Flux Envelope Analysis: A method for directly calculating the reduced-dimensional solution space and Phenotypic Phase Planes of constraint-based models

Matthew R. Long and Jennifer L. Reed

December 13, 2020

1 Abstract

Background: Genome-scale constraint-based metabolic modeling is widely used to model cellular metabolism in descriptive, predictive, and prescriptive ways. Less effort has been focused on understanding the impact that specific features of the model may have on the calculated solutions. Existing methods for calculating the Phenotypic Phase Plane (a type of sensitivity analysis) provide some insight into these nuances; however, methods for calculating these phase planes are inefficient and do not provide a high level of detail.

Results: We have developed a method called Flux Envelope Analysis which can directly and efficiently calculate all of the constraints of the solution space over a set of variables of interest. These constraints can be: plotted to generate Phenotypic Phase Planes; analyzed for the presence of alternate optimal solutions; and exported to a new optimization problem defined only over the reduced dimensions of interest, which may be more easily optimized with non-linear objectives.

Conclusions: Flux Envelope Analysis is more efficient, accurate, and detailed than existing methods for calculating the solution space of a linear program. The direct calculation of constraining facets allows for a wide variety of analyses and further calculations which could significantly improve the understanding of a variety of constraint-based genome-scale metabolic modeling predictions.

2 Introduction

Genome-scale constraint based modeling has been widely utilized for describing cellular fluxes, predicting cellular responses, and identifying cellular perturbations. For example, constraint-based modeling techniques can be used to desscribe the fluxes occurring in a cell [1], to predict drug discovery targets [2] and off-target drug effects [3], and for prescribing implementation strategies for achieving a specific metabolic engineering goal [4]. The majority of these methods are extensions or modifications of Flux Balance Analysis (FBA), which is utilized for calculating the steady-state flux distribution of reactions in a cell [5]. In FBA, each cell is assumed to be at steady-state (*i.e.*, no net accumulation or consumption of intracellular metabolites) while a specific cellular objective (*e.g.*, the biomass growth rate) is maximized or minimized [6]. This approach is sufficient for estimating all of the reaction fluxes within a cell and has been shown to be fairly accurate when compared with experimental data. Other methods are often extensions of FBA which either modify the objective functions—as in pFBA [7]—or encapsulate FBA into a larger problem—as in OptKnock [8]. Solutions from these methods can provide predictive, descriptive, and prescriptive information; however, comparatively less effort has been spent developing methods for elucidating the properties of these solutions beyond the values themselves.

One issue with linear optimization of under-determined problems, and therefore most solutions to constraint-based models, is the presence of multiple optimal solutions. Under these circumstances, the optimal value of the objective occurs at multiple extreme points. This results in an infinite number of optimal solutions, with no way for the solver to select the 'best' solution. Any SIMPLEX-type method, almost universally utilized for solving linear programs, will indiscriminately select one of the many best possible solutions. The presence of an alternate optimal solution can misguide interpretation of modeled solutions. For example, an OptKnock solution which predicts high product yield at maximum growth rate could have an alternate un-coupled solution of no yield at maximum growth. To this end, Flux Variability Analysis (FVA) is often utilized to discover whether any key decision variables may have a range of values at the optimal solution value [9, 10]. FVA is applied by fixing the objective value to some percentage of the optimal value found from another method (often 100%, but it may be decreased to identify the sensitivity of the solution to the optimal value) and then maximizing and/or minimizing the fluxes of any key variables to identify their range across possible solutions.

The entirety of the solution space can be more generally analyzed by creating a Phenotypic Phase Plane (PhPP) [11, 12]. These figures are generated by creating a mesh grid over all but one of the variables of interest (generally over one or two variables) and then solving for the maximum and/or minimum values of the remaining variable. This provides a two or more dimensional surface of the solution space and can provide insight into the various regimes that exist and border any solution. For example, PhPPs readily show the presence of alternate optimal solutions as being perpendicular to the objective, and can identify different cellular operating conditions such as oxygen and glucose limited growth in yeast. PhPP methods have been implemented and released as part of the COBRA toolbox [13] and Cameo [14] as well as other independent packages [10]. Calculating a PhPP via traditional techniques is not particularly efficient since it requires solving two problems per grid point (a maximum and minimum value) even though many of the solutions may be bounded by the same solution space facet. The execution time can be reduced by disabling the pre-solve step—which finds an initial valid point—and utilizing previous solutions to 'warm-start' the solver—starting at a point that should be close to the actual solution instead of starting from any valid point [10]; however, this does not reduce the number of problems which must be solved. Similarly, constructing a PhPP by utilizing a grid may average together or fail to detect different regimes due to features being smaller than the step-size of the grid. Finally, the different phases which exist in a PhPP may not be readily apparent and require additional analysis (*i.e.*, the delineation between different phases must be calculated after-the-fact and are not guaranteed to be precise). Techniques to create PhPPs which extract more information about the solution space than is contained in the target variables themselves have utilized dual variable values to delineate different phases [12]. In linear optimization, the solver will return both the values of variables and constraints at the optimal solution, referred to as primal values, as well as information about the impact that changing particular variable values or loosening certain constraints would have on the objective value, referred to as dual values. These dual values are particularly susceptible to the presence of degenerate optimal solutions. A degenerate optimal solution is when there is more than one choice for the combination of variables and/or constraints which yield the optimal solution value. An obvious example of this scenario is when there are alternate optimal solutions where moving to a different point yields different values with the same objective value (*i.e.*, an alternate solution). Since PhPP objective functions are generally defined over only a single variable, it is possible that the combination of other variables and constraints which limit the objective function could be changed, thus changing the dual values for those solutions. There is another form of degeneracy which can exist even if the optimal solution occurs only at a single point. If there are more than the required amount of constraints that intersect at the optimal solution point, the solver will pick any combination of these constraints and the various variable and constraint duals may change depending upon which of the constraints are chosen. While these types of degeneracies can be detected by closely examining the combination of variable and constraint primal and dual values, most existing work in constraint-based genome-scale modeling has generally avoided using dual values directly and have focused on checking for alternate optimal solutions for key decision variables.

To overcome these issues, we have developed a new method for efficiently and explicitly finding the solution space over a set of N dimensions of interest from any larger linear program. This method, called Flux Envelope Analysis (FEA), identifies each of the facets which exist in the solution space for the $\|N\|$ dimensions of interest as well as their defining vertices, edges, planes, and any higher dimensional faces. This method allows for both plotting the solution spaces—similarly to PhPPs—as well as creating new linear programs defined over only $\|N\|$ dimensions while fully encoding the behavior of the original larger problem. FEA is also resilient to degenerate solutions over the dimensions of interest.

3 Methods

3.1 Problem Definition

FEA is designed to calculate an $\|N\|$-dimensional solution space from a larger $\|M\|$-dimensional problem by identifying the facets which bound the reduced dimensional problem. The original $\|M\|$-dimensional linear program can be represented in standard form as:

$$\max_{\boldsymbol{v}} \sum_{m \in M} c_m \boldsymbol{v}_m \tag{1a}$$

$$\text{s.t.} \sum_{m \in M} a_{g,m} \boldsymbol{v}_m \leq b_g \qquad \forall g \in G \tag{1b}$$

where the variables, $\boldsymbol{v}$, are defined over the original dimensions defined in set M with constraints defined over set G. Parameters c, a, and b contain various coefficients.

