Bioinformatics, 31(8), 2015, 1328–1330
doi: 10.1093/bioinformatics/btu814
Advance Access Publication Date: 10 December 2014
Applications Note

Systems biology

MpTheory Java library: a multi-platform Java library for systems biology based on the Metabolic P theory

Luca Marchetti[1,2,3,*] **and Vincenzo Manca**[1,2]

[1]Department of Computer Science, University of Verona, 37134 Verona, Italy, [2]Center for BioMedical Computing, University of Verona, 37134 Verona, Italy and [3]The Microsoft Research – University of Trento, Centre for Computational and Systems Biology (COSBI), 38068 Rovereto (TN), Italy

*To whom correspondence should be addressed.
Associate Editor: Alfonso Valencia

Received on February 13, 2014; revised on November 14, 2014; accepted on December 7, 2014

Abstract

Summary: MpTheory Java library is an open-source project collecting a set of objects and algorithms for modeling observed dynamics by means of the Metabolic P (MP) theory, that is, a mathematical theory introduced in 2004 for modeling biological dynamics. By means of the library, it is possible to model biological systems both at continuous and at discrete time. Moreover, the library comprises a set of regression algorithms for inferring MP models starting from time series of observations. To enhance the modeling experience, beside a pure Java usage, the library can be directly used within the most popular computing environments, such as MATLAB, GNU Octave, Mathematica and R.
Availability and implementation: The library is open-source and licensed under the GNU Lesser General Public License (LGPL) Version 3.0. Source code, binaries and complete documentation are available at http://mptheory.scienze.univr.it.
Contact: luca.marchetti@univr.it, marchetti@cosbi.eu
Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Systems biology has been brought to the forefront of life-science research (Choi, 2007; Kaneko, 2008; Kitano, 2002). At the core of systems biology lies the development of models describing biological systems (Bailey, 1998; Surridge, 2002). The definition of such models is often a laborious task, which involves the use of numerical, statistical and mathematical analysis (Fisher and Henzinger, 2007; Heath and Kavraki, 2009). The Java library herein presented implements a Metabolic P (MP) solution to this task, by combining an initial intuition based on rewriting rules with linear algebra methods and statistical regression. The presented software extends the modeling functionalities provided by *MetaPlab*, an open-source standalone Java application based on MP theory, that has been firstly released in 2008 (http://mplab.sci.univr.it). With respect to MetaPlab, the library provides the implementation of new regression algorithms and supports new types of MP models (e.g. based on differential equations, both Ordinary Differential Equations (ODEs) and Delay Differential Equations (DDEs)).

The MP theory is a mathematical theory introduced in 2004 for modeling biological dynamics (Manca, 2013; Manca *et al.*, 2005). The letter 'P' comes from Păun's P systems, a particular type of computational model inspired to the functioning of biological cells (Păun, 2000). MP models inherit from P systems a native similitude with the functioning of a living cell. This closeness permits to define models that are comprehensible (for both mathematicians and biologists) and consistent with the intrinsic structure of the phenomena under examination. The MP theory has been initially defined for modeling metabolic systems, however, it has then been extended along different directions and now it constitutes a framework for modeling any kind of dynamics (Bollig-Fischer *et al.*, 2014; Manca *et al.*, 2011; Marchetti and Manca, 2012; Marchetti *et al.*, 2014).

2 Methods and functionalities

2.1 Modeling and simulation

The library supports three different types of MP models (see Manca, 2013, Supplementary Material and online documentation of the software for details): *MPF models* (MP models based on fluxes), *MPR models* (MP models based on reaction maps) and *MPDE models* (MP models based on differential equations, ODEs or DDEs). Regardless of the chosen type, every model comprises a set of *variables* that represent quantities of interest. The evolution of variables is then specified by means of a set of *rules* collected in an *MP grammar*, as in the following example:

$$\begin{array}{ll} r_0 : \emptyset \rightarrow A & \varphi_0 : 0.047 + 0.087 \cdot A \\ r_1 : A \rightarrow B & \varphi_1 : 0.002 \cdot A + 0.0002 \cdot A \cdot C \\ r_2 : A \rightarrow C & \varphi_2 : 0.002 \cdot A + 0.0002 \cdot A \cdot B \\ r_3 : B \rightarrow \emptyset & \varphi_3 : 0.04 \cdot B \\ r_4 : C \rightarrow \emptyset & \varphi_4 : 0.04 \cdot C, \end{array} \tag{1}$$

