# Estimating Cellular Goals from High-Dimensional Biological Data

Laurence Yang*[†]
lyang@eng.ucsd.edu
University of California, San Diego

Michael A. Saunders
saunders@stanford.edu
Stanford University

Jean-Christophe Lachance
jelachance@eng.ucsd.edu
Université de Sherbrooke

Bernhard O. Palsson
palsson@ucsd.edu
University of California, San Diego

José Bento*
jose.bento@bc.edu
Boston College

## ABSTRACT

Optimization-based models have been used to predict cellular behavior for over 25 years. The constraints in these models are derived from genome annotations, measured macromolecular composition of cells, and by measuring the cell's growth rate and metabolism in different conditions. The *cellular goal* (the optimization problem that the cell is trying to solve) can be challenging to derive experimentally for many organisms, including human or mammalian cells, which have complex metabolic capabilities and are not well understood. Existing approaches to learning goals from data include (a) estimating a linear objective function, or (b) estimating linear constraints that model complex biochemical reactions and constrain the cell's operation. The latter approach is important because often the known reactions are not enough to explain observations; therefore, there is a need to extend automatically the model complexity by learning new reactions. However, this leads to nonconvex optimization problems, and existing tools cannot scale to realistically large metabolic models. Hence, constraint estimation is still used sparingly despite its benefits for modeling cell metabolism, which is important for developing novel antimicrobials against pathogens, discovering cancer drug targets, and producing value-added chemicals. Here, we develop the first approach to estimating constraint reactions from data that can scale to realistically large metabolic models. Previous tools were used on problems having less than 75 reactions and 60 metabolites, which limits real-life-size applications. We perform extensive experiments using 75 large-scale metabolic network models for different organisms (including bacteria, yeasts, and mammals) and show that our algorithm can recover cellular constraint reactions. The recovered constraints enable accurate prediction of metabolic states in hundreds of growth environments not seen in training data, and we recover useful cellular goals even when some measurements are missing.

*Both authors contributed equally to this project and are corresponding authors.
[†]Current address: Department of Chemical Engineering, Queen's University, Canada.

## KEYWORDS

nonconvex optimization; distributed optimization; metabolism; computational biology

**ACM Reference Format:**
Laurence Yang, Michael A. Saunders, Jean-Christophe Lachance, Bernhard O. Palsson, and José Bento. 2019. Estimating Cellular Goals from High-Dimensional Biological Data. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3292500.3330775

## 1 INTRODUCTION AND RELATED WORK

Engineered microbial and mammalian cells are used as production platforms to synthesize commodity or specialty chemicals and pharmaceuticals. Accurate computational models of cell metabolism and protein expression are important to design cell factories and to maximize product yield and quality [26, 43]. Similarly, mathematical models of cell metabolism have been used to identify strategies to improve the efficacy of existing antibiotics [8].

The ability of engineers to predict microbial behavior was facilitated in 1990 by the observation that overflow metabolism—an industrially-relevant metabolic behavior—in *Escherichia coli* could be predicted by a relatively simple network of reactions with capacity-constrained flows [28]. Since then, this constrained optimization model of cellular metabolism (often called COBRA) has been applied to over 78 species across the tree of life [30]. Metabolic reconstructions today are "genome-scale"—i.e., they account for the majority of metabolic genes annotated in the organism's genome—and consist of hundreds to thousands of biochemical reactions and metabolites. For example, the most recent reconstruction of human metabolism includes 13,543 metabolic reactions involving 4,140 metabolites [7], while the latest multiscale model of metabolism and protein expression for *E. coli* [27] consists of 12,655 reactions and 7,031 components including macromolecules like protein, RNA (ribonucleic acid), and ribosome.

As of mid-2013, over 640 published studies used COBRA for experimental investigation in various domains of engineering and health sciences [5]. Successful applications of COBRA include engineering microbes to produce commodity or valuable chemicals [43], developing novel antimicrobials against infectious disease [8], and discovering new drug targets against cancer [15].

In its basic form, COBRA predicts $n$ reaction fluxes $\mathbf{v} \in \mathbb{R}^n$ (reaction rate normalized by dry weight of biomass) in a metabolic network, consisting of $m$ metabolites and $n$ reactions, by solving a linear program that models cellular goal:

$$\max_{\mathbf{v}} \mathbf{c}^T\mathbf{v} \text{ subject to } S\mathbf{v} = \mathbf{b},\ \mathbf{l} \leq \mathbf{v} \leq \mathbf{u}, \tag{1}$$

where $\mathbf{c} \in \mathbb{R}^n$ is a vector of objective coefficients, $(\cdot)^T$ denotes the transpose, $S \in \mathbb{R}^{m \times n}$ is a matrix of stoichiometric coefficients (one column per biochemical reaction), $\mathbf{b} \in \mathbb{R}^m$ is a vector of metabolite accumulation or depletion rates, and $\mathbf{l}, \mathbf{u} \in \mathbb{R}^n$ are lower and upper flux bounds.

In order to make accurate predictions, these models require an accurate $S$ matrix, which is nowadays reliably reconstructed from extensive knowledgebases of enzyme biochemistry, and genome annotations for thousands of species and strains organisms. The other key ingredient is the cellular objective function, $\mathbf{c}^T\mathbf{v}$. Other objectives, including nonlinear functions, have been tested [36]. For many microbes cultured in nutrient-limiting conditions, maximization of cell growth rate (per unit of limiting nutrient) is an accurate objective function [36]. This particular function corresponds to a $\mathbf{c}$ that is an indicator vector with a 1 in the component associated with the reaction that corresponds to the cell growth, and zero everywhere else.

Currently, system-level understanding of living organisms requires the analysis of large-scale measurements originating from disparate biological processes operating at multiple length and time scales. Such measurements are collectively referred to as "omics", as they involve measuring the complete makeup of a given biological variable (e.g., proteomics attempts to measure the complete protein composition of a cell). Analysis of such omics measurements has shown that, despite the complexity inherent in living systems, relatively simple models are accurate enough for several biological studies. E.g., the gene expression profiles of various cell types can be described in terms of relatively few biological functions [21].

Similarly, the metabolic fluxes in a cell can be predicted by assuming that the cell is solving an optimization problem shaped by its evolutionary history [14]. The problem includes constraints that model biochemical "reactions" consuming and producing metabolites at specific stoichiometric ratios, and an objective that depends on the fluxes through the reactions. For example, the metabolism of fast-growing microbes (mid-log phase) is predicted accurately by a linear problem such as (1). As another example, the metabolism of mammalian cells (e.g., hybridoma), is explained well by minimization of the flux through the reaction that makes reactive oxygen species [14]. Studies have identified alternative objective functions that best predict microbial metabolism under different growth conditions [36]. These objectives include maximizing ATP yield per flux unit and maximizing ATP or biomass yield.

While the aforementioned studies have focused on using predefined optimization problems, a number of studies have investigated data-driven estimation of cellular goals. Two types of method exist. Methods of the first type estimate how important the different chemical reactions are for the cell's operation, i.e., estimate $\mathbf{c}$ in (1). For example, ObjFind [9] does this through a nonconvex optimization formulation, and the more recent invFBA [44] solves a linear program. The second type, and the focus of our work, estimate the stoichiometric coefficients of a new chemical reaction, i.e., estimate a new column for matrix $S$ in (1). This approach is important because often the known biochemical reactions are not enough to explain observed data. We want to extend the model complexity automatically by learning new chemical reactions, i.e., new columns for $S$.

The main drawback of estimating new reactions is that it requires solving nonconvex optimization problems. Currently, only the BOSS tool [17] does this. BOSS was shown to recover known biomass reactions successfully in synthetic experiments involving less than 70 reactions and 60 metabolites. This is not large enough for real-life applications, which can involve thousands of reactions/metabolites. Note that BOSS (a) uses an off-the-shelf solver (e.g. MINOS) that cannot exploit parallelism, (b) cannot induce sparsity on the learnt reaction, and (c) has not been tested on large problems.

Here we address these important limitations. Our main contributions are the following.

1. We develop BIG-BOSS (https://github.com/laurenceyang33/cellgoal), the first tool that can learn biochemical reactions in large models (up to 4,456 metabolites and 6,663 reactions).
2. BIG-BOSS is based on the alternating direction method of multipliers (ADMM). It employs an adaptive penalty and a preconditioning scheme, and hence requires no tuning. Further, it can exploit multiple CPUs to speed up computations on large problems.
3. BIG-BOSS uses a sparsity-inducing regularization to allow users to control the complexity of inferred reactions.
4. We test and report the accuracy of BIG-BOSS on 75 genome-scale models of cell metabolism including prokaryotes, eukaryotes, and mammalian cells, using both clean and corrupted input datasets.

## 2 REACTION ESTIMATION AS AN OPTIMIZATION PROBLEM

We estimate a cellular goal using measured fluxes (cf. $\mathbf{v}$ in (1)) as input. These fluxes are measured for different cell growth environments, e.g., growth on glucose, or with/without oxygen. The part of the cellular goal that we estimate here is one new metabolic reaction, in particular its stoichiometric coefficients. In other words, we want to learn a new column for $S$ in (1) in order to explain observations better. We focus on one reaction for simplicity, but BIG-BOSS can be used to estimate simultaneously the stoichiometric coefficients of multiple new biochemical reactions.

