MetaPlab: A Computational Framework for Metabolic P Systems

Alberto Castellini and Vincenzo Manca

Verona University, Computer Science Dept.,
Strada LeGrazie 15, 37134 Verona, Italy.
{alberto.castellini, vincenzo.manca}@univr.it

Abstract. In this work the formalism of Metabolic P systems has been employed as a basis of a new computational plugin-based framework for modeling biological networks. This software architecture supports MP systems dynamics in a virtual laboratory, called MetaPlab. The Java implementation of the software is outlined and a specific plugin at work is described to highlight the internal functioning of the entire architecture.

1 Introduction

Systems biology copes with the quantitative analysis of biological systems by means of computational and mathematical models which assist biologists in developing experiments and testing hypothesis for complex systems understanding [12,26]. On the other hand, new mathematical and computational techniques have been conceived to infer coherent theories and models from the huge amount of available data.

P systems were introduced by Gh. Păun in [23] as a new computational model inspired by the structure and functioning of the living cell. This approach is rooted in the context of formal language theory and it is essentially based on *multiset* rewriting and *membranes*. In the P systems theory many computational universality results have been achieved [24]. P systems seem especially apt to model biological systems, however their original mathematical setting was too abstract for expressing real biological phenomena.

Metabolic P systems, namely *MP systems*, are a class of P systems proved to be significant and successful for modeling biological phenomena related to metabolism (matter transformation, assimilation and expulsion in living organisms). They were conceived in [19] and subsequently extended in many works [4,5,15,16,17,18]. MP system dynamics is computed by a deterministic algorithm based on the *mass partition principle* which defines the transformation rate of object populations, according to a suitable generalization of chemical laws. This kind of rewriting-based and bio-inspired modeling overcomes some drawbacks of traditional Ordinary Differential Equations (ODE) allowing a new insight about biological processes, which cannot be achieved by using the "glasses" of classical mathematics [2].

Equivalence results have been proved, in [9] and [7,8], between MP systems and, respectively, autonomous ODE and Hybrid Functional Petri nets. The dynamics of several biological processes has been effectively modeled by means of MP systems, among them: the Belousov-Zhabotinsky reaction (in the Brusselator formulation) [4,5], the Lotka-Volterra dynamics [4,19], the SIR (Susceptible-Infected-Recovered) epidemic [4], the Protein Kinase C activation [5], the circadian rhythms, the mitotic cycles in early amphibian embryos [18], a Pseudomonas quorum sensing model [1,6] and the *lac* operon gene regulatory mechanism in glycolytic pathway [7]. In order to simulate MP systems we developed a Java computational tool called *MPsim* [3]. The current release of the software, available at [10], is based on the theoretical framework described above, and it enables the graphical definition of MP models, their simulation and plotting of dynamics curves.

Recent work aims at deducing MP models, for given metabolic processes, from a suitable macroscopic observation of their behaviors along a certain number of steps. Indeed, the search of efficient and systematic methods to define MP systems from experimental data is a crucial point for their use in complex systems modeling. The solution of this *reverse-engineering* task is supported, into the MP systems framework, by the *Log-gain* theory [14,15] which roots in allomeric principle [27]. The main result of this theory is the possibility of computing *reaction fluxes* at each step by solving a suitable linear equations system which combine stoichiometric information with other regulation constraints (by means of a sophisticated method for squaring and making univocally solvable the systems). This means that the knowledge of substances and parameters at each step provides the evaluation of reaction fluxes at that step. In this way, time-series of system states generate corresponding flux series, and from them, by standard regression techniques, the final regulation maps are deduced. This approach turned to be very effective in many cases and recently [20] it provided a model of a photosyntesis phenomenon, deduced by experimental time-series.

What seems to be peculiar of Log-gain theory is the strong connection with biological phenomena and its deep correlation with the allomeric principle, a typical concept of systems biology. Other general standard heuristics or evolutive techniques, already employed to estimate model structures and parameters [25], could be usefully combined with Log-gain method, in fact, the biological inspiration of this theory could add particular specificity to the wide spectrum potentiality of heuristics/evolutionary techniques by imposing constraints able to orientate the search of required solutions.

