Mixed-Integer Optimization for Loopless Flux Distributions in Metabolic Networks

Hannah Troppens ✉ 🆔
Zuse Institute Berlin, Germany

Mathieu Besançon ✉ 🆔
Université Grenoble Alpes, Inria, LIG, CNRS, France

St. Elmo Wilken 🆔
Institute of Quantitative and Theoretical Biology, Heinrich Heine University Düsseldorf, Germany

Sebastian Pokutta 🆔
Zuse Institute Berlin, Germany
Technische Universität Berlin, Germany

── Abstract ────

Constraint-based metabolic models can be used to investigate the intracellular physiology of microorganisms. These models couple genes to reactions, and typically seek to predict metabolite fluxes that optimize some biologically important metric. Classical techniques, like Flux Balance Analysis (FBA), formulate the metabolism of a microbe as an optimization problem where growth rate is maximized. While FBA has found widespread use, it often leads to thermodynamically infeasible solutions that contain internal cycles (loops). To address this shortcoming, Loopless-Flux Balance Analysis (ll-FBA) seeks to predict flux distributions that do not contain these loops. ll-FBA is a disjunctive program, usually reformulated as a mixed-integer program, and is challenging to solve for biological models that often contain thousands of reactions and metabolites.
In this paper, we compare various reformulations of ll-FBA and different solution approaches.

Overall, the combinatorial Benders' decomposition is the most promising of the tested approaches with which we could solve most instances. However, the model size and numerical instability pose a challenge to the combinatorial Benders' method.

1 Introduction

In systems biology, detailed annotations of a microorganism's genome can be leveraged to construct a genome-scale metabolic model (GSMM). These models connect genes to proteins that catalyze specific reactions within the metabolism of a microbe. One essential

and challenging task is predicting the *flux* channeled by these metabolic reactions, i.e., the amount of molecules consumed by the different reactions. Reaction kinetics drive the flux of individual reactions, but due to challenges in their measurement and estimation, encoding them into models remains highly impractical, leaving them under-determined. Constraint-based models have been proposed to address these issues and improve flux estimation by reformulating the model as an optimization problem with reaction fluxes as variables.

In the following, we briefly introduce key biological concepts. The chemical reactions that take place in an organism, known as *metabolism*, determine the central functions of the cell [18]. Understanding metabolism is, therefore, crucial for understanding cellular behavior. *Metabolites* are small molecules involved in biochemical reactions, and each *reaction* converts a set of *reactants* into *products*, with defined *stoichiometry* indicating the number of molecules involved. *Enzymes* are important for cell metabolism as they *catalyze* reactions, meaning they accelerate reactions without being consumed. We differentiate between *internal reactions*, which involve only internal metabolites, and *exchange reactions*, which interact with the environment. A *metabolic network* captures this structure, often modeled as a directed hypergraph, where nodes represent metabolites and hyperedges represent reactions.

By assuming that the cell operates at metabolic steady state, the network of feasible reaction fluxes can be captured in a polytope. Finally, the addition of a biologically relevant objective function, e.g. growth rate maximization, yields a linear program (LP) that can be solved efficiently. This approach, coined Flux Balance Analysis (FBA), can be summarized in the following LP:

$$
\begin{aligned}
\max_{\mathbf{v}} \quad & \mathbf{c}^{\mathsf{T}}\mathbf{v} \\
\text{s.t.} \quad & \mathbf{S}\mathbf{v} = \mathbf{0} \\
& \mathbf{l} \leq \mathbf{v} \leq \mathbf{u}.
\end{aligned}
\tag{FBA}
$$

The structure of the reaction network is captured in the stoichiometric matrix $\mathbf{S} \in \mathbb{R}^{m \times n}$, where m is the number of internal metabolites and n is the number of reactions. The fluxes $\mathbf{v} \in \mathbb{R}^n$ are bounded by $\mathbf{l}, \mathbf{u} \in \mathbb{R}^n$. The linear system $\mathbf{S}\mathbf{v} = \mathbf{0}$ ensures that the model is mass-balanced at steady state. FBA is inspired by viewing evolutionary selection pressure as a mathematical optimization problem to maximize fitness (biomass growth) [17]. Importantly, FBA enabled systems biologists to predict cellular behavior in accordance with observations in experiments [16]. However, solutions of FBA often violate thermodynamic principles, as they may yield solutions that contain internal cycles, which are biologically unrealistic. An internal loop or internal cycle is a nonzero flux vector $\boldsymbol{\ell}$ such that the internal network is at steady-state: $\mathbf{S}_{\mathcal{I}}\boldsymbol{\ell} = \mathbf{0}$ [15]. Loopless-Flux Balance Analysis (ll-FBA) is a constraint-based approach to predict flux distributions that removes internal loops [21], thus optimizing over a subset of the FBA polytope. Unlike FBA, ll-FBA is an $\mathcal{NP}$-hard disjunctive program and is challenging to solve in theory and practice for biological models that often contain thousands of reactions and metabolites. In addition to tractability challenges, ll-FBA instances create numerical issues for solvers, motivating the design of robust and stable methods.

Besides ll-FBA, other methods exist to obtain loopless or thermodynamically feasible flux distributions. For example, thermodynamic Flux Balance Analysis incorporates thermodynamic constraints directly using metabolite concentrations, improving realism but requiring additional data [7]. Parsimonous FBA [12] and CycleFreeFlux [6] are post-processing approaches that remove cycles after solving FBA, offering computational efficiency but potentially sacrificing optimality.

In this paper, we tackle the ll-FBA problem with mixed-integer optimization techniques, aiming to bridge the gap between model biological realism and tractability even on genome-scale instances. We derive disjunctive formulations of ll-FBA that notably avoid introducing

some of the artificially large bounds on continuous variables. We then design a combinatorial Benders' (CB) decomposition for the disjunctive problem, exploiting a natural separation between the flux and edge activation on one side, and thermodynamic feasibility on the other side. We evaluate the different formulations and algorithms on realistic genome-scale metabolic model examples from the systems biology literature. To demonstrate the flexibility of our framework, we also conduct experiments incorporating enzyme constraints. Computational experiments show that our CB algorithm performs much better than the alternatives and reveal choices of the algorithm's parameters that work well on the whole instance set, specifically studying the number of cuts to add per Benders' iteration and how to select these cuts. Our implementations of the various methods for ll-FBA are open-source and rely on open-source solvers and libraries to be directly utilized by the systems biology community.

Notation

A vector $\mathbf{v} = (v_1, ..., v_n) \in \mathbb{R}^n$ is identified as a column vector and printed in bold. With $\mathbf{v} \leq \mathbf{w}$ we denote element-wise inequality. The 1-vector is written as $\mathbf{1} := (1, 1, ..., 1) \in \mathbb{R}^n$, the 0-vector as $\mathbf{0} := (0, 0, ..., 0) \in \mathbb{R}^n$. The *support* of $\mathbf{v}$ is the set of indices i with $v_i \neq 0$ and is denoted by $\mathrm{supp}(\mathbf{v})$. With $\mathrm{sign}(\mathbf{v})$ the element-wise sign function is applied to $\mathbf{v}$.

The zero matrix is denoted by $\mathbf{0}_{m,n}$ with m rows and n columns. With $\mathrm{diag}(\mathbf{v})$ we denote the quadratic matrix with $\mathbf{v}$ on the diagonal and 0 for all other entries. A *basis B* of a vector space V is a set of vectors $(\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_n)$ that are linearly independent and every $\mathbf{v} \in V$ can be written as a linear combination of vectors in B. The *nullspace* of a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ is defined as $\mathrm{null}(\mathbf{A}) := \{\mathbf{x} \in \mathbb{R}^n : \mathbf{A}\mathbf{x} = \mathbf{0}\}$.