### 2.1 Bilevel optimization problem

We formulate our problem as a bilevel optimization problem [2]. We start from fluxes $\{\tilde{\mathbf{v}}^{(i)}\}_{i=1}^k$ measured across $k$ growth conditions, which is accomplished by feeding an organism carbon-13 labeled nutrients and tracing their distribution using spectroscopy/spectrometry [1]. Each vector $\tilde{\mathbf{v}}^{(i)}$ contains the fluxes through a subset of all $n$ reactions taking place in the cell. We also fix the flux $z^{(i)} \geq 0$, $z^{(i)} \in \mathbb{R}$, through the new undefined reaction whose stoichiometric coefficients $\mathbf{y} \in \mathbb{R}^m$ we want to estimate. Given a set of coefficients $\mathbf{y}$, and for each growth condition $i$, we assume that the cell fluxes

$\mathbf{v}^{(i)} \in \mathbb{R}^n$ optimize a standard COBRA model:

$$\mathbf{v}^{(i)} \in C^{(i)}(\mathbf{y}) \triangleq \arg\max_{\mathbf{w}^{(i)}} \quad \mathbf{c}^T\mathbf{w}^{(i)} + d^{(i)}z^{(i)} \tag{2}$$

$$\text{subject to} \quad S\mathbf{w}^{(i)} + z^{(i)}\mathbf{y} = \mathbf{b}^{(i)}, \tag{3}$$

$$\mathbf{l}^{(i)} \leq \mathbf{w}^{(i)} \leq \mathbf{u}^{(i)}, \tag{4}$$

where $\mathbf{l}^{(i)}$ and $\mathbf{u}^{(i)}$ (known) are the lower and upper flux bounds, $\mathbf{b}^{(i)}$ (known) is the rate of accumulation or depletion of different metabolites, $\mathbf{c}$ (known) is the relative importance of the different known chemical reactions in the model, $d^{(i)} \in \mathbb{R}$ (known) is the relative importance of the new chemical reaction we want to learn, and $S$ (known) are the stoichiometric coefficients of the known chemical reactions in the model.

Our goal is to find sparse reaction coefficients $\mathbf{y}$ (unknown) for the new reaction, such that the resulting fluxes $\{\mathbf{v}^{(i)}\}_{i=1}^k$ explain the measured fluxes $\{\tilde{\mathbf{v}}^{(i)}\}_{i=1}^k$ as well as possible. For each growth condition $i$, we keep track of which of the $n$ fluxes we measure in a binary matrix $F^{(i)}$ that, when applied to $\mathbf{v}^{(i)}$, selects the correct measured components. This leads to the formulation that BIG-BOSS solves:

$$\min_{\{\mathbf{v}^{(i)}\}_{i=1}^k, \mathbf{y}} \frac{1}{k}\sum_{i=1}^k \left\| F^{(i)}\mathbf{v}^{(i)} - \tilde{\mathbf{v}}^{(i)} \right\|_2^2 + \delta \|\mathbf{y}\|_1 \tag{5}$$

$$\text{subject to } \mathbf{v}^{(i)} \in C_i(\mathbf{y}),$$

where $\delta \geq 0$ controls sparsity. Larger $\delta$ encourages reactions having few reactants and products.

We reformulate this bilevel optimization problem as a single-level optimization problem.

Theorem 2.1. *Problem* (5), *can be written as*

$$\min_{\{\mathbf{v}^{(i)}\}, \mathbf{y}, \{\boldsymbol{\omega}^{(i)}\}, \{\boldsymbol{\mu}^{(i)}\}, \{\boldsymbol{\eta}^{(i)}\}} \frac{1}{k}\sum_{i=1}^k \left\| F^{(i)}\mathbf{v}^{(i)} - \tilde{\mathbf{v}}^{(i)} \right\|_2^2 + \delta \|\mathbf{y}\|_1 \tag{6}$$

$$\textit{subject to} \quad S\mathbf{v}^{(i)} + z^{(i)}\mathbf{y} = \mathbf{b}, \forall i, \tag{7}$$

$$S^T\boldsymbol{\omega}^{(i)} - \boldsymbol{\mu}^{(i)} + \boldsymbol{\eta}^{(i)} = \mathbf{c}, \forall i, \tag{8}$$

$$\mathbf{y}^T\boldsymbol{\omega}^{(i)} \geq d^{(i)}, \forall i, \tag{9}$$

$$\mathbf{c}^T\mathbf{v}^{(i)} + z^{(i)} = \mathbf{b}^T\boldsymbol{\omega}^{(i)} - \mathbf{l}^{(i)T}\boldsymbol{\mu}^{(i)} + \mathbf{u}^{(i)T}\boldsymbol{\eta}^{(i)}, \forall i, \tag{10}$$

$$\mathbf{l}^{(i)} \leq \mathbf{v}^{(i)} \leq \mathbf{u}^{(i)}, \forall i, \tag{11}$$

$$\boldsymbol{\mu}^{(i)}, \boldsymbol{\eta}^{(i)} \geq 0, \forall i, \tag{12}$$

*where $\boldsymbol{\omega}^{(i)}$ is the dual variable for constraint* (3), *and $\boldsymbol{\mu}^{(i)}$ and $\boldsymbol{\eta}^{(i)}$ are dual variables for the upper and lower bounds in* (4).

The proof of Theorem 2.1 follows the same ideas as in [17]. In a nutshell, we write the Karush-Kuhn-Tucker (KKT) conditions for the convex problem (2), and add constraint (10) to say that the primal and dual objective values for (2) are equal, which is true by strong duality.

The bilinear constraints (9) make the problem nonconvex. Furthermore, all of the constraints are coupled by the variable $\mathbf{y}$, the new chemical reaction that we want to learn. If it were not for (9), we could decouple problem (6) into independent generalized LASSO problems [41]. To solve (6)–(12), we use the Alternating Direction Method of Multipliers (ADMM) [4, 6, 11].

**Table 1: Nomenclature of variables and parameters**

| Variable | Space | Description |
|---|---|---|
| | | *Model variables* |
| n | $\mathbb{N}$ | number of fluxes in the cell |
| m | $\mathbb{N}$ | number of metabolites in the cell |
| k | $\mathbb{N}$ | number of experimental conditions |
| p | $\{1, \dots, n\}$ | number of measured fluxes |
| $i$ | $\{1, \dots, k\}$ | condition $i$ |
| $\mathbf{v}, \mathbf{v}^{(i)}$ | $\mathbb{R}^n$ | all fluxes in model |
| $\tilde{\mathbf{v}}, \tilde{\mathbf{v}}^{(i)}$ | $\mathbb{R}^p$ | measured fluxes |
| $\mathbf{c}$ | $\mathbb{R}^n$ | relative importance of reaction |
| $S$ | $\mathbb{R}^{m\times n}$ | stoichiometric coefficients |
| $\mathbf{b}, \mathbf{b}^{(i)}$ | $\mathbb{R}^m$ | rate of accumulation or depletion of metabolites (typically zero) |
| $\mathbf{l}, \mathbf{l}^{(i)}$ | $\mathbb{R}^n$ | flux lower bounds |
| $\mathbf{u}, \mathbf{u}^{(i)}$ | $\mathbb{R}^n$ | flux upper bounds |
| $\mathbf{y}$ | $\mathbb{R}^m$ | coefficients of the new reaction, which we want to learn from data |
| $z, z^{(i)}$ | $\mathbb{R}$ | fixed flux through new reaction |
| $\boldsymbol{\omega}, \boldsymbol{\omega}^{(i)}$ | $\mathbb{R}^m$ | dual variables for linear constraints |
| $\boldsymbol{\mu}, \boldsymbol{\mu}^{(i)}$ | $\mathbb{R}^n$ | dual variables for flux lower bounds, $l$ |
| $\boldsymbol{\eta}, \boldsymbol{\eta}^{(i)}$ | $\mathbb{R}^n$ | dual variables for flux upper bounds, $u$ |
| $d, d^{(i)}$ | $\mathbb{R}$ | relative importance of the new reaction |
| $\delta$ | $\mathbb{R}$ | sparsity regularization weight |
| | | *Factor-graph and ADMM variables* |
| QP | | Quadratic function-node |
| BI$^{(i)}$ | | Bi-linear function-node |
| BD | | Boundary function-node |
| SP | | L1-norm function-node |
| FX | | Fluxes variable-node |
| NR | | New reaction Variable-node |
| DB | | Dual variable for flux bounds variable-node |
| DE$^{(i)}$ | | Dual variable for equilibrium constraint variable-node |
| $\Psi$ | | Primal iterate |
| $\Gamma$ | | Consensus iterate |
| $\Xi$ | | Dual iterate |
| **Prox** | | Proximal operator (PO) map |
| $\rho$ | | ADMM PO tuning parameter |
| $\gamma$ | | ADMM over-relaxation tuning parameter |
| $\alpha$ | | ADMM step-size tuning parameter |
| $\mathbf{n}, \mathbf{n}^{(i)}$ | | Input to PO map |

## 3 SUMMARY OF THE NOTATION USED

All of the variables and parameters that we use are defined in Table 1. Vectors are written in bold, matrices are capitalized, and scalars are lower-case.

A few variables are defined locally, and their definitions only hold inside the section, or subsection, where they are defined. These are the only variables not listed in the notation tables.

