Improving Extreme Pathway Computations

Steven L. Bell[1] **and Bernhard Ø. Palsson**[2]

Keywords: metabolic networks, stoichiometric matrix, sparse matrices, convex cone

1 Introduction.

The abundance of genomic data available today allows for construction of genome-scale metabolic networks for many organisms. The topology of the type of networks considered here is determined by an $m \times n$ stoichiometric matrix, $\mathbf{S}$, whose rows and columns represent the system's metabolites and reactions, respectively. The dynamics of the system is given by $\dot{\mathbf{x}}(t) = \mathbf{S}\mathbf{v}$, where $\mathbf{x}$ is the m-dimensional vector of metabolite concentrations, "$\dot{}$" denotes time-derivative, and $\mathbf{v}$ is a vector of fluxes which we assume is independent of concentrations and time.

Under the assumption that the system is in steady-state, we have that $\mathbf{S}\mathbf{v} = \mathbf{0}$, and to obtain biologically feasible solutions to this equation, we also impose the condition that $\mathbf{v} \geq \mathbf{0}$. The solution set is a so-called *convex cone* which can be generated by a finite (and unique up to a multiple) number of vectors, i.e., each biologically feasible flux vector (when the system is in steady state) can be expressed as a non-negative linear combination of these *extreme pathways* [2]. The extreme pathways are the edges of the convex cone, or more precisely, they are *conically independent*, i.e., no such vector can be expressed as a non-negative linear combination of any other vectors in the cone.

Given a metabolic network, where the metabolites are represented by the nodes and the edges represent the associated reactions, we compute the extreme pathways using an algorithm presented in [3] (see also [4]). The algorithm uses matrix operations similar to those used in the well-known Gaussian elimination algorithm. Such operations require frequent access to memory, significantly degrading performance of the algorithm if large matrices are stored. Furthermore, the computational time (and the number or extreme pathways) typically grows exponentially as the size of the network grows linearly. Existing implementations work well for relatively small networks, but are of limited use for genome-scale systems. Here, we propose two means to improve the performance of computing extreme pathways: an efficient sparse matrix storage and computational procedure and a scheme to select pivoting columns which we will refer to as the *exponential threshold method*.

2 Description of the algorithm

We now give a simplified version of the extreme pathway algorithm. The algorithm may be described as a sequence of tableaux $T^0, T^1, \ldots, T^N$, where the initial tableau is given by $T^0 = [\,\mathbf{I} \quad \mathbf{S}'\,]$, and the final tableau $T^N = [\,\mathbf{P} \quad \mathbf{0}\,]$. In the initial tableau, $\mathbf{S}$ is the $m \times n$ stoichiometric matrix, "prime" denotes transpose, and $\mathbf{I}$ is the $n \times n$ identity matrix (and hence T^0 is an $n \times (n + m)$ matrix). The final tableau, T^N, for some $0 < N \leq m$, consists of the matrix $\mathbf{P}$ whose rows are the extreme pathways, and the zero matrix, $\mathbf{0}$, which has m columns. Converting the right hand matrix $\mathbf{S}'$ to the zero matrix is done column by column

[1]Department of Bioengineering, University of California, San Diego, 9500 Gilman Drive, San Diego, CA. 92093-0412 E-mail: `sbell@ucsd.edu`

[2]Department of Bioengineering, University of California, San Diego, 9500 Gilman Drive, San Diego, CA. 92093-0412 E-mail: `bpalsson@bioeng.ucsd.edu`

using elementary row operations, each tableaux corresponding to a column. For $1 \leq i \leq N$, the tableau T^i is obtained from T^{i-1} by first choosing a pivoting column of the right hand matrix (originating from $\mathbf{S}'$) to zero out, column j, say. Suppose there are `pos` positive, `neg` negative, and `z` zero elements in column j. First, the `z` rows of T^{i-1} containing a zero in column j are copied to T^i. Then each of the `pos` rows which contain positive elements in column j is combined (using an elementary row operation) with each of the `neg` rows which contain negative elements in column j so that a zero is produced in column j of T^i. More precisely, if $T^{i-1}_{s,j} > 0$ and $T^{i-1}_{t,j} < 0$ for some s and t, then $|T^{i-1}_{t,j}|\,T^{i-1}_{s} + |T^{i-1}_{s,j}|\,T^{i-1}_{t}$ is the new row to be added to T^i. (Here, T^{i-1}_s denotes the s^{th} row, $T^{i-1}_{s,j}$ is the (s,j)-element in the tableau T^{i-1}, and $|x|$ is the absolute value of x.) Finally, all rows which are not conically independent are deleted from T^i. Hence, the number of rows in T^i is at most $\mathtt{z} + \mathtt{neg} * \mathtt{pos}$.

3 Sparse matrices and the exponential threshold method

The stoichiometric matrix contains few non-zero elements (about 5%) and although the final pathway matrix is less sparse (about 25% non-zero elements), we believe that sparse matrix methods used with success in similar algorithms, such as Gaussian elimination, can also benefit implementations of the extreme pathway algorithm. Memory is conserved since only non-zero entries of matrices are stored, and computational performance is improved since row operations use only non-zero elements of the vectors. There are many storage schemes for sparse matrices each with its own type of data structures, and the problem usually dictates which particular scheme is employed ([1], pg. 37). From Section 2, we see that for the extreme pathway algorithm it is important that individual column elements and whole rows can be accessed efficiently, so the storage method must be designed with these objectives in mind.

In Section 2 we saw that the aim of the extreme pathway algorithm is to zero out the columns of the tableaux using elementary row operations. Furthermore, for a fixed iteration, the number of rows to be added to the next tableau depends on `pos` and `neg`, the number of positive- and- negative elements, respectively, in the pivoting column. Our proposed method dictates that a column is processed only if $\mathtt{pos} * \mathtt{neg} < \mathbf{T}$, where $\mathbf{T}$ is some threshold. When all the columns of a tableau have been tested, the threshold is raised (exponentially) and the process starts anew, if there are any remaining columns to be zeroed out. The rationale for the threshold method is that the sparse columns are processed first since they require less computation, and postponing processing the denser columns may result in some of their non-zero elements being zeroed out by the elementary row operations performed in the earlier iterations, i.e., doing less work early may reduce the amount of work that has to be done later, resulting in less overall work. Preliminary results seem to indicate that this is indeed the case.

References

[1] Duff, I. S., Erisman, A. M. and Reed, J. K. 1986. *Direct Methods for Sparse Matrices.* New York: Oxford University Press.

[2] Rockafellar, R. 1970. *Convex Analysis.* Princeton: Princeton University Press.

[3] Schilling, C. H., Letscher, D. and Palsson, B. Ø. 2000. Theory for the systemic definition of metabolic pathways and their use in interpreting metabolic function from a pathway-oriented perspective. *Journal of Theoretical Biology* 203:249–283.

[4] Schuster, R. and Schuster, S. 1993. Refined algorithm and computer program for calculating all non-negative fluxes admissible in steady states of biochemical reaction systems with or without some flux rates fixed. *Computational and Applied Bioscience* 9:79–85.