Bioinformatics, 2014, 1–8
doi: 10.1093/bioinformatics/btu733
Advance Access Publication Date: 6 November 2014
Original Paper

OXFORD

Systems biology

TreeEFM: calculating elementary flux modes using linear optimization in a tree-based algorithm

Jon Pey[1], Juan A. Villar[2], Luis Tobalina[1], Alberto Rezola[1], José Manuel García[2], John E. Beasley[3] and Francisco J. Planes[1,*]

[1]CEIT and TECNUN, University of Navarra, Manuel de Lardizabal 15, 20018 San Sebastian, Spain, [2]Computer Engineering Department, School of Computer Science, POB 30100 University of Murcia, Spain and [3]Mathematical Sciences, Brunel University, Kingston Lane, UB8 3PH Uxbridge, UK

*To whom correspondence should be addressed.

Associate Editor: Igor Jurisica

Received on February 27, 2014; revised on September 16, 2014; accepted on October 31, 2014

Abstract

Motivation: Elementary flux modes (EFMs) analysis constitutes a fundamental tool in systems biology. However, the efficient calculation of EFMs in genome-scale metabolic networks (GSMNs) is still a challenge. We present a novel algorithm that uses a linear programming-based tree search and efficiently enumerates a subset of EFMs in GSMNs.

Results: Our approach is compared with the EFMEvolver approach, demonstrating a significant improvement in computation time. We also validate the usefulness of our new approach by studying the acetate overflow metabolism in the *Escherichia coli* bacteria. To do so, we computed 1 million EFMs for each energetic amino acid and then analysed the relevance of each energetic amino acid based on gene/protein expression data and the obtained EFMs. We found good agreement between previous experiments and the conclusions reached using EFMs. Finally, we also analysed the performance of our approach when applied to large GSMNs.

Availability and implementation: The stand-alone software TreeEFM is implemented in C++ and interacts with the open-source linear solver COIN-OR Linear program Solver (CLP).

Contact: fplanes@ceit.es

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Many advances in medicine and biology would not be possible without a detailed understanding of the functional mechanisms of cellular metabolism (Hsu and Sabatini, 2008). However, this is a very complex task as there are intricate regulatory events controlling cellular response. Novel technologies, referred to with the suffix *omics*, are emerging in the last two decades and providing us with a cellular picture of unprecedented coverage and resolution (Joyce and Palsson, 2006). Unfortunately, it is often difficult to detect the mechanisms behind an observed alteration using the data provided by *omics* technologies. The scientific community needs improved tools and methods to identify these mechanisms in different scenarios (Werner, 2008).

Several mathematical methods modeling metabolism are emerging that are able to incorporate datasets provided by different *omics* technologies. Many of these methods are encompassed within constraint-based models (Price *et al.*, 2004), in which a set of mathematical constraints are defined using a genome-scale metabolic network (GSMN) reconstruction as a starting point. Several curated GSMNs can be found in the literature (Thiele and Palsson, 2010). However, being able to automatically characterize the biochemical reactions present in a particular metabolism through *omics* data truly constitutes a challenge (Schmidt *et al.*, 2013).

Typically, constraint-based models account for two principal constraints, namely the mass balance equation at steady state, and

thermodynamic-based irreversibility constraints. If, given a set of enzymes, all of them are essential to fulfill these two constraints, the set is referred to as an elementary flux mode (EFM; Schuster *et al.*, 2000). In other words, EFMs are solutions with the minimum support necessary to operate in stoichiometric steady-state balance with all reactions in the appropriate direction. The advantages of analysing metabolic networks based on EFMs have been shown in different works (e.g. de Figueiredo *et al.*, 2008; Gebauer *et al.*, 2012; Trinh *et al.*, 2009; Rezola *et al.*, 2014). However, their use has been limited because enumerating them is computationally demanding. Algorithms have been developed to enumerate all the EFMs in medium-size metabolic networks (Terzer and Stelling, 2008; Urbanczik and Wagner, 2005; Von Kamp and Schuster, 2006). However, despite the development of novel methods using state of the art computational techniques expediting their application in larger networks (Hunt *et al.*, 2014), this family of algorithms fails on GSMNs using standard computers, because of the combinatorial explosion in the number of EFMs (Klamt and Stelling, 2002). In this light, several methods have been recently proposed to determine a subset of EFMs in GSMNs (de Figueiredo *et al.*, 2009; Kaleta *et al.*, 2009; Pey and Planes, 2014; Rezola *et al.*, 2011, 2013). Continuing with this effort, we present here a novel algorithm that uses a linear programming-based tree search for EFM numeration, denoted as TreeEFM. We benchmark the performance of our new approach with EFMEvolver (Kaleta *et al.*, 2009), currently the most efficient framework to find EFMs in GSMNs. The results show that TreeEFM outperforms EFMEvolver.

After confirming the efficiency of the new approach presented here, we calculate a large set of EFMs to study a relevant biological question, the acetate overflow metabolism. The aerobic production of acetate causes serious economic losses in bacteria-based biofactories (Nakano *et al.*, 1997). Finding the mechanisms that prevent this scenario constitutes an open question in the field (Valgepea *et al.*, 2010). Here we analyse the relevance of eight energetic amino acids in the overflow metabolism using our approach to compute a large set of EFMs. We then apply the methodology presented in Rezola *et al.* (2013) to score each EFM based on expression data, using the gene/protein expression data provided by Valgepea *et al.* (2010). On the basis of a literature review, we could validate the conclusions reached using EFMs.