## 4 SOLUTION PROCEDURE USING ADMM

BIG-BOSS's inner workings are based on the over-relaxed, distributed consensus, ADMM algorithm [6, 33]. This choice is based

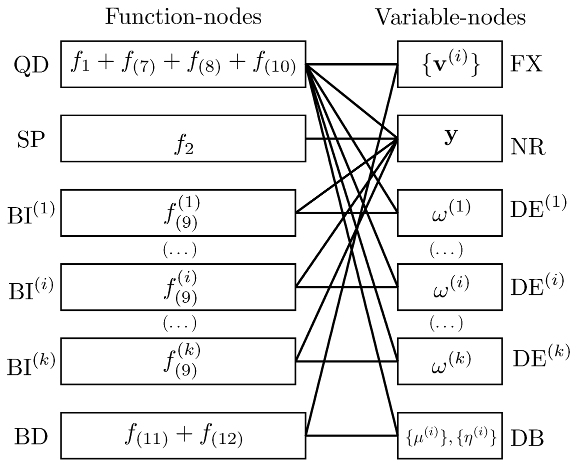

**Figure 1: Representation of (6)-(12) as a factor-graph. Each function-node, and each variable-node, is labeled by two capital letters.**

on the fact that ADMM (i) allows us to easily parallelize BIG-BOSS, (ii) has good empirical performance for several nonsmooth, nonconvex problems [4, 11, 12, 20, 29, 45], and (iii) has global optimality guarantees (under mild convexity assumptions) and a convergence rate that (under proper tuning) equals the convergence rate of the fastest possible first-order method [16], although itself not a first-order method.

Let $f_1(\{\mathbf{v}^{(i)}\}) = \frac{1}{k}\sum_{i=1}^{k}\left\|F^{(i)}\mathbf{v}^{(i)} - \tilde{\mathbf{v}}^{(i)}\right\|_2^2$ and $f_2(\mathbf{y}) = \delta\|\mathbf{y}\|_1$. Let $f_{(7)}$ be a function that takes the value $+\infty$ if the constraint (7) is not satisfied, and zero if it is. Let $f_{(8)}, \ldots, f_{(12)}$ be defined analogously. For the special case of constraints (9), we define one function $f_{(9)}^{(i)}$ for each condition $i$ in (9). To explain how we solve problem (6)–(12) using ADMM, it is convenient to perform two simple rerepresentations of (6)–(12) using the functions $f_i$.

First, we rewrite problem (6)–(12) as

$$\min f_1 + f_2 + f_{(7)} + f_{(8)} + f_{(9)}^{(1)} + \cdots + f_{(9)}^{(k)} + f_{(11)} + f_{(12)}.$$

Second, we represent this unconstrained problem in a *factor-graph* form, i.e., a bipartite graph connecting objective functions (*function-nodes*) to the variables that they depend on (*variable-nodes*), as shown in Figure 1. There are $3 + k$ function-nodes and variable-nodes in total.

We interpret ADMM as an iterative scheme that operates on iterates that live on the edges/nodes of the factor-graph in Figure 1, similar to the approaches in [11, 12, 20]. The function-nodes have the following labels: *quadratic* (QP), *sparse* (SP), *bi-linear* (BI$^{(1)}, \ldots,$ BI$^{(k)}$), and *bound* (BD). The variable-nodes have the following labels: *flux* (FX), *new reaction* (NR), *dual equilibrium* (DE$^{(1)}, \ldots,$ DE$^{(k)}$), and *dual bound* (DB). Each edge (function-node, variable-node), such as (BD, DE$^{(i)}$), has two associated iterates: a primal iterate $\Psi$ and a dual iterate $\Xi$. Both iterates are vectors whose dimension matches the variable in the variable-node that they connect to. Each edge also has an associated parameter $\rho > 0$. For example, the dual iterate and parameter associated with the edge (BD, DE$^{(i)}$) are $\Xi_{\text{BD},\text{DE}^{(i)}}$ (dimension $m$) and $\rho_{\text{BD},\text{DE}^{(i)}}$ respectively. Each variable-node has one associated iterate: a consensus iterate $\Gamma$ whose dimension matches the variable associated with the variable-node. For example, the consensus iterate associated with node DB is $\Gamma_{\text{DB}}$ (dimension $2kn$).

Essential to ADMM is the concept of a *proximal operator* (PO) of a function $f : \mathbb{R}^s \mapsto \mathbb{R}$ with parameter $\rho$. This is a map $\mathbf{Prox}_{f,\rho} : \mathbb{R}^s \mapsto \mathbb{R}^s$ that takes as input $\mathbf{n} \in \mathbb{R}^s$ and outputs

$$\mathbf{Prox}_{f,\rho}(\mathbf{n}) \in \arg\min_{\mathbf{w}\in\mathbb{R}^s} f(\mathbf{w}) + \frac{\rho}{2}\|\mathbf{n} - \mathbf{w}\|_2^2. \tag{13}$$

We denote the POs for the function in each function-node as $\mathbf{Prox}_{\text{QP}}$, $\mathbf{Prox}_{\text{SP}}$, $\mathbf{Prox}_{\text{BI}}$, and $\mathbf{Prox}_{\text{BD}}$ respectively. All POs have uniquely defined outputs.

Given this setup, the over-relaxed distributed-consensus ADMM algorithm leads directly to Algorithm 1, which is the basis of BIG-BOSS. In Algorithm 1, if $Y$ is a variable-node, $\partial Y$ represents all edges incident on $Y$. Similarly, if $X$ is a function-node, $\partial X$ represents all edges incident on $X$. Note that each for-loop in Algorithm 1 can be computed in parallel.

**Algorithm 1** Algorithm for BIG-BOSS

```
1:  while !stopping criteria do
2:      for $E = (X, Y) \in$ edges of factor-graph do
3:          $\Xi_E \leftarrow \Xi_E + \alpha(\gamma\Psi_E - \Gamma_Y + (1-\gamma)\Gamma_Y^{\text{old}})$
4:      end for
5:      for $X \in$ function-nodes of factor-graph do
6:          $\{\Psi_E\}_{E\in\partial X} \leftarrow \mathbf{Prox}_{X,\rho_X}(\{\Gamma_Y - \Xi_E\}_{E\in\partial X})$
7:      end for
8:      for $Y \in$ variable-nodes of factor-graph do
9:          $\Gamma_Y^{\text{old}} \leftarrow \Gamma_Y$
10:         $\Gamma_Y \leftarrow (1-\gamma)\Gamma_Y + \frac{\sum_{E\in\partial Y}\rho_E(\gamma\Psi_E + \Xi_E)}{\sum_{E\in\partial Y}\rho_E}$
11:     end for
12: end while
```

The rate of convergence of ADMM is influenced by the multiple $\rho$'s, $\gamma$, and $\alpha$, and there are various methods for choosing good values or adapting them while ADMM is running [6, 11, 22, 37, 40–42]. As described in Section 4.6, we implement the residual balancing method [37]. Although this scheme is not provably optimal, the algorithm converges very quickly in practice, and the BIG-BOSS user need not worry about tuning in general. In BIG-BOSS, all of the $\rho$'s corresponding to edges incident on the same function-node $X$ in the factor-graph have the same value, $\rho_X$, throughout ADMM's execution. See Section 4.6 for more details. In this work, we run ADMM until all of the iterates are identical between iterations to a numerical absolute tolerance of $10^{-9}$. This is our *stopping criterion.*

Most updates in Algorithm 1 are linear. The exceptions are updates involving POs, which require solving (13) for different functions $f$. A vital part of our work was to find efficient solvers for each PO. Our decomposition of problem (6)-(12) into the specific form of Figure 1 is not accidental. It leads to a very small number of POs, all of which can be evaluated quickly. ($\mathbf{Prox}_{\text{QP}}$ amounts to solving a linear system, and the others can be computed in closed form.) We expand on this in the next sections.

## 4.1 Proximal operator for node QP

The function-node QP is a quadratic objective subject to some linear constraints. To explain how we compute $\mathbf{Prox}_{\text{QP}}$, we start by defining $\mathbf{w}^{(i)} = \{\mathbf{v}^{(i)}, \boldsymbol{\omega}^{(i)}, \boldsymbol{\mu}^{(i)}, \boldsymbol{\eta}^{(i)}, \mathbf{y}\} \in \mathbb{R}^{2(n+m)+m}$ for each condition $i$, writing

$$f_1(\{\mathbf{w}^{(i)}\}) = \frac{1}{k}\sum_{i=1}^{k} \left\| P^{(i)}\mathbf{w}^{(i)} - \mathbf{q}^{(i)} \right\|_2^2,$$

and writing the constraints $f_{(7)} + f_{(8)} + f_{(10)} < +\infty$ as $C^{(i)}\mathbf{w}^{(i)} = \mathbf{e}^{(i)}$, $\forall i \in \{1,\ldots,k\}$, where

$$P^{(i)} = \begin{bmatrix} F^{(i)} & 0 & 0 & 0 & 0 \end{bmatrix}, \; \mathbf{q}^{(i)} = \begin{bmatrix} \tilde{\mathbf{v}}^{(i)} \end{bmatrix},$$

$$C^{(i)} = \begin{bmatrix} S & 0 & 0 & 0 & z^{(i)} \\ 0 & S^T & -I & I & 0 \\ \mathbf{c}^T & -\mathbf{b}^T & \mathbf{l}^{(i)T} & -\mathbf{u}^{(i)T} & 0 \end{bmatrix}, \text{ and } \mathbf{e}^{(i)} = \begin{bmatrix} \mathbf{b}^{(i)} \\ \mathbf{c} \\ -z^{(i)} \end{bmatrix}.$$