FEA will calculate the constraints for a linear program over a set of target dimensions. These target dimensions are defined as set N, where N is a subset of M. Internally, FEA will utilize the following linear program:

$$\max_{\boldsymbol{v},\mathcal{O},\mathcal{F}} \; \mathcal{O} = \sum_{n \in N} c_n \boldsymbol{v}_n \tag{2a}$$

$$\text{s.t. } \mathcal{F}_f = \sum_{n \in N} \hat{k}_{f,n} \boldsymbol{v}_n - l_f = \epsilon \qquad \forall f \in F^{active} \tag{2b}$$

$$\alpha_n \leq \boldsymbol{v}_n \leq \beta_n \qquad \forall n \in N \tag{2c}$$

$$\sum_{m \in M} a_{g,m} \boldsymbol{v}_m \leq b_g \qquad \forall g \in G \tag{2d}$$

Note that the constraints from the original linear program as defined in Eq. (1b) are incorporated in Eq. Eq. (2d). The objective (Eq. (2a)) is denoted as $\mathcal{O}$ with the specific objective being defined by the parameter c which is defined by the FEA algorithm. Variable limits (Eq. (2c)) of α and β are added since FEA requires that the solution be an enclosed polytope over the variables in set N and is thus not capable of solving an unbounded problem. FEA automatically defines a series of additional equality constraints, denoted as $\mathcal{F}$ (Eq. (2b)), which are parallel to a known facet at some small distance, ϵ. The set of active $\mathcal{F}$ constraints, contained in F^{active}, is chosen by FEA. A facet of the solution space will be a single face of the $\|N\|$-polytope. This may be represented as a closed halfspace in $\mathbb{R}^{\|N\|}$ which may be mathematically represented by a hyperplane inequality constraint:

$$\mathcal{H}_f(\boldsymbol{v}) = \sum_{n \in N} \hat{k}_{f,n} \boldsymbol{v}_n - l_f \geq 0 \tag{3}$$

where $\hat{k}$ is the unit normal vector of the facet and l is a constant offset. For consistency, FEA will always define the unit normal vector of each constraining facet to point towards the center of the polytope. The $\mathcal{F}$ constraints (Eq. (2b)) are defined over these same parameters; however, they are parallel to the actual facet (Eq. (3)) and moved some small positive distance towards the center of the polytope, ϵ. Note that $\mathcal{O}$ and $\mathcal{F}$ as well as the variable limits are defined only over the target dimensions.

3.2 Solving for the Constraining Facet

When any linear program is solved with a SIMPLEX type algorithm, the values of variables and constraints (referred to as primal values) are returned as well as additional values which provide information about the impact that variables or constraints have on the objective (referred to collectively as duals or marginals and separately as reduced costs for variables and shadow prices for constraints). Of particular note to FEA, the dual values from constraints are the change in the value of the objective with respect to 'loosening' the right hand side of an active constraint; allowing them to be treated similarly to a derivative.

When Eq. (2) is solved, the duals from $\mathcal{F}$, denoted as $m^{\mathcal{F}}$, can be utilized to calculate the properties of the unkown facet which is constraining the optimal solution. The dual for a single constraint can be treated as:

$$m_f^{\mathcal{F}} = \frac{d\mathcal{O}}{d\mathcal{F}_f} \tag{4}$$

$$m_f^{\mathcal{F}} \frac{d\mathcal{F}_f}{d\boldsymbol{z}_f} = \frac{d\mathcal{O}}{d\mathcal{F}_f}\frac{d\mathcal{F}_f}{d\boldsymbol{z}_f} = \frac{d\mathcal{O}}{d\boldsymbol{z}_f} \tag{5}$$

where $\boldsymbol{z}_f$ is some arbitrary variable. To avoid trivial solutions, the value of $\frac{d\mathcal{O}}{d\boldsymbol{z}_f}$ must be non-zero. We can simply define this to be 1 since the choice of $\boldsymbol{z}_f$ is arbitrary:

$$\frac{d\mathcal{O}}{dz_f} = \sum_{n\in N} c_n \frac{d\boldsymbol{v}_n}{dz_f} \equiv 1 \tag{6}$$

The derivative of $\mathcal{F}_f$ with respect to $\boldsymbol{z}_f$ is:

$$\frac{d\mathcal{F}_f}{d\boldsymbol{z}_f} = \sum_{n\in N} d_{f,n} \frac{d\boldsymbol{v}_n}{d\boldsymbol{z}_f} \tag{7}$$

Substituting Eqs. (6) and (7) into Eq. (5) and re-arranging yields:

$$\sum_{n\in N} d_{f,n} \frac{d\boldsymbol{v}_n}{d\boldsymbol{z}_f} = \left(m_f^{\mathcal{F}}\right)^{-1} \tag{8}$$

Equations (6) and (8) yield two equations with $\|N\|$ unknowns, requiring an additional $(\|N\| - 2)$ equations to fully determine and solve the system of equations for $\frac{d\boldsymbol{v}_n}{d\boldsymbol{z}_f}$. Since $\mathcal{F}$ contains equality constraints, if there are a total of $(\|N\| - 1)$ constraints in F, then FEA can utilize one of the constraints for its dual and the remaining constraints to fully determine the system of equations. Each of the remaining constraints given by Eq. (2b) is shifted by

$\frac{d\boldsymbol{v}_n}{d\boldsymbol{z}_f}$, thus giving:

$$\sum_{n \in N} d_{f,n} \left(\boldsymbol{v}_n + \frac{d\boldsymbol{v}_n}{d\boldsymbol{z}_f} \right) - e_f = \epsilon \tag{9}$$

$$\sum_{n \in N} d_{f,n} \frac{d\boldsymbol{v}_n}{d\boldsymbol{z}_f} + \cancel{\left(\sum_{n \in N} d_{f,n} \boldsymbol{v}_n - e_f - \epsilon \right)} = 0 \tag{10}$$

$$\sum_{n \in N} d_{f,n} \frac{d\boldsymbol{v}_n}{d\boldsymbol{z}_f} = 0 \tag{11}$$

Combining Eq. (6) for the objective, Eq. (8) for one of the $\mathcal{F}$ constraints, and Eq. (11) for the remaining $\mathcal{F}$ constraints; the following system of equations may be solved for $\frac{d\boldsymbol{v}_n}{dz_f}$:

$$\begin{bmatrix} c_1 & c_2 & \cdots & c_{\|N\|} \\ d_{1,1} & d_{1,2} & \cdots & d_{1,\|N\|} \\ d_{2,1} & d_{2,2} & \cdots & d_{2,\|N\|} \\ \vdots & \vdots & \ddots & \vdots \\ d_{f,1} & d_{f,2} & \cdots & d_{f,\|N\|} \\ \vdots & \vdots & \ddots & \vdots \\ d_{\|F\|,1} & d_{\|F\|,2} & \cdots & d_{\|F\|,\|N\|} \end{bmatrix} \begin{bmatrix} \frac{d\boldsymbol{v}_1}{d\boldsymbol{z}_f} \\ \frac{d\boldsymbol{v}_2}{d\boldsymbol{z}_f} \\ \vdots \\ \frac{d\boldsymbol{v}_{\|N\|}}{d\boldsymbol{z}_f} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ \left(m_f^{\mathcal{F}}\right)^{-1} \\ \vdots \\ 0 \end{bmatrix} \forall f \in F \tag{12}$$

where $\frac{d\boldsymbol{v}_n}{d\boldsymbol{z}_f}$ can be thought of as a ray originating at the optimal solution point and directed such that it lies: perpendicular to the $\mathcal{F}$ constraint whose marginal was utilized, on each of the other $\mathcal{F}$ constraints, and most importantly on the unknown constraining facet.