In this work, we propose a new plugin-based architecture that transforms the software MPsim from a simple simulator to a proper *virtual laboratory* which will be called *MetaPlab*. It assists biologists to understand internal mechanisms of biological systems and to forecast, in silico, their response to external stimuli, environmental condition alterations and structural changes. The Java implementation of MetaPlab ensures the cross-platform portability of the software, which will be released under the GPL open-source license.

Several tools for modeling biological pathways are already available on-line. The most of them are based on ODE, such as *COPASI* [11], which enables to simulate biochemical networks and to estimate ODE parameters. It is a very powerful tool but its usage requires a deep knowledge of molecular kinetics, because the involved differential equations have an intrinsically microscopic nature. Petri nets have been employed by *Cell Illustrator*^{TM} [22], a software which graphically represents biological pathways by graphs and computes their temporal dynamics by a specific evolution algorithm [8]. Unfortunately, this tool can be used just to simulate biological behaviors, but it does not provide any support for the parameter estimation and the analysis of models. The new computational framework we propose in the following, instead, is based on an extensible set of plugins, namely Java tools for solving specific tasks relevant in the framework of MP systems, such as parameter estimation for regulative mechanisms of biological networks, simulation, visualization, graphical and statistical curve analysis, importation of biological networks from on-line databases, and possibly other aspects which would result to be relevant for further investigations.

In Section 2 we introduce some basic principles of MP systems and MP graphs, and we discuss a few biological problems which can be tackled by this modeling framework. Section 3 describes the new plugin-based architecture for a systematic management of these problems, and finally, a plugin for computing MP systems dynamics is presented in Section 4 with a complete description of its functioning.

2 MP systems: model and visualization

MP systems are deterministic P systems developed to model dynamics of biological phenomena related to metabolism. The notion of MP system we consider here generalizes the one given in [15,18].

Definition 1 (MP system) *An MP system is a discrete dynamical system specified by a construct [14]:*

$$M = (X, R, V, Q, \Phi, \nu, \mu, \tau, q_0, \delta)$$

where X, R, V are finite sets of cardinality $n, m, k \in \mathbb{N}$ (the natural numbers) respectively.

1. *$X = \{x_1, x_2, \ldots, x_n\}$ is a set of **substances** (the types of molecules);*
2. *$R = \{r_1, r_2, \ldots, r_m\}$ is a set of **reactions** over X. A reaction r is represented in the arrow notation by a rewriting rule $\alpha_r \rightarrow \beta_r$ with α_r, β_r strings over X. The **stoichiometric matrix** $\mathbb{A}$ stores reactions stoichiometry, that is, $\mathbb{A} = (\mathbb{A}_{x,r} \mid x \in X, r \in R)$ where $\mathbb{A}_{x,r} = |\beta_r|_x - |\alpha_r|_x$, and $|\gamma|_x$ is the number of occurrences of the symbol x in the string γ;*
3. *$V = \{v_1, v_2, \ldots, v_k\}$ is a set of **parameters** (such as pressure, temperature, volume, pH, ...) equipped with a set $\{h_v : \mathbb{N} \rightarrow \mathbb{R} \mid v \in V\}$ of **parameter evolution functions**, where, for any $i \in \mathbb{N}$, $h_v(i) \in \mathbb{R}$ (the real numbers) is the value of parameter v at the step i;*