Finally, we analysed the performance of our approach for handling even larger networks than traditional GSMNs, paving the way for forthcoming models involving an ever-increasing number of reactions and metabolites.

2. Methods

2.1 Mathematical definitions

Assume we have a metabolic network comprising C metabolites and R reactions. For each reaction r ($r=1,\ldots,R$), we assign a continuous flux variable v_r representing its activity. EFMs assume that the concentration of the internal metabolites (I) remain constant. This is considered in the mass-balance equation, assuming the steady-state condition, as in Equation (1). Note that S_{cr} is the stoichiometric coefficient associated with metabolite c ($c=1,\ldots,C$) in reaction r ($r=1,\ldots,R$). As usual in the literature, e.g. Schuster *et al.* (2000), substrates have a negative stoichiometric coefficient, whilst products have a positive stoichiometric coefficient.

$$\sum_{r=1}^{R} S_{cr} v_r = 0, \quad \forall c \in I \tag{1}$$

In addition, EFMs must satisfy thermodynamic constraints in the form of irreversible reactions (Irr). Equation (2) prevents fluxes in Irr from being negative.

$$v_r \geq 0, \ \forall r \in Irr \tag{2}$$

Equations (1) and (2) define a non-pointed polyhedral cone (Rezola *et al.*, 2011). To calculate EFMs with the methodology presented here, it is necessary to transform the non-pointed cone into a particular pointed cone, referred to as P. For that, each reversible reaction is split into two irreversible reactions, as typically done in the literature (de Figueiredo *et al.*, 2009; Pey *et al.*, 2014).

We search for EFMs including at least one reaction from a set L. This is achieved by cutting P with the hyper-plane in Equation (3).

$$\sum_{r \in L} v_r \geq 1 \tag{3}$$

Please note here that the 1 in the right-hand side of Equation (3) can be arbitrarily modified because any hyper-plane intersecting the pointed cone defined by Equations (1) and (2) can be considered. Further details can be found in Pey and Planes (2014).

Equations (1)–(3) generate a convex feasible space whose extreme points are the EFMs of interest (Pey and Planes, 2014). This idea has been exploited by previous approaches, e.g. Kaleta *et al.* (2009). Efficient methods to find extreme points in terms of a given objective function have been developed using linear programming and the Simplex algorithm (Dantzig *et al.*, 1955), which iterates through extreme points. The Simplex method is efficiently implemented in tools such as COIN CLP (Lougee-Heimer, 2003).

Note that any feasible solution satisfying (1)–(3) calculated using the Simplex algorithm is an EFM (Pey and Planes, 2014). Therefore, any linear objective function can be used, as optimality is not necessary but feasibility is. In particular, we do not impose any objective function to identify the first EFM.

2.2 Tree-based search

We have defined the model to calculate a single EFM based on linear programming. Now we introduce our tree-based algorithm to enumerate a subset of EFMs, referred to as TreeEFM. Each node of our tree represents a linear program (LP) containing a potential EFM different from the one that was calculated in its ancestor node. To guarantee this, let us assume that an EFM calculated at an arbitrary node comprises m active reactions. Without loss of generality, assume that $v_1, v_2, \ldots, v_m > 0$ and $v_{m+1}, v_{m+2}, \ldots, v_R = 0$, so that the first m reactions are active. We then create m additional tree nodes, each of which removes one of the m active reactions in the parent node, namely the first node has $v_1 = 0$, the second node $v_2 = 0$, and so on (Fig. 1).Thus, we prevent the method from recalculating the preceding solution by recurrently removing reactions.

This process is repeated at each node, carrying forward any fluxes set to zero in the ancestor node, as typically done in tree-based search algorithms. Our LP-tree-based search procedure is similar to the branch-and-bound approaches to solve integer

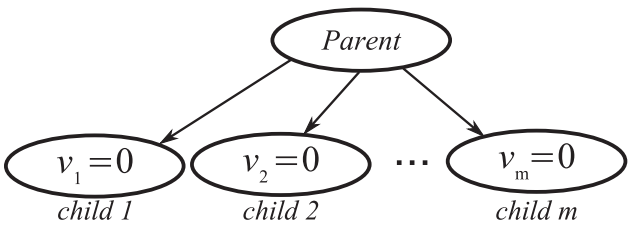

Fig. 1. Connections between the parent node and its children

programming models (Land and Doig, 1960). However, our branching strategy is completely different and bounds are not used here. In particular, in many branch-and-bound approaches two children emerge from each parent node, whilst here TreeEFM may create as many child nodes as active reactions in the solution obtained in the antecessor node. Note also that during the branch-and-bound process integrality is promoted by including an inequality in each child besides the constraints in the parent node. TreeEFM is slightly different since a constraint deactivating a variable is included in the child node. Nevertheless, both approaches share the idea of carrying forward constraints from the parent to the child node.

