gEFM: An Algorithm for Computing Elementary Flux Modes Using Graph Traversal

Ehsan Ullah, Shuchin Aeron, and Soha Hassoun

Abstract—Computational methods to engineer cellular metabolism promise to play a critical role in producing pharmaceutical, repairing defective genes, destroying cancer cells, and generating biofuels. Elementary Flux Mode (EFM) analysis is one such powerful technique that has elucidated cell growth and regulation, predicted product yield, and analyzed network robustness. EFM analysis, however, is a computationally daunting task because it requires the enumeration of all independent and stoichiometrically balanced pathways within a cellular network. We present in this paper an EFM enumeration algorithm, termed graphical EFM or gEFM. The algorithm is based on graph traversal, an approach previously assumed unsuitable for enumerating EFMs. The approach is derived from a pathway synthesis method proposed by Mavrovouniotis et al. The algorithm is described and proved correct. We apply gEFM to several networks and report runtimes in comparison with other EFM computation tools. We show how gEFM benefits from network compression. Like other EFM computational techniques, gEFM is sensitive to constraint ordering; however, we are able to demonstrate that knowledge of the underlying network structure leads to better constraint ordering. gEFM is shown to be competitive with state-of-the-art EFM computational techniques for several networks, but less so for networks with a larger number of EFMs.

Index Terms—Pathway analysis, network analysis, graph algorithms, metabolic networks, biochemical networks, elementary flux modes, elementary modes, flux modes, EFMs

1 Introduction

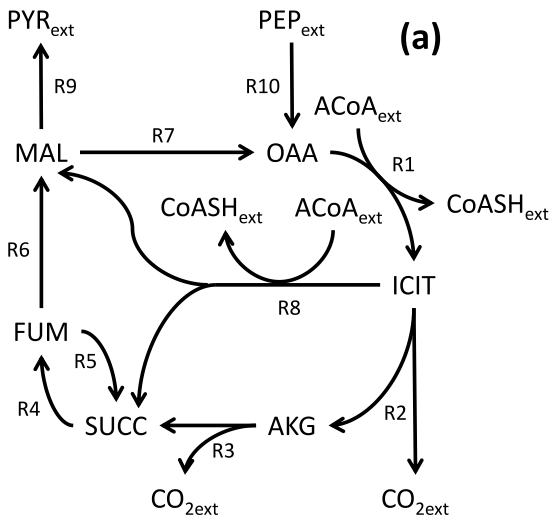
The continued success of designing many industrially relevant micro-organisms [1] and developing synthetic biology applications [2] requires efficient computational tools for modeling, analysis, and design optimization [3]. One particular powerful computational technique for analyzing cellular metabolism is elementary flux mode (EFM) analysis. A "flux mode" represents a steady-state flux pattern where the proportions of fluxes are fixed while their absolute magnitudes are indeterminate [4]. EFM analysis decomposes a metabolic network into routes that have three properties: *thermodynamic feasibility*, *quasi steady-state operation* and *independence* of other pathways [4]. Thermodynamic feasibility imposes that each irreversible reaction proceeds to have a non-negative flux (turnover) rate. Quasi steady-state operation ensures that metabolites internal to the network are neither accumulated nor depleted. Mutual independence of other pathways, together with the other two properties, guarantees that the EFM decomposition is unique. Several applications have benefited from EFM analysis. Example applications include validation of metabolic model construction [5], analyzing and understanding metabolic network including robustness and cellular regulation [6], [7], [8], [9], [10], analyzing competitive microbial strategies [11], increasing product yield [12], [13], and assessing plant fitness and agricultural productivity [14].

- *E. Ullah and S. Hassoun are with the Department of Computer Science, Tufts University, Medford, MA 02421. E-mail: {ehsan.ullah, soha.hassoun}@tufts.edu.*
- *S. Aeron is with the Department of Electrical and Computer Engineering, Tufts University, Medford, MA 02421. E-mail: shuchin@ece.tufts.edu.*

Manuscript received 11 Oct. 2014; revised 7 Mar. 2015; accepted 9 Apr. 2015. Date of publication 6 May 2015; date of current version 3 Feb. 2016.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TCBB.2015.2430344

Computing EFMs has been shown equivalent to computing the extreme rays of a convex pointed cone [15]. More precisely, once each reversible reaction is split into a forward and a reverse reaction, the steady-state operation and irreversibility constraints define a pointed convex cone that lies in the positive quadrant of the space defined by the network reactions. Any steady-state flux vector for the network lies within this convex cone and can be expressed as a non-negative linear combination of the extreme rays (edges) of the cone. Algorithms for computing the generating vectors of a convex polyhedral cone can be utilized to compute the elementary modes. To compute EFMs, Schuster and Hilgetag [4] applied one such algorithm [16] where rows of the transposed stoichiometric matrix augmented by the identity matrix are combined pairwise to generate the elementary modes. This method was later elaborated by adding a dependency test criterion to eliminate redundant modes [17]. The earlier Schuster and Hilgetag approach [4] is referred to as the 'Canonical Basis' approach [15]. Wagner [18] and Urbanczik and Wagner [19] proposed to first derive the basis vectors of the null space of all steady-state conditions, and then calculate the elementary modes by linearly combining the basis vectors. This algorithm is referred to as the "Null Space" approach, and provides significant speed up over the Canonical Basis approach [19]. Gagneur and Klamt [15] showed that the two approaches, the Canonical Basis and the Null Space, are variants of the double-description method, used to enumerate the extreme rays of a convex cone (see [20], [21], [22], [23] for a description of this method). The two most widely used EFM tools, Metatool [24] and EFMTool [25], are based on the Null-Space approach.

We revisit in this paper the Canonical Basis approach as described by [17], but from a graph-traversal perspective. The basic underlying idea is from the works of Mavrovouniotis et al. on the synthesis of metabolic pathways from a given



(b)

	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10
OAA	-1	0	0	0	0	0	1	0	0	1
ICIT	1	-1	0	0	0	0	0	-1	0	0
AKG	0	1	-1	0	0	0	0	0	0	0
SUCC	0	0	1	-1	1	0	0	1	0	0
FUM	0	0	0	1	-1	-1	0	0	0	0
MAL	0	0	0	0	0	1	-1	1	-1	0

Fig. 1. Example network. (a) Network graph. (b) Stoichiometric matrix.

substrate(s) to a given product(s) [26], [27], [28]. Conceptually, the Mavrovouniotis approach iteratively incorporates the set of stoichiometric constraints associated with each metabolite, and transforms an initial set of reactions (one-step pathways) into a final set of pathways that satisfy all the constraints. Schuster et al. [17] argued that row elimination within the Canonical Basis approach is equivalent to the Mavrovouniotis approach; however, it was suggested that using matrix formalism instead of graph theory is more elegant when tackling structural analysis of metabolic networks.

We present in this paper an algorithm, gEFM, that relies on graph traversal to compute the EFMs. The algorithm combines the Mavrovouniotis pathway synthesis approach [26] with the dependency test identified by Schuster and Hilgetag [17]. An earlier version of gEFM was presented [29] without a correctness proof nor the implementation details, and the preliminary results differ significantly from the ones presented in this paper. The runtimes reported here are different from those reported earlier [29] for two reasons. First, gEFM was updated with the rank condition mentioned in Section 2.4.1. Second, here we utilize a faster computing platform with larger memory (2.3 GHz AMD Opteron 6176 CPU with a 512 KB for the conference paper versus 2.83 GHz Intel Xeon E5440 CPU with a 12 MB cache). The main contribution of this paper is showing that graph-based approaches are viable for computing Elementary Flux Modes. Naturally, this contribution extends to enumerating rays of a convex cone. Because it retains the network structure, gEFM is accessible and intuitive. Our results show that the runtime of gEFM compares well or exceeds that of comparable tools, namely Metatool [24] and EFMTool [25], for a number of networks. We also examine the impact of network compression and constraint ordering, and show that there is a preferable constraint ordering based on the analysis of the underlying network structure that benefits gEFM.

2 Methods

2.1 Definitions

Before presenting the details of the gEFM algorithm, we provide some definitions to clarify the exposition. Boldface capital letters denote matrices. Boldface lower case letters denote column vectors. The ith entry of a vector, $\mathbf{p}$, is referred to using the notation, $\mathbf{p}[i]$.

Definition 1. *An $m \times n$ stoichiometric matrix $\mathbf{S}$ represents the structure of a biochemical network, where m is the number of metabolites internal to the network and n is the number of reactions.*

The network reactions include exchange reactions that connect the network to a set of external metabolites not captured in the stoichiometric matrix. The i,jth entry of the matrix $\mathbf{S}$, denoted by $\mathbf{S}_{ij}$, is negative (positive) if metabolite m_i is a reactant (product) participating in reaction r_j. A zero entry $\mathbf{S}_{ij}$ indicates that metabolite m_i does not participate in reaction r_j. Each reversible reaction is split into a forward reaction and a reverse reaction. The pair will be referred to as an *irreversible reaction pair*, or an *irreversible pair*.

Metabolites in the network are classified as either external or internal [30]. Internal metabolites do not accumulate when the network is operating under steady-state conditions; however, external metabolites, sometimes referred to as pool metabolites or sources/sinks, can accumulate. The next definition formalizes metabolite production/consumption under steady-state conditions.

