Metatool 5.0: fast and flexible elementary modes analysis

Axel von Kamp* and Stefan Schuster
Department of Bioinformatics, Friedrich-Schiller-University Jena, 07743 Jena, Germany

ABSTRACT
Summary: Elementary modes analysis is a powerful tool in the constraint-based modeling of metabolic networks. In recent years, new approaches to calculating elementary modes in biochemical reaction networks have been developed. As a consequence, the program Metatool, which is one of the first programs dedicated to this purpose, has been re-implemented in order to make use of these new approaches. The performance of Metatool has been significantly increased and the new version 5.0 can now be run inside the GNU octave or Matlab environments to allow more flexible usage and integration with other tools.

Availability: The script files and compiled shared libraries can be downloaded from the Metatool website at http://pinguin.biologie.uni-jena.de/bioinformatik/networks/index.html. Metatool consists of script files (m-files) for GNU octave as well as Matlab and shared libraries. The scripts are licensed under the GNU Public License and the use of the shared libraries is free for academic users and testing purposes. Commercial use of Metatool requires a special contract.

Contact: kamp@minet.uni-jena.de

Elementary modes analysis has become an important method for the study of metabolic networks (Schuster et al., 2000, 2002). It allows one to systematically enumerate all independent minimal pathways through a network that are stoichiometrically and thermodynamically feasible. This has been applied to various biochemical systems in view of medical and biotechnological applications (Schwender et al., 2004; Carlson and Sireci, 2004; Papin et al., 2004). As inputs, the reaction stoichiometries and reversibilities have to be known. In addition, the metabolites have to be classified as either external or internal. External metabolites are assumed to be buffered while internal metabolites have to be balanced by production and consumption reactions in the network. The stoichiometric coefficients are collected into a matrix $N$, where rows correspond to internal metabolites and columns to reactions. Note that the reaction reversibilities are not integrated into the matrix $N$, but are taken into account later.

Elementary modes are flux distributions of the metabolic network in steady state (expressed by the equation $Nv = 0$) with the fluxes through irreversible reactions going in the appropriate direction. In addition, elementary modes have to be independent of each other, which means that the reactions of any elementary mode must not be a subset of the reactions of any other mode.

In the previous versions of Metatool, the algorithm described in Schuster et al. (2000) was used to calculate the elementary modes. In contrast, the current version is based on the algorithm proposed by Urbanczik and Wagner (2005), which empirically shows a higher performance. Both algorithms can be seen as variants of the double-description method for enumerating all extreme rays of a polyhedral cone (Gagneur and Klamt, 2004). Moreover, they are similar to methods proposed in the theory of Petri nets (Colom and Silva, 1991). The current implementation (version 5.0) differs from the description by Urbanczik and Wagner (2005) in that the test of candidate modes for their mutual independence is now performed by an algebraic test.

The previous versions of Metatool were stand-alone programs compiled from C/C++ source code with simple text input and output. This has the drawback that it is difficult for the user to make changes to the calculation procedure because this would require advanced programming knowledge. Another drawback results from the cumbersome exchange of input and output via text files with other programs, for instance for post-processing the results. Therefore, Metatool is now embedded in the math environments GNU octave and Matlab. It consists of script files that are compatible with both programs and shared libraries that are specifically compiled for each program and operating system. To understand and modify these scripts only basic programming knowledge is required. The shared libraries are built from C++ sources that have been completely rewritten for this purpose.

As input, Metatool can still read the standard input files used by the previous versions. Additionally, a stoichiometric matrix and a vector that specifies reaction reversibilities can be directly used as inputs. Because Metatool is now embedded in a math environment, the results are returned as a data structure which can be processed with the standard commands of these environments. Furthermore, the results can be analyzed by custom scripts. Thus it is, for example, easily possible to calculate the yields of the elementary modes with respect to specified substrate/product pairs. The usage of Metatool and the format of the input/output data structures are described in more detail on the Metatool web pages.