FEA can calculate one of these rays for each marginal of the $\mathcal{F}$ constraints, providing a total of $(\|N\| - 1)$ rays that lie in the surface of the unknown constraining facet. Therefore, each of these rays will be orthogonal to the normal vector of the unknown constraining facet and the dot product between them will be zero. Since FEA defines the normal vector of a facet as pointing towards the center of the polytope and the objective vector is maximized (Eq. (2a)) and thus points outward from the polytope at the optimal solution: the normal vector must be pointed in the opposite direction of the objective vector making the dot product between these two vectors negative. Combining these properties provides a fully determined system of equations:

$$\begin{bmatrix} c_1 & c_2 & \cdots & c_{\|N\|} \\ \frac{d\boldsymbol{v}_1}{dz_1} & \frac{d\boldsymbol{v}_2}{dz_1} & \cdots & \frac{d\boldsymbol{v}_{\|N\|}}{dz_1} \\ \frac{d\boldsymbol{v}_1}{dz_2} & \frac{d\boldsymbol{v}_2}{dz_2} & \cdots & \frac{d\boldsymbol{v}_{\|N\|}}{dz_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{d\boldsymbol{v}_1}{dz_{|F|}} & \frac{d\boldsymbol{v}_2}{dz_{|F|}} & \cdots & \frac{d\boldsymbol{v}_{\|N\|}}{dz_{|F|}} \end{bmatrix} \begin{bmatrix} k_1 \\ k_2 \\ \vdots \\ k_{\|N\|} \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{13}$$

which may be solved for k, the normal vector of the unknown constraining facet.

Since FEA may repeatedly encounter the same facet, it is desirable for each facet to have a uniquely identifiable fingerprint. To this end, FEA utilizes the unit normal vector for the facet defined as:

$$\hat{k}_n = \frac{k_n}{\left[\sum_{x \in N} (k_x)^2\right]^{1/2}} \tag{14}$$

Finally, the facet offset can be calculated from the optimal solution point:

$$l = \sum_{n \in N} \hat{k}_n \boldsymbol{v}_n \tag{15}$$

This process uniquely calculates the parameters of the constraining facet and, should the same facet be found in a later iteration, will result in an identical set of parameters.

As previously mentioned, when the optimal solution is degenerate the assumptions underlying this process may break down. In a degenerate solution, there are more constraints bounding the optimal solution than required. Therefore, the solver can choose which constraints are basic (*i.e.*, bounding) in an unpredictable manner. In particular, it is possible that different variables and constraints are bounded by different unknown facets and thus the facet calculated by FEA may be invalid. FEA utilizes the $(\|N\| - 1)$ $\mathcal{F}$ equality constraints defined in Eq. (2b) to constrain the solution to a single ray which will either terminate in a single facet of the $\|N\|$-dimensional space or in the intersection of multiple original facets. In order to avoid degenerate solutions, the $\mathcal{F}$ constraints are generally parallel to a known facet at a small distance, defined by ϵ. If a degenerate solution is detected, one of the ϵ values can be decreased and the problem can be re-solved. In the process of finding facets, FEA can detect that a facet is invalid if any observed point in the solution does not comply with the facet halfspace equation (*i.e.*, an observed vertex is eliminated by a calculated halfspace, indicating that the halfspace is invalid). If this is found, the facet can be removed.

3.3 Searching N-Dimensional Solution Space

While this formulation allows for the calculation of one constraining facet, FEA solves for all constraining facets of the $\|N\|$-dimensional solution space. Therefore, the algorithm must choose equality constraints ($\mathcal{F}$), solve for the constraining facet, and then iterate to find a new facet until the entirety of the solution space has been discovered.

As previously discussed, the $\|N\|$-dimensional solution space for a linear problem will be an $\|N\|$-polytope constrained by facets which are $\|N\|$ dimensional halfspace constraints (Eq. (3)). In order to ensure that the solution space is an enclosed polytope, all target variables must have upper and lower bounds set to some finite value (Eq. (2c)). Vertices occur at the intersection of at least $\|N\|$ facets and are 0-dimensional. The intersection of $\|N\| - 1$ facets defines an edge. All other levels between 1 and $(\|N\| - 1)$ will be referred to as the l-face where l is the dimension of the element. Thus in two dimensions, facets are edges and the intersection of two adjacent lines forms a vertex. These elements can be shown as nodes on a face lattice graph, where a node of level l is defined by the intersection of $(\|N\| - l)$ facets. The exception to this rule are vertices ($l = 0$) and the null polytope ($l = -1$) which may exceed this number of facets. The directed edges of the graph, hereafter referred to as connections to distinguish them from geometric edges, indicate that all of the facets contained in a node are inherited by the child node. Figure 1A-B shows an example of a three dimensional pyramid and its corresponding face lattice graph.

In order to calculate the solution polytope and face lattice graph, FEA will choose an existing node in the lattice graph as the search node. Initially, the only node in the graph is the polytope node itself. FEA will search for a child node by identifying a facet that constrains the search node. This traversal method is similar in principal to performing a pivot in the SIMPLEX algorithm or the gift wrapping method. In order to do this, FEA will utilize each facet contained in the search node as one of the fixed $\mathcal{F}$ constraints, with a small offset distance defined by ϵ. This offset serves as a feature detection limit, with any facets and vertices that exist less than ϵ distance from a known facet remaining undiscovered. FEA will then calculate an objective vector, c, that is orthogonal to the normal vectors of each of these facets. If the search node has any known child nodes, then the dot product between the normal vector for any facets contained in the child nodes but not the search node should be positive to avoid re-discovering a previously identified facet. If there are insufficient constraints to uniquely determine an objective direction, any random objective that meets any existing constraints may be utilized. If no objective that meets these criteria can be found, the search