4. *Q is the set of **states**, seen as functions $q : X \cup V \to \mathbb{R}$ from substances and parameters to real numbers. A general state q can be identified as the vector $q = (q(x_1), \ldots, q(x_n), q(v_1), \ldots, q(v_k))$ of the values which q associates to the elements of $X \cup V$. We denote by $q_{|X}$ the restriction of q to the substances, and by $q_{|V}$ its restriction to the parameters;*
5. *$\Phi = \{\varphi_r : Q \to \mathbb{R} \mid r \in R\}$ is a set of **flux regulation maps**, where for any $q \in Q$, $\varphi_r(q)$ states the amount (moles) which is consumed/produced, in the state q, for every occurrence of a reactant/product of r. We define $U(q) = (\varphi_r(q) \mid r \in R)$ the **flux vector** at state q;*
6. *ν is a natural number which specifies the number of molecules of a (conventional) mole of M, as its **population unit**;*
7. *μ is a function which assigns to each $x \in X$, the **mass** $\mu(x)$ of a mole of x (with respect to some measure unit);*
8. *τ is the **temporal interval** between two consecutive observation steps;*
9. *$q_0 \in Q$ is the **initial state**;*
10. *$\delta : \mathbb{N} \to Q$ is the **dynamics** of the system. It can be identified as the vector $\delta = (\delta(0), \delta(1), \delta(2), \ldots)$, where $\delta(0) = q_0$, and $\delta(i) = (\delta(i)_{|X}, \delta(i)_{|V})$ is computed by the following autonomous first order difference equations:*

$$\delta(i+1)_{|X} = \mathbb{A} \times U(\delta(i)) + \delta(i)_{|X} \tag{1}$$

$$\delta(i+1)_{|V} = (h_v(i+1) \mid v \in V) \tag{2}$$

where $\mathbb{A}$ is the stoichiometric matrix of R over X, of dimension $n \times m$, while $\times$, $+$ are the usual matrix product and vector sum. We introduce the symbol $\delta_{<i}$ to identify the finite vector $(\delta(0), \delta(1), \ldots, \delta(i))$.

MP graphs, introduced in [18], are a natural representation of MP systems modeling biochemical reactions as bipartite graphs with two levels, in which the first level describes the *stoichiometry* of reactions, while the second level expresses the *regulation*, which tunes the flux of every reaction (i.e., the quantity of chemicals transformed at each step) depending on the state of the system (see for example Figure 1).

Given a metabolic process, some elements of a related MP system can be generally defined from a macroscopic observation of the system, while other elements should be computed by means of suitable mathematical techniques. For instance, if we deduce by experimental observations the set of substances (X, item 1 of Definition 1), the chemo-physical parameters (V, item 3) and the reactions (R, item 2) involved in the biological process, and if we know the mathematical laws which regulate these reactions (Φ, item 5), then the system dynamics (δ, item 10) can be computed by the equations (1) and (2).

The *dynamics computation* task, just defined, is only one of several biologically inspired mathematical problems which can be tackled by MP systems. Table 1 collects a few of these tasks focusing on the known and the unknown elements of the related MP system. The second problem we propose is the *flux discovery*, which entails the computation of flux time-series $U(\delta(0)), \ldots, U(\delta(i-1))$ that yield an observed dynamics $\delta_{<i}$ of substances and parameters. A mathematical theory for solving this problem, called Log-gain theory, has been proposed

in [14,15], and some computational tools based on it are currently under construction. A third task is related to *regulation discovery*. It is a regression problem which aims at computing functions Φ which better interpolate a (known) flux time-series $U(\delta(0)), \ldots, U(\delta(i))$. They could be calculated by traditional regression methods [20] as well as by evolutionary computing techniques, such as genetic programming [13] and neural networks [21].

Problem	**Known elements**	**Unknown elements**
Dynamics computation	X, R, V, Φ, q_0	δ
Fluxes discovery	$X, R, V, U(\delta(0)), \delta_{<i}$	$U(\delta(1)), \ldots, U(\delta(i-1))$
Regulation discovery	$R, U(\delta(0)), \ldots, U(\delta(i)), \delta_{<i}$	Φ
Dynamics analysis	$X, R, V, \delta_{<i}$	Statistical params, etc.

Table 1. Some biologically inspired problems which can be tackled within the MP systems framework. Unknown elements should be computed from known elements by means of suitable mathematical techniques and computational tools.

The last problem listed in Table 1 concerns the *dynamics analysis*, a data-mining task which involves the discovery of new biological information from observed dynamics. It is related to the discovery of statistical parameters (e.g., dynamics and flux correlations), the clustering of observed time-series to detect the main actors of a biological system, and the analysis of the dynamical behaviors occurring from different (environmental and structural) conditions.

Of course, it could be very useful to systematically attack these and further bio-inspired problems by means of a set of computational tools suitably developed to satisfy biologists' needs. The software architecture proposed in the next section answers this request by supporting an extendable set of plugins, each dedicated to a specific task.