Trees have been used previously for EFM computation. In particular, in the efmtool approach (Terzer and Stelling, 2008), the concept of bit pattern trees was introduced to optimize the iterative phase of the double description method, which is the standard algorithm to enumerate extreme rays in a polyhedral cone. This algorithm starts from an initial cone that contains the final cone, defined by Equations (1) and (2). Each time a constraint is added, the cone and extreme rays are updated. By definition, it is necessary to complete the iterative process to guarantee that each candidate is indeed an EFM. Bit pattern trees were introduced to accelerate the elementarity test in intermediate iterations. Our tree-branching approach is substantially different, as it is based on linear programming and extreme point enumeration. Each node in the tree constitutes a LP that potentially provides us with an EFM. As linear programming solvers are highly sophisticated, we can always compute a subset of EFMs even in large-size networks. This gives a substantial computational advantage to our approach relative to efmtool, which is only applicable to networks of moderate size.

To conduct the tree search more efficiently, several heuristic rules have been included in the algorithm. In particular they avoid: (i) recalculating previously obtained EFMs and (ii) solving nodes representing LPs without solution, namely infeasible models, which may substantially increase the computation time. Further details are given below.

2.3 Heuristic rules

In our approach, the majority of the computation time is spent on solving LPs. Hence, decreasing the total number of LPs required to obtain an EFM constitutes an appropriate strategy. In this light, we first explore nodes that are more likely to provide a new EFM. Let us define E as the already calculated set of EFMs (where this set will grow in size as we explore more of the tree). We also define a non-negative *score* for each node i that is a count of how many EFMs in E are feasible in i. We consider that an EFM $j \in E$ is feasible in node i if all the constraints imposed in that node are satisfied in the EFM, meaning that solving the corresponding LP in node i could retrieve EFM j. It is important to realize that unnecessary calculations are being carried out when an EFM is recalculated, to the detriment of overall computational performance. For this reason, we first explore nodes with the lowest score, as the smaller the score of a node, the higher the probability of obtaining a new EFM after solving its corresponding LP. Different strategies are used depending on that score: (i) when the score is equal to zero and (ii) when the score is one or more. A schematic representation of the procedures follows:

Input: stoichiometric matrix S, active reaction set L, Stop conditions StopConditions

Output: flux vectors of the EFMs

```
// setting the root node
root_node = buildRoot(S, L);
efm = solveLinearProgram(root_node);
feasible_children = eliminateInfeasibilities(root_node, efm);
nodes_tree = storeChildrenInTree(feasible_children);

// processing the nodes of the tree
while (checkStopConditions(StopConditions) is not True and
        NumberUnexploredNodes(nodes_tree) is not zero) do
          current_node = selectNodeLowestScore(nodes_tree);
          if (scoreOf(current_node) is zero) then
                    efm = solveLinearProgram(current_node);
          else
                    efm = searchFittingEFM(current_node);
          end
          feasible_children = eliminateInfeasibilities
           (current_node, efm);
          nodes_tree = storeChildrenInTree(feasible_children);
end
```

Description:

After the initial node is solved, the child nodes are stored in the tree. While the stop conditions (i.e. number of obtained EFMs, or expired execution time) are not met and there are still unexplored nodes, the algorithm selects from the tree the node whose score is the lowest. If the score is zero, then the LP of the node is re-solved by the CLP library. However, if the score is greater than zero, it means at least one of the obtained EFMs is likely to be returned again by the CLP library, so the algorithm just takes the existing EFM without invoking the CLP library. Lastly, the algorithm performs an analysis of the EFM of the current node to detect the reactions that may produce infeasible children, so that only feasible children are stored in the tree. If new EFMs are found by the function eliminateInfeasibilities, they will be added to E. Once the algorithm terminates, the flux vectors of the obtained EFMs are saved to disk.

Case 1: Score = 0

When the lowest score among the available nodes is zero, the LP defined in these nodes is directly solved. Two scenarios may then arise: (i) a new EFM is obtained, which can be added to E; or (ii) the LP is infeasible. In no case can we obtain an already calculated EFM, because a score of zero means that constraints in node i are not fulfilled by any EFM in E.

Note that solving an *infeasible node* implies solving an LP without obtaining a new EFM, resulting in wasted computation. To prevent this, we introduce the following strategies when the score is non-zero.

Case 2. Score > 0

Score > 0 implies there is at least one EFM in E that satisfies the constraints applicable at the current node. We refer to this EFM as one that *fits* in the current node. If the corresponding LP is solved, we run the risk of calculating again the same EFM. However, by solving only one additional LP, we can take advantage of this fitting EFM to calculate all the infeasibilities that may arise from the offspring of the current node. Let us consider the example in Figure 2.

Figure 2 represents the scenario posed above where the selected node, referred to as node i, imposes the elimination of reactions *1*, *3* and *7*. In addition, we have already calculated the EFM j activating reactions 2, 5, 8, 9 and 10. Note how EFM j fits in node i (since reactions 1, 3 and 7 are not active in it). If score > 1, we consider the most recently calculated EFM.

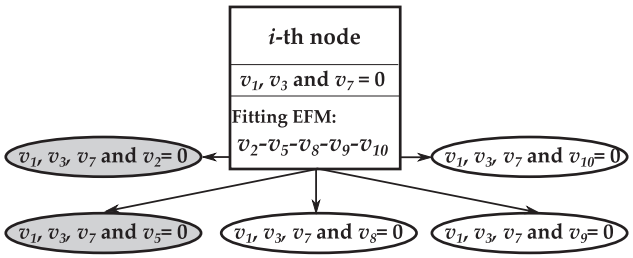

Fig. 2. A graphical representation of the example used to explain how to avoid solving unnecessary LPs to find infeasibilities. Nodes in gray correspond to infeasible nodes

As an objective for solving the LPs, we define a dynamic objective function f

$$f = \sum_{r \in K} v_r \tag{4}$$

that minimizes total flux in a set of reactions K, which are exactly those reactions active in the fitting EFM. The set K will be updated in an iterative process presented below. Returning to the toy example, $K = \{2, 5, 8, 9, 10\}$, we define f so that the activity of reactions 2, 5, 8, 9 and 10 is minimized. Thus, the LP defined in node i and amended with the objective function f is solved. Note that the LP is always feasible because there is at least one solution that satisfies the underlying constraints, i.e. EFM j.