Evaluating $\mathbf{Prox}_{\text{QP}}(\{\mathbf{n}^{(i)}\})$ for input vectors $\{\mathbf{n}^{(i)}\} \in \mathbb{R}^{2k(n+m)+m}$ now amounts to finding the unique minimizer of a strongly convex quadratic program with linear constraints, namely,

$$\min_{\{\mathbf{w}^{(i)}\} \in \mathbb{R}^{2k(n+m)+m}} \frac{1}{k}\sum_{i=1}^{k} \|P^{(i)}\mathbf{w}^{(i)} - \mathbf{q}^{(i)}\|_2^2 + \frac{\rho}{2}\sum_{i=1}^{k} \|\mathbf{w}^{(i)} - \mathbf{n}^{(i)}\|_2^2$$
$$\text{subject to } C^{(i)}\mathbf{w}^{(i)} = \mathbf{e}^{(i)}, \forall i \in \{1,\ldots,k\}.$$

We find this unique minimizer by solving a linear system obtained from the KKT conditions. We then have $k$ linear systems that are coupled by the variable $\mathbf{y}$ that is common to each $\mathbf{w}^{(i)}$. We can write each linear system in block form as

$$\begin{bmatrix} \frac{2}{k}P^{(i)T}P^{(i)} + \rho I & C^{(i)T} \\ C^{(i)} & 0 \end{bmatrix} \begin{bmatrix} \mathbf{w}^{(i)} \\ \lambda^{(i)} \end{bmatrix} = \begin{bmatrix} \frac{2}{k}P^{(i)T}\mathbf{q}^{(i)} + \rho\mathbf{n}^{(i)} \\ \mathbf{e}^{(i)} \end{bmatrix}, \quad (14)$$

where $\lambda^{(i)}$ is the Lagrange multiplier for the linear constraints $C^{(i)}\mathbf{w}^{(i)} = \mathbf{e}^{(i)}$. Finally, we stack the $k$ linear systems into one large linear system[1]. For numerical stability reasons, we add $-\beta I$, $\beta = 10^{-12}$, to the system's matrix. This ensures a unique solution. As described in Section 4.5, we also implemented a preconditioner for this linear system to improve its condition number.

To solve the linear system, we compared two different numerical libraries. One is the C library UMFPACK, which we called from Fortran using the Fortran 2003 interface in mUMFPACK 1.0 [19]. The other is the symmetric indefinite solver MA57 [13] from HSL [23]. For both solvers, the symbolic factorization needs to be computed only once. When $\rho$ is updated, only the numerical factorization is recomputed. As other groups have reported [18], we found that MA57's solve phase was faster than UMFPACK's. We therefore used MA57 for all results in this study.

## 4.2 Proximal operator for node DB

Observe that the constraint $f_{(11)} + f_{(12)} < +\infty$ is equivalent to $3k$ constraints of the type $\mathbf{e}^{(i,j)} \leq \mathbf{w}^{(i,j)} \leq \mathbf{q}^{(i,j)}$ for some constant vectors $\mathbf{e}^{(i,j)}, \mathbf{q}^{(i,j)} \in \mathbb{R}^n$: one for each condition $i$ and each of the variables $\mathbf{v}^{(i)}$ (for index $j = 1$), $\boldsymbol{\mu}^{(i)}$ ( for index $j = 3$), and $\boldsymbol{\eta}^{(i)}$ (for index $j = 4$). Hence, computing $\mathbf{Prox}_{\text{DB}}(\{\mathbf{n}^{(i,j)}\})$ for some input $\{\mathbf{n}^{(i,j)}\} \in \mathbb{R}^{3kn}$ amounts to solving $3k$ independent problems

$$\min_{\mathbf{w}^{(i,j)} \in \mathbb{R}^n} \frac{\rho}{2}\left\|\mathbf{w}^{(i,j)} - \mathbf{n}^{(i,j)}\right\|_2^2 \text{ subject to } \mathbf{e}^{(i,j)} \leq \mathbf{w}^{(i,j)} \leq \mathbf{q}^{(i,j)}.$$

[1]If $k$ is very large, we can also split the function-node QP into $k$ function-nodes, one per condition $i$. This will result in $k$ smaller linear systems, now decoupled, that can be solved in parallel.

This problem has a unique minimizer given explicitly by

$$\mathbf{w}^{(i,j)} = \min\{\mathbf{q}^{(i,j)}, \max\{\mathbf{e}^{(i,j)}, \mathbf{n}^{(i,j)}\}\},$$

where min and max are taken component-wise.

## 4.3 Proximal operator for node SP

The PO for node SP, which has $f_2(\mathbf{w}) = \delta\|\mathbf{w}\|_1$, requires finding the minimizer of

$$\min_{\mathbf{w} \in \mathbb{R}^m} \frac{\rho}{2}\|\mathbf{w} - \mathbf{n}\|_2^2 + \delta\|\mathbf{w}\|_1.$$

This minimizer is $\mathbf{w} = \mathrm{S}_{\delta/\rho}(\mathbf{n})$, where $\mathrm{S}_t(\cdot)$ is the element-wise soft-thresholding operator [6] defined, for input scalar $x$, as

$$S_t(x) = \begin{cases} 0, & \text{if } |x| \leq t \\ \text{sign}(x)(|x| - t), & \text{otherwise.} \end{cases} \quad (15)$$

## 4.4 Proximal operator for node BI$^{(i)}$

The $i$th bilinear proximal operator $\mathbf{Prox}_{\text{BI}^{(i)}}$ receives input vectors $\mathbf{n}, \mathbf{n}^{(i)} \in \mathbb{R}^m$, and outputs the minimizer of

$$\min_{\mathbf{y}, \mathbf{w}^{(i)} \in \mathbb{R}^m} \frac{\rho}{2}\|\mathbf{y} - \mathbf{n}\|_2^2 + \frac{\rho}{2}\left\|\boldsymbol{\omega}^{(i)} - \mathbf{n}^{(i)}\right\|_2^2 \quad (16)$$
$$\text{subject to } \mathbf{y}^T\boldsymbol{\omega}^{(i)} \geq d^{(i)}. \quad (17)$$

To solve this problem, we consider two cases. If $\mathbf{n}^{(i)T}\mathbf{n} \geq d^{(i)}$, we know that the minimizer is $(\mathbf{y}, \mathbf{w}^{(i)}) = (\mathbf{n}, \mathbf{n}^{(i)})$. Otherwise, we know that the constraint (17) is active, and we compute the minimizer from among the set of solutions of the KKT conditions for the (now constrained) problem. The KKT conditions are

$$\rho\mathbf{y} - \rho\mathbf{n} - \lambda\boldsymbol{\omega}^{(i)} = 0, \; \rho\boldsymbol{\omega}^{(i)} - \rho\mathbf{n}^{(i)} - \lambda\mathbf{y} = 0, \; \mathbf{y}^T\boldsymbol{\omega}^{(i)} = d^{(i)},$$

where $\lambda$ is the dual variable for the constraint $\mathbf{y}^T\boldsymbol{\omega}^{(i)} \geq d^{(i)}$. By defining $x = \lambda/\rho$ and reformulating the KKT system, we arrive at the following quartic equation that determines the value of $\lambda$:

$$d^{(i)}x^4 + (-2d^{(i)} - \mathbf{n}^T\mathbf{n}^{(i)})x^2 + (\mathbf{n}^T\mathbf{n} + \mathbf{n}^{(i)T}\mathbf{n}^{(i)})x$$
$$+ (d^{(i)} - \mathbf{n}^T\mathbf{n}^{(i)}) = 0. \quad (18)$$

We solve this quartic equation in closed form, using the Quartic Polynomial Program Fortran code [10, 32]. For each real root, $x$, we compute $\mathbf{y}$ and $\boldsymbol{\omega}^{(i)}$ as

$$\mathbf{y} = \frac{\mathbf{n} - x\mathbf{n}^{(i)}}{1 - x^2}, \quad \boldsymbol{\omega}^{(i)} = \frac{\mathbf{n}^{(i)} - x\mathbf{n}}{1 - x^2}, \quad (19)$$

and we keep the solution that minimizes the objective function (16).

## 4.5 Preconditioner

The convergence rate of the ADMM can be improved by preconditioning the data matrices associated with the node QP [37, 38].

We use the Ruiz equilibration method [34] to compute positive definite diagonal matrices, $Q$ and $R$, with which we scale all of the variables $\{\mathbf{w}^{(i)} = \{\mathbf{v}^{(i)}, \boldsymbol{\omega}^{(i)}, \boldsymbol{\mu}^{(i)}, \boldsymbol{\eta}^{(i)}, \mathbf{y}\}\}$ in our formulation, as well as the constraints $C^{(i)}\mathbf{w}^{(i)} = \mathbf{e}^{(i)}\forall i$, associated with the node QP. The new scaled variables are defined through the change of variable $\mathbf{w}^{(i)} \to Q\mathbf{w}^{(i)}$, and the new constraints then transform as

$C^{(i)}\mathbf{w}^{(i)} = \mathbf{e}^{(i)} \rightarrow RC^{(i)}Q\mathbf{w}^{(i)} = R\mathbf{e}^{(i)}\forall i$. Note that all of the copies of $\mathbf{y}$ inside each $\mathbf{w}^{(i)}$ get scaled exactly in the same way.