3 A new plugin-based framework for processing MP systems

Here, we propose a computational structure which enables MetaPlab to systematically tackle the problems introduced in Table 1. Figure 2 depicts this framework, which involves four main layers: the first deals with the model definition and visualization by *MP graphs*, the second is dedicated to the representation and storing of MP systems by a suitable data structure called *MP store*, the third concerns with the processing of these data by means of computational units called *MP plugins*, and finally, the fourth arranges a set of *vistas* which support the MP systems analysis.

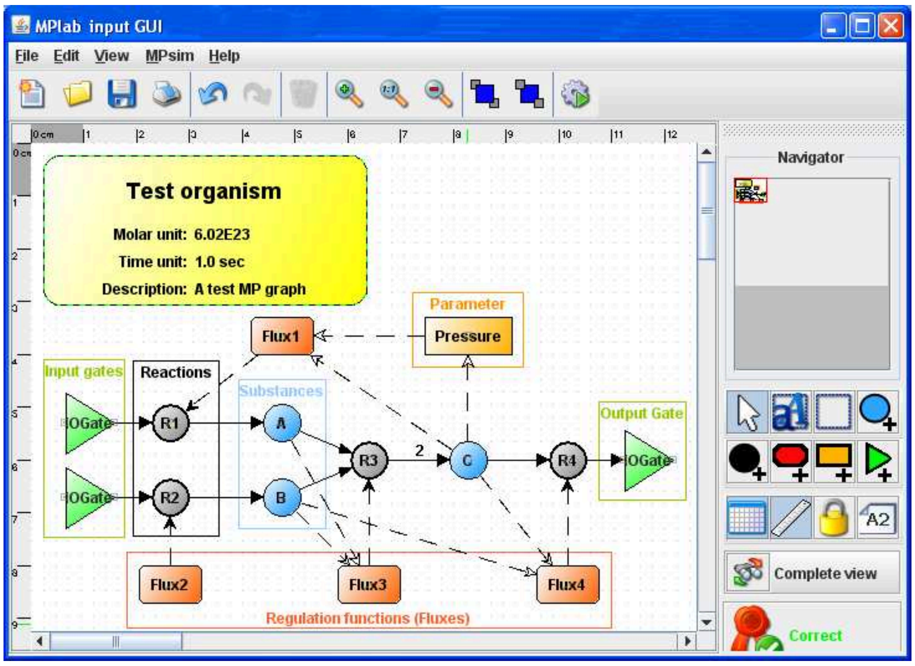


Fig. 1. An MP graph visualized by a graphical user interface of MetaPlab. Frame labels point out MP system elements in the MP graph representation. Substances, reactions and parameters describe the system stoichiometry, while fluxes express the system regulation.

MetaPlab employs this framework to extend the functionalities and to improve the performances of MPsim 3, which only coped with the MP systems simulation. The new extendable data processing layer, described below, turns MetaPlab to be a proper "virtual laboratory" wherein MP plugins act as virtual tools for processing MP systems. In the following we show some implementation details of the new software architecture depicted in Figure 2. A technical description of the whole architecture will be published soon in the MetaPlab User Guide [28].

MP graphs. The leftmost layer of Figure 2 contains the MetaPlab input GUI, also depicted in Figure 1. It is an easy-to-use graphical user interface which takes MP systems as inputs and visualizes them by means of MP graphs. The user drags MP graph elements from the right toolbar of Figure 1 to the central white panel. He or she specifies their internal parameters by filling in suitable fields, and connects the nodes by drawing arcs between them. Importation of observed time-series related to substances, parameters and fluxes dynamics is supported and a "network-oriented" visualization of MP system dynamics is provided by pop-up windows attached to each node. MP graphs loaded by this GUI are stored

