BIOINFORMATICS APPLICATIONS NOTE

Vol. 26 no. 14 2010, pages 1788–1789
doi:10.1093/bioinformatics/btq263

Systems biology

Advance Access publication May 25, 2010

A parallel algorithm to compute chemical organizations in biological networks

Florian Centler[1,*], Christoph Kaleta[2], Pietro Speroni di Fenizio[3] and Peter Dittrich[4]

[1]Department of Environmental Microbiology, UFZ – Helmholtz Centre for Environmental Research, Permoserstraße 15, D-04318 Leipzig, [2]Department of Bioinformatics, Friedrich-Schiller University Jena, Ernst-Abbe-Platz 2, D-07743 Jena, Germany, [3]Coimbra University, Engenharia Informática, Coimbra, Portugal and [4]Friedrich-Schiller University Jena, Bio Systems Analysis Group, Ernst-Abbe-Platz 2, D-07743 Jena, Germany

Associate Editor: Jonathan Wern

ABSTRACT

Summary: Analysing genome-scale *in silico* models with stoichiometry-based methods is computationally demanding. The current algorithms to compute chemical organizations in chemical reaction networks are limited to small-scale networks, prohibiting a thorough analysis of large models. Here, we introduce a parallelized version of the constructive algorithm to determine chemical organizations. The algorithm is implemented in the Standard C programming language and parallelized using the message passing interface (MPI) protocol. The resulting code can be executed on computer clusters making use of an arbitrary number of processors. The algorithm is parallelized in an embarrassing parallel manner, providing good scalability.

Availability: An implementation of the algorithm including source code can be obtained from http://www.minet.uni-jena.de/csb/prj/ot/tools

Contact: florian.centler@ufz.de

Received on April 13, 2010; revised on May 12, 2010; accepted on May 16, 2010

1 INTRODUCTION

Stoichiometry-based network analysis methods like flux balance analysis (Edwards *et al.*, 2001), elementary mode analysis (Schuster *et al.*, 1999) and chemical organization analysis (Dittrich and Speroni di Fenizio, 2007) have proven useful in gaining a better understanding of the functioning of biochemical systems at the systems level (Centler *et al.*, 2008; Price *et al.*, 2004; Trinh *et al.*, 2009). With the ongoing efforts to elucidate more and more biochemical details, genome-scale metabolic networks have become available for an increasing number of organisms (Feist *et al.*, 2009). Analysis of such networks with species and reaction numbers ranging in the thousands pose a new computational challange (Kaleta *et al.*, 2009a). To meet this challange, we introduce a parallelized version of the constructive algorithm to determine chemical organizations. A chemical organization is a set of network species constituting a subsystem of the whole network, that fulfills two properties: (i) algebraic closure and (ii) self-maintenance. The first property ensures that the species of an organization cannot generate a species that is not already contained in that set, while the second property ensures that a flux vector exists such that all species of the organization are produced at a sufficient rate for their survival. Hence, organizations represent species combinations that are likely to persist over time; novel species cannot occur due to the closure property and species do not vanish due the self-maintenance property. Every organization is also a semi-organization, a closed set that is semi-self-maintaining. Semi-self-maintenance is a weaker condition than self-maintenance. It merely requires that for each species of the set for which a reaction exists in which it is consumend, a reaction also exists in which it is produced. The set of organizations forms a hierarchy that can be visualized in a Hasse diagram. Organizations are vertically arranged according to size, with the organization containing the fewest species at the bottom. Links are drawn between two organizations, if the upper organization contains all species of the bottom organization and there is no other organization between them. Formal definitions of these concepts can be found in Dittrich and Speroni di Fenizio (2007).

*To whom correspondence should be addressed.

2 METHODS

The constructive algorithm to compute chemical organizations (Centler *et al.*, 2008) consists of two steps. In the first step, all semi-organizations of the reaction network are determined in a bottom-up fashion. The central function of the algorithm determines, given a semi-organization, the smallest semi-organizations that contain this semi-organization. This is done by adding new species to the semi-organization to form a larger semi-organization. The newly found semi-organizations are added to a list of semi-organizations which still need to be processed. The algorithm starts by initializing this list with the smallest semi-organization of the reaction network. In each iteration of the algorithm, the smallest semi-organization is taken from this list and the semi-organizations above it are determined. This is repeated until the list of semi-organizations to be processed is empty. In the second step, every computed semi-organization is tested for the property of self-maintenance. This involves the processing of a linear programming problem for each semi-organization. The parallelization of both steps of the algorithm follows an inverted client–server model. In the first step, the server keeps track of the list of semi-organizations which still needs processing, and distributes the task of determining larger semi-organizations for one semi-organization of that list to the client CPUs as they become available. In the second step, the server evenly distributes the linear programming problems to the available client CPUs. Both the computation of larger semi-organizations, and the computation of the linear programming problem can be executed in isolation, not requiring any communication with other CPUs. This allows for a parallelization in an embarrassingly parallel fashion with good scalability and an expected linear speedup.