2 Loopless-FBA

Loopless Flux Balance Analysis (ll-FBA) is a constraint-based approach that predicts flux distributions in cells that do not contain internal loops by incorporating thermodynamic information as an extension of FBA:

$$
\begin{aligned}
\max_{\mathbf{v}, \mathbf{\Delta^\mu}, \mathbf{\mu}} \quad & \mathbf{c}^\mathsf{T}\mathbf{v} \\
\text{s.t.} \quad & \mathbf{S}\mathbf{v} = \mathbf{0} \\
& \mathbf{l} \leq \mathbf{v} \leq \mathbf{u} \\
& \Delta^\mu_i v_i < 0 \lor v_i = 0 \quad \forall i \in \mathcal{I} \\
& \mathbf{\Delta^\mu} = \mathbf{S}_\mathcal{I}^\mathsf{T}\mathbf{\mu},
\end{aligned}
\tag{1}
$$

where $\mathbf{v} \in \mathbb{R}^n$, $\mathbf{\mu} \in \mathbb{R}^m$ and $\mathbf{\Delta^\mu} \in \mathbb{R}^{|\mathcal{I}|}$. The matrix $\mathbf{S}_\mathcal{I}$ is the submatrix of $\mathbf{S}$ that contains the columns of internal reactions only. $\mathbf{\mu}$ denotes Gibbs free energy, which predicts the direction of a reaction at constant pressure and temperature, and $\mathbf{\Delta^\mu}$ can be interpreted as the change in Gibbs free energy. A reaction i is thermodynamically feasible only if the flux and the corresponding change in Gibbs free energy $\mathbf{\Delta^\mu_i}$ have opposite signs. Forward flux requires $\mathbf{\Delta^\mu_i} < 0$, while backward flux requires $\mathbf{\Delta^\mu} > 0$ [2]. However, in ll-FBA only the sign($\mathbf{\Delta^\mu}$) corresponds to the actual Gibbs free energy change. $\mathbf{\Delta^\mu} = \mathbf{S}_\mathcal{I}^\mathsf{T}\mathbf{\mu}$ ensures that the reaction energies around a circuit sum up to zero, which is known as *Kirchhoff's second law*. Note that Kirchhoff's second law can also be ensured by the equation $\mathbf{B}^\mathsf{T}\mathbf{\Delta^\mu} = \mathbf{0}$, where $\mathbf{B} \in \mathbb{R}^{|\mathcal{I}| \times k}$ is a matrix with columns formed by the set of vectors $\{\mathbf{b}_i\}_{i=1}^k$ that form a basis for $\mathrm{null}(\mathbf{S}_\mathcal{I})$ [21]. Since $\mathbf{v}$ and $\mathbf{\Delta^\mu}$ have different lengths, we assume that the first entries of $\mathbf{v}$ correspond to the internal reactions, so that each index i refers to the pair (v_i, Δ^μ_i).

Problem (1) is $\mathcal{NP}$-hard and much more complicated to solve than FBA [19]: it is challenging due to the disjunction, the strict equality and the product of decision variables $\boldsymbol{\Delta^\mu}$ and $\mathbf{v}$. To simplify the model, the disjunction is rewritten by introducing the constant ϵ, and we arrive at the ll-FBA model we will use throughout the paper:

$$
\begin{aligned}
\max_{\mathbf{v},\boldsymbol{\Delta^\mu},\boldsymbol{\mu}} \quad & \mathbf{c^\mathsf{T} v} \\
\text{s.t.} \quad & \mathbf{Sv = 0} \\
& \mathbf{l \le v \le u} \\
& \left. \begin{aligned} &\left((v_i \ge 0) \wedge (\Delta_i^\mu \le -\epsilon)\right) \vee \\ &\left((v_i \le 0) \wedge (\Delta_i^\mu \ge \epsilon)\right) \end{aligned} \right\} \quad \forall i \in \mathcal{I} \\
& \boldsymbol{\Delta^\mu} = \mathbf{S}_{\mathcal{I}}^\mathsf{T} \boldsymbol{\mu}.
\end{aligned}
\tag{ll-FBA}
$$

To avoid the degenerate solution where all $\Delta_i^\mu = 0$, it is standard practice to exclude the interval $[-\epsilon, \epsilon]$ [21]. Since only the sign of $\boldsymbol{\Delta^\mu}$ is thermodynamically relevant, this scaling does not alter the biological interpretation. A solution to ll-FBA does not contain internal cycles, but ll-FBA is computationally harder as we need to solve a disjunctive program, which is usually reformulated as a mixed-integer program (MIP)[1].

As an example, let us consider the metabolic network in Figure 1. We have three internal metabolites A, B, C, two irreversible exchange reactions r_1, r_5 and three reversible internal reactions r_2, r_3, r_4. The stoichiometric matrix is:

$$
\mathbf{S} = \begin{bmatrix} 1 & -1 & 0 & -1 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 & -1 \end{bmatrix}
$$

and we assume the following bounds on the fluxes:

$$
\mathbf{l} = (0, -30, -30, -30, 0) \quad \mathbf{u} = (10, 30, 30, 30, 10)
$$
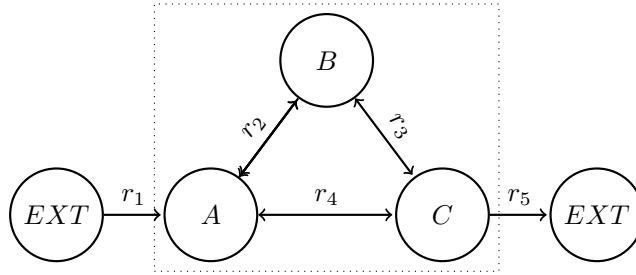
Figure 1 Simple model with internal loop.

If we maximize the flux through the internal reactions, that is $\mathbf{c} = (0, 1, 1, 1, 0)$, an optimal solution is $\mathbf{v}^* = (10, 30, 30, -20, 10)$. The solution contains an internal loop, as there is a flux of 20 going through each of the internal reactions, as seen in Figure 2.

---

[1] While ll-FBA is usually introduced in the literature directly via indicator constraints and their MIP reformulation using big-M constraints [21], we refer to this disjunctive form as ll-FBA throughout the paper and distinguish it from its reformulations used in practice.
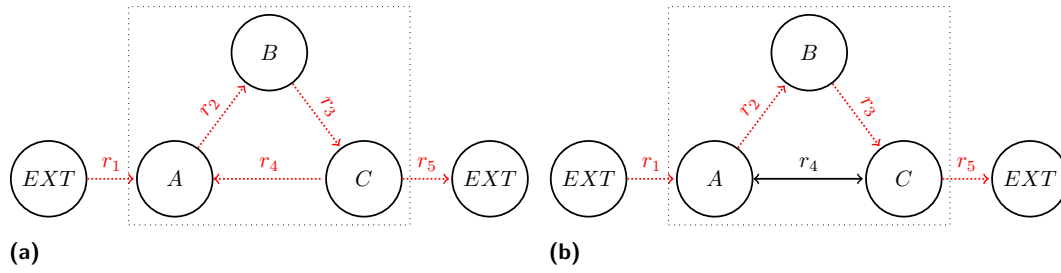
Figure 2 Used reactions in (a) the FBA solution which contains an internal cycle, and (b) the ll-FBA solution.

Enzyme Information

In classical FBA simulations, the predicted growth rate of the model is limited by the substrate uptake rate. This flux constraint on the uptake reaction is typically empirically measured, and imposed on the model. Recent theory has shown that the protein density in microbes is relatively constant, and thus a proteome limitation is a more plausible constraint to impose on the model [22]. Enzyme constrained metabolic models extend genome scale metabolic models by incorporating simplified enzyme kinetics. In short, reaction flux is assumed to be proportional to a kinetic constant, and the concentration of enzyme available to catalyze a specific reaction. The optimization goal is then to distribute enzyme resources, subject to a total enzyme capacity bound, to maximize growth. This extension can be used to model complex metabolic phenomena, like overflow metabolism. Models incorporating this extra information are called enzyme constrained flux balance analysis models [20].