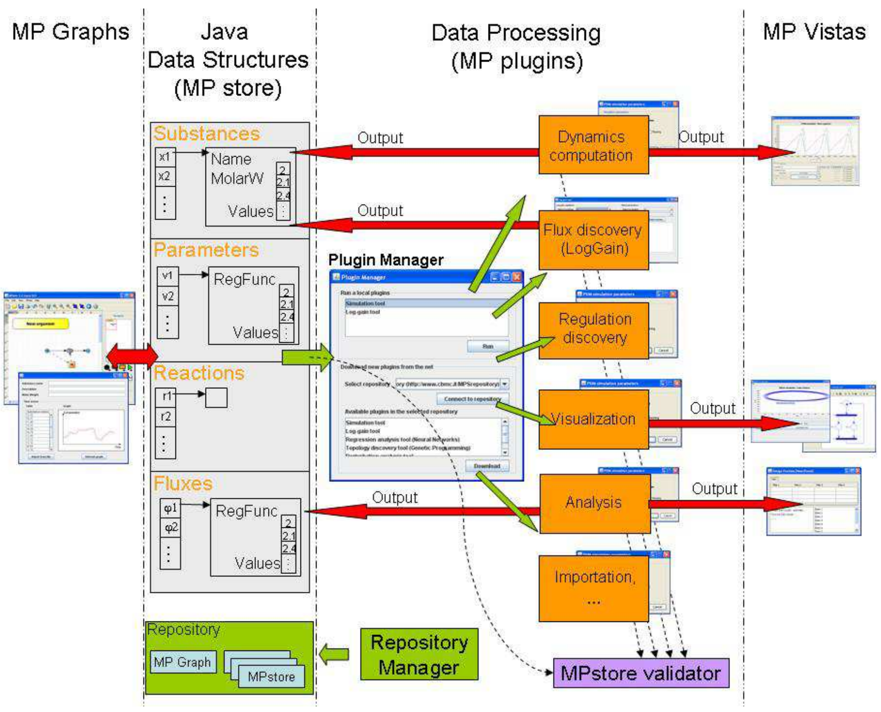


Fig. 2. The MetaPlab framework.

into MP store objects, which are defined below.

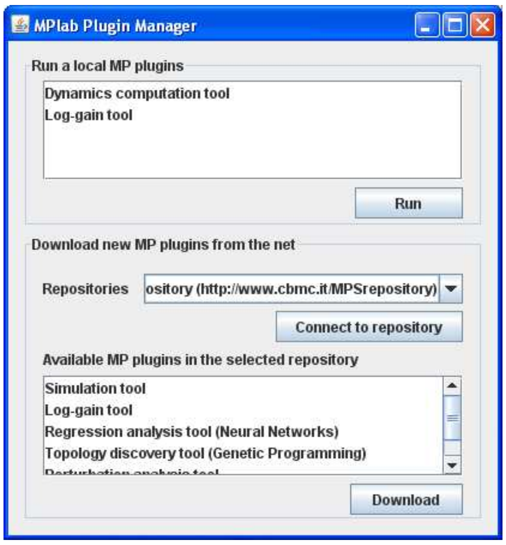
***MP store* data structure.** The second layer of Figure 2 consists of an object-oriented data structure called *MP store*. It has been designed to store all the elements of an MP system by suitable Java objects. Each *substance* $x \in X$ is mapped to an object which stores the substance name x, its molar weight $\mu(x)$ and the time-series of its dynamics $((\delta(i))(x) \mid i \in \mathbb{N})$. Each *parameter* $v \in V$ is implemented by an object having two main fields, the first stores the regulation function h_v as a string, while the second holds the time-series of its dynamics $((\delta(i))(v) \mid i \in \mathbb{N})$ as a vector of real numbers. *Flux* objects are very similar to parameters, in fact each flux stores the regulation function φ_r of a reaction r by a string field, and the related flux time-series $(\varphi_r(\delta(i)) \mid i \in \mathbb{N})$ by a vector. Finally, each *reaction* object implements a reaction rule $r \in R$ by a vector of pointers that address the substances objects involved in r. Each pointer from a reaction r to a substance x has a related multiplicity field which stores the value $\mathbb{A}_{x,r}$ of the stoichiometric matrix. MP store is a crucial point of the new software architecture. Indeed, being the standard input of every plugin, it acts as a bridge between the *MP graph visualization* and the *data processing* layer described in the following.