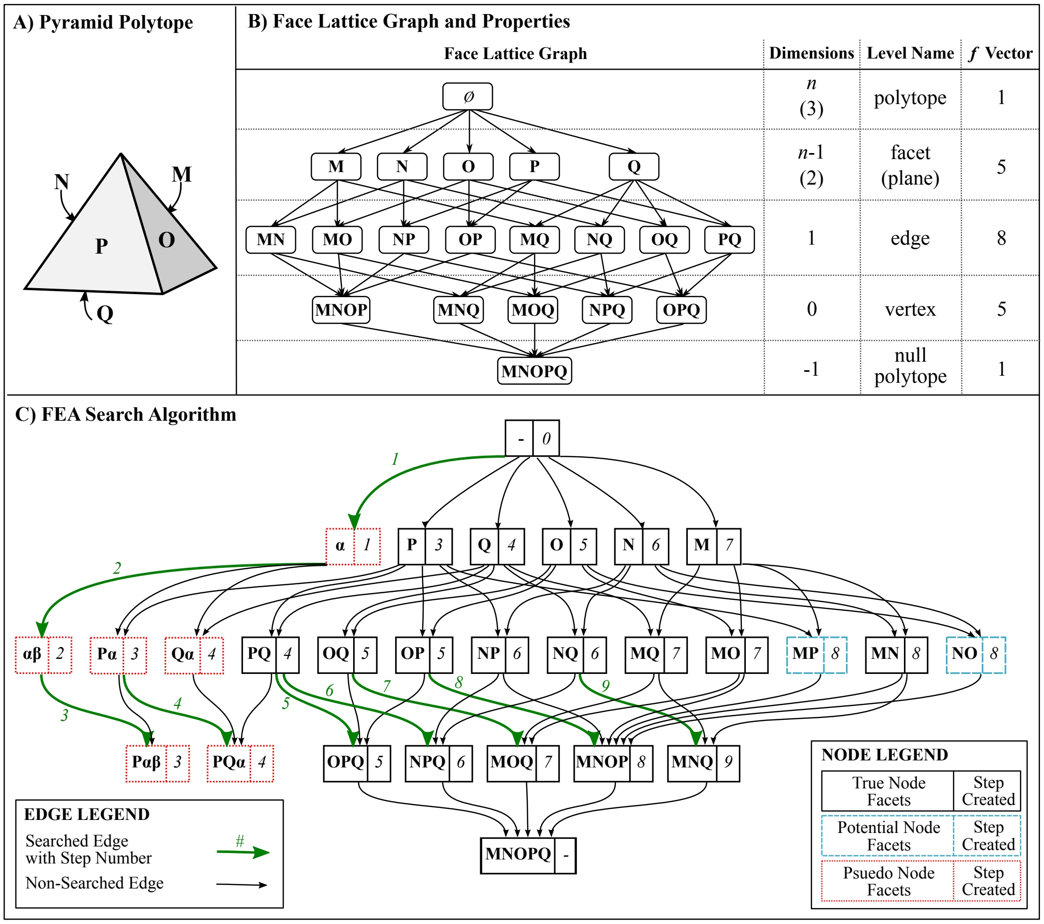

Figure 1: (A) A three dimensional ($\|N\| = 3$) pyramid polytope with labeled facets and (B) the corresponding lattice graph as defined by its facets are shown. Each node is represented by the set of facets which define it. The dimensions and level names for the nodes on the lattice graph are shown. The f-vector is the number of valid nodes for each level. (C) The facet search pattern and lattice graph generation algorithm of FEA is shown for the pyramid polytope. Search steps are shown (wide green connection arrows) along with the iteration number of the search. Each node is labeled with the facets it is defined by as well as the search step in which it was created. Psuedo facets are Greek characters while real facets are Latin characters. True (solid black), potential (dashed blue), and psuedo (red dotted) nodes are shown.

node is discarded, a new search node is chosen, and the process is repeated. If an objective can be found, the optimization problem given by Eq. (2) is solved.

The search node will only contain $(\|N\| - 1)$ $\mathcal{F}$ constraints—as required by Eq. (13) to solve for an unknown bounding facet—when an edge node is chosen. For any other higher-level node, the constraining facet cannot be calculated. Instead, a psuedo-facet is created whose normal vector is the negative objective direction at the optimal solution point. This psuedo-facet is tangent to the solution space in the context of the added $\mathcal{F}$ constraints. The psuedo-facet is appended to the search node to create a new child psuedo-node. The creation of psuedo-facets provides the additional equality constraints required until an edge psuedo-node is created and searched, at which point a real constraining facet may be found. It is important that psuedo-facets are not propagated up the face lattice, since they are only valid in the context of all of the constraints which existed when they were created.

If the search node is an edge, the constraining facet is calculated using the procedure described in Section 3.2. This facet is appended to the search node and added as a child node of the search node. In addition, all possible combinations of the child node's facets are tentatively created as hypothetical parent nodes and propagated up the lattice graph (*e.g.*, if a three-dimensional vertex if created by the intersection of three plane facets, FEA will hypothesize that each combination of two plane facets may be an edge). Not all of these nodes will necessarily exist; however, the combinations encompass the full range of possible valid parent nodes.

FEA is will then choose a new search node and the process is repeated. In general, FEA will choose the lowest level un-searched node containing the fewest psuedo-facets as the search node. An example of a completed FEA run with numbered steps is shown for a pyramid in Fig. 1C.

3.4 Verifying the N-Dimensional Solution

Since the solution to FEA is an enclosed $\|N\|$-polytope with non-zero volume, there are several statements about the properties of valid lattice graphs which may be utilized. Considering the entire lattice graph, the f-vector—containing the total number of valid nodes at each level of the lattice graph as shown in Fig. 1C—may be constructed. From the f-vector, the $\|N\|$-dimensional extension of Euler's polyhedral formula [15] by Schläfli [16] will hold for any enclosed n-polytope:

$$\sum_{l=-1}^{|N|} (-1)^l f_l = 0 \tag{16}$$

Additionally, since all enclosed, convex $\|N\|$-polytopes can be decomposed into a simplicial complex via triangulation [17], then each value of the f-vector must be greater than or equal to the f-vector of the related $\|N\|$-simplex. A simplex is a generalization of a triangle in higher dimensions, allowing any $\|N\|$-polytope to be constructed by combining multiple simplices together, making all polytopes a superset of a simplex. Conveniently, the f-vector for a $\|N\|$-simplex is the $\|N\|$th row of Pascal's Triangle and can thus be easily computed and compared to the f-vector discovered by FEA [16].

Finally, since each node of level l in the polytope with its children is itself a valid l-polytope, the same criteria can be utilized to determine the validity of any subgraph of the full face lattice graph. For these nodes, FEA only utilizes the criteria that a valid node of level l must have at least $(l + 1)$ valid child nodes since this is the minimum number of nodes necessary to be a l-simplex. At this time, we have not found it necessary to perform a more complete filtering of these subgraphs. An example of an edge node without sufficient children is the edge **NO** shown in Figure Fig. 1C. While facets **N** and **O** do intersect at vertex node **MNOP**, the edge defined by node **NO** only has a single vertex child node and thus has zero length and is invalid. It may be possible for these properties to be true before the search has been completed; however, they are guaranteed be true when the polytope is completed. Therefore, these criteria are utilized by FEA to check whether the polytope has been fully discovered only after all edge nodes have been searched.

3.5 Implementation

The code for FEA has been implemented in a Python 3 package compatible with COBRApy, Cameo, and OptLang. Both COBRApy and Cameo utilize OptLang for their underlying optimization interface with either CPLEX, GUROBI, or GLPK as the linear program solver. All results were solved using CPLEX. Systems of equations were solved with NumPy. Nonlinear optimization was performed with SciPy's COBYLA function.

The package is available at https://github.com/long-m-r/fea.

4 Results

FEA was implemented and evaluated to ensure that the method is accurate and efficient. The performance was compared to existing methods, although it is worth noting that the richness of information directly provided by FEA is different than that of any available existing method. Finally, several applications of FEA to genome-scale constraint-based metabolic modeling were evaluated and and reported.