Mathematically, if reaction j is catalyzed by enzyme E_i, the reaction rate v_j (in $\frac{\text{mmol}}{\text{gDW} \times \text{h}}$) depends on the enzyme usage e_i (in $\frac{\text{mmol}}{\text{gDW}}$) and the turnover number (kinetic parameter) k_{cat}^{ij} (in $\frac{1}{\text{h}}$) as $k_{cat}^{ij} e_i = v_j$, which is known as enzyme mass balance. To account for enzyme mass balance in a GSMM, the stoichiometric matrix $\mathbf{S}$ is extended to form $\mathbf{S}^{ENZ}$. This extension adds a row for each enzyme E_i and a column for the corresponding enzyme usage e_i, with p enzymes. The lower left submatrix contains the enzyme information on the diagonal, that is $-1/k_{cat}^{ij}$. The resulting matrix is of the form:

$$\mathbf{S}^{ENZ} = \left[\begin{array}{ccc|ccc} s_{1,1} & ... & s_{1,n} & 0 & ... & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ s_{m,1} & ... & s_{m,n} & 0 & ... & 0 \\ \hline -1/k_{cat}^{11} & ... & 0 & 1 & ... & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & ... & -1/k_{cat}^{pn} & 0 & ... & 1 \end{array} \right] = \left[\begin{array}{cc} \mathbf{S} & \mathbf{0}_{m,p} \\ \text{diag}(-1/k_{cat}^{ij}) & \mathbf{I}_p \end{array} \right].$$

In addition to the reaction rates $\mathbf{v}$, we are interested in the enzyme usage $\mathbf{e}$. We obtain the following optimization problem:

$$\begin{aligned} \max_{\mathbf{v},\mathbf{e}} \quad & \mathbf{c}^\mathsf{T} \mathbf{v} \\ \text{s.t.} \quad & \mathbf{S}^{ENZ}(\mathbf{v}, \mathbf{e}) = \mathbf{0} \\ & \mathbf{l} \le \mathbf{v} \le \mathbf{u} \\ & \mathbf{0} \le \mathbf{e} \le [\mathbf{E}], \end{aligned} \quad\quad\quad \text{(ENZ)}$$

where $[E_i]$ is the intracellular concentration of enzyme E_i.

3   Mathematical Models and Solving Methods

In this section, we present various reformulations of the ll-FBA problem in using indicator constraints and big-M constraints to model the disjunctions. Our goal is to compare the performance of

1. solving these reformulations directly,

2. decomposing the problem,

3. solving the convex hull formulation.

3.1   Big-M and Indicator Reformulations

We can reformulate ll-FBA to no longer write the disjunctions explicitly. The mixed-integer programming formulation of flux balance analysis without unbounded internal cycles, using indicator constraints, is provided by [21]:

$$
\begin{aligned}
\max_{\mathbf{v},a,\mathbf{\Delta^\mu},\mathbf{\mu}} \quad & \mathbf{c^\mathsf{T} v} \\
\text{s.t.} \quad & \mathbf{Sv = 0} \\
& \mathbf{l \le v \le u} \\
& a_i = 1 \quad \implies \quad v_i \ge 0 \qquad \forall i \in \mathcal{I} \\
& a_i = 1 \quad \implies \quad \Delta_i^\mu \le -\epsilon \quad \forall i \in \mathcal{I} \\
& a_i = 0 \quad \implies \quad v_i \le 0 \qquad \forall i \in \mathcal{I} \\
& a_i = 0 \quad \implies \quad \Delta_i^\mu \ge \epsilon \qquad \forall i \in \mathcal{I} \\
& \mathbf{\Delta^\mu = S_\mathcal{I}^\mathsf{T} \mu} \\
& a_i \in \{0,1\} \quad \forall i \in \mathcal{I},
\end{aligned}
\qquad \text{(ll-FBA (indicator))}
$$

where each binary variable a_i indicates which disjunct holds. The mixed-integer program of flux balance analysis without unbounded internal cycles using big-M constraints, introducing the constant M, takes the form [21]:

$$
\begin{aligned}
\max_{\mathbf{v},a,\mathbf{\Delta^\mu},\mathbf{\mu}} \quad & \mathbf{c^\mathsf{T} v} \\
\text{s.t.} \quad & \mathbf{Sv = 0} \\
& \mathbf{l \le v \le u} \\
& -Ma_i + \epsilon(1 - a_i) \le \Delta_i^\mu \le -\epsilon a_i + M(1 - a_i) \quad \forall i \in \mathcal{I} \\
& -M(1 - a_i) \le v_i \le Ma_i \quad \forall i \in \mathcal{I} \\
& \mathbf{\Delta^\mu = S_\mathcal{I}^\mathsf{T} \mu} \\
& a_i \in \{0,1\} \quad \forall i \in \mathcal{I}.
\end{aligned}
\qquad \text{(ll-FBA (big-M))}
$$

The big-M formulation ensures the opposite sign of v_i and Δ_i^μ if $v_i \ne 0$. If the flux through reaction i is a forward flux, that is $v_i > 0$ and $a_i = 1$, it holds that $-M \le \Delta_i^\mu \le -\epsilon$ and analogously for a backward flux. The value of ϵ affects the scale of $\mathbf{\Delta^\mu}$ (and $\mathbf{\mu}$). The value of ϵ should be bigger than the solver precision to differentiate it from 0. We set ϵ to 1 to match the formulation of [21]. The big-M constant is the maximal absolute value of the flux bounds $\mathbf{l}, \mathbf{u}$.

3.2 Decomposing ll-FBA

Instead of solving a mixed-integer reformulation of ll-FBA directly, the problem is decomposed into a master problem and a subproblem. The solving procedure involves first solving the master problem and then verifying the feasibility of the relaxed solution. The master problem is a relaxed version of ll-FBA (indicator), where the loopless constraints are ignored, but the indicator variables $\boldsymbol{a}$ are assigned:

$$
\begin{aligned}
\max_{\mathbf{v},\boldsymbol{a}} \quad & \mathbf{c^\mathsf{T} v} \\
\text{s.t.} \quad & \mathbf{Sv} = \mathbf{0} \\
& \mathbf{l} \le \mathbf{v} \le \mathbf{u} \\
& a_i = 1 \quad \implies \quad v_i \ge 0 \quad \forall i \in \mathcal{I} \\
& a_i = 0 \quad \implies \quad v_i \le 0 \quad \forall i \in \mathcal{I}.
\end{aligned}
\tag{MP}
$$

The indicator constraints can be linearized by using big-M constraints. The subproblem is the following parameterised program:

$$
\begin{aligned}
\max_{\boldsymbol{\mu},\boldsymbol{\Delta^\mu}} \quad & 0 \\
\text{s.t.} \quad & a_i^{MP} = 1 \quad \implies \quad \Delta_i^\mu \le -\epsilon \quad \forall i \in \mathcal{I} \\
& a_i^{MP} = 0 \quad \implies \quad \Delta_i^\mu \ge \epsilon \quad \forall i \in \mathcal{I} \\
& \boldsymbol{\Delta^\mu} = \mathbf{S}_\mathcal{I}^\mathsf{T} \boldsymbol{\mu},
\end{aligned}
\tag{SP}
$$

where $\boldsymbol{a}^{\mathrm{MP}}$ are the values of a solution to the master problem. If a relaxed solution is not a feasible solution to the ll-FBA problem, a cut is added to the relaxed problem that cuts off the assignment of binary variables. This procedure is repeated until a relaxed solution is a feasible solution to ll-FBA.