Definition 2. *A biochemical network is at steady state if the net production rate equals the net consumption rate for each metabolite internal to the network.*

A network can be represented using a hypergraph, G, where $\mathbf{S}$ represents the incidence matrix for G. G is specified by a set of vertices and a set of edges. A vertex corresponds to a metabolite, and an edge corresponds to a reaction. An edge in the graph may be a hyperedge (e.g. reaction R1 in Fig. 1), with potentially multiple sources and multiple sinks. A path, or pathway, in the network is defined as a connected sequence of reactions such that products of a reaction are reactants of the next reaction(s) in the sequence. Assuming e number of external metabolites, the number of vertices in G equals $m + e$, and the number of edges equals n. The terms network, graph, and hypergraph are used interchangeably, and so are the terms reaction, edge, and hyperedge.

An example metabolic network representing a subsection of the TCA cycle and where reactions are assumed

irreversible operating in one direction is illustrated in Fig. 1a. The network is assumed to have external metabolites PYR, PEP, ACoA, CoASH, and CO_2, thus the "ext" subscript in Fig. 1a, while all others are treated as internals. Fig. 1b shows the equivalent stoichiometric matrix.

A pathway can be represented in two possible ways. One representation uses a vector of reaction coefficients, $\mathbf{p} \in \mathbb{R}^n$. Each coefficient represents the reaction relative turnover rate. A pathway can also be represented using a vector of *binary values*, $\mathbf{b} \in \{0,1\}^n$, indicating reaction participation or lack thereof, along with a vector of *metabolite coefficients*, $\mathbf{c} \in \mathbb{R}^m$, representing the net metabolite balance. A positive $\mathbf{c}[i]$ coefficient indicates a net production of metabolite i; a negative value indicates a net consumption. A zero value indicates a balance between production and consumption, and the relevant metabolite is referred to as a *balanced metabolite*. The values of $\mathbf{b}$ are derived from the reaction coefficients as follows:

$$\mathbf{b}[i] = \begin{cases} 1 & \text{if } \mathbf{p}[i] > 0, \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

For example, a pathway involving R10, R1, R8, R4, R6, and R9 in Fig. 1a operating at steady-state has zero metabolite coefficients for the pertinent internal metabolites, and can be represented as:

$$\mathbf{p} = [1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 2 \quad 1]^T,$$
$$\mathbf{b} = [1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 1]^T.$$

Metabolite and reaction coefficients are related as follows:

$$\mathbf{c} = \mathbf{S}\mathbf{p}. \quad (2)$$

In the gEFM algorithm, the metabolite coefficients values are used to identify pathways that produce or consume a particular metabolite. Reaction coefficients needed to specify the EFMs are computed from the binary coefficients.

Definition 3. *A balanced pathway* $\mathbf{p}$ *induces a steady-state condition on a network* $\mathbf{S}$ *iff*

$$\mathbf{S}\mathbf{p} = \mathbf{0}. \quad (3)$$

Therefore, all internal metabolite coefficients along a balanced pathway must be zero.

Definition 4. *A pathway is decomposable or dependent if it can be represented as a non-negative linear combination of other pathways.*

A pathway $\mathbf{p_1}$ will be dependent on pathway $\mathbf{p_2}$ if it can be expressed as non-negative linear combination of $\mathbf{p_2}$ and other pathway(s). The independence of two pathways can be readily derived from their binary representation using *bitwise and* operation [23], [15].

Lemma 1. *Given two pathways* $\mathbf{p}$ *and* $\mathbf{p}'$*, with binary representations* $\mathbf{b}$ *and* $\mathbf{b}'$ *respectively,* $\mathbf{p}'$ *is dependent on* $\mathbf{p}$ *iff:*

$$\mathbf{b}\ AND\ \mathbf{b}' = \mathbf{b}. \quad (4)$$

For example, the independence of pathway $\mathbf{p_a}$ consisting of R1, R2, R3, R4, R6, and R7 and pathway $\mathbf{p_b}$ consisting of R1, R2, R3, R4, R6, R7, R8, and R9 in Fig. 1a can be verified by comparing their binary representation using Lemma 1. Here, the active reactions in $\mathbf{p_a}$ are a subset of the active reactions in $\mathbf{p_b}$, making $\mathbf{p_b}$ dependent on $\mathbf{p_a}$, and $\mathbf{p_a}$ independent of $\mathbf{p_b}$.

An *elementary flux mode*, or *flux mode*, or *elementary mode* is a steady-state flux pattern in which flux proportions are fixed while their absolute magnitudes are indeterminate [31]. A formal definition of a flux mode is provided below.

Definition 5. *Given an* $m \times n$ *stoichiometric matrix,* $\mathbf{S}$*, three conditions must be met to label a pathway* $\mathbf{p}$ *an* elementary flux mode*:*

- **C1:** The network reactions proceed in a direction dictated by thermodynamic feasibility. Each reaction coefficient in $\mathbf{p}$ must be non-negative.
- **C2:** The network is in quasi steady-state condition with no accumulation of internal metabolites in the network. Mathematically, $\mathbf{S}\mathbf{p} = \mathbf{0}$.
- **C3:** Each elementary mode must be independent from any other elementary mode in the network.

A vector $\mathbf{p}$ is thus an EFM if and only if $\mathbf{p}$ is thermodynamically feasible, satisfies quasi-steady state conditions, and there is no other non-null flux vector (up to a scaling) that satisfies both **C1** and **C2** and involves a proper subset of the reactions participating in $\mathbf{p}$.

The elementary flux modes for the example in Fig. 1 as shown in Fig. 2 are:

$$\mathbf{EFM}_1 = [1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0]^T,$$
$$\mathbf{EFM}_2 = [1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1]^T,$$
$$\mathbf{EFM}_3 = [1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 0]^T,$$
$$\mathbf{EFM}_4 = [1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 2 \quad 1]^T,$$
$$\mathbf{EFM}_5 = [0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]^T.$$

Each pathway listed above is an EFM because all reaction directions are consistent with thermodynamic feasibility as specified in the original network. Additionally, each pathway is balanced, where each metabolite can be produced and consumed without net accumulation specified by the mass-balance constraints in $\mathbf{S}$. Finally, each of the EFMs is independent of all others, as specified by the test in Lemma 1.

The benefit of the EFM decomposition is that any steady-state flux distribution in the network can be represented as a non-negative linear combination of EFMs. For example, a flux distribution of $\mathbf{p} = [3 \quad 1 \quad 1 \quad 3 \quad 0 \quad 3 \quad 3 \quad 2 \quad 2 \quad 0]^T$ can be written as the linear combination of EFM1 and EFM3, weighted by 1 and 2, respectively.

2.2 The gEFM Algorithm

The gEFM algorithm is an iterative algorithm that processes one internal metabolite at a time to construct partially balanced pathways. Each partially balanced pathway, referred to as a *partial pathway*, is balanced with respect to processed metabolites, but not necessarily balanced with respect to unprocessed internal metabolites. The pseudo code of the algorithm is presented in Algorithm 1. Initially, gEFM treats each reaction in the network as a partial pathway (line 1). When a metabolite v is selected (line 3) and processed (lines 4-9), new partial pathways are constructed by combining each partial pathway in $inputs$ that produces v with each partial pathway in $outputs$ that consumes v, and generating a set of

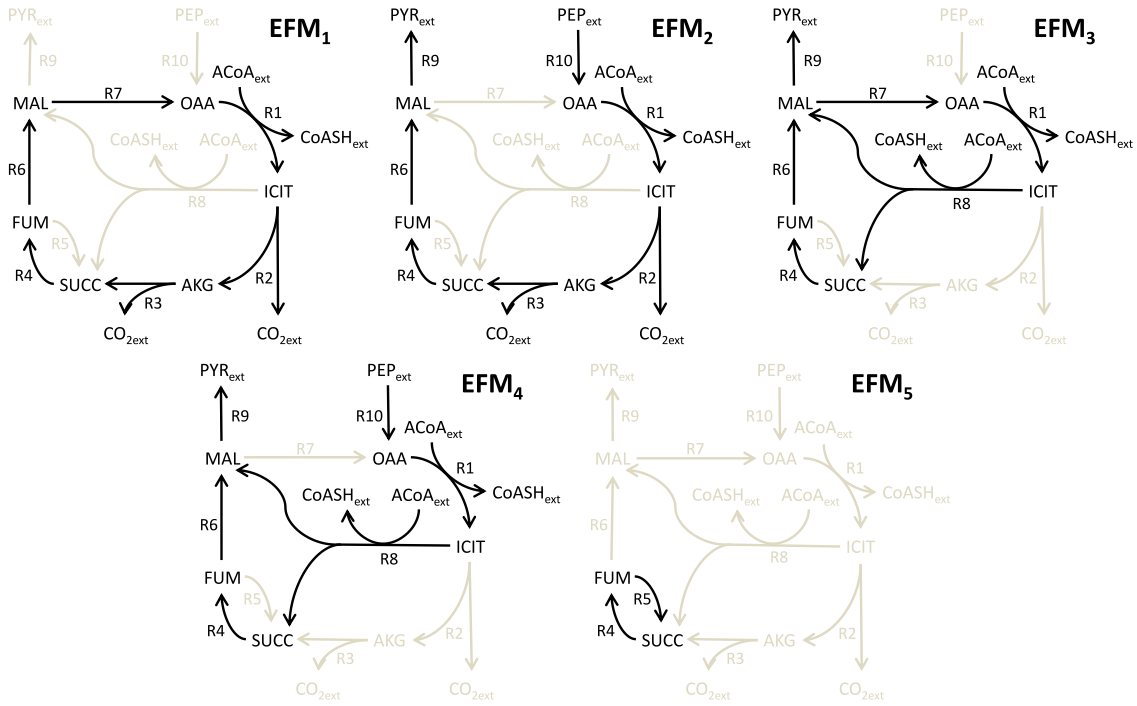


Fig. 2. EFMs of the network in Fig. 1a.

candidate pathways (line 7). Partial pathways producing or consuming v are removed from `allPathways` and the remaining pathways with no net production or consumption of v, referred to as non-participating pathways, are stored in `nonParticipatingPathways` (line 6). Candidate dependent pathways, including those containing one or more irreversible reaction pair, are identified and discarded (line 8). Dependency checking is described in detail in Section 2.4.1. `allPathways` is updated as the union of `nonParticipatingPathways` and `candPathways` (line 9). The process repeats until all internal metabolites are processed. The remaining pathways are all EFMs. In the algorithm, the superscript k refers to the iteration number. Without loss of generality, we assume that k is the index of the internal metabolite selected at step k. The details of the algorithm can be found in [29].