Once the LP is solved, we update the reactions belonging to the set K. In particular, we remove reactions from K that became inactive in the calculated EFM. We are sure that nodes corresponding to the reactions removed from K are feasible, as we obtained a new EFM fitting in those nodes. By analogy with the original definition of K, we build a new f based on the new set K. Back to the example above, let us assume that we obtain an EFM activating reactions 2, 5, 8 and 11. We now update $K = \{2, 5, 8\}$. Note that reactions *9* and *10* were removed from K because they became inactive in the obtained EFM. The updated set K automatically leads to a new objective function (4). Then, solving the LP comprising the constraints defined at node i coupled with this new objective function f provides a new EFM activating reactions 2, 5, 9, 10, 12 and 13. Again, we update $K = \{2, 5\}$ and the objective function f.

This procedure is repeated until (i) the set K is empty or (ii) two consecutive iterations obtain the same set K. The first scenario is achieved when all offspring nodes are feasible. The latter corresponds to a scenario in which infeasibilities may appear. Returning again to the toy example, we assume that the EFM involving reactions *2*, *5*, *8* and *11* is obtained, and, in consequence, K remains the same because reaction 8 was previously removed. Therefore, nodes corresponding to the reactions in K are highly likely to lead to an infeasible solution. In the example above, we assume that the nodes corresponding to having reactions 2 and 5 inactive are infeasible (Fig. 2). We decided not to explore these nodes further and move to the next ones. These nodes could be considered later e.g. if the number of requested EFMs is not reached after exploring all the non-discarded nodes. Even if they are not considered, our approach allows us to determine many EFMs quickly, which is the goal of any approach, as the full computation of the entire set of EFMs in networks of large size is currently not viable given present-day computer resources. It is important to emphasize that in the last step we may solve an LP without obtaining a new EFM. However, such an LP enables us to prevent infeasible nodes from appearing.

Note that an already calculated EFM may be retrieved in the iterative process. In that case, the EFM is not re-included in the set E. In particular, when two nodes do not belong to the same descent line, the same EFM may fit in them. For instance, recalling again the example in Figure 2, note how the EFM activating reactions 2, 5, 8 and 11 fits in two nodes, i.e. nodes deactivating reactions *9* and *10* together with reactions *1*, *3* and *7*. However, it is not possible to obtain the EFM calculated in the parent node in any child, as they belong to the same descent line.

2.4 Implementation

TreeEFM is implemented in C++ and is provided in the Supplementary Material. The LPs were solved using the open-source Simplex implementation in CLP 1.15.

2.5 *P*-value calculation

In Rezola *et al.* (2013), a novel methodology quantifying the relevance of a set of EFMs using gene/protein expression data was introduced. Using Gene–Protein-Reaction logical rules, each reaction in the metabolic network is first classified as highly, medium or lowly expressed.

For the e-th EFM that comprises T_e reactions, the probability of having i_e and j_e of them as highly and lowly expressed, respectively, is calculated using a multivariate hypergeometric distribution:

$$P(x(R_G, R_H, R_L, T_e) = (i_e, j_e)) = \frac{\binom{R_H}{i_e}\binom{R_L}{j_e}\binom{R_G - R_H - R_L}{T_e - i_e - j_e}}{\binom{R_G}{T_e}}, \tag{5}$$

where R_H and R_L are the number of highly and lowly expressed reactions in the considered metabolic network comprising R_G reactions. To promote those EFMs activating highly expressed reactions while minimizing the number of lowly expressed, Rezola *et al.* presented a P-value based on the probability above

$$P\text{-value}_e = \sum_{i=i_e}^{\min(R_H, T_e)} \sum_{\substack{j=0 \\ i+j \le T_e}}^{j_e} P(x(R_G, R_H, R_L, T_e) = (i, j)). \tag{6}$$

In this work, we consider this P-value as a metric to analyse the relevance of a calculated set of EFMs under particular gene/protein expression conditions. Note that calculating the P-value is independent of the enumeration of EFMs carried out by TreeEFM.

3 Results

3.1 Validation

The improvement achieved by our approach is shown with a side-by-side comparison with EFMEvolver, one of the most efficient frameworks currently in the literature to calculate EFMs in GSMNs (Kaleta *et al.*, 2009). We considered the cases corresponding to the GSMN of *Escherichia coli* (Feist *et al.*, 2007). In particular, two different comparisons have been carried out for the production of L-lysine, L-Threonine and L-Arginine. Table 1 shows the number of EFMs obtained in 7200 s using EFMEvolver and TreeEFM. Solving LPs is the most time-consuming task in both approaches. Therefore, we also compared the number of LPs required to obtain 2000 EFMs in each case. As discussed in section 2.3, we consider that the approach solving fewer LPs per EFM is more efficient. The results support the efficiency of TreeEFM and as such it can be considered a state-of-the-art approach to calculate EFMs in GSMNs.

