A new simulation algorithm for multienvironment probabilistic P systems

M.A. Martínez-del-Amor [#1], I. Pérez-Hurtado [#2], M.J. Pérez-Jiménez [#3], A. Riscos-Núñez [#4], M. Angels Colomer [*5]

[#] *Dpt. of Computer Science and Artificial Intelligence, University of Seville*
Avda. Reina Mercedes s/n, 41012 Sevilla, Spain
[1] mdelamor@us.es
[2] perezh@us.es
[3] marper@us.es
[4] ariscosn@us.es

[*] *Department of Mathematics, University of Lleida*
Avda. Alcalde Rovira Roure, 191, 25198 Lleida, Spain
[5] colomer@matematica.udl.es

***Abstract*—Multienvironment P systems are the base of a general framework for modeling ecosystems dynamics. On one hand, this modeling framework represents the structural and dynamical aspects of real ecosystems in a discrete, modular and compressive way. On the other hand, the inherent randomness and uncertainty of biological systems are captured by using probabilistic strategies. Nowadays, the simulation of these P systems based models is fundamental for experimentation and validation. In this paper, we introduce a new simulation algorithm, called DNDP, which performs object distribution and maximal consistency in the application of rules, that are crucial aspects of these systems.**

The paper also depicts a parallel implementation of the algorithm, and a comparison with the existing algorithm in *PLinguaCore* is provided. In order to test the performance of the presented algorithm, several experiments (simulations) have been carried out over four simple P systems with the same skeleton and different number of environments.

I. Introduction

Since Gh. Păun introduced Membrane Computing in 1998 [9], this bio-inspired computing paradigm has proved to be an active research area within Natural Computing. The aim of this area is to define computational models which abstract from the functioning (chemical reactions) and structure (chemical substances, membranes and compartments) of living cells. Devices of this model are called P systems, and they consist of a cell-like membrane structure, in the compartments of which are placed multisets of objects that evolve according to given rules in a synchronous (assuming a global clock) non-deterministic maximally parallel way.

P systems have several syntactic ingredients: a membrane structure consisting of a hierarchical arrangement of membranes embedded in a skin membrane, and delimiting regions or compartments where multisets of objects (corresponding to chemical substances) and sets of evolution rules (corresponding to reaction rules) are placed. The skin membrane delimits the internal region of the P system with the environment. Every membrane has an associated label, and depending on the P system model, also a polarization that can be modified in the computation. P systems also have two main semantic ingredients: their inherent parallelism and non-determinism.

Recent research works have been focused on using P systems as a modeling tool for biological phenomena, within the framework of computational Systems Biology and Population Dynamics [10], [1], [2], [6], [11], [12], being complementary and an alternative to more classical approaches (i.e. ODEs, Petri Nets, etc). They are used as a formalism for describing, and simulating, the behavior of biological systems, with the advantage of providing a discrete and modular formal model.

In [3], a P systems based general framework for modeling ecosystems dynamics is presented. This computational modeling has been used for real ecosystems, such as the scavenger bird in the Catalan Pyrenees [4] and the zebra mussel in Ribarroja reservoir [3]. These P system based models are used to study the ecosystem dynamics, analyzing the simultaneous evolution of a high number of species. The modularity of these models also enables to easily adding or removing new ingredients and characteristics.

The aim of this P system based modeling tool is to help the ecologists to adopt *a priori* management strategies for the real system by executing virtual experiments. However, since no *in vivo* nor *in vitro* implementations of P systems are yet available, the computation and analysis of these models is currently performed by simulators. Thus, the design of simulators and other related software tools becomes a critical point in the process of model validation, as well as for virtual experimentation.

In this sense, a software tool was developed and presented in [3]. It provides, among others, the following features: a graphical user interface (users be ecologists or model designers), definition of model and ecosystem's parameters, execution of simulations, creation of statistical data in form of tables, graphs, etc. The core of this application is *PLinguaCore* [7], a software library for Membrane Computing. The models are defined by plain-text files using P-Lingua specification language. The application loads that file, configures the corresponding

parameters, calls *PLinguaCore* to execute, and collects the results of the simulation.