Algorithm 1. gEFM Pseudocode

1 $\text{allPathways}^{(0)} \leftarrow$ All reactions in the network;
2 **for** $k = 1$ *to* m **do**
3 $\quad \text{v}^{(k)} \leftarrow$ Unbalanced internal metabolite at index k;
4 $\quad \text{inputs}^{(k)} \leftarrow$ All pathways in $\text{allPathways}^{(k-1)}$ for which $\text{v}^{(k)}$ is a reactant;
5 $\quad \text{outputs}^{(k)} \leftarrow$ All pathways in $\text{allPathways}^{(k-1)}$ for which $\text{v}^{(k)}$ is a product;
6 $\quad \text{nonParticipatingPathways}^{(k)} \leftarrow \text{allPathways}^{(k-1)} \setminus (\text{inputs}^{(k)} \cup \text{outputs}^{(k)})$;
7 $\quad \text{candPathways}^{(k)} \leftarrow \text{inputs}^{(k)} \times \text{outputs}^{(k)}$;
8 $\quad$ Remove dependant pathways from $\text{candPathways}^{(k)}$;
9 $\quad \text{allPathways}^{(k)} \leftarrow \text{nonParticipatingPathways}^{(k)} \cup \text{candPathways}^{(k)}$;
10 **end**
11 Compute reaction coefficients for each pathway in $\text{allPathways}^{(m)}$;

2.3 Correctness of the gEFM Algorithm

In this section, definitions are introduced, and several Lemmas are presented to prove the correctness of gEFM in constructing pathways that meet the three conditions in Definition 5.

Definition 6. *A stoichiometric matrix* $\mathbf{S}^{(k)}$ *is an* $k \times n$ *submatrix of* $\mathbf{S}$ *that includes the first* k *rows (metabolites) of* $\mathbf{S}$.

Metabolites that are not represented in $\mathbf{S}^{(k)}$ are considered external with respect to $\mathbf{S}^{(k)}$. The matrix $\mathbf{S}^{(\mathbf{0})}$ is a zero-row matrix representing the network without any internal metabolites.

The gEFM algorithm constructs partially balanced pathways with respect to $\mathbf{S}^{(k)}$. That is, each metabolite along a pathway $\mathbf{p}$ will have a zero metabolite coefficient. More formally,

Definition 7. *A partially balanced pathway (or partial pathway)* $\mathbf{p}^{(k)}$ *is balanced with respect to the first* k *metabolites in* $\mathbf{S}$.

We now present some Lemmas that argue the correctness of gEFM with respect to the construction incremental in the number of metabolites.

Lemma 2. *Partial pathways generated in every iteration of gEFM satisfy condition C1 in Definition 5.*

Proof. Initially, prior to the first iteration of the algorithm, all reactions operate in their specified direction. By construction, a partial pathway producing a metabolite is combined with a partial pathway that consumes the metabolite. The direction of all the reactions along the new partial pathways is consistent with earlier construction steps. Therefore, all partial pathways constructed at each step of the algorithm, and overall, satisfy condition C1 in Definition 5. □

Lemma 3. *Partial pathways generated in iteration k of gEFM satisfy condition C2 in Definition 5 for the network specified by $\mathbf{S}^{(k)}$.*

Proof. For $\mathbf{S}^{(\mathbf{0})}$, all the network reactions represent partial pathways because no metabolite is considered internal and each reaction is stoichiometrically balanced. In each iteration k of the algorithm, a new metabolite $\mathrm{v}^{(k)}$ is balanced resulting in a new partial pathway $\mathbf{p}^{(k)}$. This is accomplished by combining two balanced partial pathways in $\mathbf{S}^{(k-1)}$. During step k of the algorithm, the metabolites balanced previously during the first $k-1$ iterations remain balanced as their coefficients are already zero. The following thus holds:

$$\mathbf{S}^{(k)}\mathbf{p}^{(k)} = \mathbf{0}.$$

Therefore, condition C2 of Definition 5 for $\mathbf{S}^{(k)}$ is satisfied. □

Lemma 4. *The set of partial pathways generated after every iteration of gEFM contains only independent partial pathways that satisfy C3 in Definition 5 for the network specified by $\mathbf{S}^{(k)}$.*

Proof. At the end of each iteration, $\mathtt{allPathways}^{(\mathtt{k})}$ is computed as the union of pathways in $\mathtt{nonParticipatingPathways}^{(\mathtt{k})}$ and pathways in $\mathtt{candPathways}^{(\mathtt{k})}$. To show that pathways in $\mathtt{allPathways}^{(\mathtt{k})}$ are independent partial pathways, we show that each set of this union has only independent pathways, and that pathways within one set are independent of those in the other set.

Initially, $\mathtt{allPathways}^{(0)}$ contains all reactions in the network, which are independent of each other. By induction, we can assume that $\mathtt{allPathways}^{(\mathtt{k}-1)}$ contains only independent pathways. During each iteration, each partial pathway that has a reaction producing or consuming $\mathrm{v}^{(\mathtt{k})}$ is removed from $\mathtt{allPathways}^{(\mathtt{k})}$, and the remaining non-participating pathways are stored in $\mathtt{nonParticipatingPathways}^{(\mathtt{k})}$. Pathways in $\mathtt{nonParticipatingPathways}^{(\mathtt{k})}$ are independent from each other as they are a subset of the pathways in $\mathtt{allPathways}^{(\mathtt{k}-1)}$. Because each non-participating partial pathway has no reactions producing or consuming $\mathrm{v}^{(\mathtt{k})}$, such pathways are independent of pathways in $\mathtt{inputs}^{(\mathtt{k})}$ and $\mathtt{outputs}^{(\mathtt{k})}$, as per Lemma 1. Further, pathways in $\mathtt{nonParticipatingPathways}^{(\mathtt{k})}$ are independent of the pathways formed by combining each pathway in $\mathtt{inputs}^{(\mathtt{k})}$ and each pathway in $\mathtt{outputs}^{(\mathtt{k})}$. That is, pathways in $\mathtt{nonParticipatingPathways}^{(\mathtt{k})}$ are independent of pathways in $\mathtt{candPathways}^{(\mathtt{k})}$.

Because each pathway in $\mathtt{candPathways}^{(\mathtt{k})}$ is compared for dependency against all other pathways in $\mathtt{candPathways}^{(\mathtt{k})}$ and pathways in $\mathtt{nonParticipatingPathways}^{(\mathtt{k})}$, the resulting pathways in $\mathtt{candPathways}^{(\mathtt{k})}$ are independent from each other and those in $\mathtt{nonParticipatingPathways}^{(\mathtt{k})}$. □

Lemma 5. *gEFM produces all EFMs for the network defined by $\mathbf{S}^{(k)}$.*

Proof. At each iteration of the gEFM algorithm, all possible input/output partial pathway combinations are explored, ensuring that all possible ways of balancing a metabolite $\mathrm{v}^{(k)}$ are considered. Combined with Lemmas 2, 3, and 4, all EFMs for $\mathbf{S}^{(k)}$ are generated. □

Theorem 1. *The gEFM algorithm generates all EFMs.*

Proof. In every iteration of gEFM an internal metabolite is balanced. When all the internal metabolites are balanced, gEFM terminates and the following holds:

$$\mathbf{S} \equiv \mathbf{S}^{(m)}. \tag{5}$$

All partial pathways generated after the last iteration satisfy C1 and C2 based on Lemmas 2 and a 3, respectively. Since the set of generated pathways contains all the EFMs (Lemma 5) and all the dependent pathways are removed (Lemma 4), C3 is satisfied. Therefore, the set $\mathtt{allPathways}^{(m)}$ only contains EFMs. □

2.4 Implementation Details

2.4.1 Dependency Checking

Dependency checking is a fundamental and computationally expensive operation when generating EFMs. To ensure independence, all partial pathways generated by the algorithm must be tested for dependency against each other and against non-participating pathways. However, the implementation can be made more efficient by discarding any partial pathway with length larger than **rank S** [32]. Another method of speeding the implementation is to compare the newly generated partial pathway against the generating input and output partial pathways instead of other generated partial pathways. The details of the comparison follow Lemma 7.