First, we introduce no-good cuts. If a relaxed solution $\boldsymbol{a}^{\mathrm{MP}}$ is thermodynamically infeasible, the following constraint is added to the master problem:

$$
\sum_{i \in \mathcal{I}:\, a_i^{MP}=0} a_i + \sum_{i \in \mathcal{I}:\, a_i^{MP}=1} (1 - a_i) \ge 1,
\tag{2}
$$

where the entire infeasible relaxed solution is excluded from the solution space of the master problem.

Combinatorial Benders' Cuts

When deriving combinatorial Benders' cuts for ll-FBA, we aim to identify the variables that lead to thermodynamic infeasibility and potentially derive stronger cuts. If the subproblem is infeasible, we compute a corresponding minimal infeasible subsystem (MIS) $\mathcal{C}$. The minimal infeasible subsystem contains a minimal number of constraint indices that make the subproblem infeasible. The following combinatorial Benders' (CB) cut is added to the master problem if the subproblem is infeasible:

$$
\sum_{i \in \mathcal{C}:\, a_i^{MP}=0} a_i + \sum_{i \in \mathcal{C}:\, a_i^{MP}=1} (1 - a_i) \ge 1.
\tag{3}
$$

Note that if $\mathcal{C}$ contains all internal reactions $\mathcal{I}$, the CB cut corresponds to a no-good cut (Equation (2)).

■ **Algorithm 1** solving ll-FBA with the combinatorial Benders' approach.

Require: stoichiometric matrix $\mathbf{S}$, lower bound on fluxes $\mathbf{l}$, upper bound on fluxes $\mathbf{u}$, allowed
    number of cuts per iteration m, indices of internal reactions $\mathcal{I}$
 1: $\mathrm{MP} \leftarrow \mathsf{build\_master\_problem}(\mathbf{S}, \mathbf{l}, \mathbf{u}, \mathcal{I})$
 2: $\mathbf{v}, \mathbf{a} \leftarrow \mathsf{optimize}(\mathrm{MP})$
 3: $\mathcal{C} \leftarrow \mathsf{compute\_mis}(\mathbf{S}_{\mathcal{I}}, \boldsymbol{a}, m)$ ▷ set of minimal infeasible subsystems
 4: **while** $\mathcal{C} \neq \emptyset$ **do**
 5:     $\mathsf{add\_cut}(\mathrm{MP}, \boldsymbol{a}, \mathcal{C})$ ▷ combinatorial Benders' cut
 6:     $\mathbf{v}, \boldsymbol{a} \leftarrow \mathsf{optimize}(\mathrm{MP})$
 7:     $\mathcal{C} \leftarrow \mathsf{compute\_mis}(\mathbf{S}_{\mathcal{I}}, \boldsymbol{a}, m)$
 8: **end while**
 9: $\mathrm{SP} \leftarrow \mathsf{build\_sub\_problem}(\mathbf{S}_{\mathcal{I}}, \mathcal{I}, \boldsymbol{a})$
10: $\boldsymbol{\Delta^{\mu}}, \boldsymbol{\mu} \leftarrow \mathsf{optimize}(\mathrm{SP})$
11: **return** $(\mathbf{v}, \boldsymbol{a}, \boldsymbol{\Delta^{\mu}}, \boldsymbol{\mu})$ ▷ loopless flux distribution

---

The pseudocode for the combinatorial Benders' cut procedure is shown in Algorithm 1.
The set of minimal infeasible subsystems is computed in $\mathsf{compute\_mis}$, which identifies at
most m subsystems. The derivation of an MIS is based on the dual problem of the infeasible
subproblem, and is explained in detail in [5].
The primal problem in inequality form is:

$$\max_{\mathbf{x}} \quad 0$$
$$\text{s.t.} \quad \tilde{\mathbf{A}}\boldsymbol{\mu} \leq \tilde{\mathbf{b}},$$

where

$$\tilde{\mathbf{A}} = \begin{bmatrix} [\mathbf{S}_{\mathcal{I}}]_{*,i} & \forall i \in \mathcal{I} : a_i^{MP} = 1 \\ -[\mathbf{S}_{\mathcal{I}}]_{*,i} & \forall i \in \mathcal{I} : a_i^{MP} = 0 \end{bmatrix} \quad \tilde{\mathbf{b}} = \begin{bmatrix} -\epsilon^{|\mathcal{C}|} \end{bmatrix}.$$

The linear program to find minimal infeasible subsystems is:

$$\max_{\boldsymbol{\lambda}} \quad \sum_i w_i \lambda_i$$
$$\text{s.t.} \quad \tilde{\mathbf{A}}^{\intercal}\boldsymbol{\lambda} = \mathbf{0} \tag{4}$$
$$\boldsymbol{\lambda} \geq \mathbf{0}$$
$$\tilde{\mathbf{b}}^{\intercal}\boldsymbol{\lambda} = \mathbf{1},$$

where w_i is the weight corresponding to the dual variable λ_i. The support of each solution at
a vertex of the feasible region of (4) defines a minimal infeasible subsystem. By modifying
$\mathbf{w}$, we potentially derive several minimal infeasible subsystems to one infeasible solution $\boldsymbol{a}$.
Each solution corresponds to a minimal infeasible subsystem. The nonzero elements in $\boldsymbol{\lambda}$
correspond to a MIS with the set of reaction indices $\mathcal{C}$. We choose how many cuts k can be
added per iteration relative to the model size. For any $i \in 1, ..., k$, we set the i-th coefficient
in the objective function to zero and set all other coefficients to 1. At the end of the MIS
search, we filter the minimal infeasible subsystems found to have unique subsystems within
each iteration. To account for the model size, we set k to a percentage of the number of
reactions of the model. For instance, MIS 0.5% means that at most 0.5% minimal infeasible
subsystems are blocked per iteration. Instead of using all minimal infeasible subsystems
found, we can filter the subsystems and block only a subset in the master problem (see
Section B.2).

3.3 Convex Hull Reformulation

The feasible region of the convex hull reformulation corresponds to the convex hull of the disjunctive constraints and thus represents the tightest possible convex relaxation of the original disjunctive program [1]. However, this formulation introduces a significantly larger number of decision variables and constraints compared to the big-M approach. In particular, the number of additional constraints grows exponentially with the number of disjunctions. Each disjunction is associated with two binary variables:

$$
\begin{bmatrix} y_i \\ v_i \geq 0 \\ \Delta_i^\mu \leq -\epsilon \end{bmatrix} \vee \begin{bmatrix} y_{i+|\mathcal{I}|} \\ v_i \leq 0 \\ \Delta_i^\mu \geq \epsilon \end{bmatrix} \quad \forall i \in \mathcal{I}. \tag{5}
$$

To formulate the disjunctive constraint with a mixed-integer program, the constraint below is added to ensure the disjunction holds:

$$
y_i + y_{i+|\mathcal{I}|} = 1 \quad \forall i \in \mathcal{I}.
$$

Two continuous variables are added for each decision variable in a disjunction. The variables corresponding to v_i are denoted by v_{i_1} and v_{i_2} and the variables corresponding to Δ_i^μ are $\Delta_{i_1}^\mu$ and $\Delta_{i_2}^\mu$. The hull reformulation of the disjunction is:

$$
\begin{aligned}
v_i &= v_{i_1} + v_{i_2} \\
\Delta_i^\mu &= \Delta_{i_1}^\mu + \Delta_{i_2}^\mu \\
-v_{i_1} &\leq 0 \\
v_{i_2} &\leq 0 \\
\Delta_{i_1}^\mu &\leq -y_i \\
-\Delta_{i_2}^\mu &\leq -y_{i+|\mathcal{I}|}.
\end{aligned}
$$

4 Computational Experiments

In this section, we present the results of the computational experiments for the ll-FBA problem. We compare solving ll-FBA (big-M) and ll-FBA (indicator) with `SCIP` directly, to solving the convex hull reformulation and to decomposing the problem with a combinatorial Benders' approach.