This scaling affects the different proximal operators in different ways. The node QP, that requires computing $\mathbf{Prox}_{\text{QP}}(\{\mathbf{n}^{(i)}\})$, now requires solving a modified version of (14), namely, the following system of linear equations (coupled by $\mathbf{y}$),

$$\begin{bmatrix} Q\frac{2}{k}P^{(i)^T}P^{(i)}Q + \rho I & QC^{(i)^T}R \\ RC^{(i)}Q & 0 \end{bmatrix} \begin{bmatrix} \mathbf{w}^{(i)} \\ \lambda^{(i)} \end{bmatrix} = \begin{bmatrix} Q\frac{2}{k}P^{(i)^T}\mathbf{q}^{(i)} + \rho\mathbf{n}^{(i)} \\ R\mathbf{e}^{(i)} \end{bmatrix}.$$

For $\mathbf{Prox}_{\text{DB}}(\{\mathbf{n}^{(i,j)}\})$, let $Q^{(j)}, j = 1, 3, 4$, be the diagonal block of $Q$ that scales the variables $\mathbf{v}^{(i)}$ $(j = 1)$, $\boldsymbol{\mu}^{(i)}$ $(j = 3)$, and $\boldsymbol{\eta}^{(i)}$ $(j = 4)$. We now solve $3k$ problems of the type

$$\min_{\mathbf{w}^{(i,j)} \in \mathbb{R}^n} \frac{\rho}{2} \left\| \mathbf{w}^{(i,j)} - \mathbf{n}^{(i,j)} \right\|_2^2 \text{ subject to} \tag{20}$$

$$Q^{(j)^{-1}}\mathbf{e}^{(i,j)} \le \mathbf{w}^{(i,j)} \le Q^{(j)^{-1}}\mathbf{q}^{(i,j)}, \tag{21}$$

which has a unique minimizer

$$\mathbf{w}^{(i)} = \min\{Q^{(j)^{-1}}\mathbf{q}^{(i,j)}, \max\{Q^{(j)^{-1}}\mathbf{e}^{(i,j)}, \mathbf{n}^{(i,j)}\}\}.$$

For the node SP, let $Q^{(5)}$ be the diagonal block of $Q$ that scales the variable $\mathbf{y}$. The new $\mathbf{Prox}_{\text{SP}}(\mathbf{n})$ now outputs

$$\mathbf{w} = \mathrm{S}_{Q^{(5)^{-1}}\delta/\rho}(\mathbf{n}),$$

where, for the $r$th component of $\mathbf{n}$, we use the $r$th diagonal element of $Q^{(5)^{-1}}\delta/\rho$ in the operator S, that is applied component-wise.

For the $i$th bilinear proximal operator $\mathbf{Prox}_{\text{BI}^{(i)}}(\mathbf{n}, \mathbf{n}^{(i)})$, we now solve

$$\begin{aligned} \arg\min_{\mathbf{y}, \boldsymbol{\omega}^{(i)} \in \mathbb{R}^m} \quad & \frac{\rho}{2}\|\mathbf{y} - \mathbf{n}\|_2^2 + \frac{\rho}{2}\left\|\boldsymbol{\omega}^{(i)} - \mathbf{n}^{(i)}\right\|_2^2 \\ \text{subject to} \quad & \mathbf{y}^T Q^{(5)} Q^{(2)} \boldsymbol{\omega}^{(i)} \ge d^{(i)}, \end{aligned}$$

where $Q^{(2)}$ is the diagonal block of $Q$ that scales the variable $\boldsymbol{\omega}^{(i)}$. We modify the Ruiz equilibration method such that $Q^{(5)}Q^{(2)} = constant \times I$, and hence we can solve this PO using the same method as before.

## 4.6 Adaptive penalty

Our ADMM scheme allows for one $\rho$ per each edge in the factor-graph in Fig. 1. In BIG-BOSS, these $\rho$'s are chosen automatically, such that a user does not need to worry about tuning. All of the $\rho$'s associated with edges incident on the same function-node have the same value. Hence, we refer to the different $\rho$'s by their respective PO. We update the $\rho$'s for each PO dynamically every 1000 iterations using the *residual balancing method* [6], which we now explain.

Similar to [37, 40], we define the vector of relative primal residuals at iteration $10^3 \times t$, and for any proximal operator $X$, as

$$r_{\text{prim}}(t) = \frac{\{\Psi_E(t) - \Gamma_Y(t)\}_{E:E=(X,Y)\in\partial X}}{\max\{\{\max\{|\Psi_E(t)|, |\Gamma_Y(t)|\}\}_{E:E=(X,Y)\in\partial X}\}},$$

where, we recall, $\partial X$ are all the edges incident on node $X$.

The vector of relative dual residuals for the proximal operator $X$ = QP at iteration $10^3 \times t$ is defined as follows. Let

$$C_1 = \max_{i=1}^{k} \left\| \frac{2}{k} P^{(i)^T} P^{(i)} \{\Psi_E(t+1)\}_{E\in\partial X} \right\|_\infty,$$

$$C_2 = \|\{\Xi_E(t+1)\}_{E\in\partial X}\|_\infty, \text{ and } C_3 = \max_{i=1}^{k} \left\| -\frac{2}{k} P^{(i)^T} \mathbf{q}^{(i)} \right\|_\infty.$$

We define

$$r_{\text{dual}}(t+1) = \frac{\{\Gamma_Y(t+1) - \Gamma_Y(t)\}_{Y:(X,Y)\in\partial X}}{\max\{C_1, C_2, C_3\}}.$$

If $X$ is a PO other than QP, we define

$$r_{\text{dual}}(t+1) = \frac{\{\Gamma_Y(t+1) - \Gamma_Y(t)\}_{Y:(X,Y)\in\partial X}}{C_2},$$

since $C_1$ and $C_3$ do not exist when $X$ is not QP.

The $\rho$ update rule is then defined as follows. Once the $\infty$-norm of both the primal and the dual residuals falls below a threshold of $10^{-3}$, we start updating $\rho$ as

$$\rho(t+1) = \max\{10^{-3}, \min\{10^3, \tilde{\rho}(t+1)\}\}$$

where

$$\tilde{\rho}(t+1) = \begin{cases} \tau\rho(t) & \text{if } \left|\ln\left(\|r_{\text{prim}}(t)\|_\infty / \|r_{\text{dual}}(t)\|_\infty\right)\right| \ge \ln(\eta), \\ \rho(t) & \text{otherwise,} \end{cases}$$

where $\tau = \sqrt{\|r(t)_{\text{prim}}\|_\infty / \|r(t)_{\text{dual}}\|_\infty}$ and $\eta > 0$, which in this study we set to 2, is a threshold to prevent the $\rho$ from changing too frequently. Note that a frequently-changing $\rho$ requires many numerical re-factorization for solving the linear system in the node QP. Furthermore, a frequently-changing $\rho$ might compromise the convergence of the ADMM.

## 4.7 Step size

The step size $\alpha$ in Algorithm 1 can be adjusted to speed up convergence. We used a default step size of 1.0 for all problems. If the problem did not converge to the desired tolerance within 2 million iterations, we re-ran the problem with a simple step size heuristic. We use $\alpha = 1$ for 10,000 iterations, $\alpha = 0.1$ for 100,000 iterations, then $\alpha = 0.001$ until convergence, or the iteration limit is reached.

## 5 EXPERIMENTAL RESULTS

## 5.1 BOSS versus BIG-BOSS

We compare BIG-BOSS with BOSS in terms of running speed, and accuracy. To make a valid comparison, we set $k = 1$ and $\delta = 0$, such that BIG-BOSS is doing the same inference as BOSS [17]. BOSS requires choosing an off-the-shelf nonlinear programming solver, we choose IPOPT [39] (v24.7.1), an open-source, interior-point solver used to solve large-scale nonlinear programs.

We generate synthetic flux measurements, $\tilde{\mathbf{v}}$, using the latest metabolic network of *E. coli* [31] called iML1515. This network has 1, 877 metabolites and 2, 712 reactions. We then hide one goal reaction from the network, $\mathbf{y}$, and try to infer it from flux data.

Using a single CPU core (Intel i7, 3.4GHz), BOSS is able to learn $\mathbf{y}$ for this model in 13.3 seconds (wall time) using IPOPT to primal and dual residuals of $1.05 \times 10^{-11}$ and $7.23 \times 10^{-11}$. Using the same CPU core, BIG-BOSS is able to learn $\mathbf{y}$ in 113 seconds to primal and dual residuals of $2.73 \times 10^{-10}$ and $6.63 \times 10^{-10}$. In Appendix A.3, we report the run-time for other accuracy values. Although BIG-BOSS is slower than BOSS, it is still fast, and, more importantly, allows $k > 1, \delta > 0$, and the use of multiple cores for large models.

Now we look at how well the recovered reaction coefficients $\mathbf{y}$ compare to the true coefficients that we hide, both in BOSS and BIG-BOSS. We do so by (i) directly comparing the ground truth with the recovered $\mathbf{y}$, as well as, (ii) using the recovered $\mathbf{y}$ to predict measured fluxes (training fluxes).