Lemma 6. *Pathway dependency is transitive. If $\mathbf{p}_1$ is dependent on $\mathbf{p}_2$ and $\mathbf{p}_2$ is dependent on $\mathbf{p}_3$, then $\mathbf{p}_1$ is dependent on $\mathbf{p}_3$.*

Proof. Consider three pathways $\mathbf{p}_1$, $\mathbf{p}_2$ and $\mathbf{p}_3$ with binary representations $\mathbf{b}_1$, $\mathbf{b}_2$ and $\mathbf{b}_3$. Let $\mathbf{p}_1$ be dependent on $\mathbf{p}_2$, and $\mathbf{p}_2$ is dependent on $\mathbf{p}_3$. Using Lemma 1,

$$\begin{aligned}\mathbf{b}_1 \; AND \; \mathbf{b}_2 &= \mathbf{b}_2,\\ \mathbf{b}_2 \; AND \; \mathbf{b}_3 &= \mathbf{b}_3.\end{aligned}$$

Consider the dependency test of $\mathbf{p}_1$ and $\mathbf{p}_3$:

$$\begin{aligned}\mathbf{b}_1 \; AND \; \mathbf{b}_3 &= \mathbf{b}_1 \; AND \; (\mathbf{b}_2 \; AND \; \mathbf{b}_3)\\ &= (\mathbf{b}_1 \; AND \; \mathbf{b}_2) \; AND \; \mathbf{b}_3\\ &= \mathbf{b}_2 \; AND \; \mathbf{b}_3\\ &= \mathbf{b}_3.\end{aligned}$$

The above derivation shows that pathway $\mathbf{p}_1$ is dependent on pathway $\mathbf{p}_3$. □

Lemma 7. *If a pathway $\mathbf{p}$ is dependent on a pathway $\mathbf{p}_{combo}$, then pathway $\mathbf{p}$ is also dependent on the pathways $\mathbf{p}_{in}$ and $\mathbf{p}_{out}$ generating $\mathbf{p}_{combo}$.*

Proof. A pathway $\mathbf{p}_{combo}$ generated by combining two pathways $\mathbf{p}_{in}$ and $\mathbf{p}_{out}$ is naturally dependent on $\mathbf{p}_{in}$ and $\mathbf{p}_{out}$.

Since pathway dependency is transitive (Lemma 6), a pathway $\mathbf{p}$ is dependent on a pathway $\mathbf{p}_{combo}$, the pathway $\mathbf{p}$ is also dependent on the pathway $\mathbf{p}_{in}$ and on the pathway $\mathbf{p}_{out}$. □

Consider the set of pathways $\texttt{candPathways}^{(k)}$ is generated by combining the set of input pathways $\texttt{inputs}^{(k)}$ and the set of output pathways $\texttt{outputs}^{(k)}$ in an iteration of gEFM. The dependency test is performed for each pathway $\mathbf{p} \in \texttt{candPathways}^{(k)}$, generated by $\mathbf{p}_{in} \in \texttt{inputs}^{(k)}$ and $\mathbf{p}_{out} \in \texttt{outputs}^{(k)}$, and all pathways in $\texttt{candPathways}^{(k)} \setminus \mathbf{p}$. Using Lemma 7, the same dependency analysis results are obtained by comparing pathway $\mathbf{p}$ with each pathway $\mathbf{p}'_{in} \in \texttt{inputs}^{(k)} \setminus \mathbf{p}_{in}$ and $\mathbf{p}'_{out} \in \texttt{outputs}^{(k)} \setminus \mathbf{p}_{out}$. Within gEFM, we used the bit pattern trees [33] data structure to implement pathway dependency checking.

2.4.2 Reversible Reaction Trees

The splitting of reversible reactions into forward and backward reactions results in additional cyclical EFMs, each composed of an irreversible reaction pair. As each reaction within the split pair is treated independently during the algorithm execution, it is possible that gEFM identifies pathways containing one or more irreversible reaction pairs. Such pathways are dependent on the respective two-futile cycles [15]. The algorithm runtime benefits by rejecting such pathways as early as possible during the dependency checking step (line 8 of the Algorithm). We utilize an additional data structure, *reversible reaction trees*, to facilitate the rejection of such paths. Reversible reaction trees are similar to bit pattern trees [33]. In this section, we first explain the structure and construction of reversible trees, and then explain how they can be used to avoid the generation of spurious pathways.

A reversible tree is a binary tree representing a set of pathways. The tree is constructed by recursively splitting the set of pathways P based on a reaction r in F, the set of reactions stemming from splitting reversible reactions. Reaction r is used to split the pathways into two sets. All pathways for which the reaction is present are stored in right subtree, while the rest of the pathways are stored in the left subtree. Intermediate nodes in the tree correspond to a reaction in F and leaf nodes contain subsets of pathways in P. A balanced tree will have a higher traversal efficiency compared to a non-balanced tree.

Consider an example network with one reversible reaction R1 that is split into an irreversible reaction pair R1f and R1b. Consider a set of pathways P in the network for which a reversible reaction tree is to be constructed. F contains R1f and R1b. To build the tree, a reaction from F is selected to split the pathways. Assume R1f is selected first. All pathways containing R1f are stored in the right subtree, and all other pathways are stored in the left subtree. R1f is removed from F. The recursive construction process is repeated until F is empty. Fig. 3 shows the reversible tree for the network. Four subsets of pathways P_1, P_2, P_3 and P_4 are generated. Pathways in the subset P_4 can be discarded because each pathway in P_4 contains both R1f and R1b.

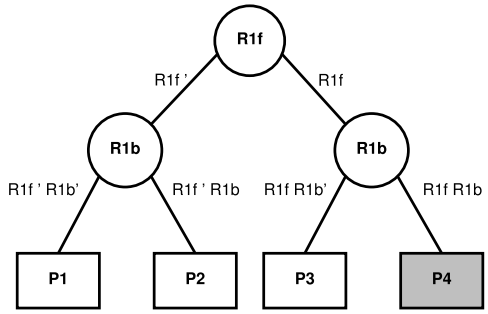


Fig. 3. Reversible tree for a network with a reversible reaction R1.

During each iteration of gEFM, a reversible tree in is built for the set of input pathways and a reversible tree out is built for the set of output pathways. New pathways are generated by recursively combining a set of pathways from the in tree, and a set of pathways from the out tree. All such combinations are considered except for the ones that result in pathways with one or more reversible reaction pairs. Consider the example in and out trees in Fig. 4. Each leaf node in in is combined with all leaf nodes of out except for the cases in which the resulting pathways will have R1f and R1b present. Combinations of sets in_2out_3 and in_3out_2 are not generated because both R1f and R1b will be present in the resulting combinations.

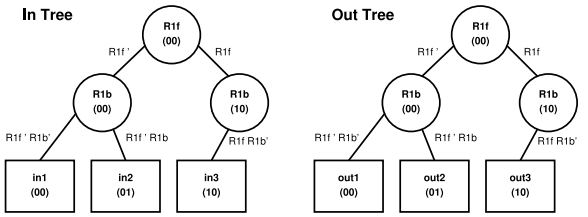
The process of recursive combination can be further improved by using a label at each intermediate node in the reversible tree. The label consists of a bit vector with each bit corresponding to a reaction in F. A bit value of '1' corresponding to a reaction indicates all the pathways in the subtrees will have that reaction. A bit value of '0' corresponding to a reaction indicates that all pathways may or may not have the reaction. In Fig. 4, the labels corresponding to each node for the above example are shown in parenthesis.

The recursive combination procedure is shown in Algorithm 2. In each recursive call, the label for the combination $comboLabel$ is calculated by computing the bit-wise logical OR of the labels of input and output tree nodes. If $comboLabel$ contains '1' for both forward and reverse reactions in a two-futile cycle, the recursion is stopped. If neither in nor out trees are leaves, then the procedure is called recursively on all possible input-output combinations. If either in or out trees are leaves, then the recursive procedure is called judiciously.

3 Results

3.1 Test Cases

To assess the performance of gEFM and compare with other tools, we selected several biochemical networks with a varying number of reactions and metabolites. The smaller test cases were culled directly from the literature. The larger test cases were culled from compartments within published genome-scale models. To ensure enumeration of all EFMs, the larger models were further limited in complexity by removing the cofactors, highly connected metabolites whose presence significantly increases the number of EFMs. Additionally, one of the models, *Escherichia coli* (*E. coli*), was



List of all possible combinations of input and output pathways

Input (Label)	Output (Label)	Combination (Label)	Valid
in1 (00)	out1 (00)	in1out1 (00)	Yes
in1 (00)	out2 (01)	in1out2 (01)	Yes
in1 (00)	out3 (10)	in1out3 (10)	Yes
in2 (01)	out1 (00)	in2out1 (01)	Yes
in2 (01)	out2 (01)	in2out2 (01)	Yes
in2 (01)	out3 (10)	in2out3 (11)	No
in3 (10)	out1 (00)	in3out1 (10)	Yes
in3 (10)	out2 (01)	in3out2 (11)	No
in3 (10)	out3 (10)	in3out3 (10)	Yes

Fig. 4. Generation of pathways using reversible reaction trees for a network with one reversible reaction R1.

modified by restricting the directionality of the reactions to be irreversible and by allowing the cell to only grow on glucose to generate two additional test cases. These two modifications allowed the examination of the impact of reducing the effective number of reactions on the number of EFMs and various performance metrics. Collectively, the test cases provide comparison points of gEFM against existing tools, and provide insights into gEFM advantages and limitations.