Table 1. Side-by-side comparison with EFMEvolver. Results for TreeEFM are the average value of ten trials

# EFMs in 7200 s	EFMEvolver	TreeEFM	Improvement (%)
L-Lysine	118 598	178 056	50.13
L -Threonine	126 491	184 425	45.80
L-Arginine	127 988	174 154	36.07
LPs for 2000 EFMs	EFMEvolver (LPs per EFM)	TreeEFM (LPs per EFM)	Improvement (%)
L-Lysine	2.23	1.38	38.12
L-Threonine	1.90	1.64	13.68
L-Arginine	1.80	1.67	7.22

Table 2. Average *P*-value provided by the methodology in Rezola *et al.* (2013)

Amino Acid	Mean *P*-value
Glycine (Gly)	0.0130
Threonine (L-Thr)	0.0238
Serine (L-Ser)	0.0940
Aspartate (L-Asp)	0.1369
Proline (L-Pro)	0.1718
Asparagine (L-Asn)	0.1812
Glutamate (L-Glu)	0.1992
Glutamine (L-Gln)	0.2168

3.2 Case study: acetate overflow

The number of industrial applications in which bacteria play a key role is increasing day by day. The competitiveness of bacterial processes is maximized by augmenting metabolic velocities. However, at a certain point, an overflow metabolism occurs. One of the main characteristics of this overflow metabolism in the *E.coli* bacteria is aerobic acetate excretion. This unwanted phenomenon impairs efficiency (Swartz, 2001). Despite the efforts of the scientific community (Valgepea *et al.*, 2010), a precise description of the mechanism triggering this overflow metabolism has not yet been reported.

In this section, we aim to identify key amino acids in the overflow scenario based on an EFM analysis. In particular, we analyse the EFMs producing the eight amino acids identified as energetic in (Gschaedler and Boudrant, 1994), as their role in the context of acetate overflow was experimentally analysed in (Han *et al.*, 2002). For that, we calculated 1 million EFMs per energetic amino acid (see first column in Table 2) to which an individual *P*-value was assigned using the method presented in (Rezola *et al.*, 2013) and revisited in section 2.5. These EFMs were calculated from the GSMN reconstruction of *E.coli* presented by Feist *et al.* (2007). In addition, we replicated the minimal growth medium with glucose as the sole carbon source as used in Valgepea *et al.* (2010). As the objective of this study was precisely to classify the global relevance of each energetic amino acid, we consider EFMs not necessarily excreting acetate. Finally, we used the reaction classification reported in Pey *et al.* (2013), which was obtained from the experimental data presented in Valgepea *et al.* (2010).

In this section, we first study the computational performance of the TreeEFM approach when calculating the aforementioned 1 million EFMs. Then, we analyse the obtained results after projecting the experimental data in Valgepea *et al.* (2010) into the computed EFMs.

3.2.1 Computational performance

The methodology presented here provides us with a large number of EFMs for GSMNs. As an illustration of the efficiency of the TreeEFM approach, we calculated 1 million EFMs producing each energetic amino acid in Table 2.

The time complexity of the TreeEFM framework is not linear because the computational cost of enumerating a new solution increases with the number of already calculated EFMs. This is mainly due to internal processes that increase in complexity when the number of calculated EFM increases. Nevertheless, any unwanted nonlinear behavior is not significant after 1 million EFMs are calculated, as illustrated in Figure 3.

Figure 3 shows the time evolution during the process of the calculation of 1 million EFMs corresponding to the best and worst scenario found, namely glutamate (L-Glu) and serine (L-Ser), respectively. In addition, Figure 3 contains the average time consumption for the eight energetic amino acids. Note that the computation time required to obtain a new EFM is not dramatically affected, even when a million of them have already been calculated. To quantify this, we performed a fitting of the data corresponding to the process computing EFMs producing L-Glu and L-Ser using the function in (7):

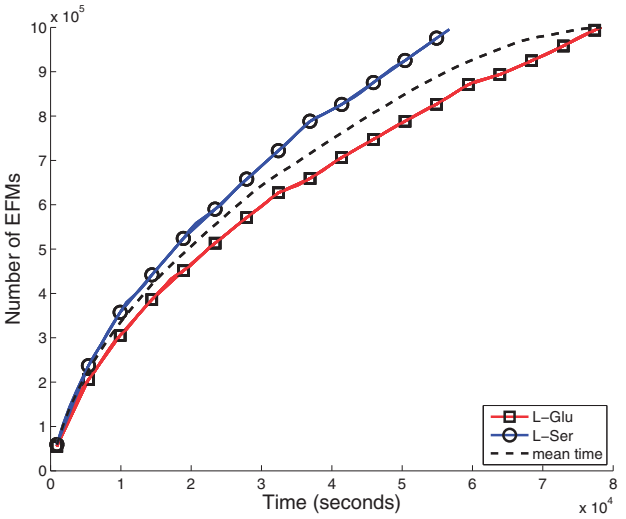

Fig. 3. Computation time for the calculation of 1 million EFMs

$$f(x) = \beta x^{\alpha} \tag{7}$$

where x is the number of calculated EFMs and $f(x)$ is the accumulated time for the calculation of x EFMs.