which implements a simple synthetic oscillator, called *Sirius* (Manca, 2013), developed within the MP theory. This uniformity of representation provides important modeling advantages and permits an easy comparison of models that refer to different definitions. In MP grammars each rule is defined by: (i) a *reaction*, which specifies variable transformation (using standard arrow notation, e.g. $2 \cdot H + O \rightarrow H_2O$) and (ii) a *regulator*, which is a formula for deterministically computing the speed of the reaction. In the MP grammar (1) there are five rules over three variables (A, B and C). The first rule introduces matter of type A in the system (input rule), rules one and two apply some transformations and the last two rules expel matter of type B and C, respectively (output rules).

Regulators are used during simulation according to different strategies depending on the model type (MPF, MPR or MPDE). The simpler way of considering regulators is that implemented in MPF models, where regulators directly compute the fluxes of rules in the current state.

The definition of MP model given above has been widely extended to improve the modeling capabilities of the user. Among the most interesting features there is the possibility of adding *parameters* and the possibility of specifying *delays* in regulator formulas. Moreover, Java objects are equipped with a comprehensive Javadoc documentation and they have been designed to maintain the modeling phase simple and consistent with mathematical definitions. The following code defines the MP grammar (1):

```
MpGrammar gr = new MpGrammar();
gr.add(new MpRule("0->A","0.047+0.087*A"));
gr.add(new MpRule("A->B","0.002*A+0.0002*A*C"));
gr.add(new MpRule("A->C","0.002*A+0.0002*A*B"));
gr.add(new MpRule("B->0","0.04*B"));
gr.add(new MpRule("C->0","0.04*C"));
```

Once a model has been created, the library is equipped with state of the art simulators for computing the system dynamics, which have been specifically developed within the MP theory (see Manca, 2013, Supplementary Material and online documentation of the software). The library is also equipped with a complete set of graphical methods for plotting simulation results.

2.2 Regression

Most relevant algorithms implemented in the library are devoted to the inference of models starting from time series of observations.

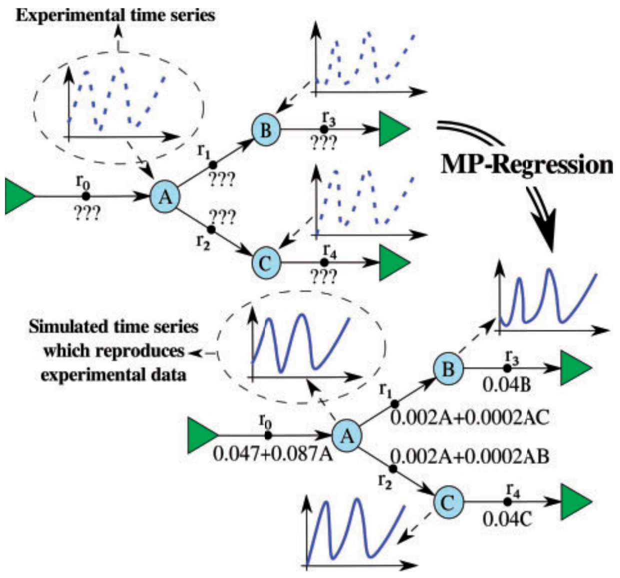

Fig. 1. The regression of an MP model. The library provides a comprehensive set of algorithms specifically developed for inferring models starting from time series of observations. Question marks stand for specific additional information that the user has to provide according to the chosen regression method

Such algorithms apply to different modeling scenarios according to the number of available data and to the previous knowledge that the user wants to encode in the model (see Fig. 1).

In particular, the library provides an implementation of a regression algorithm specifically developed within the MP theory, called *Log-gain Stoichiometric Stepwise regression* (Manca and Marchetti, 2012) and some state of the art algorithms based on non-linear optimization techniques. Among them it is provided an implementation of the *Covariance Matrix Adaptation Evolution Strategy* (a stochastic evolutionary algorithm, Hansen, 2006) and of the Powell's *BOBYQA* algorithm (an iterative procedure particularly well suited for high dimensional problems where derivatives are not available, Powell, 2009).