4.1 Experimental Setup

We implemented the different solving methods[2] in Julia 1.9.0. We use `JuMP` [14] and `MathOptInterface` [11] to build the mathematical models. The `DisjunctiveProgramming` package is used to get the hull formulation. The MIP solver used in the experiments is `SCIP` 9.0 [3, 4]. The LP solver used is `HiGHS` 1.8.1 [8]. We use `COBREXA` [10] to load the biological model data. The experiments were carried out on a 64-core compute node equipped with an Intel Xeon Gold 6338 2GHz CPU and 512GB RAM. The CPU memory was limited to 3000MB, and the time limit is 1800 seconds unless specified otherwise.

---

[2] available at
https://github.com/hannahtro/Loopless_Fluxes_with_Mixed_Integer_Optimization

Metabolic Model Instances

We test the entire set of metabolic networks of the *biochemical, genetic, and genomic* (BiGG) database[3] [9]. The models cover different model sizes, ranging from the smallest model e_coli_core with 72 metabolites and 95 reactions to the largest model Recon3D with 5835 metabolites and 10600 reactions.

We use a subset of metabolic networks of yeast models made available by [13]. The models selected are similar in size to the largest BiGG models used here (see Table 3).

When adding enzyme constraints to the model, the flux rate depends on the enzymes and the flux bounds $\mathbf{l}, \mathbf{u}$ are only required to restrict the direction of a reaction. Due to the limited availability of curated enzyme data in the BiGG database, we use randomly generated enzyme parameters for the BiGG models. Our aim is not to ensure biological realism in these cases, but rather to demonstrate the flexibility of our framework in handling enzyme constraints. We use `COBREXA` to build the enzyme models, and the enzyme data (i.e. the turnover numbers and protein molar masses) is generated randomly as specified below. The turnover numbers differentiate for the forward and backward direction of a reaction and therefore we split each reversible reaction into one forward and one backward reaction. At least one enzyme is mapped to each reaction including the turnover numbers for the reaction in the forward and the turnover number for the backward direction are defined. The turnover numbers are taken independently from a standard normal distribution. The protein concentrations have to be in the interval $[0, 1000]$. An enzyme is made up of one or more proteins. Each protein is randomly associated with either mass group A or B, and the product mass of each group is bounded by 0.5 $\frac{\text{mmol}}{\text{gDW}}$. For each protein we assign a product molar mass randomly from a uniform distribution.

4.2   Results

We begin by comparing the performance of three approaches for solving ll-FBA (big-M) for the BiGG instances: solving it directly with `SCIP`, solving its convex hull reformulation, and applying combinatorial Benders' cuts. Figure 3 shows that the convex hull reformulation solves the fewest instances. Even though the feasible region of the hull reformulation is the convex hull, the reformulation introduces more constraints and decision variables, resulting in increased running times. In contrast, when comparing the MIP reformulations to solving ll-FBA (big-M) with combinatorial Benders' cuts, we observe that the latter achieves superior performance. Specifically, it solves the majority of instances and does so significantly faster.

We next present detailed computational results for decomposing ll-FBA with a particular focus on combinatorial Benders's cuts. We compare the performance of solving ll-FBA (big-M) and ll-FBA (indicator) with `SCIP` directly on the BiGG instances to two decomposition approaches: solving the decomposition of ll-FBA (big-M) with no-good cuts and solving it with the combinatorial Benders' cuts, all within a time limit of 1800 seconds. For the combinatorial Benders' method, we experiment with three different formulations of the master problem. In each iteration of Algorithm 1, we add a single cut. The formulations we consider are:
1. the indicator formulation,
2. the big-M formulation,
3. a combined approach using both the indicator and big-M formulations (i.e. adding both groups of constraints).

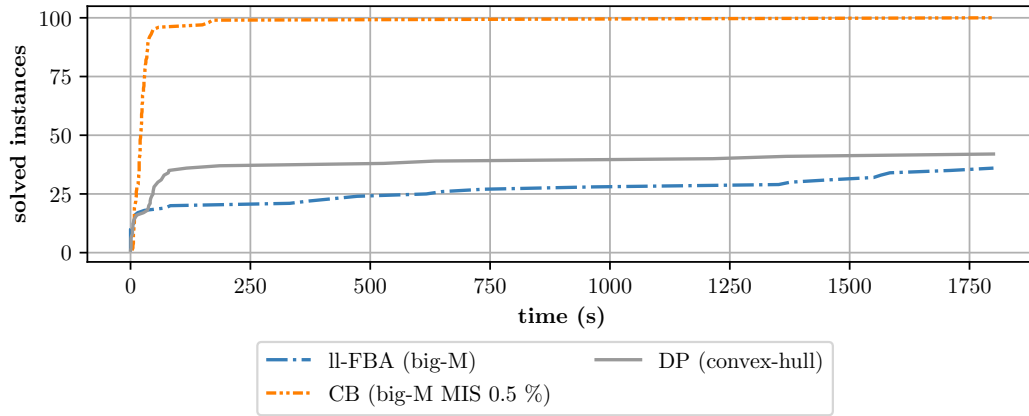---

[3] last accessed on Sep 25, 2024

Figure 3 Comparing the number of BiGG instances optimally solved by the big-M and convex hull formulations, by solving ll-FBA (big-M) directly, and by solving the problem with CB (CB (big-M MIS 0.5 %)). Note that some instances remain unsolved by the combinatorial Benders' approach, as shown in Table 1 and Table 2.

Figure 4 shows that the combinatorial Benders' method outperforms the other approaches, solving most instances within the time limit. The fastest combinatorial Benders' method is the big-M formulation, followed by the combined indicator and big-M formulation, and last the indicator formulation. Although the running time varies among the different formulations, the number of iterations remains similar across methods, suggesting that the running time is primarily determined by the time spent on solving the master problem. The combinatorial Benders' method with indicator constraints solves the fewest instances among the tested setups, yet we continue our experiments with it, because it solves a different set of instances to optimality compared to the method with big-M constraints.

In contrast, directly solving any of the ll-FBA variants results in significantly fewer optimally solved instances within the time limit. The poorest performance is observed with the no-good cuts approach, which solves only 8% of the instances. We see that combinatorial Benders' cuts are much stronger than no-good cuts. This poor performance was expected since no-good cuts only remove exactly one solution per iteration, in contrast to Benders' cuts which block a cycle and thus produces a constraint that removes a large portion of the binary hypercube.

Table 1 shows that the combinatorial Benders' method with big-M constraints solves most instances (93 %) while needing the shortest average solving time (58 seconds). For the combinatorial Benders' decomposition of ll-FBA (big-M), the instances `iAM_Pc455`, `iAM_Pf480` and `iCN718` error during the solution process, even though solving ll-FBA (big-M) returns an optimal solution. If we solve `iAM_Pc455` and `iAM_Pf480` with a tighter tolerance, we find an optimal solution within the time limit. The combinatorial Benders' method with indicator constraints also solves 93% of the instances to optimality, but the average solving time is 227 seconds.