4.1 Validation and Performance

In order to evaluate the performance of FEA, a series of random original linear programs were created. The constraints for these problems were created by generating a random unit-normal vector for each constraint and setting the right hand side of the constraint to be randomly distributed between -1 and 1. Each constraint was defined as greater-than-or-equal-to or less-than-or-equal-to such that the origin was valid. This procedure was utilized so that the number of variables and constraints in the original problem can be varied independently of the target variables for the FEA reduced-dimensional solution space.

First, the FEA solution spaces were tested against those calculated utilizing an FVA-based PhPP routine. The PhPP routine creates an $(\|N\| - 1)$ dimensional grid and fixes all but one of the variables to a position on the grid. It then maximizes and minimizes the remaining variable to calculate the range of that variable. When comparing this solution to that calculated by FEA, each of the PhPP routine's calculated points should be within the detection limit distance, ϵ, of a facet found by FEA over the same reduced variable set, since the PhPP solution should be bounded by the facets found by FEA. For random original original problems spanning between 5 and 100 variables and reduced problems containing between 1 and 5 dimensions, the facets calculated by FEA included all points calculated by FVA. A subset of PhPP solutions plotted against FEA polytopes are shown in Fig. 2.

FEA was further tested to ensure that it successfully captures all of the information of a higher order $(\|N\| > 3)$ solution space. To this end, random five-dimensional problem were generated and solved directly for various four- three- and two-dimensional spaces. Each of the three- and four-dimensional solution spaces calculated by FEA were converted into new linear programs, allowing FEA to be applied again and new two- and three-dimensional solution spaces were calculated. Regardless of whether a solution space was calculated from the original five-dimensional problem or from one of the various reduced-dimensional spaces, every FEA solution space over the same set of variables were identical. This indicates that all of the information con-

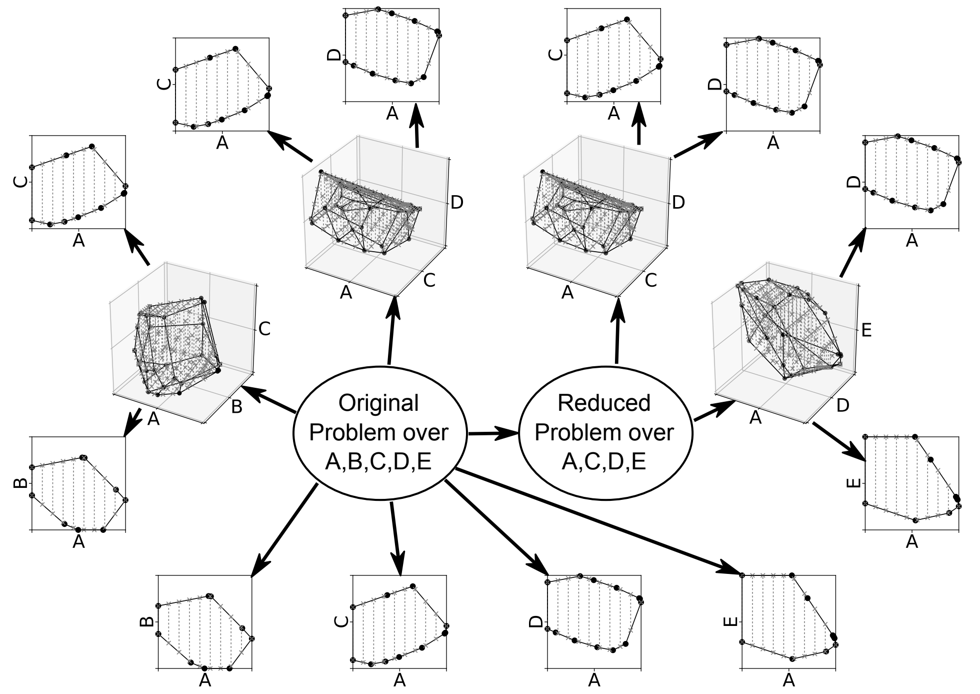

Figure 2: Plots of the FEA reduced solution spaces from a single original problem of five bounded variables (A, B, C, D and E) with ten random halfspace constraints. FEA calculated vertices are shown as black dots and edges are shown as black line segments. Each arrow indicates the FEA reduction from the origin dimensions to the target dimensions. In cases where the origin was already reduced, a new linear program was created from the facet halfspaces of the origin FEA solution. FVA results are also plotted, with the maximum and minimum points shown by blue 'x'es connected by a dashed blue line. FVA was performed with ten evenly spaced increments for each independent variable.

tained in the original problem with regards to the reduced variable sets are encoded in the reduced FEA solution. These spaces, as well as the paths taken to calculate them are shown in Fig. 2.

While these results show the validity of the solution space calculated by FEA, the performance of FEA is also critical for its usage. The existing PhPP routine for calculating the solution space requires solving two optimization problems (minimum and maximum) for each point in an $(\|N\| - 1)$ dimensional mesh. This results in solving $2m^{\|N\|-1}$ different optimization problems, assuming m grid points are created for each dimensions. Conversely, FEA requires solving $\|N\|$ optimization problems to find the first facet, another $(\|N\| - 1)$ problems to find the first vertex, and a single problem to find each remaining vertex. FEA requires more computation per iteration since each optimization solution requires solving for the constraining facet by solving $\|N\|$, $\|N\|$-dimensional systems of equations. If there are v vertices, FEA will require solving $(v + 2\|N\| - 2)$ optimization problems; however, the number of vertices cannot be known *a priori*. As previously discussed, a simplex is the simplest form of an $\|N\|$-polytope and has a total of $\|N\| + 1$ vertices, requiring $(3\|N\| - 1)$ optimization steps to solve. On the other hand, a sphere can similarly be generalized in n-dimensions with an infinite number of constraining facets and vertices. As a sphere would require infinitely many constraints to be represented by the original problem, it is likely that the solution polytope will be more similar to a simplex than a sphere. Therefore, in the best case scenario FEA will require a linear increase in steps as the number of dimensions increases while FVA increases exponentially; however, in the worst-case scenario FEA may require an infinite number of steps to complete. On the other hand, since FEA directly and explicitly calculates the constraining facets, the richness of the data returned by FEA compared to FVA is substantially more detailed and precise and may therefore be utilized for more complex analyses afterward.

This performance was tested by generating a variety of random linear programs with varying numbers of original variables, original constraints, and reduced variables. Overall, FEA utilizes close to the theoretical minimum number of solutions per problem (calculated once the solution was found and the number of vertices was known); however, the time scales exponentially with the number of optimization problems. This is likely due to the time required by the various optimization and linear equation solving steps increasing exponentially as a function of the number of dimensions (problems with more dimensions also require more theoretical steps, thus making it difficult to differentiate these two phenomena). These results are shown in Fig. 3.