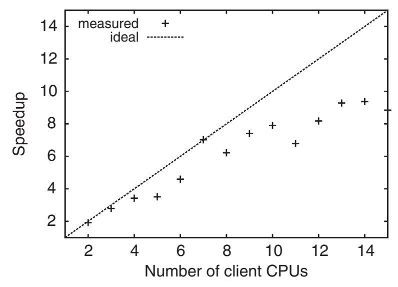

Fig. 1. Speedup for computing the first 100 organizations of a genome-scale network model of *E.coli*.

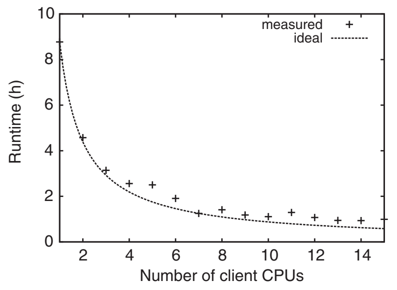

Fig. 2. Runtime for computing the first 100 organizations of a genome-scale network model of *E.coli*.

3 IMPLEMENTATION AND PERFORMANCE

The constructive algorithm was implemented in the Standard C programming language, and parallelization was realized using the message passing interface (MPI) protocol (Message Passing Interface Forum, 1994). The program reads a reaction network model in SBML format (Finney and Hucka, 2003) using libSBML (Bornstein *et al.*, 2008) and can compute organizations as well as connected organizations (Centler *et al.*, 2008). For solving linear programming problems, either the lpsolve library (Berkelaar *et al.*, 2005) or GLPK (Makhorin, 2003) can be used. To avoid processing the same set of species twice when trying to expand a semi-organization, a hash structure is used to keep track of already processed species sets. This hash structure is, however, kept local at the client CPUs to avoid excessive communication between processors. The drawback is that the effectiveness of the hash facility decreases if using many CPUs, as the same species set might then be processed by more than one client CPU. The code can be configured to write restart files and terminate after a predefined execution time for cluster systems where runtime is restricted. To assess the performance of the parallelized algorithm, we record the runtime to compute the first 100 organizations of a genome-scale reaction network model of *Escherichia coli* (Scenario 1 in Centler *et al.*, 2008) using 1–16 CPUs. The speedup is almost ideal for up to about ten client CPUs, but slows down for higher numbers of processors as communication demand increases and efficiency of the hash facility decreases (Fig. 1). Nevertheless, absolute runtime stays close to ideal parallelization, even for large numbers of CPUs (Fig. 2), indicating good scalability.

4 CONCLUSION

The computational time required to compute the organizations for a given reaction network model does not solely depend on the network size, but also its structure (Centler *et al.*, 2008). Being NP-hard, the computation of all organizations will remain unfeasable for certain networks, even when using the parallelized version of the algorithm on a computer cluster. Nevertheless, the parallelized version reduces runtime to practical limits for large models, especially if many CPUs are available. For example, all connected organizations of three previously not tractable networks (Kaleta *et al.*, 2009b) could be determined using the parallelized algorithm.

Funding: This work was supported by the German Research Foundation (DFG) Grant Di 852/4. Additional funding was provided by the Helmholtz Association Grant VG-NG-338 ('GReaT MoDE').

Conflict of Interest: none declared.

REFERENCES

Berkelaar,M. *et al.* (2005) lp solve: open source (mixed-integer) linear programming system, version 5.5. Available at http://tech.groups.yahoo.com/group/lp_solve/ (last accessed date February 9, 2009).

Bornstein,B.J. *et al.* (2008) LibSBML: an API library for SBML. *Bioinformatics*, **24**, 880–881.

Centler,F. *et al.* (2008) Computing chemical organizations in biological networks. *Bioinformatics*, **24**, 1611–1618.

Dittrich,P. and Speroni di Fenizio,P. (2007) Chemical organization theory. *Bull. Math. Biol.*, **69**, 1199–1231.

Edwards,J.S. *et al.* (2001) In silico predictions of *Escherichia coli* metabolic capabilities are consistent with experimental data. *Nat. Biotechnol.*, **19**, 125–130.

Feist,A.M. *et al.* (2009) Reconstruction of biochemical networks in microorganisms. *Nat. Rev. Microbiol.*, **7**, 129–143.

Finney,A. and Hucka,M. (2003) Systems Biology Markup Language: level 2 and beyond. *Biochem. Soc. Trans.*, **31**(Pt 6), 1472–1473.

Kaleta,C. *et al.* (2009a) Can the whole be less than the sum of its parts? pathway analysis in genome-scale metabolic networks using elementary flux patterns. *Genome Res.*, **19**, 1872–1883.

Kaleta,C. *et al.* (2009b) Using chemical organization theory for model checking. *Bioinformatics*, **25**, 1915–1922.

Makhorin,A. (2003) GLPK (GNU linear programming kit). Available at http://www.gnu.org/software/glpk/ (last accessed date May 10, 2010).

Message Passing Interface Forum (1994) MPI: a message-passing interface-standard. *Technical Report UT-CS-94-230*, University of Tennesee, Knoxville, Tennessee.

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

Schuster,S. *et al.* (1999) Detection of elementary flux modes in biochemical networks: a promising tool for pathway analysis and metabolic engineering. *Trends Biotechnol.*, **17**, 53–60.

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.