolecule components. While overproduction is seen *in vivo* in cells, this phenomenon would not be selected as the optimal solution. Furthermore, using inequality constraints greatly expands the size of the possible solution space significantly increasing the time required to solve the optimization. Reformulating the model using equality constraints thus resulted in a reduced ME-matrix with the coupling coefficients embedded directly into the reaction in which they are used (**Figure 3**). This process also removes the majority of "coupling constraints" and associated variables in the original formulation, which makes the models much smaller (i.e. the imposition of each macromolecule coupling constraint in *i*LE1678-ME requires at least one less reaction and constraint (metabolite) than previous ME-model versions).

The elimination of most inequality constraints greatly reduces the space of feasible fluxes at suboptimal growth rates. At an optimum, a ME-model will not waste resources, and, as a result, all the computed values are pushed up against their inequality constraints, 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. A more thorough description of coupling constraints and their implementation can be found in the **COBRAme Documentation**.

Reaction Lumping

As a consequence of using only equality constraints in the COBRAme formulation, reactions which occur in a number of individual steps or sub-reactions (i.e., ribosome formation, translation, etc.) can be easily 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 simplifies the process of adding in new processes. The information of the individual subprocesses involved in these reactions (i.e. translation elongation for translation reactions) is maintained by splitting the ME-model into information storage and reaction components (detailed in **COBRAme Me-model Structure**). This allows this information to be queried, edited and updated throughout the model and greatly reduces the number of model reactions.

Nonequivalent Changes

Unlike the replacement of inequality constraints and reaction lumping, some of the changes made in the COBRAme 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 COBRAme formulation. Additionally, in COBRAme, 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 COBRAme formulation will require translation of these carriers in order for them to participate in reactions. Further non-equivalent changes to the ME-model can be found in the **Supplementary Text**.

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 1, 6). This ME-matrix is nonlinear only in the variable μ, but remains quasi-convex [23], 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 [6]. 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 [6].

To perform the binary search, the following procedure was implemented in COBRApy [17]. First, each symbolic coefficient or reaction bound was compiled into a function by SymPy [19]. 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 [24], ME-models are very ill-scaled [6]. Therefore, the SoPlex solver (version 2.2) was integrated with cobrapy and used [21]. 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.

Results and Discussion

Model Overview

The COBRAme framework was used to reconstruct a reformulated version of *i*JL1678-ME, called *i*LE1678-ME. This produced a model with 12,683 reactions and 7,035 metabolites, a marked improvement over *i*JL1678-ME which contained 79,871 reactions and 70,751 metabolites. As a result, *i*LE1678-ME has a matrix with ~90% 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. The resulting reduced model was simulated in glucose aerobic minimal media in silico conditions and verified to give very similar solutions on a transcription, translation and metabolic level compared to *i*OL1650-ME (**Figure 4**). The reformulated ME-model cannot be expected to give completely identical solutions as *i*OL1650-ME due to some of the nonequivalent changes and model corrections described in **Nonequivalent Changes**.