4.2 Phenotypic Phase Plane Analysis

FEA was utilized to calculate the PhPP with respect to glucose uptake rate (GUR), oxygen uptake rate (OUR), and biomass growth rate (μ) for *Saccharomyces cerevisiae* utilizing the iMM904 [18] genome-scale metabolic network. Based soley on these three variables, the PhPP shown in Figure Fig. 4a

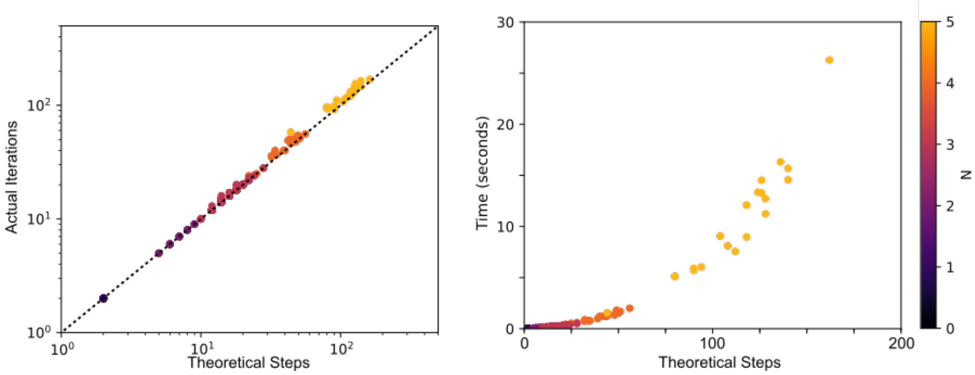

Figure 3: The (A) comparison between number of solved linear programs to the theoretical steps required by FEA and (B) the execution time compared to the theoretical number of steps required. Both plots vary the number of dimensions between one and five.

is similar to that calculated Duarte *et al.* [12]. The extra edges found by FEA may be the result of either our utilization of a more recent metabolic model for *S. cerevisiae* or due to the increased sensitivity of FEA (*i.e.*, several of the edges are close together and may have been missed by a traditional grid-based approach). By adding the ethanol production rate as the fourth dimension for FEA, a four dimensional space was generated and directly analyzed for the interaction between growth and ethanol production. These results are shown in Fig. 4B-C. By directly calculating all of our variables of interest, the relationship of ethanol production in a single facet of the original PhPP is shown to be non-continuous, with new edges dividing previously identified facets into different regimes. Additionally, there are points in the 3D space defined by GUR, OUR, and μ which have alternate solutions for ethanol production. For example, the point defined by a GUR and OUR of 20 mmol/gDW/hr and a μ of zero has two vertices located at ethanol production rates of 0 and 33.33, indicating that the ethanol production rate can be anywhere between those two values. This result shows the dangers of directly utilizing variable and constraint marginals, since there may be alternate optimal solutions at various locations within a single facet. Therefore, it is important to directly calculate the PhPP for any variables of interest instead of extrapolating results based upon variable or constraint values or marginals. It is worth noting that all of the plots found in Fig. 4B-C can be generated from a single run of FEA.

The PhPP for *S. cerevisiae* grown in a chemostat with a dilution rate of 0.1hr^{-1} under various levels of oxygen limitation. This PhPP was compared to the experimentally measured data from Weusthuis *et al.* [19]. This PhPP is shown in Fig. 5. These data all lie within a single edge of the PhPP, indicating the yeast actively minimize both the glucose uptake rate and the oxygen uptake rate and that ethanol production is largely dependent upon achieving these two goals. Conveniently, since FEA calculates the unit-normal vectors of each constraining facet, it is trivial to identify the constraining vector for a given node as well as the objective vector which, if maximized, would result in the solution being constrained by the given node. This vector can be found by adding the unit-normal vectors of all facets contained in a node and, optionally, re-normalizing the vector.

In addition to yeast, the PhPP for *Escherichia coli* was calculated utilizing the iJO1366 model [20]. Figure 6 shows this PhPP for various growth rates,

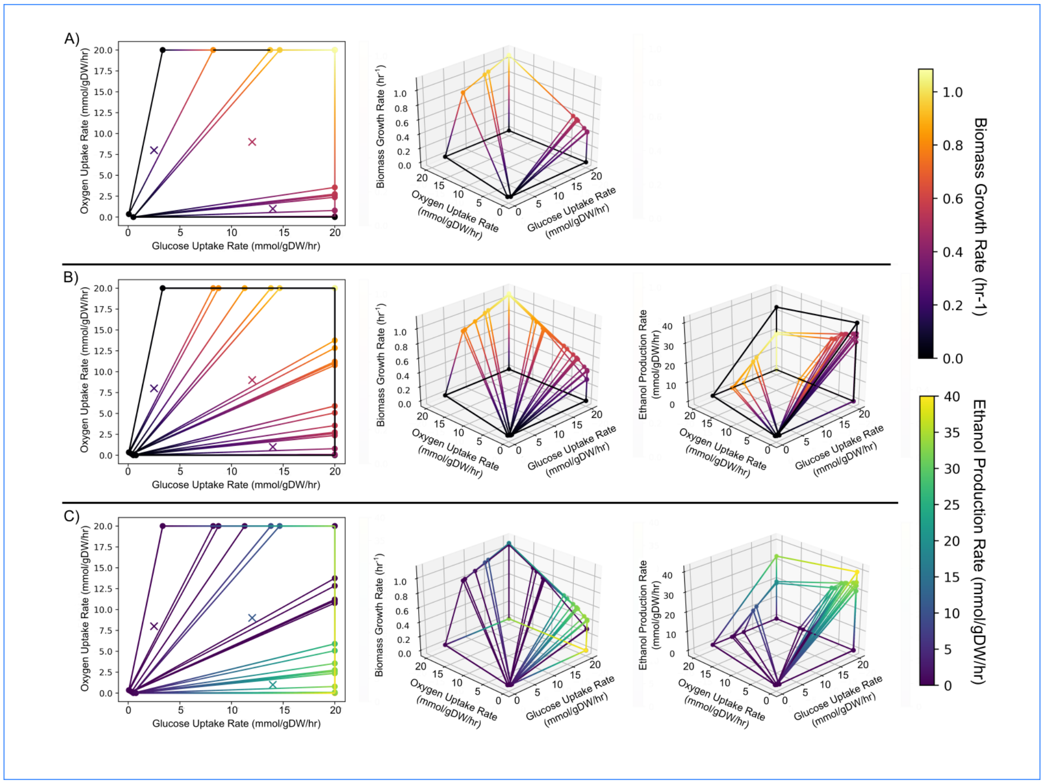

Figure 4: Various PhPPs for *S. cerevisiae* with respect to glucose uptake rate (GUR), oxygen uptake rate (OUR), ethanol production rate (EPR), and/or biomass growth rate (μ) are shown. The black-to-yellow color scale corresponds to μ and the blue-to-green color scale corresponds to EPR. (A) the PhPP calculated with respect to only GUR, OUR, and μ is shown as both a top-down projection and as a three dimensional plot. Experimental data points [12] from distinct regimes of yeast growth are shown as 'x'es on the top-down projection and correspond to distinct facets of the PhPP. (B) The PhPP was again calculated using all four variables and all edges and vertices were plotted for GUR and OUR against either EPR or μ along with a top down projection. Including the EPR in the calculation shows that there are substantially more facets on the surface of the PhPP with respect to growth than previously identified. (C) The same PhPPs are shown as panel B; however, the color scale has been changed to reflect the EPR instead of μ.