Algorithm 2. Recursive Generation of Combinations Using Reversible Trees

```
1  void generateCombinations (RevTree in, RevTree out);
2  begin
3    comboLabel ← in.label OR out.label;
4    if comboLabel is valid then
5      if (in is not leaf )and(out is not leaf )then
6        generateCombinations (in.left, out.left);
7        generateCombinations (in.left, out.right);
8        generateCombinations (in.right, out.left);
9        generateCombinations (in.right, out.right);
10     end
11     else if (in is not leaf) and (out is leaf) then
12       generateCombinations (in.left, out);
13       generateCombinations (in.right, out);
14     end
15     else if (in is leaf) and (out is not leaf) then
16       generateCombinations (in, out.left);
17       generateCombinations (in, out.right);
18     end
19     else
20       for each pathway i in in.pathwaysdo
21         for each pathway o in out.pathwaysdo
22           generate combination of i and o
23         end
24       end
25     end
26   end
27 end
```

The first network represents adipocyte central carbon metabolism, and was used for flux profiling and modularity analysis [34]. The second network is a reduced model capturing central carbon metabolism [35] of the Chinese Hamster Ovarian (CHO) cell [36]. The next network is a model of *E. coli* that was utilized when engineering a minimal *E. coli* cell for the efficient production of ethanol from hexoses and pentoses [13]. In *E. coli*(irrev), all reactions in the network are made irreversible by forcing reversible reactions to operate only in the forward direction, as specified by the default reaction listing. In *E. coli*(gluc), glucose is considered as the only carbon source for the production of ethanol. The next two cases were models of a human gastric pathogen, *Helicobacter pylori* (*H. pylori*) [37], and of *Saccharomyces cerevisiae* (*S. cerevisiae*) iND750 [38]. For each of these test cases, we considered only the cytosol compartment. Cofactors, including ATP, ADP, NAD, NADP, NADH, NADPH, and AMP, were removed along with non-organic compounds including O_2, sodium, ammonia, and nitrate. The last model represents primary metabolism in *Chlamydomonas reinhardtii* (*C. reinhardtii*) [39], a single celled green alga. Reactions in mitochondria are considered with phosphate and water removed.

Several compression methods provided by EFMTool [40] were utilized to minimize the size of the test cases. The dead-end metabolite (DEM) removal method eliminates internal metabolites that are either only produced or only consumed. Reactions associated with such metabolites are also eliminated. The coupled-zero compression method removes all reactions that always carry zero flux at steady state. The coupled-contradicting compression method removes negatively coupled reactions (CR). The unique-flows compression method removes metabolites that are produced by only one reaction and consumed by only one other reaction by combining the producing and consuming reactions. The coupled-combine compression method removes all flux-coupled reactions, ones for which their relative flux is always constant, except one representative reaction. Flux values for reactions removed using these compression techniques are computed based on flux values of the retained reactions.

Table 1 reports test case statistics for both uncompressed and compressed models. The left most column lists the test case name. The second column labeled "Uncompressed" reports the number of metabolites and number of reactions, with the number of reversible reactions shown in parenthesis. The next six columns report the network size for the compressed models in the following order: when all compression techniques are utilized, and when each of the following individual five compression techniques are applied: dead-end, coupled-zero, coupled-contradicting, unique-flow, and coupled-combine. The techniques were applied in the order listed in the table. Compression reduces the network size considerably. The resulting number of reactions and metabolites were identical when applying all compression techniques and when applying the unique-flow reductions; however, the resulting network topologies, and thus the **S** matrices, were different. In the last column, the table lists the number of EFMs for each test case. As there is a one-to-one correspondence between pathways before and after compression, the number of EFMs is the same in the uncompressed and compressed networks.

While Table 1 shows the results of compressing the test cases, Table 2 shows the inconsistencies present in the uncompressed models. These results were obtained using

TABLE 1
Statistics for the Test Case Networks

Test Case	Uncompressed		Compressed												EFMs
			All		Dead-end		Coupled-zero		Coupled-contradicting		Unique-flows		Coupled-combine		
	Mets	Rxns (Rev)	Mets	Rxns (Rev)	Mets	Rxns (Rev)	Mets	Rxns (Rev)	Mets	Rxns (Rev)	Mets	Rxns (Rev)	Mets	Rxns (Rev)	
Adipocyte	26	34 (0)	7	15 (0)	26	34 (0)	26	34 (0)	26	34 (0)	7	15 (0)	20	27 (0)	78
CHO	26	34 (10)	12	21 (9)	24	33 (10)	24	33 (10)	25	34 (10)	12	21 (9)	21	30 (9)	1,431
E. coli	52	70 (19)	26	44 (12)	52	70 (19)	52	70 (19)	52	70 (19)	26	44 (12)	32	50 (12)	429,276
E. coli (irrev)	52	70 (0)	12	26 (0)	50	66 (0)	51	68 (0)	51	67 (0)	12	26 (0)	30	47 (0)	840
E. coli (gluc)	47	60 (19)	26	39 (12)	47	60 (19)	47	60 (19)	47	60 (19)	26	39 (12)	31	44 (12)	33,220
H. pylori	287	413 (0)	23	140 (0)	162	275 (0)	203	323 (0)	287	413 (0)	23	140 (0)	287	413 (0)	753,664
S. cerevisiae	147	179 (19)	29	62 (19)	147	179 (19)	147	179 (19)	147	179 (19)	29	62 (19)	52	84 (19)	4,535,802
C. reinhardtii	39	121 (0)	28	110 (0)	38	120 (0)	38	120 (0)	39	121 (0)	28	110 (0)	32	114 (0)	4,152,658

MC^3 [41], a steady-state model and constraint consistency checker that uses the stoichiometric matrix and flux variability analysis to determine model inconsistencies. MC^3 identifies and lists all model inconsistencies in the original model related to the number of single-ended metabolites (SEM), dead-end metabolites, coupled reactions, inconsistent coupling (RCR), zero flux reactions (ZFR), and unsatisfied reversible reactions (UR). For example, per Table 2, MC^3 identifies six single-ended metabolites and eight dead-end metabolites in *H. Pylori* in the uncompressed model. As reported in Table 1 in the Dead-end column, compression reduces the number of metabolites to 162 (from the original 287 metabolites), and to 275 reactions (from the original 413 reactions). When we examined all inconsistencies identified by MC^3 in the uncompressed models, we found two unsatisfied reversible reactions, one for *E. coli* and one for *E. coli* (gluc), that were not removed in the compressed models. Overall, compression is effective in removing inconsistencies from the uncompressed models.

3.2 Computing Platform

We have bench-marked gEFM against Metatool [24] and EFMTool [25]. The MATLAB implementation of MetaTool 5.1 is used with MATLAB 2013. The Java implementation of EFMTool is used with Java runtime environment 1.6. Because gEFM does not currently have a multi-threaded implementation and to provide a fair comparison, we disabled multi-threading for all tools. We have performed all experiments on a 2.83 GHz Intel Xeon E5440 CPU with 6 MB cache running Red Hat Linux.

TABLE 2
Results of Running MC^3 on the Test Cases

Test Case	SEM	DEM	CR	RCR	ZFR	UR
Adipocyte	0	0	10	0	0	0
CHO	2	2	3	1	2	0
E. coli	0	0	13	0	0	4
E. coli (irrev)	1	2	32	1	17	0
E. coli (gluc)	0	0	9	0	0	4
H. pylori	6	8	40	43	14	0
S. cerevisiae	0	0	34	0	0	0
C. reinhardtii	1	1	7	0	1	0

MC^3 reports the number of single-ended metabolites, dead-end metabolites, coupled reactions, inconsistent coupling (RCR), zero flux reactions, and unsatisfied reversible reactions.

3.3 Runtime Analysis

The runtimes (in seconds) for the uncompressed and compressed models are reported in Table 3. The first column lists the model names. The next three columns list the runtime in seconds for all three tools for the uncompressed models, and the following three columns report the runtimes for the compressed models. When run times were less than 0.01 seconds, they were reported as < 0.01. Several entries are labeled as TO (timeout), where the computation did not complete within a given time frame. We utilized a different number of seconds for the timeouts depending on network size, consistently allowing for a timeout window of at least $2\times$ that of the fastest running tool.

We make the following observations regarding the results. Consistently, Metatool 5.1 computes the EFMs for the smaller examples for both the compressed and uncompressed models, but the larger examples do not complete as Metatool 5.1 crashes without reporting any errors. EFMTool and gEFM compute EFMs for all compressed and uncompressed test cases. gEFM outperforms EFMTool on the first six test cases except for *E. coli* compressed model, while EFMTool outperforms gEFM on the last two test cases. The network size of *H. pylori* is larger than the last two test cases (*S. cerevisiae* and *C. reinhardtii*), but *H. pylori* has a significantly smaller number of EFMs.

The difference in runtimes between gEFM and EFMTool is dependent on several factors. EFMTool is based on the Null Space approach whereas gEFM is based on the Canonical Basis approach. The number of algorithmic iterations in the Null Space approach is thus always less than or equal to that in the Canonical basis approach, as shown in Table 4. Because gEFM and EFMTool process differing (in number and type) constraints, the number of combinations generated and the number of comparisons performed during dependency checking vary for each technique. Moreover, the order of constraint processing impacts the number of resulting combinations and comparisons. These differences are explored in more detail in the following sections.

3.4 Comparisons Performed

To better understand how EFMtool and gEFM differ in runtimes, we quantify the cumulative amount of work performed by each tool. In Table 4, we list the number of iterations, the cumulative number of combinations generated, and the cumulative number of comparisons performed by each tool for the compressed and uncompressed models.