With the help of the Matlab compiler, a stand-alone version of Metatool can be created. Such a program will require some additional Matlab libraries that can be distributed freely so that it is not necessary to have a Matlab installation in order to use it. By default, this program reads a standard Metatool input file and produces an output file in a format similar to the previous versions. Optionally, the parser of Metatool can be used separately. One of the previous versions of Metatool has been embedded in the Systems Biology Workbench (Sauro et al., 2003). The new SBW–Matlab interface (Wellock et al., 2005) opens up the possibility to easily integrate the current version of Metatool into the SBW.
The central routines can now optionally be used by the FluxAnalyzer/CellNetAnalyzer (Klamt et al., 2003), which thereby profits from the increased performance of Metatool.

The previous versions of Metatool came in two variants, one for integer and one for floating-point calculations. The current version internally distinguishes stoichiometric matrices that only contain integers from those that also contain floating-point numbers. When only integers are present in the input, the results will also be integers. In both cases, floating-point numbers with double precision are used to represent numbers because this is the default type of octave and Matlab. In the case of integer calculations, this allows for an exact representation of numbers up to \(2^{53} - 1\). To prevent the explosion of integer coefficients through successive multiplications, the integer vectors are normalized by their greatest common divisor after each operation.

As mentioned above, Metatool 5.0 is based on the new algorithm proposed by Urbanczik and Wagner (2005). This has empirically been found to be faster for biochemical networks than the previously used algorithm presented in Schuster et al. (2000, 2002). One reason for the increased performance is the smaller initial tableau, because it contains the null-space of the stoichiometry matrix rather than the matrix itself. In addition, more vectors that are elementary already are carried over from the current tableau to the next one at each iteration step.

In contrast to the Urbanczik and Wagner (2005) algorithm, the test for mutual independence of elementary modes is made by checking the rank of a submatrix of the stoichiometry matrix. This algebraic test shows a significantly higher performance than the originally used combinatorial test (Klamt et al., 2005). The reason for this is that when using the combinatorial test each candidate mode has to be compared with all other modes. Therefore, this test scales, for each examined mode, with the number of preliminary modes, which is usually growing while the algorithm is running. The algebraic test on the other hand is independent of other modes and its worst-case time complexity is limited by parameters that are constant during run time.

To the best of our knowledge, Metatool is currently the fastest program for calculating elementary modes. The performance stems not only from algorithmic innovations but also from an efficient implementation in C++ that can be optimized well by modern compilers. The increase in speed and the reduction in memory requirements allows one to tackle larger reaction systems than before. For example, for a system of 112 reactions and 89 internal metabolites, describing the central metabolism in *Escherichia coli*, Metatool computes 2 450 787 elementary modes. This takes 87 min on a 2.4 GHz PC (Klamt et al., 2005). It was impossible to calculate such systems with earlier versions of Metatool.

Like earlier versions of Metatool, version 5.0 computes also other structural invariants besides elementary modes, such as conservation relations (cf. Heinrich and Schuster, 1996) and enzyme subsets (Pfeiffer et al., 1999) and fits a power law to the connectivity distribution of metabolites (Jeong et al., 2000).

The current reimplementation of Metatool basically produces the same output as the previous versions, but it is now possible to adapt the script files to one’s needs with only basic programming skills. The enormous reduction in running time allows one to tackle larger reaction systems than before. This is of interest in view of the current efforts in analyzing genome-scale and whole-cell models.

Current developments to make use of distributed computing as described in Klamt et al. (2005) have led to a preliminary but already usable implementation. The use of distributed computation will not only speed up the calculations by using several processors in parallel, but will also make it possible to calculate even larger systems because each subprocess requires less resources (especially memory) than the complete task. Because the scripts for distributed computing are still in development, they are at present not included in Metatool.

An interesting question for future studies is whether the algorithm developed by Urbanczik and Wagner (2005) is faster for all chemical reaction systems than that proposed by Schuster et al. (2000). The fact that biochemical systems have arisen from biological evolution might imply special properties [e.g. the property to be scale-free (Jeong et al., 2000)] that favor one algorithm over another.

ACKNOWLEDGEMENTS
We would like to thank Mihail Pachkov for his contributions to the Metatool program as well as Steffen Klamt and Julien Gagneur for helpful comments.

Conflict of Interest: none declared.

REFERENCES