All the implemented algorithms can be customized and support multi-threading. Moreover, results are equipped with additional measures for evaluating the reliability of models according to statistics (confidence interval and standard deviation of regression parameters, error estimates and Pearson correlations are provided for each model). Models can also be compared according to the information theory Akaike criterion (standard and corrected one) and to the Bayesian information criterion.

2.3 Supported computing environments

To enhance the modeling capabilities of the user, beside a pure Java usage, the library can be directly used within the most popular computing environments, such as MATLAB, GNU Octave, Mathematica and R. In fact, once the library has been imported in the chosen framework, all the implemented objects and methods can be directly used as any other native data structure or command.

The user guide provides a comprehensive set of examples that have been translated for being used in all of the supported environments. Moreover, the library has been also equipped with specific methods for saving and loading models, which can be used to exchange models between different computing environments by enhancing the transfer of knowledge between research groups that are familiar with different modeling platforms.

3 Conclusion

We presented a new open-source Java library for modeling biological dynamics by means of the MP theory. The software is distributed with a Javadoc documentation and a 100 pages user guide that introduces different modeling scenarios and discusses a comprehensive set of examples. The library can be directly used within the most popular computing environments, such as MATLAB, GNU Octave, Mathematica and R.

Funding

This work was supported by the Center for Biomedical Computing (CBMC) of the University of Verona, Italy.

Conflict of Interest: none declared.

References

Bailey,J. (1998) Mathematical modeling and analysis in biochemical engineering: past accomplishments and future opportunities. *Biotechnol. Prog.*, **14**, 8–20.

Bollig-Fischer,A. *et al.* (2014) Modeling time-dependent transcription effects of HER2 oncogene and discovery of a role for E2F2 in breast cancer cell-matrix adhesion. *Bioinformatics*, in press.

Choi,S. (2007) *Introduction to Systems Biology.* Humana Press, Clifton.

Fisher,J. and Henzinger,T. (2007) Executable cell biology. *Nat. Biotechnol.,* **25**, 1239–1249.

Hansen,N. (2006) The CMA evolution strategy: a comparing review. In: Lozano,J. *et al.* (eds) *Towards a New Evolutionary Computation. Advances on Estimation of Distribution Algorithms*. Springer, Berlin Heidelberg, pp. 75–102.

Heath,A. and Kavraki,L. (2009) Computational challenges in systems biology. *Comput. Sci. Rev.*, **3**, 1–17.

Kaneko,K. (2008) *Life: An Introduction to Complex Systems Biology.* Springer, Berlin Heidelberg.

Kitano,H. (2002) Computational systems biology. *Nature*, **420**, 206–210.

Manca,V. (2013) *Infobiotics: Information in Biotic Systems.* Springer, Berlin Heidelberg.

Manca,V. and Marchetti,L. (2012) Solving dynamical inverse problems by means of Metabolic P systems. *BioSystems*, **109**, 78–86.

Manca,V. *et al.* (2005) Evolutions and oscillations of P systems: theoretical considerations and application to biological phenomena. *LNCS*, **3365**, 63–84.

Manca,V. *et al.* (2011) MP modelling of glucose–insulin interactions in the intravenous glucose tolerance test. *Int. J. Nat. Comput. Res.,* **2**, 13–24.

Marchetti,L. and Manca,V. (2012) A methodology based on MP theory for gene expression analysis. *LNCS*, **7184**, 300–313.

Marchetti,L. *et al.* (2014) MP modelling for systems biology: two case studies. In: Frisco,P. *et al.* (eds) *Applications of Membrane Computing in Systems and Synthetic Biology, Series: Emergence, Complexity and Computation,* vol 7, 223–245.

Păun,G. (2000) Computing with membranes. *J. Comput. Syst. Sci.*, **61**, 108–143.

Powell,M. (2009) The BOBYQA algorithm for bound constrained optimization without derivatives. *DAMTP 2009/NA06 Report*. Centre for Mathematical Sciences, Cambridge.

Surridge,C. (2002) Computational biology. *Nature*, **420**, 205.