Inside the *PLinguaCore* library, many simulation algorithms are defined for the different P system models. Specifically, for probabilistic P systems, the implemented simulation algorithm is based on binomial distribution and blocks of rules. This simulation algorithm is efficient enough for small and medium instance sizes, but it lacks maximal consistency application of rules and calculation of probability functions. In this paper, we propose a new simulation algorithm, inspired on the algorithm presented by V. Nguyen et al. in [8], that removes the concept of block of rules, and solves the above mentioned restrictions. We also analyze and validate the algorithm and study the new features and constraints.

The rest of the paper is structured as follows. Section II describes the modeling framework based on probabilistic P systems. Section III depicts the details of the proposed simulation algorithms. In section IV we show some results by using a toy P system, validating the algorithm. The paper ends with some conclusions and ideas for future work.

II. The P system based framework

Definition 1: A multienvironment functional probabilistic P system with active membranes of degree (q, m) with $q \geq 1,\ m \geq 1$, taking T time units, $T \geq 1$, is a tuple

$$(G, \Gamma, \Sigma, R_E, \Pi, \{f_{r,j} :\ r \in R_\Pi, 1 \leq j \leq m\}, \{\mathcal{M}_{ij} :\ 0 \leq i \leq q-1,\ 1 \leq j \leq m\})$$

where:

- $G = (V, S)$ is a directed graph such that $(x, x) \in S$, for each $x \in V$. Let $V = \{e_1, \ldots, e_m\}$ whose elements are called environments;
- Γ is the working alphabet and $\Sigma \subsetneqq \Gamma$ is an alphabet representing the objects that can be present in the environments;
- R_E is a finite set of communication rules between environments of the form

  $$(x)_{e_j} \xrightarrow{p_{(x,j,j_1,\ldots,j_h)}} (y_1)_{e_{j_1}} \ldots (y_h)_{e_{j_h}}$$

  where $x, y_1, \ldots, y_h \in \Sigma$, $(e_j, e_{j_l}) \in S$ $(l = 1, \ldots, h)$ and $p_{(x,j,j_1,\ldots,j_h)}(t) \in [0, 1]$, for each $t = 1, \ldots, T$. If $p_{(x,j,j_1,\ldots,j_h)}(t) = 1$, for each t, then we omit the probabilistic function. These rules verify the following:
  - ⋆ for each e_j and for each x, the sum of functions associated with the rules from R_E whose left–hand side is $(x)_{e_j}$ coincide with the constant function equal to 1.
- $\Pi = (\Gamma, \mu, R_\Pi)$ where
  - μ is a membrane structure consisting of q membranes, with the membranes injectively labeled with $0, \ldots, q-1$. The skin membrane is labeled with 0. We also associate electrical charges from the set $\{0, +, -\}$ with membranes; and
  - R_Π is a finite set of evolution rules of the form $r : u[\ v\ ]_i^\alpha \to u'[\ v'\ ]_i^{\alpha'}$ where $u, v, u', v' \in M(\Gamma),\ i \in \{0, 1, \ldots, q-1\}$, and $\alpha, \alpha' \in \{0, +, -\}$;
- For each $r \in R_\Pi$ and for each j, $1 \leq j \leq m$, $f_{r,j}$ is a computable function whose domain is $\{1, 2, \ldots, T\}$ and its range is contained in $[0, 1]$, verifying the following:
  - ⋆ For each $u, v \in M(\Gamma)$, $i \in \{0, \ldots, q-1\}$ and $\alpha \in \{0, +, -\}$, if $r_1, \ldots, r_z$ are the rules from R_Π whose left–hand side is $u[v]_i^\alpha$, then $\sum_{j=1}^{z} f_{r_j}(t) = 1$, for each $t,\ 1 \leq t \leq T$.
- For each j $(1 \leq j \leq m)$, $\mathcal{M}_{0j}, \ldots, \mathcal{M}_{q-1,j}$are strings over Γ, describing the multisets of objects initially placed in the q regions of μ.