For BOSS, the recovered goal reaction coefficients $\mathbf{y}$ closely resemble the true reaction coefficients, with Spearman rank correlation, $r^{\text{Spearman}} = 0.996$ (Fig. 2a). With the recovered $\mathbf{y}$ back into the model, we then solve (1) to simulate fluxes. For BOSS, the estimated goal reaction has zero flux, resulting in simulated fluxes that do not resemble the training fluxes (Fig. 2c). To help BOSS, we make $\mathbf{y}$ sparse by zeroing coefficients with absolute value smaller than a fixed threshold $\theta$. Since this cannot be done in a real setting, it shows a limitation in BOSS. We tested thresholds between 0 to 1 and found that values between $10^{-3}$ to $10^{-1}$ result in accurate flux predictions ($0.98 \leq R^2 \leq 1.0$).

We perform the same experiment using BIG-BOSS, which estimates $\mathbf{y}$ with $r^{\text{Spearman}} = 0.871$ (Fig. 2b). Fluxes predicted using this new reaction are accurate on the training data to $R^2 = 0.994$ without zeroing additional coefficients (Fig. 2d).

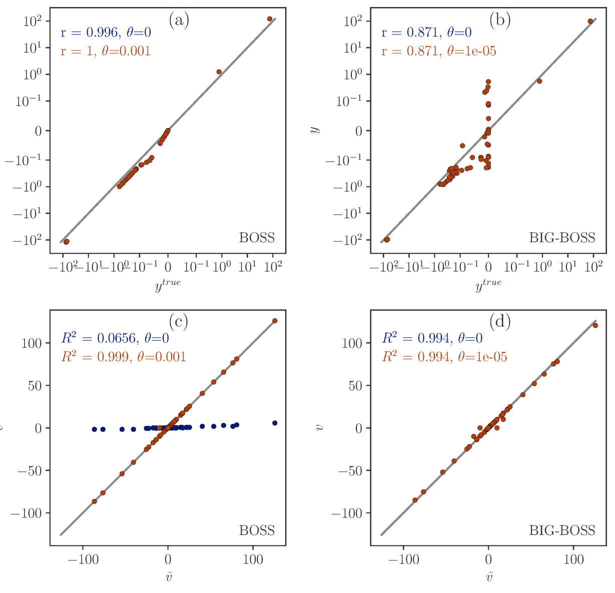

**Figure 2: Performance of BOSS and BIG-BOSS. Coefficients of $\mathbf{y}$ estimated using BOSS (a) and BIG-BOSS (b) compared against true coefficients. Fluxes predicted using the goal reaction estimated by BOSS (c) and BIG-BOSS (d) compared against training fluxes $\tilde{\mathbf{v}}$. $R^2$: coefficient of variation. $r$: Spearman rank correlation. $\theta$: threshold for zeroing out elements of $\mathbf{y}$ smaller than $\theta$.**

## 5.2 Experiments with 75 large-scale models

We test our algorithm's performance on 75 large-scale metabolic networks in the BiGG database [25]. Model sizes ranged from 95 to 10, 600 reactions and 72 to 5, 835 metabolites. The median model size is 2, 727 reactions and 1, 942 metabolites. For each model, we generate synthetic training fluxes by solving Problem (1) using the existing cellular goal reaction. To generate test data, we do as above but in a different simulated nutrient environment. This is accomplished by changing the bounds $\mathbf{l}$ and $\mathbf{u}$ on nutrient transport reactions. Our algorithm then receives as input the training fluxes, the bounds, and an incomplete stoichiometric matrix, which is missing the column corresponding to the "ground truth" cellular goal reaction that we want to estimate.

We evaluate our algorithm's performance based on three metrics: (i) how well the estimated reaction allows to predict training fluxes, (ii) how well the estimated reaction allows to predict testing fluxes, and (iii) how close the estimated reaction coefficients are to the ground truth coefficients. To evaluate metrics (i) and (ii), we insert the estimated reaction into the metabolic network, having first removed the true goal reaction, and then simulate fluxes.

Based on the coefficient of determination ($R^2$) between measured and predicted fluxes, both training and test performances were high (median $R^2$ of 0.993 and 0.977) (Table 2). To assess the significance of these $R^2$ values, we compare them with fluxes generated from 1, 000 random goal reactions. These random goal reactions are obtained by randomly shuffling the coefficients of the ground truth coefficients. We say that the $R^2$ based on our algorithm is significant if it exceeds 95% of the $R^2$ values that were computed using the 1,000 random goal reactions. Overall, $R^2$ values are significant for 70.1% of the models for training data and 67.2% of the models for test data (Table 2). The ground truth goal reaction is recovered accurately: across 75 models, the median Pearson correlation was 1.0 (Table 3). The Spearman rank correlation was lower, with median 0.48. The fact that predicted fluxes are accurate indicates that metabolic flux predictions are relatively insensitive to the precise recovery of the true coefficients in $\mathbf{y}$.

In biological experiments, the majority of fluxes are unmeasured, so 50% missing flux measurements is common [1]. We test the effect of missing flux measurements by randomly removing 10% and 50% of measurements, repeated five times. The main consequence of missing measurements is that recovering the ground truth goal is more difficult. With 10% and 50% of missing measurements, median Pearson correlations of −0.023 and 0.0012 are obtained (Table 2). The accuracy of predicted fluxes also deteriorate when measurements are missing (Table 2). However, depending on which reactions are measured, fluxes can be predicted with $R^2 > 0.90$ for 6.67% of the cases with 10% of missing measurements, and for 3.16% of the cases with 50% of missing measurements, in the case of test data. This result shows that certain fluxes are more important than others, and that BIG-BOSS can recover goal reactions when those important fluxes are measured.

## 5.3 Effect of multiple data types

Next, we investigate if including data types other than metabolic fluxes can improve reaction estimation. In particular, the concentration of over 95% of cell proteins and transcripts (RNA) can be measured [35]. These data types have been used to identify statistically enriched biological functions [21] but not to estimate reactions.

**Table 2: Performance for 75 BiGG models. $R^2$: coefficient of determination. By 'significant' we mean that the $R^2$ based on our algorithm's goal reaction exceeds 95% of the $R^2$ values that are based on 1,000 random goal reactions.**

| | % missing | $R^2$ | | % cases, | % cases, |
|---|---|---|---|---|---|
| Data | fluxes | mean | median | $R^2$ > 0.90 | significant |
| Train | 0.0 | 0.711 | 0.993 | 63.8 | 70.1 |
| | 10.0 | -6.89 | 0.56 | 12.2 | 19.7 |
| | 50.0 | -38.9 | 0.604 | 4.98 | 6.45 |
| Test | 0.0 | 0.666 | 0.977 | 59.4 | 67.2 |
| | 10.0 | -4.7 | 0.33 | 6.67 | 15 |
| | 50.0 | -29.2 | 0.431 | 3.16 | 6.56 |

**Table 3: Goal reaction recovery accuracy for 75 BiGG models.**

| % missing fluxes | Metric | min | max | mean | median |
|---|---|---|---|---|---|
| 0.0 | Pearson | 0.021 | 1 | 0.96 | 1 |
| | Spearman | 0.1 | 0.76 | 0.48 | 0.48 |
| 10.0 | Pearson | -0.74 | 1 | 0.063 | -0.023 |
| | Spearman | -0.12 | 0.9 | 0.59 | 0.63 |
| 50.0 | Pearson | -0.92 | 0.82 | -0.032 | 0.0012 |
| | Spearman | -0.27 | 0.76 | 0.38 | 0.4 |

Thus, to include protein concentrations in our algorithm, we extend (1), similar to [3]. (Details in Supplement A.1.)

We estimated goal reactions with three different data sets: only metabolic fluxes, only proteins, and both. To reflect realistic data, we include only metabolic fluxes for central metabolism, and include all proteins—this is achievable using proteomics, or with RNA sequencing where RNA is used to estimate protein concentrations. We then use the estimated goal reaction to predict all metabolic fluxes and protein concentrations (including values not seen by the algorithm) in left-out conditions to assess the test performance.

When only metabolic fluxes are provided, the prediction accuracy is low (mean $R^2 = 0.281$) (Table 4). When only proteins are provided, the accuracy is even lower (mean $R^2 = -16.4$). One reason for the low accuracy is that protein concentrations are not always directly proportional to flux—they only provide an upper bound. When both fluxes and proteins are provided, the average test accuracy is $R^2 = 0.567$, which is double the accuracy with only flux measurements.

**Table 4: Test performance of flux and protein predictions given different coverage of protein and flux measurements. Coverage refers to the percentage of fluxes or proteins that are measured and used as input to BIG-BOSS.**

| Coverage (%) | | $R^2$ | | |
|---|---|---|---|---|
| Flux | Protein | mean | min | max |
| 57 | 100 | 0.567 | 0.108 | 0.77 |
| 57 | 0 | 0.281 | 0.225 | 0.322 |
| 0 | 100 | -16.375 | -23.556 | -11.293 |

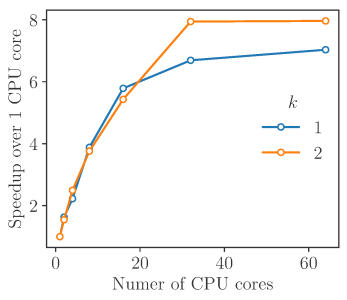

**Figure 3: Speedup with multiple CPU cores for number of conditions $k = 1$ and 2 using the fine-grained implementation of BIG-BOSS.**

## 5.4 Multi-core speedup

Problem (5) can grow rapidly in size and can benefit from parallel computing. BOSS cannot exploit parallel computational resources, but BIG-BOSS can. We thus tested how BIG-BOSS scales with multiple processors, using the same metabolic network as in Section 5.1. To maximize parallelism, we implemented a fine-grained version of our algorithm, similar to [20]. The fact that we are using ADMM is important to achieve this. This implementation involves splitting node QP: one for every constraint. By combining this fine-grained version of QP with BD and BI, we have our fine-grained BIG-BOSS. This implementation of BIG-BOSS gave 1.6× speedup with 2 CPUs, and 7× speedup with 64 CPUs (Fig. 3). With $k = 2$ the speedup was greater, achieving 8× with 32 and 64 CPUs.