On the other hand, directly solving any of the ll-FBA variants leads to significantly fewer optimally solved instances within the time limit. Specifically solving ll-FBA (indicator) directly leads to 17% optimally solved instances, while solving ll-FBA (big-M) directly leads to 33% optimally solved instances. Instances `iSB619` and `iCN718` are said to be infeasible when solving ll-FBA (indicator) directly and are therefore excluded from further experiments with indicator constraints.
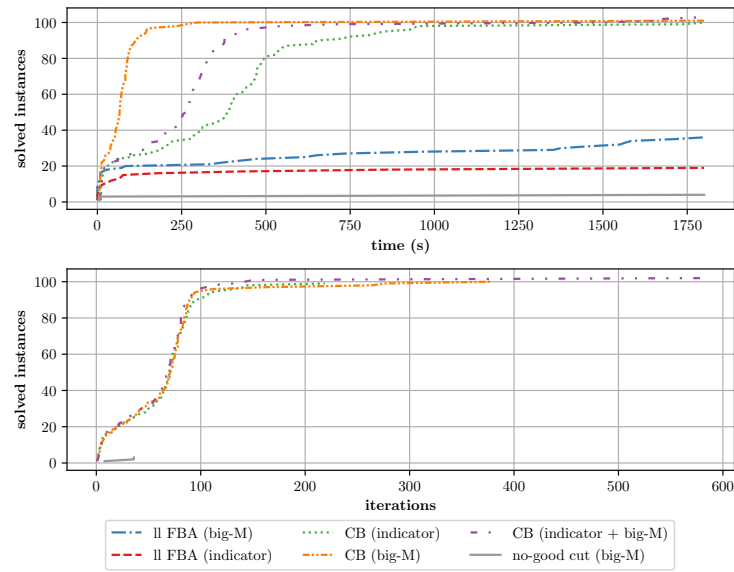
■ Figure 4 Performance of CB master problem variants to directly solving ll-FBA.
Comparing the number of optimally solved BiGG instances of ll-FBA (big-M), ll-FBA (indicator) directly, solving the decomposition with no-good cuts and solving the combinatorial Benders' decomposition of ll-FBA within the time limit of 1800 seconds. We experiment with different formulations of the master problem: big-M reformulation, indicator constraints, and indicator and big-M constraints. Per iteration of the combinatorial Benders' approach, one minimal infeasible subsystem is computed.

However, in instances where solving ll-FBA (big-M) takes less than 10 seconds, it achieves an average solving time of just one second, outperforming all other methods. This shows there is a small overhead to designing dedicated algorithms compared to passing a single model to a MIP solver.

■ Table 1 Comparing the number of optimally solved BiGG instances and the average running time of solving ll-FBA (indicator) directly, of solving ll-FBA (big-M) directly and solving their combinatorial Benders' decomposition within a time limit of 1800 seconds. Per iteration of the combinatorial Benders' approach, one minimal infeasible subsystem is computed. The BiGG instances are divided depending on the shortest running time required to solve ll-FBA (big-M) with any of the listed methods. The time column shows the average time in seconds using the geometric mean.

solved in (s)	# inst.	ll-FBA (big-M)		ll-FBA (indicator)		CB (big-M)		CB (indicator)	
		% solved	time (s)	% solved	time (s)	% solved	time (s)	% solved	time (s)
0-10	22	91	**8**	64	72	**95**	10	**95**	19
10-60	23	43	1139	4	1442	**100**	**42**	**100**	201
60-100	44	7	1671	7	1531	**100**	**80**	95	505
100-600	12	17	1542	0	1800	**100**	**137**	92	639
600-1800	6	0	1800	0	1800	0	1800	**33**	**996**
all	107	33	517	17	829	**93**	**58**	93	227

So far, we added one combinatorial Benders' cut per iteration of Algorithm 1. In Figure 5, we compare the performance of directly solving ll-FBA to the combinatorial Benders' method for both indicator and big-M constraints, experimenting with the number of cuts added per iteration. In the following, all potential cuts found per iteration are added. We investigate different cut selection strategies in B.2.

As expected, increasing the number of cuts per iteration leads to reduced running times. Any combinatorial Benders' setup using multiple cuts per iteration outperforms the single-cut approach. However, setups using big-M constraints and adding MIS 2% or more cuts per iteration perform significantly worse than combinatorial Benders' with one cut per iteration, with fewer instances being solved within the time limit. On the contrary, when using indicator constraints, the CB algorithm maintains its capacity to solve most instances across all choices of the number of cuts per round.
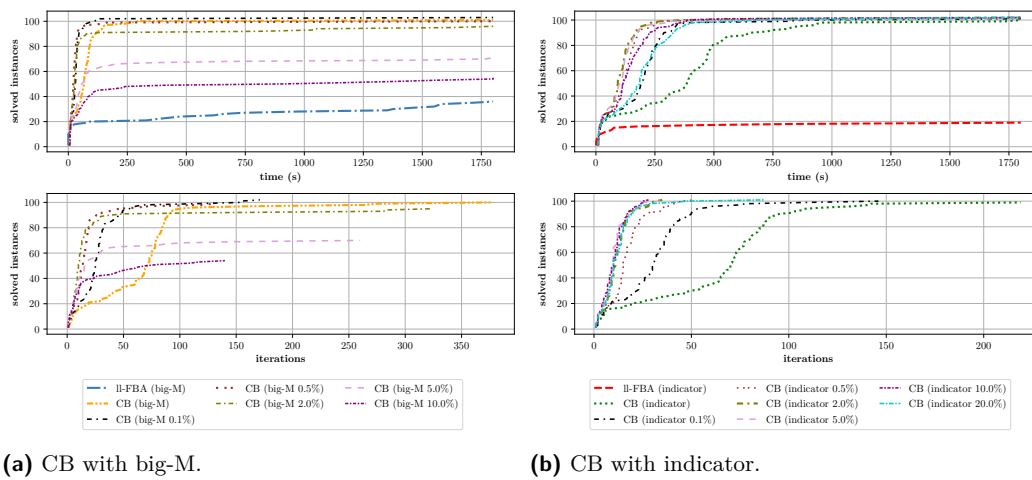

(a) CB with big-M.   **(b)** CB with indicator.

Figure 5 Performance of CB with multiple cuts, comparing the number of optimally solved BiGG instances of solving ll-FBA (indicator/big-M) directly and running CB. We experiment with the number of cuts added per iteration of CB depending on the instance size. MIS 0.5% means that at most k cuts are added per iteration, where k is 0.5% of the number of reactions of the model.

Table 2 highlights that the combinatorial Benders' decomposition ll-FBA (big-M) achieves the best performance, solving most instances in the shortest average time. Specifically, blocking up to 0.1% of the number of reactions in the per iteration allows to solve 95% of the instances, while needing only around 28 seconds in average for the solving process when blocking up to 0.5% of the number of reactions. However, adding too many cuts per iteration can negatively impact the overall performance, leading to more iterations and longer running times. For instance, blocking up to 5% of the number of reactions per iteration results in an average running time of 130 seconds, which is higher than the performance of solving it with indicator constraints (86 seconds) and higher than adding just one cut per iteration (58 seconds). Additionally, the instance `iAM_Pv561` is said to be infeasible, even though we find a solution with other combinatorial Benders' setups. If we tighten the tolerance of the solving process, the instance is solved to optimality showing that too many cuts can increase numerical instability.

After experimenting with various formulations of the master problem and different numbers of cuts per iteration in Algorithm 1, we test both solving strategies on yeast models, which are larger than most models of the BiGG database. Neither solving strategy performs well on the yeast instances: only one strategy is able to solve a single instance out of the 11 instances tested.

■ **Table 2** Comparing the number of optimally solved instances and average running time of solving ll-FBA (indicator) and ll-FBA (big-M) with the combinatorial Benders' method on the BiGG instances within a time limit of 1800 seconds. We experiment with the number of cuts added per iteration of the combinatorial Benders' approach depending on the instance size. MIS 0.5% means that at most k cuts are added per iteration, where k is 0.5% of the number of reactions of the model. The time column shows the geometric mean. The last row shows the results of the CB methods adding one cut per iteration.