Data processing. The third layer of Figure 2 represents the core of the new architecture, in fact, it concerns with a plugin-based module coping with the MP systems data processing. This layer is composed by *i)* an extendable set of Java *plugins*, listed on the right of the third layer, each equipped with specific input and (auxiliary) output GUIs, and *ii)* a *Plugin Manager*, depicted on the left of the third layer, which automatically loads MP plugins and makes them available to be launched. *MP plugins* are the MetaPlab processing units. Each of them is involved in a specific computational task, such as the dynamics computation of an MP system, the estimation of its regulation functions, the analysis of its dynamics, or the importation of metabolic pathways from databases. To accomplish one of these (or further) tasks, a plugin gets two possible **inputs**: an MP store object, which addresses the model visualized by the input GUI, and a set of auxiliary data, coming from a plugin-specific input GUI (if the plugin provides it). The plugin **outputs** may be saved into one or more MP store objects or they can be displayed by plugin-specific output GUIs (the *MP vistas* described below).

A plugin can be implemented by one or more Java classes, having methods for accomplishing some basic functions, such as, to return the plugin name and its description, to acquire the input, to perform the data processing, and to return the output. Further Java methods manage the plugin synchronization with the rest of the application. Once all the required methods have been implemented, the plugin is ready to be launched by means of MetaPlab. In this way, the compiled (.jar) file of the plugin should be placed into a specific folder, called *plugin directory*, in order to be automatically recognized and loaded by the Plugin Manager.

Figure 3 depicts the *Plugin Manager* GUI which enables the user to choose plugins from a list and to run them. The *upper side* of this window displays the available plugins. Each of them can be launched by selecting the related entry in the list and by clicking the underlying *Run* button. If a plugin saves its output as an MP store data structure, then further plugins can work on this output, getting it as an input. Whenever a plugin computation stops, the Plugin Manager is displayed, in order to give the user the chance to run another plugin. The *lower side* of the Plugin Manager is instead dedicated to deliver new plugins. In fact, due to the intrinsic open and easy structure, MP plugins can be implemented, following a few simple rules, by whoever wants to attack a specific modeling problem by MP systems. From this perspective, the forthcoming on-line repositories will enable the exchange of these computational tools among the MetaPlab users, thus encouraging their reuse. When an on-line repository is selected by the first text field, the subsequent text box automatically shows the list of plugins which can be downloaded from the repository. Soon, a web site dedicated to MetaPlab [28] will support the download of new plugins and it will provide a complete documentation of the software.

MP vistas. The fourth level concerns with other ways of representing MP structures and MP dynamics, which can support the analysis of specific aspects


MPlab Plugin Manager
Run a local MP plugins
Run
Download new MP plugins from the net
Connect to repository
Available MP plugins in the selected repository
Download

Dynamics computation tool
Log-gain tool

Repositories: ository (http://www.cbmc.it/MPSrepository)

Simulation tool
Log-gain tool
Regression analysis tool (Neural Networks)
Topology discovery tool (Genetic Programming)

Fig. 3. The MetaPlab Plugin Manager. In the upper side the user can run plugins by choosing them from a list. The lower side allows the user to download new plugins from forthcoming on-line repositories.

of the modeling process. Some examples of these vistas are the jointly tracing of substances and parameters curves, the plotting of phase diagrams, and the visualization of statistical indexes.

Auxiliary modules. Two further modules are displayed in the bottom of Figure 2: the *MP store validator* and the *repository manager*. The first is a Java library which assists plugin designers to check the MP store consistency. The second manages the systematic storage and retrieval of the experiments related to a specific MP system.