A multienvironment probabilistic functional extended P system with active membranes of degree (q, m) taking T time units

$$(G, \Gamma, \Sigma, R_E, \Pi, \{f_{r,j} :\ r \in R_\Pi, 1 \leq j \leq m\}, \{\mathcal{M}_{ij} :\ 0 \leq i \leq q-1,\ 1 \leq j \leq m\})$$

can be viewed as a set of m environments $e_1, \ldots, e_m$ linked between them by the arcs from the directed graph G. Each environment e_j contains a functional probabilistic P system with active membranes of degree q, each of them with the *same* skeleton, Π, and such that $\mathcal{M}_{0j}, \ldots \mathcal{M}_{q-1,j}$ describe their initial multisets.

When a communication rule between environments

$$(x)_{e_j} \xrightarrow{p_{(x,j,j_1,\ldots,j_h)}} (y_1)_{e_{j_1}} \ldots (y_h)_{e_{j_h}}$$

is applied, object x pass from e_j to $e_{j_1}, \ldots, e_{j_h}$ possibly modified into objects $y_1, \ldots, y_h$, respectively. In any moment t, $1 \leq t \leq T$, at which an object x is in environment e_j, one and only one rule will be applied according to its probability which is given by $p_{(x,j,j_1,\ldots,j_h)}(t)$.

We assume that a global clock exists, marking the time for the whole system (for its compartments), that is, all membranes and the application of all rules are synchronized.

The tuple of multisets of objects present at any moment in the m environments and at each of the regions of the P systems located within them, and the polarizations of the membranes in each P system, constitutes a *configuration* of the system at that moment. At the initial configuration of the system we assume that all environments are empty and all membranes have a neutral polarization.

The P system can pass from one configuration to another by using the rules from $R = R_E \cup \bigcup_{j=1}^{m} R_{\Pi_j}$ as follows: at each transition step, the rules to be applied are selected according to the probabilities assigned to them, and all applicable rules are simultaneously applied and all occurrences of the left–hand side of the rules are consumed, as usual.

III. Binomial block based simulation Algorithm

In this section we describe the first simulation algorithm developed for the model presented in [3], multienvironment

probabilistic P systems. It is available in the current release of *PLinguaCore* library [7], and it is implemented following a strategy based on the binomial distribution and blocks of rules.

Let us consider a multienvironment probabilistic functional extended P system with active membranes of degree (q, m) with $q \geq 1,\ m \geq 1$, taking T time units, $T \geq 1$, $\Pi = (\Gamma, \Sigma, G, R_E, \Pi, \{f_{r,j} :\ r \in R_\Pi, 1 \leq j \leq m\}, \{M_{ij} :\ 0 \leq i \leq q-1,\ 1 \leq j \leq m\})$, as defined in section II, where $\{f_{r,j} :\ r \in R_\Pi, 1 \leq j \leq m\}$ are constant functions with range in $[0, 1]$. The computation of the system is a sequence of configurations C_t, $0 \leq t \leq T$ constructed by the application of rules from $R = R_E \cup \bigcup_{j=1}^{m} R_{\Pi_j}$.

The algorithm presented below shows the pseudocode of the binomial block based simulation algorithm. Roughly speaking, it is divided into two main phases: *selection* and *execution*. In the first one, the rules are selected, determining the number of times that they will be applied in the simulated step, according to their left-hand sides and the available objects in the configuration. In the second phase, the chosen rules are applied the elected number of times by adding the multisets of the right-hand side to the configuration, and possibly changing the polarization of membranes.

Let us now describe the algorithm in detail.

The input data for selection phase consists of the configuration in time t, C_t, and the set of defined rules R. The output data of this stage is a multiset of the form $R_{sel} = \{\langle r, n_r \rangle\}$, where $r \in R$ and $n_r \in \mathbb{N}$ is the number of times to be executed. Note that only in the execution phase, the right-hand side of rules are added to the configuration. Adding new objects before finishing selection phase leads to inconsistent states, since the simulation would select and execute rules from different transitions of the system.