In these experiments, we did not perform preconditioning or adaptive penalty to prevent factors besides CPU count from affecting speed. To parallelize our code, we used OpenACC and the PGI compiler 18.10. These tests were performed on Google Compute Engine virtual machines with 32-core (64-thread) Intel Xeon 2.30 GHz processors and 64 GB RAM.

## 6 CONCLUSION

In this work, we addressed the problem of estimating cellular goal reactions from measured fluxes in the context of constraint-based modeling of cell metabolism. This problem was previously addressed by the BOSS algorithm [17], which successfully demonstrated the viability of this approach on a model of *Saccharomyces cerevisiae* central metabolism composed of 60 metabolites and 62 reactions. Here, we developed a new method that extends BOSS and demonstrates its performance on 75 metabolic networks having up to 10,600 reactions and 5,835 metabolites.

Our method successfully estimated goal reactions that enabled accurate prediction of metabolic fluxes in new nutrient environments (median coefficient of determination, $R^2 = 0.98$). Furthermore, the stoichiometric coefficients of the estimated reactions matched those of the ground truth reactions that were used to generate the training fluxes (median Pearson correlation, $r = 0.96$).

As with the original BOSS, our algorithm involves a nonconvex optimization problem with bilinear constraints. ADMM is an efficient approach for solving problems with many bilinear constraints, and more generally, nonconvex quadratic constraints [24]. For our problem, the number of bilinear constraints can be large, as it increases with the number of conditions in the data. Besides scalability, BIG-BOSS allows for regularization, modularity—enabling extensions, and can use multiple data types (fluxes and protein concentrations), which improves prediction accuracy.

Genome-scale metabolic network models have been invaluable in numerous studies including infectious disease and cancer metabolism [5]. The success of these studies depends on the availability of a reaction representing cellular goals. For many important cellular systemss, such as human tissue or microbiomes, such reactions are still challenging to determine experimentally [14]. Our study shows that a data-driven approach for cellular goal estimation is promising. This approach is most effective when we have high coverage flux measurements or complementary data types, such as flux measurements for parts of the network and protein or RNA abundance measurements for most of the metabolic genes. BIG-BOSS is thus suited for analyzing large gene expression data sets, e.g., from cancer tissue. BIG-BOSS is available at https://github.com/laurenceyang33/cellgoal.

## 7 ACKNOWLEDGEMENTS

This work was supported by the NIH grants 2R01GM057089 and 1U01AI124302, NSF grant IIS-1741129, and Novo Nordisk Foundation grant NNF10CC1016517.

## REFERENCES

[1] M. R. Antoniewicz. 2015. Methods and advances in metabolic flux analysis: a mini-review. *J Ind Microbiol Biot* 42, 3 (2015), 317–325.

[2] J. F. Bard. 2013. *Practical bilevel optimization: algorithms and applications*. Vol. 30. Springer Science & Business Media.

[3] Q. K. Beg, A. Vazquez, J. Ernst, M. A. de Menezes, Z. Bar-Joseph, A.-L. Barabasi, and Z. N. Oltvai. 2007. Intracellular crowding defines the mode and sequence of substrate uptake by *Escherichia coli* and constrains its metabolic activity. *Proc. Natl. Acad. Sci. USA* 104 (2007), 12663–12668.

[4] J. Bento, N. Derbinsky, J. Alonso-Mora, and J. S. Yedidia. 2013. A message-passing algorithm for multi-agent trajectory planning. In *Adv Neur In.* 521–529.

[5] A. Bordbar, J. M. Monk, Z. A. King, and B. O. Palsson. 2014. Constraint-based models predict metabolic and associated cellular functions. *Nat Rev Genet* 15 (2014), 107–120.

[6] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, et al. 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine learning* 3, 1 (2011), 1–122.

[7] E. Brunk, S. Sahoo, D. C. Zielinski, A. Altunkaya, A. Dräger, N. Mih, F. Gatto, A. Nilsson, G. A. Preciat Gonzalez, M. K. Aurich, et al. 2018. Recon3D enables a three-dimensional view of gene variation in human metabolism. *Nat Biotechnol* 36, 3 (2018), 272.

[8] M. P. Brynildsen, J. A. Winkler, C. S. Spina, I. C. MacDonald, and J. J. Collins. 2013. Potentiating antibacterial activity by predictably enhancing endogenous microbial ROS production. *Nat Biotechnol* 31, 2 (2013), 160–165.

[9] A. P. Burgard and C. D. Maranas. 2003. Optimization-based framework for inferring and testing hypothesized metabolic objective functions. *Biotechnol Bioeng* 82 (2003), 670–677.

[10] R. L. Carmichael. [n. d.]. Public Domain Aeronautical Software. http://www.pdas.com. Accessed: 2018-08-12.

[11] N. Derbinsky, J. Bento, V. Elser, and J. S. Yedidia. 2013. An improved three-weight message-passing algorithm. *arXiv preprint arXiv:1305.1961* (2013).

[12] N. Derbinsky, J. Bento, and J. S. Yedidia. 2013. Methods for integrating knowledge with the three-weight optimization algorithm for hybrid cognitive processing. *arXiv preprint arXiv:1311.4064* (2013).

[13] I. S. Duff. 2004. MA57—a code for the solution of sparse symmetric definite and indefinite systems. *ACM T Math Soft* 30, 2 (2004), 118–144.

[14] A. M. Feist and B. O. Palsson. 2016. What do cells actually want? *Genome biology* 17, 1 (2016), 110.

[15] O. Folger, L. Jerby, C. Frezza, E. Gottlieb, E. Ruppin, and T. Shlomi. 2011. Predicting selective drug targets in cancer through metabolic networks. *Mol Syst Biol* 7, 1 (2011), 501.

[16] G. França and J. Bento. 2016. An explicit rate bound for over-relaxed ADMM. In *IEEE International Symposium on Information Theory*. IEEE, 2104–2108.

[17] E. P. Gianchandani, M. A. Oberhardt, A. P. Burgard, C. D. Maranas, and J. A. Papin. 2008. Predicting biological system objectives de novo from internal state measurements. *BMC bioinformatics* 9, 1 (2008), 43.

[18] N. I. M. Gould, J. A. Scott, and Y. Hu. 2007. A numerical evaluation of sparse direct solvers for the solution of large sparse symmetric linear systems of equations. *ACM T Math Software* 33, 2 (2007), 10.

[19] L Hanyk. [n. d.]. UMFPACK Fortran interface. http://geo.mff.cuni.cz/~lh/Fortran/UMFPACK/. Accessed: 2018-08-18.

[20] N. Hao, A. Oghbaee, M. Rostami, N. Derbinsky, and J. Bento. 2016. Testing fine-grained parallelism for the ADMM on a factor-graph. In *IEEE International Parallel and Distributed Processing Symposium Workshops*. IEEE, 835–844.

[21] Y. Hart, H. Sheftel, J. Hausser, P. Szekely, N. B. Ben-Moshe, Y. Korem, A. Tendler, A. E. Mayo, and U. Alon. 2015. Inferring biological tasks using Pareto analysis of high-dimensional data. *Nature Methods* 12 (2015), 233–235.

[22] B. S. He, H. Yang, and S. L. Wang. 2000. Alternating direction method with self-adaptive penalty parameters for monotone variational inequalities. *Journal of Optimization Theory and Applications* 106, 2 (2000), 337–356.

[23] HSL. 2013. A collection of Fortran codes for large scale scientific computation. http://www.hsl.rl.ac.uk

[24] K. Huang and N. D. Sidiropoulos. 2016. Consensus-ADMM for general quadratically constrained quadratic programming. *IEEE Transactions on Signal Processing* 64, 20 (2016), 5297–5310.

[25] Z. A. King, J. Lu, A. Dräger, P. Miller, S. Federowicz, J. A. Lerman, A. Ebrahim, B. O. Palsson, and N. E. Lewis. 2015. BiGG Models: A platform for integrating, standardizing and sharing genome-scale models. *Nucleic Acids Res* 44, D1 (2015), D515–D522.

[26] C. Kuo, A. W. T. Chiang, I. Shamie, M. Samoudi, J. M. Gutierrez, and N. E. Lewis. 2018. The emerging role of systems biology for engineering protein production in CHO cells. *Curr Opin Biotechnol* 51 (2018), 64–69.

[27] C. J. Lloyd, A. Ebrahim, L. Yang, Z. A. King, E. Catoiu, E. J. O'Brien, J. K. Liu, and B. O. Palsson. 2018. COBRAme: A computational framework for genome-scale models of metabolism and gene expression. *PLoS Comput Biol* 14, 7 (2018), e1006302.

[28] R. A. Majewski and M. M. Domach. 1990. Simple constrained-optimization view of acetate overflow in *E. coli*. *Biotechnol Bioeng* 35 (1990), 732–738.