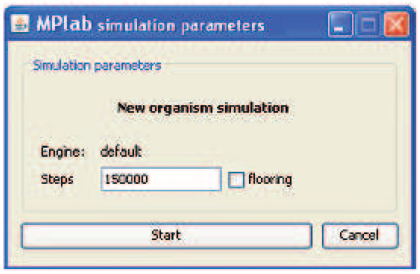
4 A plugin for computing MP system dynamics

In this section we propose a plugin example to highlight the mechanisms underlying the plugin-based framework described above. The plugin we propose is a *simulator* which computes the dynamics of an MP system by applying the recurrent equations (1) and (2) of Definition 1. We remark that the plugin is simply a rearrangement of the stand-alone software MPsim 3. The main difference between the stand-alone simulator and the relative plugin version is that, the latter satisfies some structural requirements which allow it to be automatically loaded by the Plugin Manager, to communicate with the MetaPlab input GUI and to exchange data with other plugins. All the details about these simple requirements will be published in the forthcoming MetaPlab Developer Guide [28].

Plugin functioning. At the beginning of the modeling process, we define an MP system by dragging substance, parameter and reaction nodes from the right toolbar of the input GUI (Figure 1). Then, we draw stoichiometric arcs, and we state both regulation functions and initial conditions. For example, let us imagine to define an MP graph for a typical metabolic process, as the mitotic oscillator already simulated by the stand-alone tool in [3].

After this input stage we open the Plugin Manager (Figure 3) which lists all the available plugins. We select the *dynamics computation tool* by choosing the related entry from the upper list, and we click the *Run* button, in order to start the plugin. The graphical user interface depicted on the left side of Figure 4 appears on the screen. By this window we state the number of steps to perform and then, we launch the dynamics computation process by the start button. When the process finishes, the substance, parameter and flux time-series, computed by the plugin, are automatically saved by an MP store which updates the MP graph displayed by the input GUI. Furthermore, the dynamics is plotted by the plugin output interface, depicted on the right of Figure 4, which shows the typical mitotic oscillations.

We finally remark that the just computed dynamics can be processed again by further plugins, as in a pipeline, because MP store objects preserve the format compatibility among all these tools. From this perspective MetaPlab widely increases the computational power of MPsim.



MPlab simulation parameters

Simulation parameters

New organism simulation

Engine: default

Steps 150000 ☐ flooring

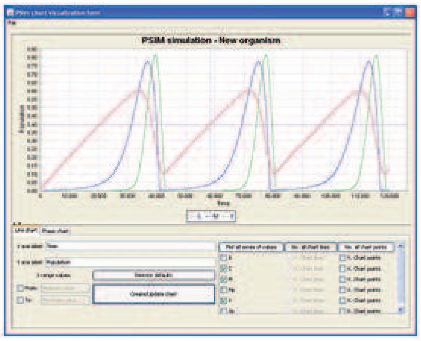


Fig. 4. On the left: The input graphical user interface of the dynamics computation plugin. **On the right:** the output graphical user interface of the dynamics computation plugin.

5 Conclusions and future works

This work has shown that several problems related to the modeling of biological networks can be systematically tackled by means of a set of computational tools integrated in a virtual laboratory, based on the MP systems theory.

The power of this laboratory tightly depends on the flexibility of the plugins architecture and will increase as much as we collect new plugins enriching the basic functionalities of our system. For example, at present we are almost ready to add a new plugin, based on the Log-gain theory [14], which compute the fluxes of a given MP system, deduced by a temporal series of observed states. We plan also to develop other plugins based on traditional regression techniques, neural networks and genetic programming, for obtaining flux maps from fluxes.

Other important functionalities of our virtual laboratory will be topics of further extensions. In particular we want to mention: *i)* plugins which compute suitable statistic coefficients and analyze the system stability when biological parameters change, *ii)* plugins dedicated to the network importation from the main on-line databases (by the SBML standard), *iii)* plugins able to map MP systems to other formalisms, such as ODE or Petri Nets, and to visualize MP models by means of alternative vistas.