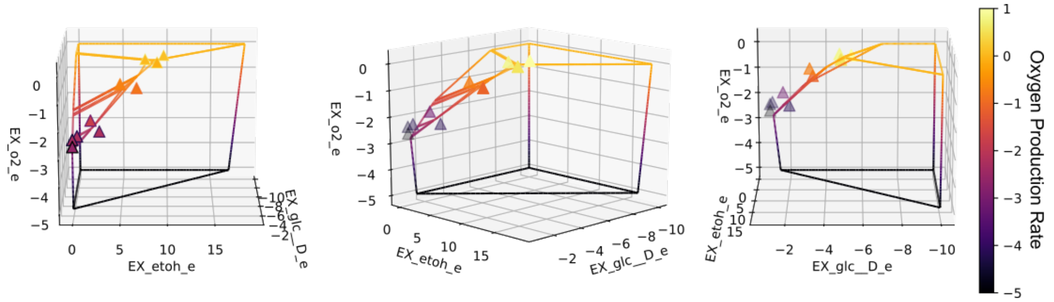

Figure 5: The PhPP for *S. cerevisiae* grown in a chemostat with a dilution rate/biomass growth rate of $0.1hr^{-1}$ was calculated with respect to the glucose uptake rate, ethanol production rate, and oxygen uptake rate. The same plot is shown rotated from three different angles. The color corresponds to the oxygen uptake rate. Experimental data for various levels of oxygen feed rate are plotted as triangles [19]. Note that the experimental results align with the ethanol production rates defined by the edge(s) which have the combined minimum oxygen and glucose uptake rates

oxygen production rates, and glucose uptake rates. Experimental data from Ishii *et al.* for aerobic chemostat growth is also plotted [21]. Overall, these results indicate that the strain is actively minimizing the glucose uptake rate for a given growth rate (as defined by the dilution rate of the chemostat). Furthermore, *E. coli* MC4100 was modelled by utilizing the iJO1366 metabolic model [20] with genes deleted which are known to be missing in *E. coli* MC4100 [22]. The PhPP for *E. coli* MC4100 was compared with experimental data from a chemostat with a dilution rate of 0.3hr^{-1} over various oxygen uptake rates [23]. This PhPP is shown in Fig. 7. The aerobic data points were clustered to a single edge where the glucose uptake rate is minimized and no ethanol is produced; however, the micro-aerobic data points (OUR of 5.9 and 5.0 mmol/gDW/hr) were not clearly defined by a

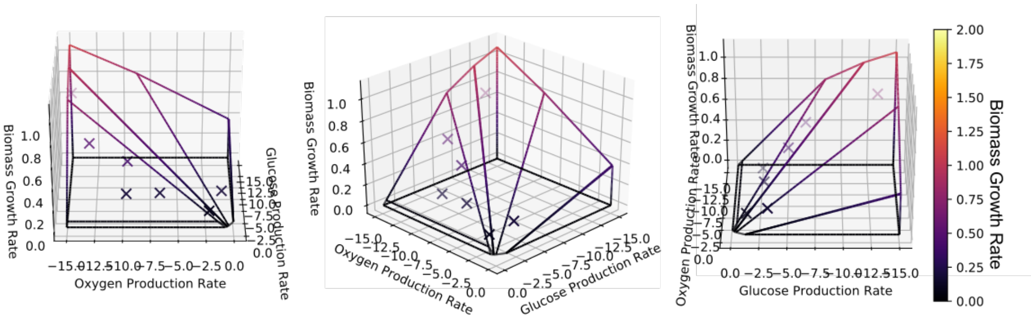

Figure 6: The PhPP for *E. coli* grown in a chemostat with a dilution rate/biomass growth rate varying between $0.1hr^{-1}$ and $0.7hr^{-1}$. The PhPP was calculated with respect to the glucose uptake rate, oxygen uptake rate, and the biomass growth rate. The same plot is shown rotated from three different angles. The color corresponds to the biomass growth rate. Experimental data for various biomass growth rates are indicated by 'x' [21]. Note that there are three replicate data points at a growth rate of $0.2hr^{-1}$ which have very different oxygen uptake rates.

single facet with respect to GUR, OUR, and ethanol production rate. By further calculating the PhPP with respect to ethanol, formate, and acetate secretion, it becomes apparent that the micro-aerobic data is balancing it's utilization of these three products. The constraining facet of these results is most bounding to acetate secretion, followed by ethanol secretion and finally formate. This suggests that the cells may prioritize acetate secretion over ethanol and both over formate.

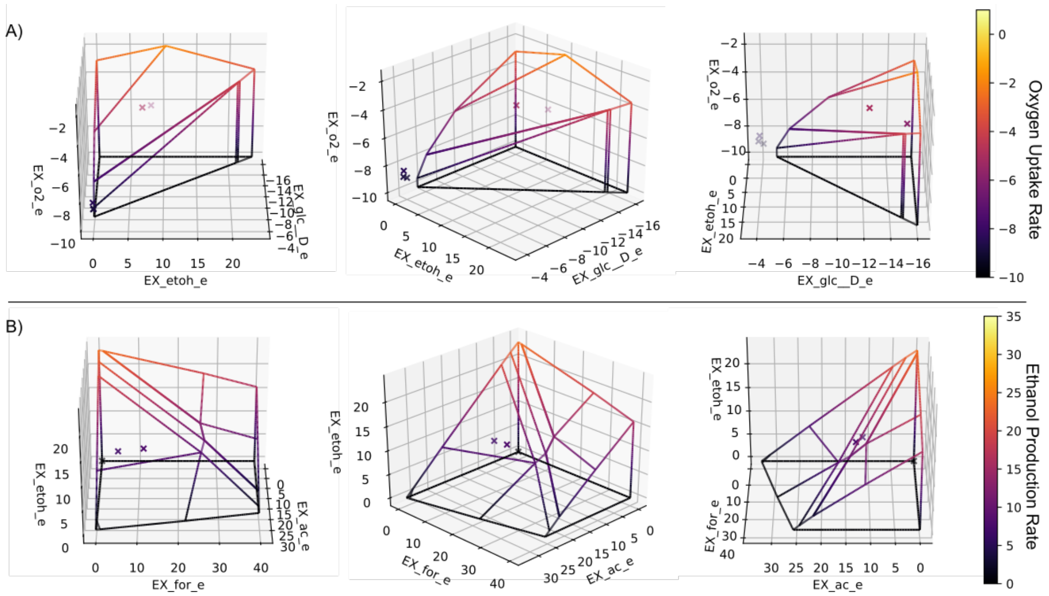

Figure 7: The PhPP for *E. coli* MC4100 grown in a chemostat with a dilution rate/biomass growth rate of $0.3hr^{-1}$. Experimental data for various oxygen concentrations (ranging from 0.5 to 10% of air saturation) are indicated by 'x' [23]. Each plot is shown rotated from three different angles. (A) The PhPP was calculated with respect to the glucose uptake rate, ethanol production rate, and oxygen uptake rate. The color corresponds to the oxygen uptake rate. The fully aerobic data points are clustered at the minimum glucose uptake rate whereas the micro-aerobic data are not clearly limited by a single facet. (B) The PhPP with respect to formate, acetate, and ethanol production rates was also calculated. The color here refers to the ethanol production rate. While the aerobic data points produce none of these products, the micro-aerobic data is bounded by a single facet whose normal vector indicates that the strain is prioritizing acetate production over ethanol production with formate production being the least important.