As expected, the exponent α in Equation (7) is smaller than one, being 0.5659 for L-Glu and 0.5964 for the L-Ser process. For both curves, the R-square values were above 0.99, highlighting the goodness of fit of the obtained regressions. Hence, this is a totally acceptable scenario allowing us to continue increasing the number of calculated EFMs.

Regarding memory usage, storing a flux array v with a precision of 4 bytes requires around 12 kb per EFM in the GSMN of *E.coli* presented in Feist *et al.* (2007), comprising 3234 reactions and 1668 metabolites. Therefore, directly storing in memory 1 million EFMs requires approximately 12 GB, which exceeds the capacity of an ordinary desktop computer. Nevertheless, TreeEFM incorporates several strategies for optimizing the memory usage, such as removing nodes with the worst scores to make space available for new nodes.

3.2.2 Acetate overflow

Having a large number of EFMs does not constitute a relevant contribution *per se*. It is essential to develop methods and tools considering EFMs as a starting point toward answering relevant biological questions. Several methods quantifying the relevance of an EFM can be found in the literature such as the one presented in Rezola *et al.* (2013) wherein a score is assigned to each EFM using experimental data. In essence, a *P*-value is calculated using a multivariate hypergeometric distribution based on a three-level enzyme classification defined from a given gene/protein expression dataset. Further details are given in section 2.5. We consider here the gene and protein expression data corresponding to the acetate overflow metabolisms provided by Valgepea *et al.* (2010). Final reaction classification is calculated as in Pey *et al.* (2013).

Table 2 summarizes the results for the mean *P*-values for the 1 million EFMs corresponding to each energetic amino acid. We can differentiate two groups: the first group comprises glycine (Gly) and threonine (L-Thr) while the second group comprises Proline (L-Pro), L-Ser, L-Glu, glutamine (L-Gln), Aspartate (L-Asp) and Asparagine (L-Asn). Note that the average *P*-values in the first group are between 4 and 17.6 times smaller than the second group.

The same classification of the energetic amino acids was previously reported by Han *et al.* (2002). In brief, Han and coworkers identified Gly and L-Thr as the unique energetic amino acids that delay the overflow metabolism when supplied into the growth media. Therefore, it is reasonable to assume that there is a saturated metabolic mechanism involving Gly and L-Thr. These two metabolites are close to each other because the enzyme *L-Threonine aldolase* (THRAi) consumes L-Thr to produce a molecule of Gly and a molecule of acetaldehyde (AcAld). In addition, considering the expression dataset in Valgepea *et al.* (2010), THRAi is classified as a highly expressed reaction. Conversely, AcAld can be directly transformed into acetate through the enzyme *Aldehyde dehydrogenase*, which is also highly expressed.

Escherichia coli has several mechanisms producing Gly, with *serine hydroxymethyltransferase* (SHMT) being preferred when glucose is the sole carbon source (Liu *et al.*, 1998). However, THRAi has been highlighted as an alternative mechanism producing Gly when SHMT is not properly operating (Liu *et al.*, 1998). Therefore, when biomass is produced at a very high rate, SHMT may become saturated, leading to the activation of alternative mechanisms aiming to increase the biomass production rate. THRAi, despite being a less efficient pathway, can provide the required support to meet cellular demand for Gly. As a by-product, a molecule of AcAld is produced, which is subsequently transformed into acetate and excreted outside of the cell, which is the most representative phenotype of the acetate overflow metabolism (Vemuri *et al.*, 2006).

Further experimental analyses are essential to validate this hypothesis, but for now, we can conclude that the obtained EFMs properly capture key properties of the acetate overflow metabolism and may lead us to pose novel metabolic mechanisms.

3.3 TreeEFM in larger networks

The size of metabolic network reconstructions is increasing day-by-day. This certainly leads to more accurate simulations but entails a growth in the computation time. In this section, we study the performance of TreeEFM in the context of three of the largest metabolic networks available: (i) Recon2, a highly comprehensive human generic metabolic reconstruction (Thiele *et al.*, 2013); (ii) the SEED model database that accounts for a pool of reactions present in a large variety of organisms (Henry *et al.*, 2010) and (iii) the multi-scale *E.coli* model of metabolism and macromolecular synthesis, referred to as MacroModel (Thiele *et al.*, 2012). After splitting reversible reactions into two irreversible steps, Recon2, SEED and MacroModel involved 11 175, 21 167 and 94 038 reactions and 5051, 17 648 and 62 212 metabolites, respectively. In each of these metabolic models, the time evolution during the calculation of 10 000 EFMs consuming glucose was studied, as shown in Figure 4.