References

1. F. Bernardini, M. Gheorghe, and N. Krasnogor. Quorum sensing P systems. *Theoretical Computer Science*, 371(1-2):20–33, 2007.
2. F. Bernardini and V. Manca. Dynamical aspects of P systems. *BioSystems*, 70:85–93, 2003.
3. L. Bianco and A. Castellini. Psim: a computational platform for Metabolic P systems. In *LNCS 4860*, pages 1–20. Springer, 2007.
4. L. Bianco, F. Fontana, G. Franco, and V. Manca. P systems for biological dynamics. In G. Ciobanu, G. Păun, and M. J. Pérez-Jiménez, editors, *Applications of Membrane Computing*, Natural Computing Series, pages 81–126. Springer, Berlin, 2006.
5. L. Bianco, F. Fontana, and V. Manca. P systems with reaction maps. *International Journal of Foundations of Computer Science*, 17(1):27–48, 2006.
6. L. Bianco, D. Pescini, P. Siepmann, N. Krasnogor, F. J. Romero-Campero, and M. Gheorghe. Towards a P systems pseudomonas quorum sensing model. In *LNCS 4361*, pages 197–214. Springer, 2006.
7. A. Castellini, G. Franco, and V. Manca. Toward a representation of Hybrid Functional Petri Nets by MP systems. In *Proceedings of the 2nd International Workshop on Natural Computing, IWNC 2007, Nagoya University, Japan.* Springer Verlag Tokyo. To appear.
8. A. Castellini, G. Franco, and V. Manca. Hybrid functional petri nets as MP systems. 2008. Submitted.
9. F. Fontana and V. Manca. Discrete solutions to differential equations by metabolic P systems. *Theoretical Computer Science*, 372(2-3):165–182, 2007.
10. Center for BioMedical Computing web site. Url: http://www.cbmc.it.
11. S. Hoops, S. Sahle, R. Gauges, C. Lee, J. Pahle, N. Simus, M. Singhal, L. Xu, P. Mendes, and U. Kummer. COPASI a COmplex PAthway SImulator. *Bioinformatics*, 22(24):3067–3074, 2006.
12. H. Kitano. Computational systems biology. *Nature*, 420, 2002.
13. J.R. Koza, M.A. Keane, M.J. Streeter, W. Mydlowec, J. Yu, and G. Lanza. *Genetic Programming IV : Routine Human-Competitive Machine Intelligence (Genetic Programming).* Springer, 2003.

14. V. Manca. Log-gain principles for Metabolic P systems. In *G. Rozenberg Festschrift*. To appear.
15. V. Manca. The Metabolic Algorithm: Principles and applications. *Theoretical Computer Science*. http://dx.doi.org/10.1016/j.tcs.2008.04.015. In print.
16. V. Manca. Metabolic P systems for biochemical dynamics. *Progress in Natural Sciences*, 17(4):384–391, 2007.
17. V. Manca. Discrete simulations of biochemical dynamics. In *LNCS 4848*, pages 231–235. Springer, 2008.
18. V. Manca and L. Bianco. Biological networks in metabolic P systems. *BioSystems*, 91(3):489–498, 2008.
19. V. Manca, L. Bianco, and F. Fontana. Evolutions and oscillations of P systems: Applications to biochemical phenomena. In *LNCS 3365*, pages 63–84. Springer, 2005.
20. V. Manca, R. Pagliarini, and S. Zorzan. Toward an MP model of Non Photochemical Quenching. In *Pre-Proceeding of the 9-th Workshop on Membrane Computing*, 2008.
21. D. Michie, D. J. Spiegelhalter, and C. C. Taylor, editors. *Machine learning, neural and statistical classification*. Ellis Horwood, Upper Saddle River, NJ, USA, 1994.
22. M. Nagasaki, A. Doi, H. Matsuno, and S. Miyano. Genomic object net: I. A platform for modelling and simulating biopathways. *Applied Bioinformatics*, 2(3):181–184, 2004.
23. G. Păun. Computing with membranes. *Journal of Computer and System Sciences*, 61(1):108–143, 2000.
24. G. Păun. *Membrane Computing. An Introduction*. Springer, Berlin, 2002.
25. F. J. Romero-Campero, H. Cao, M. Camera, and N. Krasnogor. Structure and parameter estimation for cell systems biology models. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2008*. ACM Publisher, 2008. To appear.
26. E. Voit, A. R. Neves, and H. Santos. The intricate side of systems biology. *PNAS*, 103(25):9452–9457, June 20 2006.
27. L. von Bertalanffy. *General systems theory: foundations, developments, applications*. George Braziller Inc., New York, NY, 1967.
28. MetaPlab website. Url: http://mplab.sci.univr.it.