[29] C. J. M. Mathy, F. Gonda, D. Schmidt, N. Derbinsky, A. A. Alemi, J. Bento, F. M. Delle Fave, and J. S. Yedidia. 2015. SPARTA: Fast global planning of collision-avoiding robot trajectories. In *NIPS 2015 Workshop on Learning, Inference, and Control of Multi-agent Systems*.

[30] J. Monk, J. Nogales, and B. O. Palsson. 2014. Optimizing genome-scale network reconstructions. *Nat Biotechnol* 32 (2014), 447–452.

[31] J. M. Monk, C. J. Lloyd, E. Brunk, N. Mih, A. Sastry, Z. King, R. Takeuchi, W. Nomura, Z. Zhang, H. Mori, et al. 2017. iML1515, a knowledgebase that computes Escherichia coli traits. *Nat Biotechnol* 35, 10 (2017), 904.

[32] A. H. Morris Jr. 1993. *NSWC library of mathematics subroutines*. Technical Report. NAVAL SURFACE WARFARE CENTER DAHLGREN VA.

[33] R. Nishihara, L. Lessard, B. Recht, A. Packard, and M. I. Jordan. 2015. A general analysis of the convergence of ADMM. (2015). arXiv:1502.02009

[34] D. Ruiz. 2001. *A scaling algorithm to equilibrate both rows and columns norms in matrices*. Technical Report. CM-P00040415.

[35] A. Schmidt, K. Kochanowski, S. Vedelaar, E. Ahrné, B. Volkmer, L. Callipo, K. Knoops, M. Bauer, R. Aebersold, and M. Heinemann. 2016. The quantitative and condition-dependent Escherichia coli proteome. *Nat Biotechnol* 34 (2016), 104–110.

[36] R. Schuetz, L. Kuepfer, and U. Sauer. 2007. Systematic evaluation of objective functions for predicting intracellular fluxes in *Escherichia coli*. *Mol Syst Biol* 3 (2007), 119.

[37] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. 2017. OSQP: An Operator Splitting Solver for Quadratic Programs. (Nov. 2017). arXiv:math.OC/1711.08013

[38] R. Takapoui and H. Javadi. 2016. Preconditioning via diagonal scaling. (2016). arXiv:1610.03871

[39] A. Wächter and L. T. Biegler. 2006. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math Program* 106, 1 (2006), 25–57.

[40] B. Wohlberg. 2017. ADMM penalty parameter selection by residual balancing. (2017). arXiv:1704.06209

[41] Y. Xu, M. Liu, Q. Lin, and T. Yang. 2017. ADMM without a Fixed Penalty Parameter: Faster Convergence with New Adaptive Penalization. In *Adv Neur In.* 1267–1277.

[42] Z. Xu, M. A. T. Figueiredo, and T. Goldstein. 2016. Adaptive ADMM with spectral penalty parameter selection. (2016). arXiv:1605.07246

[43] H. Yim, R. Haselbeck, W. Niu, C. Pujol-Baxley, A. Burgard, et al. 2011. Metabolic engineering of Escherichia coli for direct production of 1,4-butanediol. *Nat Chem Biol* 7 (2011), 445–452.

[44] Q. Zhao, A. I. Stettner, E. Reznik, I. Ch. Paschalidis, and D. Segrè. 2016. Mapping the landscape of metabolic goals of a cell. *Genome Biol* 17, 1 (2016), 109.

[45] D. Zoran, D. Krishnan, J. Bento, and B. Freeman. 2014. Shape and illumination from shading using the generic viewpoint assumption. In *Adv Neur In.* 226–234.

## A DETAILS FOR REPRODUCIBILITY

### A.1 Model formulation including proteins

Our model formulation for including protein concentrations is based on [3] and [27]. First, we denote $\mathbf{p}$ as the vector of protein concentrations, whose length is the number of proteins in the metabolic network. Each protein has a molecular weight, which is stored in $\mathbf{a}$. One or more proteins combine to form enzyme complexes, $\mathbf{e}$, which are ultimately the molecules that catalyze metabolic reactions. The rate of a metabolic reaction is limited by the catalytic efficiency of an associated enzyme and that enzyme's concentration. Since multiple enzymes can sometimes catalyze the same reaction, we have a matrix of catalytic efficiencies, $K$ mapping reactions to all possible enzymes. The total protein mass of a cell, $t$, can be measured, and this quantity imposes an upper bound on the sum of individual protein concentrations multiplied by their molecular weights, $\mathbf{a}$. Finally, each protein can be part of one or more enzyme complexes, and each enzyme complex is comprised of a fixed number of a particular protein. This protein to enzyme mapping is encoded in the matrix $E$.

By adding these relationships to (1), we have the following linear problem:

$$\begin{aligned} \max_{\mathbf{v},\mathbf{p},\mathbf{e}} \quad & \mathbf{c}^T\mathbf{v} + \mathbf{f}^T\mathbf{p} \\ \text{subject to} \quad & S\mathbf{v} = \mathbf{b}, \\ & \mathbf{v} - K\mathbf{e} + \sigma^{(1)} = 0, \\ & \mathbf{a}^T\mathbf{p} + \sigma^{(2)} = t, \\ & E\mathbf{e} - \mathbf{p} + \sigma^{(3)} = 0, \\ & \mathbf{l} \le \mathbf{v} \le \mathbf{u}, \\ & \mathbf{p}, \mathbf{e} \ge 0, \\ & \sigma^{(j)} \ge 0, \forall j \in \{1,2,3\}, \end{aligned} \tag{22}$$

where $\mathbf{f}$ is the cell's relative importance of different proteins, and $\sigma^{(j)}$ are non-negative slack variables.

By defining $\mathbf{w} = \{\mathbf{v}, \mathbf{p}, \mathbf{e}, \sigma^{(1)}, \sigma^{(2)}, \sigma^{(3)}\}$,

$$\bar{S} = \begin{bmatrix} S & 0 & 0 & 0 & 0 & 0 \\ I & 0 & -K & I & 0 & 0 \\ 0 & -I & E & 0 & 0 & I \end{bmatrix},$$

$\bar{\mathbf{c}} = \{\mathbf{c}, \mathbf{f}\}$, $\bar{\mathbf{l}} = \{\mathbf{l}, 0, 0, 0, 0, 0\}$, $\bar{\mathbf{u}} = \{\mathbf{u}, +\infty, +\infty, +\infty + \infty, +\infty\}$, and $\bar{\mathbf{b}} = \{\mathbf{b}, 0, t, 0\}$, we can write (22) as

$$\max_{\mathbf{w}} \bar{\mathbf{c}}^T\mathbf{w} \text{ subject to } \bar{S}\mathbf{w} = \bar{\mathbf{b}},\ \bar{\mathbf{l}} \le \mathbf{w} \le \bar{\mathbf{u}}, \tag{23}$$

which is (1) with a change of variables. Therefore, BIG-BOSS can be applied directly to the problem with proteins and fluxes.

The $K$ matrix is typically not known fully. Here, we used a default value of 65 second$^{-1}$, which are in units of enzyme turnover rate and represents an average catalytic efficiency for all enzymes in *E. coli*. Values for matrix $E$ were determined from the gene-protein-reaction mappings that are included with each metabolic model in the BiGG database [25]. Without additional information we assumed that each complex requires one unit of each protein in the gene-protein mapping. For $t$, we chose a default value of 0.30, indicating that the model accounts for 30% of the cell's protein mass. For molecular weights $\mathbf{a}$, we used a default value of 77 kDa for every protein, based on the average molecular weight of proteins in *E. coli*. (Note that $\mathbf{a}$ can also be determined directly from the protein's amino acid sequence.)

### A.2 Online Resources

Code and documentation for BIG-BOSS are available at https://github.com/laurenceyang33/cellgoal. Detailed installation instructions are available there. Briefly, users will need a Fortran compiler, CMake, and a linear system solver. We tested our algorithm with both UMFPACK, which is part of SuiteSparse, and MA57 [13] from HSL [23]. For parallel computing features, users will need OpenACC and the PGI Compiler. We have tested all code using PGI Community Edition version 18.10. In the code repository, we include test suites to run BIG-BOSS on metabolic network models that users can download from the BiGG database [25].

### A.3 Details on solution time

We report details on the run time for BIG-BOSS when solving the problem shown in Fig. 2. Solutions having modest accuracy are found quickly, as is expected for ADMM (Fig. 4).

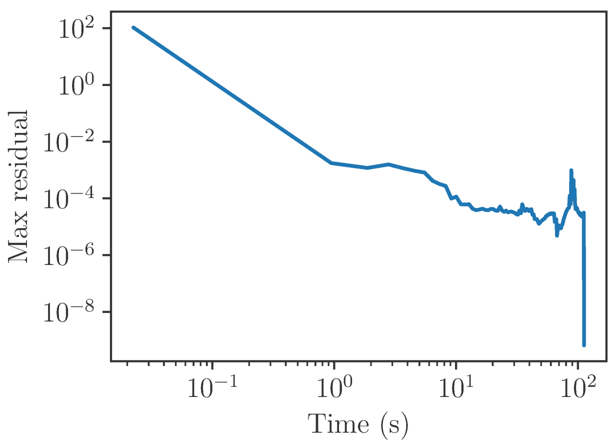

**Figure 4: Maximum of primal and dual residuals versus the wall time of running BIG-BOSS.**