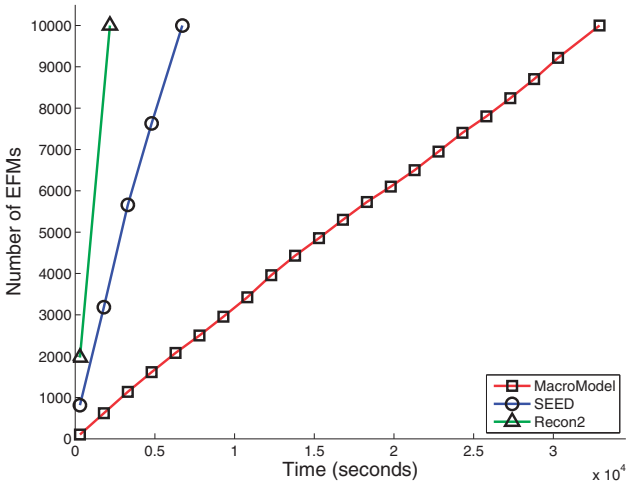

Fig. 4. Computation time for calculating 10 000 EFMs in Recon2, SEED and MacroModel

It can be observed that the computation time grows linearly with the number of calculated EFMs ($R^2 = 0.9924$, 0.9954 and 0.9993 for Recon2, SEED and MacroModel, respectively). Therefore, we can consider that in these three metabolic networks, the time required to enumerate a new EFM remains constant even after calculating 10 000 of them. At some point, as observed in Figure 3, we would expect the computation time to grow exponentially with the number of solutions, but in Figure 4 that scenario is not yet reached and more EFMs could be computed in a reasonable time.

We also analysed how the computation time evolves when the number of reactions increases for the calculation of 1 million EFMs. Let τ be the total time consumed in a simulation divided by the number of reactions in the GSMN and the number of obtained EFMs. If Simplex has an average-case polynomial complexity, the value τ in each network is expected to be similar. For Recon2, SEED and MacroModel networks a τ equal to 1.95×10^{-5}, 3.18×10^{-5} and 3.49×10^{-5} s/(EFM·reaction) was obtained when we calculated 1 million EFMs.

We can conclude that there is not a combinatorial explosion in complexity with the number of reactions. This is clearly illustrated with the MacroModel, in which, although quadrupling the number of reactions in SEED database, τ increases by less than 10%. Overall, this analysis shows that our new approach is particularly suitable for large metabolic networks.

4 Discussion

The methodology presented here based on linear programming and an efficient tree search procedure, is a general one for enumerating extreme points in a polytope. In this work, we focused on the computation of EFMs. Our approach showed a computational advantage with respect to EFMEvolver, which is a key issue in the analysis of GSMNs involving thousands of reactions.

The usefulness of the TreeEFM approach is validated with the successful enumeration of 1 million EFMs in the genome-scale metabolic reconstruction of *E.coli* (Feist *et al.*, 2007). Our analysis shows that the computation time is not a limiting resource but that effective strategies are needed to deal with the memory required for the large amount of generated data. Although TreeEFM implements effective strategies for reducing memory requirement, for computation of EFMs in large GSMNs, it may be necessary to develop effective strategies for compressing the EFMs.

The analysis in section 3.3 shows that TreeEFM is able to adapt, with an affordable computation cost, to future increasingly large metabolic models. For instance, metagenomic studies, in which several organisms are modeled at a time, will provide us with meta-networks encompassing several times the number of reactions and metabolites present in current GSMNs. TreeEFM is a suitable tool for such scenarios.

EFMs are an elegant theoretical concept that provide insight into different metabolic questions, as illustrated here with acetate overflow. To improve the relevance of the conclusions provided by an EFM-based study, it is essential to contextualize the analysis in the light of *omics* data. However, the required methodology to take advantage of large and diverse sets of EFMs is currently emerging, for example, for gene expression data integration (Rezola *et al.*, 2014). The open availability of TreeEFM will certainly play a role, as such approaches improve their predictions when they are provided with a more diverse set of EFMs.

Acknowledgements

The authors thank Kaspar Valgepea for providing us with the gene/protein expression data in the acetate overflow metabolism, as well as Juliane Gebauer and Christoph Kaleta for kindly providing us with the data in Table 1. The authors thank the anonymous referees for their valuable comments and suggestions, which improved the manuscript significantly.

Funding

The work of Jon Pey and Luis Tobalina was supported by the Basque Government. In addition, this work was partially supported by the Fundación Séneca (Agencia Regional de Ciencia y Tecnología, Región de Murcia) [15290/PI/2010]; the Spanish MEC and European Commission FEDER [TIN2012-31345]; and the Minister of Economy and Competitiveness of Spain [BIO2013-48933].

Conflict of interest: none declared.

References

Dantzig,G.B. *et al.* (1955) The generalized simplex method for minimizing a linear form under linear inequality restraints. *Pacific J. Math.* **5**, 183–195.

De Figueiredo,L.F. *et al.* (2008) Can sugars be produced from fatty acids? A test case for pathway analysis tools. *Bioinformatics* **24**, 2615–2621.

De Figueiredo,L.F. *et al.* (2009) Computing the shortest elementary flux modes in genome-scale metabolic networks. *Bioinformatics* **25**, 3158–3165.

Feist,A.M. *et al.* (2007) A genome-scale metabolic reconstruction for *Escherichia coli* K-12 MG1655 that accounts for 1260 ORFs and thermodynamic information. *Mol. Syst. Biol.* **3**, 121.

Gebauer, J. *et al.* (2012) Detecting and investigating substrate cycles in a genome - scale human metabolic network. *FEBS J.* **279**, 3192–3202.

Gschaedler,A. and Boudrant, J. (1994) Amino acid utilization during batch and continuous cultures of *Escherichia coli* on a semi-synthetic medium. *J. Biotechnol.* **37**, 235–251.

Han,L. *et al.* (2002) Effect of glycine on the cell yield and growth rate of *Escherichia coli*: evidence for cell-density-dependent glycine degradation as determined by ^{13}C NMR spectroscopy. *J. Biotechnol.* **92**, 237–249.