MIS %	CB (indicator)		CB (big-M)	
	% solved	time (s)	% solved	time (s)
0.1	93	137.74	**95**	**30.91**
0.5	**94**	96.11	93	**27.55**
2.0	**94**	84.06	90	**38.63**
5.0	**94**	**86.0**	67	129.19
10.0	**94**	**101.57**	50	275.82
CB	**93**	227.12	**93**	**57.75**

Finally, we experiment with solving enzyme models using the combinatorial Benders' approach. We compare directly solving ll-FBA (big-M) to solving it with the combinatorial Benders' decomposition, as the big-M formulation showed the best performance on previous experiments. We compare blocking one cycle per iteration to blocking k cycles per iteration, where k is 0.5% of the total number of reactions in the model. The results are presented in Figure 6. The results indicate that both variants of the combinatorial Benders' approach perform similarly and outperform solving ll-FBA (big-M) directly. Both combinatorial Benders' variants require only a few iterations for most instances. As the reaction rate is limited by enzyme abundance, the relaxed solution already contains fewer loops, requiring fewer cuts to obtain an optimal solution.

However, the combinatorial Benders' approach is less stable for enzyme models. While around 70% of instances are solved to optimality, 18% of instances have their solution process terminated due to numerical errors. Indeed, the enzyme models are numerically challenging, most errors arise from the master problem at the first iteration not resulting in the same objective value as FBA within a tolerance of 10^{-3}. In our experiments, we are able to solve most enzyme instances within a few seconds. Since the enzyme data is randomly generated, we do not make any claims on the general performance on enzyme models. However, the results show the flexibility of the combinatorial Benders' approach, which can easily be extended with additional constraints. They also hint that enzyme constraints regularize FBA models and remove some thermodynamically infeasible solutions from the original polytope.

5 Conclusion

In this paper, we compared the performance of different formulations of ll-FBA. We compared solving different reformulations of ll-FBA with the MIP solver `SCIP` directly including the big-M reformulation and the convex hull formulation. We then experimented with different solution methods in order to solve more instances than directly solving ll-FBA. We experimented with decomposing the ll-FBA problem into a master and a subproblem, and compared separation approaches using no-good cuts and combinatorial Benders' cuts. The Combinatorial Benders' approach performs much better on BiGG instances than the big-M formulation of ll-FBA. However, the performance of the combinatorial Benders' approach depends strongly on the number of cuts added per iteration: too many cuts per iteration slow down the solution process significantly. On the yeast models, which are larger than most
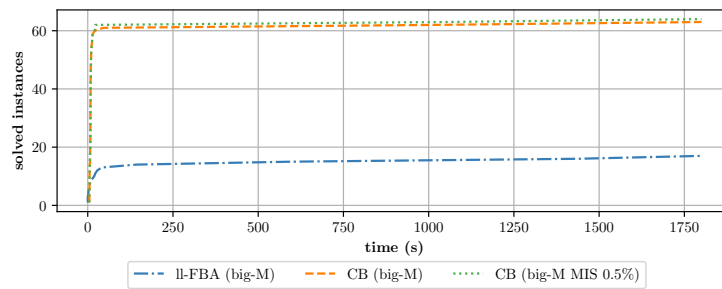
■ **Figure 6** Performance of the combinatorial Benders' method on enzyme models. Comparing the number of optimally solved instances with enzyme constraints of solving ll-FBA (big-M) directly, and solving the problem with the combinatorial Benders' approach within the time limit of 1800 seconds. We experiment with adding a single cut per iteration of the combinatorial Benders' approach and with adding at most k cuts per iteration, where k is 0.5% of the number of reactions in the model.

BiGG models, neither with the big-M reformulation nor with most combinatorial Benders' setups are we able to solve instances to optimality, showing the need for further work on scalable algorithms for ll-FBA. The combinatorial Benders' approach is flexible and can be extended, as long as we can still decompose the problem into a master and a subproblem, we illustrate this with additional enzymatic constraints. The models with enzyme constraints are experimentally much faster to solve. This opens the interesting question of their tractability: under suitable assumptions, is the ll-FBA with enzyme constraints hard?

References

1. Egon Balas. Disjunctive programming. In P.L. Hammer, E.L. Johnson, and B.H. Korte, editors, *Discrete Optimization II*, volume 5 of *Annals of Discrete Mathematics*, pages 3–51. Elsevier, 1979. `doi:10.1016/S0167-5060(08)70342-X`.

2. Daniel A. Beard, Eric Babson, Edward Curtis, and Hong Qian. Thermodynamic constraints for biochemical networks. *Journal of Theoretical Biology*, 228(3):327–333, 2004. `doi:10.1016/j.jtbi.2004.01.008`.

3. Ksenia Bestuzheva, Mathieu Besançon, Wei-Kun Chen, Antonia Chmiela, Tim Donkiewicz, Jasper van Doornmalen, Leon Eifler, Oliver Gaul, Gerald Gamrath, Ambros Gleixner, Leona Gottwald, Christoph Graczyk, Katrin Halbig, Alexander Hoen, Christopher Hojny, Rolf van der Hulst, Thorsten Koch, Marco Lübbecke, Stephen J. Maher, Frederic Matter, Erik Mühmer, Benjamin Müller, Marc E. Pfetsch, Daniel Rehfeldt, Steffan Schlein, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Boro Sofranac, Mark Turner, Stefan Vigerske, Fabian Wegscheider, Philipp Wellner, Dieter Weninger, and Jakob Witzig. Enabling Research through the SCIP Optimization Suite 8.0. *ACM Trans. Math. Softw.*, 49(2), 2023. `doi:10.1145/3585516`.

4. Suresh Bolusani, Mathieu Besançon, Ksenia Bestuzheva, Antonia Chmiela, João Dionísio, Tim Donkiewicz, Jasper van Doornmalen, Leon Eifler, Mohammed Ghannam, Ambros Gleixner, et al. The SCIP Optimization Suite 9.0, 2024. `arXiv:2402.17702`.

5. Gianni Codato and Matteo Fischetti. Combinatorial Benders' Cuts for Mixed-Integer Linear Programming. *Operations Research*, 54(4):756–766, 2006. Publisher: INFORMS. `doi:10.1287/opre.1060.0286`.

6. Abdelmoneim Amer Desouki, Florian Jarre, Gabriel Gelius-Dietrich, and Martin J. Lercher. CycleFreeFlux: Efficient Removal of Thermodynamically Infeasible Loops from Flux Distributions. *Bioinformatics*, 31(13):2159–2165, 2015. `doi:10.1093/bioinformatics/btv096`.

7. Christopher S. Henry, Linda J. Broadbelt, and Vassily Hatzimanikatis. Thermodynamics-Based Metabolic Flux Analysis. *Biophysical Journal*, 92(5):1792–1805, 2007. `doi:10.1529/biophysj.106.093138`.

8   Q. Huangfu and J. A. J. Hall. Parallelizing the dual revised simplex method. *Mathematical Programming Computation*, 10(2):119–142, 2018. `doi:10.1007/s12532-017-0130-5`.

9   Zachary A. King, Justin Lu, Andreas Dräger, Philip Miller, Stephen Federowicz, Joshua A. Lerman, Ali Ebrahim, Bernhard O. Palsson, and Nathan E. Lewis. BiGG Models: A platform for integrating, standardizing and sharing genome-scale models. *Nucleic Acids Research*, 44(D1):D515–D522, October 2015. `doi:10.1093/nar/gkv1049`.

10   Miroslav Kratochvíl, Laurent Heirendt, St Elmo Wilken, Taneli Pusa, Sylvain Arreckx, Alberto Noronha, Marvin van Aalst, Venkata P Satagopam, Oliver Ebenhöh, Reinhard Schneider, Christophe Trefois, and Wei Gu. COBREXA.jl: constraint-based reconstruction and exascale analysis. *Bioinformatics*, 38(4):1171–1172, 2021. `doi:10.1093/bioinformatics/btab782`.

11   Benoît Legat, Oscar Dowson, Joaquim Dias Garcia, and Miles Lubin. MathOptInterface: a data structure for mathematical optimization problems. *INFORMS Journal on Computing*, 34(2):672–689, 2021. `doi:10.1287/ijoc.2021.1067`.