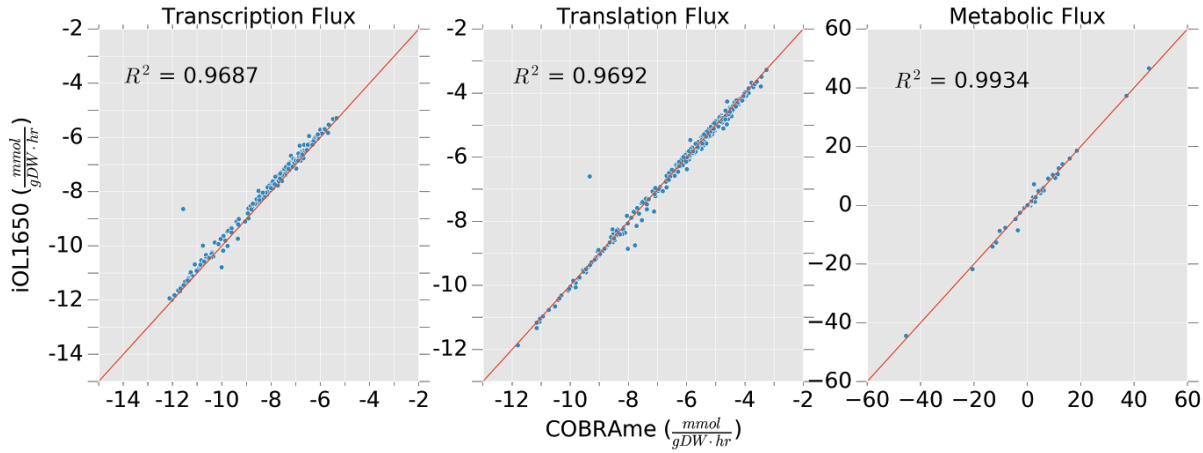

Figure 4: Comparison of the simulated fluxes of *i*OL1650-ME to the COBRAme generated version of the same model at Transcription, Translation and Metabolic flux scales. The transcription and translation flux panels are both log log plots. At each level, the models provided comparable flux predictions, indicated by the pearson correlations above 0.96. The models cannot be expected to give completely identical flux predictions due to the ME-model updates outlined in **Nonequivalent Changes**. Since *i*LE1678-ME does not contain membrane surface area constraints, *i*OL1650-ME was used for comparison.

Perhaps more importantly, this reformulation and reduced size makes the models more understandable to its human users. By lumping cellular processes into individual model reactions, the organization of the ME-model reactions is able to more closely resemble the central dogma of biology. For instance, the translation of a given gene, <gene_id> occurs in a

single model reaction, “translation_<gene_id>” where all components and coupling constraints are applied in one place (**Figure 3**) as opposed to occurring in multiple separate reactions. In addition to being more easily understandable by the user the reformulation makes the model more amenable to visualization tools like escher [18], further easing the process of interpreting simulation results.

Manipulating a Constructed ME-model

The separation of COBRAme ME-models into information storing ProcessData classes and functional MEReactions allows many components of a finalized ME-model to be easily queried, edited and updated throughout the ME-model. This is especially useful for a few reasons: 1) Certain processes occur repeated throughout the process of expressing the genes and proteins within a cell. Therefore if the user wants to edit a parameter associated with one of these processes, such as the GTP cost of translation elongations, this can be done by editing the ProcessData instance that defines this process and update it through the model. 2) Aspects of processes involved in gene expression can often be interrelated. For instance, the nucleotide sequence of an mRNA being translated dictates the compositions of the charged tRNAs that are incorporate, the number of GTP driven elongation steps that must occur, etc. COBRAme allows the user to edit the sequence in one place and use the TranslationReaction’s update method to apply the changes throughout the reaction. Examples of making edits to an ME-model can be found in the **COBRAme Documentation**.

Availability and Future Directions

Both the COBRAme and ECOLIme software packages are required to construct *i*LE1678-ME and are currently available on the Systems Biology Research Group’s github page (github.com/SBRG). Installation procedures as well as all necessary documentation required to build, simulate and manipulate ME-models are present in the repositories as well. The soplex solver can be found at (http://soplex.zib.de/) and is freely available to all academic instutitions. The soplex_cython package contains instructions to compile the soplex solver with 80-bit precision capabilities in addition to the necessary code required to solve *i*LE1678-ME with SoPlex. These software packages will be actively maintained and improved. The COBRAme documentation can be found at readthedocs.

Enable New ME-Model Reconstructions

We anticipate that the presented software tools will facilitate the reconstruction of many new ME-models beyond *i*LE1678-ME for *Escherichia coli* K-12 MG1655. While the COBRAme code was constructed to be readily generalizable to many different organisms, it is likely that some organisms will require additional features for their ME-model reconstruction that we did not originally anticipate. It is our priority to continue to update and improve the code to enhance its

usability to model new, diverse organisms. Future efforts will be also be made to create standards to govern how ME-models are reconstructed, structured and shared within the scientific community.

Acknowledgements

We would like to thank 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. We thank the Novo Nordisk Foundation [NNF16CC0021858] and the National Institute of General Medical Science of the National Institute of Health (award U01GM102098) for funding this project. CJL was supported by the National Science Foundation Graduate Research Fellowship under Grant no. DGE-1144086

References

1. Bordbar A, Aarash B, Monk JM, King ZA, Palsson BO. Constraint-based models predict metabolic and associated cellular functions. Nat Rev Genet. 2014;15: 107–120.

2. O'Brien EJ, Monk JM, Palsson BO. Using Genome-scale Models to Predict Biological Capabilities. Cell. 2015;161: 971–987.

3. Lewis NE, Nagarajan H, Palsson BO. Constraining the metabolic genotype–phenotype relationship using a phylogeny of in silico methods. Nat Rev Microbiol. 2012;10: 291–305.

4. McCloskey D, Palsson BØ, Feist AM. Basic and applied uses of genome-scale metabolic network reconstructions of Escherichia coli. Mol Syst Biol. 2013;9: 661.