Henry,C.S. *et al.* (2010) High-throughput generation, optimization and analysis of genome-scale metabolic models. *Nat. Biotechnol.* **28**, 977–982.

Hsu,P.P. and Sabatini,D.M. (2008) Cancer cell metabolism: Warburg and beyond. *Cell* **134**, 703–707.

Hunt,K.A. *et al.* (2014) Complete enumeration of elementary flux modes through scalable, demand-based subnetwork definition. *Bioinformatics* **30**, 1569–1578.

Joyce,A.R. and Palsson,B.Ø. (2006) The model organism as a system: integrating 'omics' data sets. *Nat. Rev. Mol. Cell Biol.* **7**, 198–210.

Kaleta,C. *et al.* (2009) EFMEvolver: computing elementary flux modes in genome-scale metabolic networks. *Lect. Notes Inf.* **P-157**, 179–189.

Klamt,S. and Stelling,J. (2002) Combinatorial complexity of pathway analysis in metabolic networks. *Mol. Biol. Rep.* **29**, 233–236.

Land,A.H. and Doig,A.G. (1960) An automatic method of solving discrete programming problems. *Econometric Soc.* **28**, 497–520.

Liu,J. *et al.* (1998) Gene cloning, biochemical characterization and physiological role of a thermostable low-specificity L-threonine aldolase from *Escherichia coli*. *Eur. J. Biochem.* **255**, 220–226.

Lougee-Heimer,R. (2003) The common optimization interface for operations research: promoting open-source software in the operations research community. *IBM J. Res. Dev.* **47**, 57–66.

Nakano,K., *et al.* (1997) Influence of acetic acid on the growth of *Escherichia coli* K12 during high-cell-density cultivation in a dialysis reactor. *Appl. Microbiol. Biotechnol.* **48**, 597–601.

Pey,J. and Planes,F.J. (2014) Direct calculation of Elementary Flux Modes satisfying several biological constraints in genome-scale metabolic networks. *Bioinformatics* **30**, 2197–2203.

Pey,J., Planes, F.J. and Beasley,J.E. (2014) Refining carbon flux paths using atomic trace data. *Bioinformatics* **30**, 975–980.

Pey,J. *et al.* (2013) Integrating gene and protein expression data with genome-scale metabolic networks to infer functional pathways. *BMC Syst. Biol.* **7**, 134.

Price,N. D. *et al.* (2004) Genome-scale models of microbial cells: evaluating the consequences of constraints. *Nat. Rev. Microbiol.* **2**, 886–897.

Rezola,A. *et al.* (2011) Exploring metabolic pathways in genome-scale networks via generating flux modes. *Bioinformatics* **27**, 534–540.

Rezola,A. *et al.* (2013) Selection of human tissue-specific elementary flux modes using gene expression data. *Bioinformatics* **29**, 2009–2016.

Rezola,A. *et al.* (2014) Advances in network-based metabolic pathway analysis and gene expression data integration. *Brief. Bioinf.* doi: 10.1093/bib/bbu009.

Schmidt,B.J., *et al.* (2013) GIM3E: condition-specific models of cellular metabolism developed from metabolomics and expression data. *Bioinformatics* **29**, 2900–2908.

Schuster,S. *et al.* (2000) A general definition of metabolic pathways useful for systematic organization and analysis of complex metabolic networks. *Nat. Biotech.*, **18**, 326–332.

Swartz,J.R. (2001) Advances in *Escherichia coli* production of therapeutic proteins. *Curr. Opin. Biotech.* **12**, 195–201.

Terzer,M. and Stelling,J. (2008) Large-scale computation of elementary flux modes with bit pattern trees. *Bioinformatics* **24**, 2229–2235.

Thiele,I. and Palsson,B.Ø. (2010) A protocol for generating a high-quality genome-scale metabolic reconstruction. *Nat. Protoc.* **5**, 93–121.

Thiele,I. *et al.* (2012) Multiscale modeling of metabolism and macromolecular synthesis in *E. coli* and its application to the evolution of codon usage. *PLoS One*, **7**, e45635.

Thiele,I. *et al.* (2013) A community-driven global reconstruction of human metabolism. *Nat. Biotechnol.* **31**, 419–425.

Trinh,C.T. *et al.* (2009) Elementary mode analysis: a useful metabolic pathway analysis tool for characterizing cellular metabolism. *Appl. Microbiol. Biotechnol.* **81**, 813–826.

Urbanczik,R. and Wagner,C. (2005) An improved algorithm for stoichiometric network analysis: theory and applications. *Bioinformatics* **21**, 1203–1210.

Valgepea,K. *et al*. (2010) Systems biology approach reveals that overflow metabolism of acetate in *Escherichia coli* is triggered by carbon catabolite repression of acetyl-CoA synthetase. *BMC Syst. Biol.* **4**, 166.

Vemuri,G.N. *et al*. (2006) Overflow metabolism in *Escherichia coli* during steady-state growth: transcriptional regulation and effect of the redox ratio. *Appl. Environ. Microbiol.* **72**, 3653–3661.

Von Kamp,A. and Schuster,S. (2006) Metatool 5.0: fast and flexible elementary modes analysis. *Bioinformatics* **22**, 1930–1931.

Werner,T. (2008). Bioinformatics applications for pathway analysis of microarray data. *Curr. Opin. Biotech.* **19**, 50–54.