12   Nathan E. Lewis, Kim K. Hixson, Tom M. Conrad, Joshua A. Lerman, Pep Charusanti, Ashoka D. Polpitiya, Joshua N. Adkins, Gunnar Schramm, Samuel O. Purvine, Daniel Lopez-Ferrer, Karl K. Weitz, Roland Eils, Rainer König, Richard D. Smith, and Bernhard Ø. Palsson. Omic Data from Evolved *E. coli* Are Consistent with Computed Optimal Growth from Genome-Scale Models. *Molecular Systems Biology*, 6(1):390, 2010. `doi:10.1038/msb.2010.47`.

13   Hongzhong Lu, Feiran Li, Le Yuan, Iván Domenzain, Rosemary Yu, Hao Wang, Gang Li, Yu Chen, Boyang Ji, Eduard J Kerkhoven, and Jens Nielsen. Yeast metabolic innovations emerged via expanded metabolic network and gene positive selection. *Molecular Systems Biology*, 17(10):e10427, 2021. `doi:10.15252/msb.202110427`.

14   Miles Lubin, Oscar Dowson, Joaquim Dias Garcia, Joey Huchette, Benoît Legat, and Juan Pablo Vielma. JuMP 1.0: Recent Improvements to a Modeling Language for Mathematical Optimization. *Mathematical Programming Computation*, 2023. `doi:10.1007/s12532-023-00239-3`.

15   Elad Noor, Nathan E. Lewis, and Ron Milo. A proof for loop-law constraints in stoichiometric metabolic networks. *BMC Systems Biology*, 6(1):140, 2012. `doi:10.1186/1752-0509-6-140`.

16   Jeffrey D. Orth, Ines Thiele, and Bernhard Ø. Palsson. What is flux balance analysis? *Nature Biotechnology*, 28(3):245–248, 2010. `doi:10.1038/nbt.1614`.

17   Bernhard Ø. Palsson. *Systems Biology: Constraint-based Reconstruction and Analysis*. Cambridge University Press, 2015. `doi:10.1017/CBO9781139854610`.

18   Karthik Raman. *An Introduction to Computational Systems Biology: Systems-Level Modelling of Cellular Networks*. Chapman and Hall/CRC, 2021. `doi:10.1201/9780429486951`.

19   Arne C. Reimers. *Metabolic Networks, Thermodynamic Constraints, and Matroid Theory*. Phd dissertation, Freie Universität Berlin, 2014. URL: `https://refubium.fu-berlin.de/handle/fub188/11116`.

20   Benjamín J Sánchez, Cheng Zhang, Avlant Nilsson, Petri-Jaan Lahtvee, Eduard J Kerkhoven, and Jens Nielsen. Improving the phenotype predictions of a yeast genome-scale metabolic model by incorporating enzymatic constraints. *Molecular Systems Biology*, 13(8):935, 2017. `doi:10.15252/msb.20167411`.

21   Jan Schellenberger, Nathan E. Lewis, and Bernhard Ø. Palsson. Elimination of Thermodynamically Infeasible Loops in Steady-State Metabolic Models. *Biophysical Journal*, 100(3):544–553, 2011. `doi:10.1016/j.bpj.2010.12.3707`.

22   Matthew Scott, Carl W. Gunderson, Eduard M. Mateescu, Zhongge Zhang, and Terence Hwa. Interdependence of Cell Growth and Gene Expression: Origins and Consequences. *Science*, 330(6007):1099–1102, 2010. `doi:10.1126/science.1192588`.

23   Hannah Troppens, Mathieu Besançon, St. Elmo Wilken, and Sebastian Pokutta. hannahtro/Loopless_Fluxes_with_Mixed_Integer_Optimization. Software, swhId: `swh:1:dir:9c41495c7f2ecd9e7459fa0c51d4be32fa0c033e` (visited on 2025-07-03). URL: `https://github.com/hannahtro/Loopless_Fluxes_with_Mixed_Integer_Optimization`, `doi:10.4230/artifacts.23821`.

A   Model Data

Table 3 Model size of used yeast models.

model	# metabolites	# reactions
Hanseniaspora_uvarum	2464	3569
yHMPu5000035696_Hanseniaspora_singularis	2460	3534
yHMPu5000034963_Hanseniaspora_clermontiae	2464	3573
yHMPu5000035695_Hanseniaspora_pseudoguilliermondii	2476	3559
yHMPu5000035684_Kloeckera_hatyaiensis	2465	3582
Eremothecium_sinecaudum	2549	3471
yHMPu5000035659_Saturnispora_dispora	2598	3378
Tortispora_caseinolytica	2693	3597
Starmerella_bombicola_JCM9596	2695	3735
Eremothecium_gossypii	2556	3555
Ashbya_aceri	2553	3623

B   Additional Computational Experiments

B.1   Experiments with Yeast Instances

Table 4 Comparing the termination status of solving yeast instances with ll-FBA (big-M) directly and solving ll-FBA (big-M) with CB. We experiment with the indicator and big-M formulations of the master problem. We compare adding a single cut per iteration of CB and adding multiple cuts. MIS 0.5% means that at most k cuts are added per iteration, where k is 0.5% of the number of reactions of the model.

solving strategy	# optimal	# time limit	# error
ll-FBA (big-M)	0	11	0
CB (big-M)	0	11	0
CB (indicator)	0	11	0
CB (big-M MIS 0.1 %)	0	11	0
CB (big-M MIS 0.2 %)	0	11	0
CB (big-M MIS 0.5 %)	1	10	0
CB (big-M MIS 2.0 %)	0	11	0
CB (indicator MIS 0.1 %)	0	11	0
CB (indicator MIS 0.2 %)	0	11	0
CB (indicator MIS 0.5 %)	0	11	0
CB (indicator MIS 2.0 %)	0	10	1

B.2   Cut Selection

Instead of blocking the entire set of cycles found per iteration of te combinatorial Benders' approach, we experiment with selecting a subset of cycles. We experiment with different strategies:

1. blocking distinct cycles,
2. selecting the k smallest cycles,
3. selecting cycles with a density limit.

Figure 7 shows that all tested strategies outperform blocking one cycle per iteration and directly solving ll-FBA. Most strategies perform similarly regarding the running time and the number of optimally solved instances. However, deriving an excessive number of MIS per iteration leads to increased running time and fewer solved instances. In the case of CB with indicator constraints, since more time is spent solving the master problem compared to CB with the big-M formulation, the time spent on cut selection impacts the overall running time. We observe that being too aggressive in cut filtering, i.e. removing useful cuts, can increase the running time.
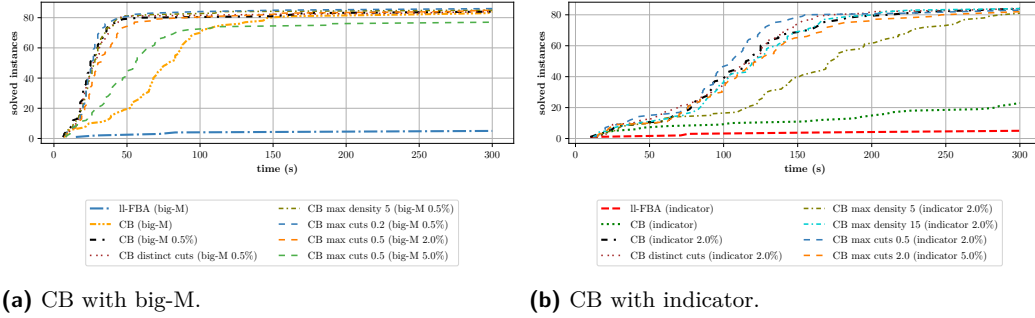

(a) CB with big-M.

(b) CB with indicator.

Figure 7 Performance of CB (big-M/indicator) with different cut selection strategies. We experiment with adding distinct cuts, adding the cuts with a maximal density and adding at most k cuts (max cuts). All strategies are applied with different parameters and a different size of maximally detected number of MIS per iteration.