5. Karr JR, Sanghvi JC, Macklin DN, Gutschow MV, Jacobs JM, Bolival B, et al. A Whole-Cell Computational Model Predicts Phenotype from Genotype. Cell. 2012;150: 389–401.

6. O'Brien EJ, Lerman JA, Chang RL, Hyduke DR, Palsson BØ. Genome-scale models of metabolism and gene expression extend and refine growth phenotype prediction. Mol Syst Biol. 2013;9: 693.

7. Liu JK, O Brien EJ, Lerman JA, Zengler K, Palsson BO, Feist AM. Reconstruction and modeling protein translocation and compartmentalization in Escherichia coli at the genome-scale. BMC Syst Biol. 2014;8: 110.

8. Lerman JA, Hyduke DR, Latif H, Portnoy VA, Lewis NE, Orth JD, et al. In silico method for modelling metabolism and gene product expression at genome scale. Nat Commun. 2012;3: 929.

9. Thiele I, Fleming RMT, Que R, Bordbar A, Diep D, Palsson BO. Multiscale Modeling of Metabolism and Macromolecular Synthesis in E. coli and Its Application to the Evolution of Codon Usage. PLoS One. Public Library of Science; 2012;7: e45635.

10. O'Brien EJ, Palsson BO. Computing the functional proteome: recent progress and future prospects for genome-scale models. Curr Opin Biotechnol. 2015;34: 125–134.

11. LaCroix RA, Sandberg TE, O'Brien EJ, Utrilla J, Ebrahim A, Guzman GI, et al. 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. Am Soc Microbiol; 2015;81: 17–30.

12. Thiele I, Jamshidi N, Fleming RMT, Palsson BØ. Genome-scale reconstruction of Escherichia coli's transcriptional and translational machinery: a knowledge base, its mathematical formulation, and its functional characterization. PLoS Comput Biol. 2009;5: e1000312.

13. Orth JD, Conrad TM, Na J, Lerman JA, Nam H, Feist AM, et al. A comprehensive genome-scale reconstruction of Escherichia coli metabolism--2011. Mol Syst Biol. 2011;7:

535.

14. Schellenberger J, Que R, Fleming RMT, Thiele I, Orth JD, Feist AM, et al. Quantitative prediction of cellular metabolism with constraint-based models: the COBRA Toolbox v2.0. Nat Protoc. 2011;6: 1290–1307.

15. Gelius-Dietrich G, Desouki AA, Fritzemeier CJ, Lercher MJ. sybil -- Efficient constraint-based modelling in R. BMC Syst Biol. 2013;7: 1–8.

16. Agren R, Liu L, Shoaie S, Vongsangnak W, Nookaew I, Nielsen J. The RAVEN Toolbox and Its Use for Generating a Genome-scale Metabolic Model for Penicillium chrysogenum. PLoS Comput Biol. 2013;9: e1002980.

17. Ebrahim A, Lerman JA, Palsson BO, Hyduke DR. COBRApy: COnstraints-Based Reconstruction and Analysis for Python. BMC Syst Biol. 2013;7: 74.

18. King ZA, Dräger A, Ebrahim A, Sonnenschein N, Lewis NE, Palsson BO. Escher: A Web Application for Building, Sharing, and Embedding Data-Rich Visualizations of Biological Pathways. PLoS Comput Biol. dx.plos.org; 2015;11: e1004321.

19. Joyner D, Čertík O, Meurer A, Granger BE. Open source computer algebra systems: SymPy. ACM Commun Comput Algebra. ACM; 2012;45: 225–234.

20. Cock PJA, Antao T, Chang JT, Chapman BA, Cox CJ, Dalke A, et al. Biopython: freely available Python tools for computational molecular biology and bioinformatics. Bioinformatics. 2009;25: 1422–1423.

21. Wunderling R. SOPLEX: the sequential object-oriented simplex class library. ZIB; 1997.

22. Yang L, Ma D, Ebrahim A, Lloyd CJ, Saunders MA, Palsson BO. solveME: fast and reliable solution of nonlinear ME models. BMC Bioinformatics. 2016;17: 391.

23. Laurence Yang, Ding Ma, Ali Ebrahim,Colton J. Lloyd, Michael A. Saunders, Bernhard O. Palsson. solveME: fast and reliable solution of nonlinear ME models.

24. Ebrahim A, Almaas E, Bauer E, Bordbar A, Burgard AP, Chang RL, et al. Do genome-scale models need exact solvers or clearer standards? Mol Syst Biol. EMBO Press; 2015;11: 831.