TABLE 3
Runtime Comparison for Metatool, EFMTool, and gEFM for Uncompressed and Compressed Models

Test Case	Uncompressed			Compressed		
	Metatool 5.1	EFMTool	gEFM	Metatool 5.1	EFMTool	gEFM
Adipocyte	0.11	0.07	< 0.01	0.09	0.05	< 0.01
CHO	3.50	0.50	0.04	3.17	0.38	0.04
E. coli	TO	14,062.05	1,636.58	TO	54.68	1,154.36
E. coli(irrev)	0.81	0.57	< 0.01	0.73	0.19	< 0.01
E. coli(gluc)	982.05	39.63	2.70	660.60	2.23	1.82
H. pylori	TO	5,138.54	1,717.17	TO	4,813.18	646.14
S. cerevisiae	TO	27,029.99	534,375.00	TO	1,094.89	22,553.10
C. reinhardtii	TO	153,132.05	471,316.00	TO	104,169.10	150,017.00

Runtime is reported in seconds. TO indicates timeout, where the computation did not complete in a time period of twice or more the time of the fastest running tool.

The last three columns list the numbers relative to gEFM. For gEFM, the number of iterations corresponds to the number of internal metabolites, m, that must be balanced. For EFMTool, the number of iterations corresponds to the number of constraints on the steady-state operation derived after computing the null-space kernel, and is equal to $n - k$ [15], where n is the number of reactions, and k is the size of the null-space kernel matrix (which is bounded by m).

Combinations correspond to pathways generated by balancing an internal metabolite in gEFM whereas they correspond to rays generated after processing a constraint in EFMTool. In each iteration of gEFM, combinations are compared to the set of (input, output and non-participating) pathways; therefore, the number of comparisons is equal to the product of the number of combinations and the number of pathways. Similarly in each iteration of EFMTool, the number of comparisons performed in each iteration is equal to the product of the number of combinations and the number of rays (positive, negative, and zero rays). The number of iterations and the number of combinations generated in each iteration were recorded and the number of comparisons was computed for each test case.

For both compressed and uncompressed models, gEFM has a higher or equal number of iterations than EFMtool, as typically biochemical networks are underdetermined (fewer constraints than metabolites). For the uncompressed models, gEFM provides significantly fewer combinations and comparisons than EFMTool for *E. coli*, *E. coli*(irrev), and *E. coli*(gluc), while EFMTool provides significantly fewer combinations and comparisons for *S. cerevisiae* and *C. reinhardtii*. The runtimes for gEFM are smaller than for EFMTool for examples with similar number of iterations and slightly smaller number of combinations and comparisons. For the compressed models, EFMTool generates a substantially smaller number of combinations and comparisons, except for *E. coli*, *E. coli*(irrev) and *C. reinhardtii*.

TABLE 4
The Number of Iterations, Cumulative Number of Generated Combinations, Cumulative Number of Comparisons for gEFM, EFMTool, for (a) Uncompressed, and (b) Compressed Models

(a) Uncompressed

Test Case	gEFM			EFMTool			EFMTool to gEFM ratios		
	Iterations	Combinations	Comparisons	Iterations	Combinations	Comparisons	Iterations	Combinations	Comparisons
Adipocyte	26	9.36×10^{02}	4.88×10^{04}	26	4.43×10^{02}	1.84×10^{04}	1.00	0.47	0.38
CHO	26	3.09×10^{05}	4.14×10^{08}	26	8.78×10^{04}	6.21×10^{07}	1.00	0.28	0.15
E. coli	52	3.41×10^{10}	8.43×10^{15}	47	4.06×10^{10}	1.14×10^{15}	0.90	1.19	1.35
E. coli(irrev)	52	2.31×10^{03}	4.10×10^{05}	47	2.29×10^{05}	1.93×10^{08}	0.90	99.17	470.57
E. coli(gluc)	52	2.58×10^{07}	3.07×10^{11}	42	5.40×10^{07}	1.27×10^{12}	0.81	2.10	4.14
H. pylori	287	4.76×10^{09}	1.83×10^{15}	281	4.29×10^{09}	1.15×10^{15}	0.98	0.90	0.63
S. cerevisiae	147	4.98×10^{11}	9.08×10^{17}	143	2.13×10^{10}	4.37×10^{16}	0.97	0.04	0.05
C. reinhardtii	39	8.58×10^{11}	2.12×10^{18}	38	1.62×10^{11}	1.75×10^{17}	0.97	0.19	0.08

(b) Compressed

Test Case	gEFM			EFMTool			EFMTool to gEFM ratios		
	Iterations	Combinations	Comparisons	Iterations	Combinations	Comparisons	Iterations	Combinations	Comparisons
Adipocyte	7	3.62×10^{02}	1.55×10^{04}	7	1.20×10^{02}	4.60×10^{03}	1.00	0.33	0.30
CHO	12	7.11×10^{04}	9.14×10^{07}	12	8.14×10^{03}	9.92×10^{06}	1.00	0.11	0.11
E. coli	26	3.41×10^{10}	8.43×10^{15}	21	3.82×10^{10}	1.25×10^{16}	0.81	1.12	1.48
E. coli(irrev)	12	1.86×10^{03}	3.13×10^{05}	9	4.21×10^{03}	2.07×10^{06}	0.75	2.26	6.60
E. coli(gluc)	26	1.72×10^{07}	2.12×10^{11}	21	1.43×10^{06}	2.40×10^{10}	0.81	0.08	0.11
H. pylori	23	3.04×10^{09}	1.17×10^{15}	21	9.98×10^{08}	5.26×10^{14}	0.91	0.33	0.45
S. cerevisiae	29	2.94×10^{10}	3.15×10^{16}	26	7.00×10^{08}	1.16×10^{15}	0.90	0.02	0.04
C. reinhardtii	28	7.76×10^{10}	1.88×10^{17}	27	8.84×10^{10}	3.09×10^{17}	0.96	1.14	1.64

TABLE 5
The Best of and Average Runtimes of 10 Runs Where Metabolites/Constraints Are Randomly Ordered

	gEFM		EFMTool	
	Best of 10 random	Average of 10 random	Best of 10 random	Average of 10 random
Adipocyte	-	-	0.76	0.91
CHO	1.50	1.53	0.78	0.91
E. coli	5.79	80.75	0.51	0.97
E. coli(irrev)	-	-	0.82	1.00
E. coli(gluc)	4.14	57.84	0.34	0.77

The runtimes are normalized to their respective runtimes using the default heuristics. The "-" indicates that the runtimes were all less than < 0.01 seconds.

3.5 Impact of Compression

Comparing the number of comparisons and constraints for the uncompressed and compressed models from Table 4, it is clear that both EFMTool and gEFM benefit from compression. The same number of EFMs is identified in each case. gEFM benefits from compression in two ways. First, some compression methods reduce the number of reactions in the network, which may reduce the size of the bit vectors used for storing the reactions and in turn reduce the runtime associated with processing the bit vectors. Each bit vector is a 32-bit integer array, large enough to represent each reaction with a bit. The reduction in the number of reactions will be beneficial only if the number of integers needed to represent the bit vectors is reduced. Second, compression reduces the number of metabolites in the network, which reduces the number of iterations within the gEFM algorithm. All compression methods can reduce the number of reactions in the network whereas only some compression methods (i.e. dead-end metabolites, unique flows, and coupled-combine methods) can reduce the number of metabolites. Metabolites removed by compression have a very small number of reactions associated with them, and would have been balanced in the early iterations of gEFM when applied to the uncompressed network. The reduction in the number of comparisons for these metabolites is therefore small. For the *E. coli*(gluc) network, compression reduces the number of comparisons by 23 percent whereas for the *C. reinhardtii* network, the reduction in the number of comparisons is 91 percent. EFMTool benefits from compression techniques due to the decreased number of reactions, which in turn reduces the number of constraints, resulting in substantial runtime savings. EFMtool does not directly benefit from reducing the number of metabolites.

3.6 Impact of Metabolite/Constraint Ordering

gEFM uses a simple heuristic, originally proposed by Mavrovouniotis et al. [26], to select a metabolite to process during each iteration of the algorithm. For each unprocessed metabolite, the potential number of new candidate pathways is calculated as the product of the number of input and output partial pathways. The metabolite with the smallest number of combinations is selected. In EFMTool, several heuristics are utilized including selecting a row with the largest number of zeros, thus generating the smallest number of combinations.

The impact of metabolite/constraint ordering on runtime performance of gEFM and EFMTool was investigated by comparing the runtimes using each algorithm's heuristic ordering against the best of 10 random orderings. For gEFM, the metabolite to be balanced was chosen at random. For EFMTool, the rows of the null-space kernel matrix were randomly ordered before applying the double-description method. For each compressed test case, Table 5 lists the best and average runtimes for the 10 runs normalized to the runtimes reported in Table 3. For gEFM, the fastest runtime among the 10 runs is always larger than the original runtime of gEFM. On average, random metabolite ordering significantly ($> 10\times$) increases the runtime as showing for *E. coli* and *E. coli*(gluc). For EFMTool, the best runtime among the 10 random runs always decreases the runtime by 18 to 66 percent. The average of the 10 random runs is at most $1.65\times$ the runtime of the original heuristic. Both tools are thus sensitive to metabolite/constraint ordering. However, the ordering heuristic for gEFM provides the smallest runtime among the 10 random constraint ordering experiments, whereas the EFMTool ordering heuristic did not.