5 Discussion

6 Conclusions

FEA can directly calculate a system of linear equations over a reduced number of dimensions from an original linear program. This algorithm requires solving a number of optimization problems; however, the number of problems increases linearly with the number of dimensions and vertices which exist in the reduced space. Results from FEA can be utilized to interpret the properties of the solution space, construct new optimization problems, and generate figures such as the Phenotypic Phase Plane.

FEA is also able to identify when a complete solution has been identified. Under certain conditions, FEA may fail to identify all of the facets, edges, and/or vertices of a solution and be unable to identify a new search step; however, it can detect that the solution is incomplete and simply re-running the algorithm often produces a complete answer since FEA begins by searching randomly. Further development to identify and eliminate any situations where FEA may fail to completely identify the solution is being performed and, as the project is being released as an open source package, public contribution is possible. It may further be possible to decrease the computational requirement of FEA by utilizing information from intermediate tableau of the SIMPLEX method to calculate bounding facets; however, such a procedure would require either implementing a SIMPLEX solver or devising a more direct interface to the already available solver packages.

Overall, FEA allows for the direct calculation, analysis, and plotting of linear programs over reduced dimensions and should be applicable to a wide variety of problems in constraint-based genome-scale metabolic modeling as well as all linear programming problems in general.

7 Bibliography

References

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

[2] J. Kim, J. L. Reed, and C. T. Maravelias, "Large-scale bi-level strain design approaches and mixed-integer programming solution techniques.," *PloS one*, vol. 6, p. e24162, jan 2011.

[3] R. L. Chang, L. Xie, L. Xie, P. E. Bourne, and B. Ø. Palsson, "Drug off-target effects predicted using structural analysis in the context of a metabolic network model.," *PLoS computational biology*, vol. 6, no. 9, p. e1000938, 2010.

[4] M. R. Long, W. K. Ong, and J. L. Reed, "Computational methods in metabolic engineering for strain design," *Current Opinion in Biotechnology*, vol. 34, pp. 135–141, aug 2015.

[5] E. T. Papoutsakis, "Equations and calculations for fermentations of butyric acid bacteria.," *Biotechnology and bioengineering*, vol. 26, pp. 174–87, feb 1984.

[6] R. Schuetz, L. Kuepfer, and U. Sauer, “Systematic evaluation of objective functions for predicting intracellular fluxes in Escherichia coli.,” *Molecular systems biology*, vol. 3, p. 119, jan 2007.

[7] N. E. Lewis, K. K. Hixson, T. M. Conrad, J. A. Lerman, P. Charusanti, A. D. Polpitiya, J. N. Adkins, G. Schramm, S. O. Purvine, D. Lopez-Ferrer, K. K. Weitz, R. Eils, R. König, R. D. Smith, and B. Ø. Palsson, “Omic data from evolved E. coli are consistent with computed optimal growth from genome-scale models.,” *Molecular systems biology*, vol. 6, p. 390, jul 2010.

[8] A. P. Burgard, P. Pharkya, and C. D. Maranas, “Optknock: a bilevel programming framework for identifying gene knockout strategies for microbial strain optimization.,” *Biotechnology and bioengineering*, vol. 84, pp. 647–57, dec 2003.

[9] R. Mahadevan and C. Schilling, “The effects of alternate optimal solutions in constraint-based genome-scale metabolic models,” *Metabolic Engineering*, vol. 5, no. 4, pp. 264–276, 2003.

[10] S. Gudmundsson and I. Thiele, “Computationally efficient flux variability analysis.,” *BMC bioinformatics*, vol. 11, p. 489, sep 2010.

[11] A. Varma, B. W. Boesch, and B. O. Palsson, “Stoichiometric interpretation of Escherichia coli glucose catabolism under various oxygenation

rates.," *Applied and environmental microbiology*, vol. 59, pp. 2465–73, aug 1993.

[12] N. C. Duarte, B. Ø. Palsson, and P. Fu, "Integrated analysis of metabolic phenotypes in Saccharomyces cerevisiae.," *BMC Genomics*, vol. 5, p. 63, sep 2004.

[13] A. Ebrahim, J. A. Lerman, B. O. Palsson, and D. R. Hyduke, "COBRApy: COnstraints-Based Reconstruction and Analysis for Python," *BMC Systems Biology*, vol. 7, p. 74, aug 2013.

[14] N. Sonnenschein, J. G. R. Cardoso, H. Redestig, K. Jensen, E. Özdemir, M. E. Beber, and S. Galkina, "Biosustain/Cameo: 0.11.0," jan 2017.

[15] L. Euler, "Elementa doctrine solidorum.," *Novi commentarii academiae scientiarum imperialis petropolitanae,*, vol. 4, pp. 109–160, 1752.

[16] H. S. M. Coxeter, *Regular polytopes.* Dover, 1973.

[17] K. H. Rosen, "Discrete and Computational Geometry," in *Handbook of Discrete and Combinatorial Mathematics* (K. H. Rosen, ed.), ch. 13, pp. 797–888, 2000.

[18] M. L. Mo, B. O. Palsson, and M. J. Herrgård, "Connecting extracellular metabolomic measurements to intracellular flux states in yeast.," *BMC Syst. Biol.*, vol. 3, p. 37, jan 2009.

[19] R. A. Weusthuis, W. Visser, J. T. Pronk, W. A. Scheffers, and J. P. van Dijken, “Effects of oxygen limitation on sugar metabolism in yeasts: a continuous-culture study of the Kluyver effect,” *Microbiology*, vol. 140, pp. 703–715, apr 1994.

[20] J. D. Orth, T. M. Conrad, J. Na, J. A. Lerman, H. Nam, A. M. Feist, and B. Ø. Palsson, “A comprehensive genome-scale reconstruction of Escherichia coli metabolism–2011.,” *Molecular systems biology*, vol. 7, p. 535, jan 2011.

[21] N. Ishii, K. Nakahigashi, T. Baba, M. Robert, T. Soga, A. Kanai, T. Hirasawa, M. Naba, K. Hirai, A. Hoque, P. Y. Ho, Y. Kakazu, K. Sugawara, S. Igarashi, S. Harada, T. Masuda, N. Sugiyama, T. Togashi, M. Hasegawa, Y. Takai, K. Yugi, K. Arakawa, N. Iwata, Y. Toya, Y. Nakayama, T. Nishioka, K. Shimizu, H. Mori, and M. Tomita, “Multiple high-throughput analyses monitor the response of E. coli to perturbations.,” *Science (New York, N.Y.)*, vol. 316, pp. 593–7, apr 2007.

[22] J. E. Peters, T. E. Thate, and N. L. Craig, “Definition of the Escherichia coli MC4100 genome by use of a DNA array.,” *Journal of bacteriology*, vol. 185, pp. 2017–21, mar 2003.

[23] M. R. de Graef, S. Alexeeva, J. L. Snoep, and M. J. Teixeira de Mattos, “The steady-state internal redox state (NADH/NAD) reflects the external redox state and is correlated with catabolic adaptation in Escherichia coli.,” *Journal of bacteriology*, vol. 181, pp. 2351–7, apr 1999.