4 Discussion

We adopt in this paper the algorithm originally proposed in 1990 by Mavrovouniotis et al. [26] for pathway synthesis to compute the Elementary Flux Modes. While earlier work [17] adapted this algorithm for EFM computation by identifying a test to remove dependent pathways, it was suggested that a matrix-based implementation is a more appropriate realization of the algorithm than graph traversal. In the present study, the proposed algorithm, gEFM, utilizes graph traversal in constructing the EFMs. We show that graph traversal provides a viable approach for computing EFMs. The gEFM implementation is shown to be competitive with state-of-the-art EFM computational techniques for several test cases, but less so for networks with a larger number of EFMs.

EFMs correspond to the extreme rays of a pointed polyhedral cone. gEFM is rooted in the double-description method, which establishes two equivalent characterization of a pointed convex cone: one based on the constraints that describe the hyperplanes forming the convex cone, and another based on the rays spanning the cone. gEFM implements the double-description method (see [20], [21], [22], [23] for a description of this method). In each iteration of gEFM, a constraint on balancing an internal metabolite is processed, and new extreme rays are identified. The gEFM algorithm combines the input and output pathways of the metabolite to generate intermediate pathways that lie on the hyperplane associated with the constraint. Removal of

dependent pathways in gEFM (as well as in prior implementations) identifies the extreme rays. The large number of intermediate pathways and dependency checking is currently the major implementation bottleneck in gEFM and in other approaches.

All methods based on the double-description have been reported sensitive to the ordering of the constraints, and that dynamic ordering methods are not necessarily superior [23]. Urbanczik and Wagner observe that it is difficult to remove such dependency [19]. Our results demonstrate this sensitivity for both gEFM and EFMTool. Importantly, we show that *there is* a clear profitable ordering for gEFM. We have explored several metabolite selection heuristics including selecting a metabolite at random, or with a maximum or minimum number of candidate pathways. Consistently, the best results were based on the smallest number of combinations, which is the metabolite selection scheme originally suggested in the Mavrovouniotis pathway synthesis approach [26]. Knowledge of the network structure allows for a better ordering heuristic, leading to reduced runtimes. In contrast, network topology information is lost when computing the null space for techniques such as EFMTool and Metatool.

The runtime of gEFM directly correlates first and foremost with the overall number of generated rays (combinations) and comparisons needed for dependency checking. Specifically, the runtimes in Table 3 correlate with the number of cumulative comparisons in Table 4. For the same number of comparisons, gEFM benefits when employing a smaller number of iterations. For example, the number of comparisons is comparable for the compressed and uncompressed models for *H. pylori* while the number of iterations is smaller for the compressed model, and the runtime for *H. pylori* is significantly reduced for the compressed model. The compression techniques we utilized were specifically developed for EFMTool, and they enabled some runtime savings, with an average runtime savings of 55 percent for EFMTool and 17 percent for gEFM for our set of test cases.

Enumerating elementary modes is a computationally intractable problem. Even when enumerating EFMs with a given reaction in their support, it was proved that there is no polynomial time algorithm in the number of reactions unless P = NP [42]. Analysis of larger biochemical networks will thus require alternate pathway analysis methods. One option is constraining the solution space based on biological information (e.g., a specified flux distribution [43], thermodynamic feasibility [44], regulatory mechanisms [45], and branching properties [46]) or structural significance (e.g., K-shortest EFMs [47]). Other options include restricting EFM analysis to a subnetwork [48], identifying EFMs that contain only particular reactions [49], and enumerating the EFMs in a lower-dimensional space [50]. Another option involves sampling the EFM solution space [51]. One other option is to consider searching for pathways with particular desirable properties that may not necessarily be EFMs [52], [53].

5 Conclusion

We presented in this paper an algorithm, gEFM, for computing the elementary flux modes within a biochemical network. The algorithm is iterative, processing one metabolite-balancing constraint at a time to generate partial pathways that are vetted for independence against all prior generated such pathways. The algorithm implements the canonical approach, which in turn is a variant of the double-description method. The paper demonstrates that graph-based approaches are viable for computing EFMs, and by natural extension, for computing the extreme rays of a convex cone. The main advantage of gEFM is utilizing the underlying structural information to derive a constraint ordering that leads to improved performance. When applied to several test cases, gEFM is found to be competitive for several test cases when compared to other EFM computational methods. Computing EFMs, however, remains a computationally intractable problem. Inexact or partial decomposition methods or alternate pathway analysis methods might be the best option when analyzing pathways within large biochemical networks. The C++ implementation of the gEFM algorithm is available via GitHub at https://github.com/eullah01/gEFM. Additionally, supplementary text file Appendix A outlines the classes and methods used for the implementation.

Appendix A

Classes and methods used for the implementation.

Acknowledgments

This work was supported by the US National Science Foundation under Grant no. 0829899. The authors wish to thank the anonymous reviewers for their valuable feedback on the submitted manuscript.

References

[1] Y.-S. Jang, J. M. Park, S. Choi, Y. J. Choi, D. Y. Seung, J. H. Cho, and S. Y. Lee, "Engineering of microorganisms for the production of biofuels and perspectives based on systems metabolic engineering approaches," *Biotechnol. Adv.*, vol. 30, pp. 989–1000, 2011.

[2] W. C. Ruder, T. Lu, and J. J. Collins, "Synthetic biology moving into the clinic," *Science*, vol. 333, pp. 1248–1252, 2011.

[3] D. Densmore and S. Hassoun, "Design automation for synthetic biological systems," *IEEE Des. Test Comput.*, vol. 29, no. 3, pp. 7–20, Jun. 2012.

[4] S. Schuster and C. Hilgetag, "On elementary flux modes in biochemical reaction systems at steady state," *J. Biol. Syst*, vol. 2, pp. 165–182, 1994.

[5] V. Acuña, F. Chierichetti, V. Lacroix, A. Marchetti-Spaccamela, M.-F. Sagot, and L. Stougie, "Modes and cuts in metabolic networks: complexity and algorithms," *Bio Syst.*, vol. 95, pp. 51–60, 2009.

[6] J. Stelling, S. Klamt, K. Bettenbrock, S. Schuster, and E. D. Gilles, "Metabolic network structure determines key aspects of functionality and regulation," *Nature*, vol. 420, no. 6912, pp. 190–193, 2002.

[7] N. Vijayasankaran, R. Carlson, and F. Srienc, "Metabolic pathway structures for recombinant protein synthesis in escherichia coli," *Appl. Microbiol. Biotechnol.*, vol. 68, no. 6, pp. 737–746, 2005.

[8] F. Llaneras and J. Pic, "An interval approach for dealing with flux distributions and elementary modes activity patterns," *J. Theoretical Biol.*, vol. 246, pp. 290–308, 2007.

[9] S. Klamt and E. D. Gilles, "Minimal cut sets in biochemical reaction networks," *Bioinformatics*, vol. 20, no. 2, pp. 226–234, 2004.

[10] A. P. Burgard, E. V. Nikolaev, C. H. Schilling, and C. D. Maranas, "Flux coupling analysis of genome-scale metabolic network reconstructions," *Genome Res.*, vol. 14, no. 2, pp. 301–12, 2004.

[11] R. P. Carlson, "Decomposition of complex microbial behaviors into resource-based stress responses," *Bioinformatics*, vol. 25, pp. 90–97, 2009.

[12] R. Carlson and F. Srienc, "Fundamental escherichia coli biochemical pathways for biomass and energy production: Creation of overall flux states," *Biotechnol. Bioeng.*, vol. 86, no. 2, pp. 149–62, 2004.

[13] C. T. Trinh, P. Unrean, and F. Srienc, "Minimal *Escherichia coli* cell for the most efficient production of ethanol from hexoses and pentoses," *Appl. Environ. Microbiol.*, vol. 74, no. 12, pp. 3634–3643, 2008.
[14] J. Schwender, F. Goffman, J. B. Ohlrogge, and Y. Shachar-Hill, "Rubisco without the calvin cycle improves the carbon efficiency of developing green seeds," *Nature*, vol. 432, no. 7018, pp. 779–82, 2004.
[15] J. Gagneur and S. Klamt, "Computation of elementary modes: A unifying framework and the new binary approach," *BMC Bioinformatics*, vol. 5, p. 175, 2004.
[16] F. Noiłka, J. Guddat, H. Hollatz, and B. Bank, *Theorie der linearen parametrischen Optimierung*, vol. 56, L. Collatz, Ed. Berlin, Germnay: Akademie-Verlag, 1976.
[17] S. Schuster, C. Hilgetag, J. Woods, and D. Fell, "Elementary modes of functioning in biochemical reaction networks. Aspects of interpretation and application," in *Computation in Cellular and Molecular Biological Systems*, R. Cuthbertson, M. Holcombe, and R. Paton, Eds. Singapore: World Scientific, 1996, pp. 151–165.
[18] C. Wagner, "Nullspace approach to determine the elementary modes of chemical reaction systems," *J. Phys. Chemistry B*, vol. 108, no. 7, pp. 2425–2431, 2004.
[19] R. Urbanczik and C. Wagner, "An improved algorithm for stoichiometric network analysis: Theory and applications," *Bioinformatics*, vol. 21, no. 7, pp. 1203–1210, 2005.
[20] T. S. Motzkin, H. Raiffa, G. L. Thompson, and R. M. Thrall, "The double description method," 1953.
[21] N. V. Chernikova, "Algorithm for finding a general formula for the non-negative solutions of a system of linear inequalities," *Zh. Vychisl. Mat. Mat. Fiz.*, vol. 5, no. 2, pp. 334–337, 1965.
[22] H. Edelsbrunner, *Algorithms in Combinatorial Geometry*. New York, NY, USA: Springer-Verlag, 1987.
[23] K. Fukuda and A. Prodon, "Double description method revisited," *Combinatorics Comput. Sci.*, vol. 1, pp. 91–111, 1996.
[24] A. von Kamp and S. Schuster, "Metatool 5.0: Fast and flexible elementary modes analysis," *Bioinformatics*, vol. 22, no. 15, pp. 1930–1, 2006.
[25] M. Terzer and J. Stelling, "Elementary flux modes state-of-the-art implementation and scope of application," *BMC Systems Biology*, vol. 1, p. P2, 2007.
[26] M. Mavrovouniotis, G. Stephanopoulos, and G. Stephanopoulos, "Computer-aided synthesis of biochemical pathways," *Biotechn. Bioeng*, vol. 36, pp. 1119–1132, 1990.
[27] M. Mavrovouniotis, "Synthesis of reaction mechanisms consisting of reversible and irreversible steps. 1. A synthesis approach in the context of simple examples," *Ind. Eng. Chem*, vol. 31, pp. 1625–1637, 1992.
[28] M. Mavrovouniotis, "Synthesis of reaction mechanisms consisting of reversible and irreversible steps. 2. Formalization and analysis of the synthesis algorithm," *Ind. Eng. Chem*, vol. 31, pp. 1637–1653, 1992.
[29] E. Ullah, C. Hopkins, S. Aeron, and S. Hassoun, "Decomposing biochemical networks into elementary flux modes using graph traversal," in *Proc. ACM Int. Conf. Bioinformatics, Comput. Biol. Biomedical Informat.*, 2013, pp. 211–218.
[30] S. Schuster, D. A. Fell, and T. Dandekar, "A general definition of metabolic pathways useful for systematic organization and analysis of complex metabolic networks," *Nat. Biotechnol.*, vol. 18, no. 3, pp. 326–332, 2000.
[31] S. Schuster, C. Hilgetag, J. Woods, and D. Fell, "Reaction routes in biochemical reaction systems: Algebraic properties, validated calculation procedure and example from nucleotide metabolism," *J. Math. Biol.*, vol. 45, pp. 153–181, 2002.
[32] M. Terzer, "Large scale methods to enumerate extreme rays and elementary modes," Ph.D. dissertation, Dept. Biosyst. Sci. Eng., Swiss Federal Inst. Technol., Zurich, Zurich, Switzerland, 2009.
[33] M. Terzer and J. Stelling, "Accelerating the computation of elementary modes using pattern trees," *Algorithms Bioinformatics*, vol. 4175, pp. 333–343, 2006.
[34] Y. Si, S. Palani, A. Jayaraman, and K. Lee, "Effects of forced uncoupling protein 1 expression in 3t3-l1 cells on mitochondrial function and lipid metabolism," *J. Lipid Res.*, vol. 48, no. 4, pp. 826–36, 2007.
[35] R. Nolan and K. Lee, "Dynamic model of CHO cell metabolism," *Metab. Eng.*, vol. 13, no. 1, pp. 108–24, Jan. 2011.
[36] L. Quek, S. Dietmair, J. Kromer, and L. Nielsen, "Metabolic flux analysis in mammalian cell culture," *Metab. Eng.*, vol. 12, no. 2, pp. 161–71, 2010.
[37] I. Thiele, T. D. Vo, N. D. Price, and B. Palsson, "Expanded metabolic reconstruction of *Helicobacter pylori* (iIT341 GSM/GPR): an in silico genome-scale characterization of single- and double-deletion mutants," *J. Bacteriol.*, vol. 187, no. 16, pp. 5818–5830, 2005.
[38] N. C. Duarte, M. J. Herrgard, and B. Palsson, "Reconstruction and validation of Saccharomyces cerevisiae iND750, a fully compartmentalized genome-scale metabolic model," *Genome Res.*, vol. 14, no. 7, pp. 1298–1309, 2004.
[39] N. R. Boyle and J. A. Morgan, "Flux balance analysis of primary metabolism in chlamydomonas reinhardtii," *BMC Syst Biol.*, vol. 3, no. 4, 2009.
[40] M. Terzer and J. Stelling. (2009, Dec.). Elementary flux mode tool (efmtool) [Online]. Available: http://www.csb.ethz.ch/tools/efmtool
[41] M. Yousofshahi, E. Ullah, R. Stern, and S. Hassoun, "MC^3: A steady-state model and constraint consistency checker for biochemical networks," *BMC Syst. Biol.*, vol. 7, pp. 129–136, 2013.
[42] V. Acuña, A. Marchetti-Spaccamela, M.-F. Sagot, and L. Stougie, "A note on the complexity of finding and enumerating elementary modes," *Bio Syst.*, vol. 99, no. 3, pp. 210–214, 2010.
[43] K. Ip, C. Colijn, and D. S. Lun, "Analysis of complex metabolic behavior through pathway decomposition," *BMC Syst. Biol.*, vol. 5, no. 1, p. 91, 2011.
[44] S. J. Jol, A. Kümmel, M. Terzer, J. Stelling, and M. Heinemann, "System-level insights into yeast metabolism by thermodynamic analysis of elementary flux modes," *PLoS Comput. Biol.*, vol. 8, no. 3, p. e1002415, 2012.
[45] C. Jungreuthmayer, D. E. Ruckerbauer, and J. Zanghellini, "regEfmtool: Speeding up elementary flux mode calculation using transcriptional regulatory rules in the form of three-state logic," *Biosystems*, vol. 113, no. 1, pp. 37–39, 2013.
[46] S. H. J. Chan, C. Solem, P. R. Jensen, and P. Ji, "Estimating biological elementary flux modes that decompose a flux distribution by the minimal branching property," *Bioinformatics*, vol. 30, no. 22, pp. 3232–3239, 2014.
[47] L. F. de Figueiredo, S. Schuster, C. Kaleta, and D. A. Fell, "Can sugars be produced from fatty acids? a test case for pathway analysis tools," *Bioinformatics*, vol. 25, no. 1, pp. 152–158, 2009.
[48] C. Kaleta, L. F. de Figueiredo, and S. Schuster, "Can the whole be less than the sum of its parts? pathway analysis in genome-scale metabolic networks using elementary flux patterns," *Genome Res.*, vol. 19, no. 10, pp. 1872–1883, 2009.
[49] L. David and A. Bockmayr, "Computing elementary flux modes involving a set of target reactions," *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, vol. 11, no. 6, pp. 1099–1107, Nov. 2014.
[50] S.-A. Marashi, L. David, and A. Bockmayr, "Analysis of metabolic subnetworks by flux cone projection," *Algorithms Molecular Biol.*, vol. 7, p. 17, 2012.
[51] D. Machado, Z. Soons, K. R. Patil, E. C. Ferreira, and I. Rocha, "Random sampling of elementary flux modes in large-scale metabolic networks," *Bioinformatics*, vol. 28, no. 18, pp. i515–i521, 2012.
[52] E. Ullah, K. Lee, and S. Hassoun, "An algorithm for identifying dominant-edge metabolic pathways," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, 2009, pp. 144–150.
[53] E. Ullah, M. Walker, K. Lee, and S. Hassoun, "PreProPath Algorithm: An uncertainty-aware algorithm for identifying predictable profitable pathways in biochemical networks," *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, vol. PP, no. 99, p. 1, 2015.

Ehsan Ullah received the master's degree in electrical engineering from the University of Engineering and Technology, Lahore, Pakistan, and the PhD degree in computer science from Tufts University. His research interests include computational tools and algorithms for systems biology. He is a member of the ACM, IET, IEEE, and IEEE Computer Society.

Shuchin Aeron received the PhD degree in ECE from Boston University in 2009. He is currently an assistant professor in the Department of ECE, Tufts University. From 2009 to 2011, he was a postdoctoral research fellow at Schlumberger-Doll Research (SDR), Cambridge, Mass., where he worked on signal processing answer products for borehole acoustics. He has several patents and the proposed workflows are currently implemented in logging while drilling tools. He received the Best Thesis Award from both the College of Engineering and the Department of ECE in 2009. He received the Center of Information and Systems Engineering (CISE) award from Boston University in 2006 and a Schlumberger-Doll Research grant in 2007. His current research interests include statistical signal processing, machine learning and pattern recognition, and geophysical signal processing. He is a member of the IEEE.

Soha Hassoun received the PhD degree from the Department of Computer Science and Engineering, University of Washington, Seattle, and the master's degree from the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology. She is a professor and the chair in the Department of Computer Science, Tufts University. Her research interests are electronic design automation and computational tools for Systems Biology. She received the US National Science Foundation (NSF) CAREER Award. She received ACM/SIGDA Distinguished Service Awards for creating the PhD forum at the Design Automation Conference (DAC), and the CADAthlon at ICCAD. She serves and has served on a number of technical and executive committees for several conferences and workshops including serving as the technical program chair for the Design Automation Conference (DAC) in 2012 and 2011, and the International Conference on Computer-Aided Design (ICCAD) in 2005. She is a Tau Beta Pi fellow. She is a member of the ACM, AiChE, and a senior member of the IEEE.