The selection mechanism starts from the assumption that rules in R can be classified into blocks of rules having the same left-hand side, following the definitions 2 and 3 below. Recall that, according to the semantics of the model, the sum of probabilities of all the rules from a block is always equal to 1 – in particular, rules with probability equal to 1 form individual blocks. Note that rules with overlapping (but different) left-hand sides are classified in different blocks.

Definition 2: Given a rule $r \in R_\Pi$ of the form $r : u[v]_h^\alpha \rightarrow u'[v']_h^{\alpha'}$:

- The left-hand side of the rule r is defined as $LHS(r) = u[v]_h^\alpha$, corresponding to the multiset u in the parent membrane of h, the multiset v in the membrane h, and the membrane h with charge α.
- The right-hand side of the rule r is defined as $RHS(r) = u'[v']_h^{\alpha'}$, corresponding to the multiset u' in the parent membrane of h, the multiset v' in the membrane h, and the membrane h with charge α'.

Definition 3: Two rules r_1 and r_2 have the same left-hand side if $LHS(r_1) = u_1[v_1]_{h_1}^{\alpha_1}$, $LHS(r_2) = u_2[v_2]_{h_2}^{\alpha_2}$ and the following holds:

- $h_1 = h_2$ and $\alpha_1 = \alpha_2$
- the strings v_1 and v_2 (resp. u_1 and u_2) represent the same multiset of objects placed in the region h_1 (resp. in the parent region of h_1).

The rules selection mechanism iterates through the randomly ordered set of blocks of rules, and within each block, rules are selected in a maximal way (i.e. they will consume as many objects from the configuration as possible). More precisely, given a block, the number of times that a rule r is applied is determined according to a binomial distribution $B(N, p_r)$ (see definition 4 below) where N is the number of copies of the multisets in $LHS(r)$ contained in the configuration, and p_r is the probability associated with r.

Definition 4: In probability theory, the binomial distribution $B(n, p)$ is the discrete probability distribution of the number of successes in a sequence of n independent Bernoulli experiments (success/failure experiment), each of which yields success with probability p.

In order to guarantee that we do not consume more resources than what it is available, after determining the number of applications for the first rule of the block ($N_r = B(N, p_r)$), the first parameter of the binomial distribution is reduced so that for the next rule the maximum number of applications matches the available objects ($N \leftarrow N - N_r$). This is done for all rules in the block (randomly ordered) except for the last one, which skips the binomial distribution and takes directly all the remaining applications. It is worth noting that this process is equivalent to calculating a multinomial distribution (definition 5).

Definition 5: In probability theory, the multinomial distribution $M(n, p_1, p_2, \ldots, p_k), \sum_{i=1}^{k} p_i = 1$ is a generalization of the binomial distribution, where each independent experiment can have $k > 0$ different results. The output of this function is composed by n_i variables, with $1 \leq i \leq k$, and $\sum_{i=1}^{k} n_i = n$.

When the selection phase finishes, it returns the multiset R_{sel} containing the applicable rules and the number of times they will be executed in the simulated step. Execution phase iterates through the selected rules, and adds the corresponding right-hand sides to the configuration (taking into account the number of applications).

Next we show the pseudocode for the binomial block based algorithm.

Input: A multienvironment functional P system with active membranes of degree (q, m) with $q \geq 1,\ m \geq 1$, taking T time units, $T \geq 1$.

1: **for** $t \leftarrow 0$ to $T - 1$ **do**
2: $\quad R_{sel} \leftarrow$ Selection-phase (C_t, R)
3: $\quad C_{t+1} \leftarrow$ Execution-phase (C_t, R_{sel})
4: $\quad$ **print** C_{t+1}
5: **end for**

Selection phase

1: Rules from $R = R_E \cup R_{\Pi_j}, 1 \leq j \leq m$ are classified into sets (blocks) B_l so that all the rules belonging to a block have the same left-hand side.

Let $F_b(N,p)$ be a function that returns a random natural number using the binomial distribution $B(N,p)$.
A random order on the family of all blocks of rules is considered.
for all blocks of rules $B_l = \{r_1, \ldots, r_s\}$, according to the selected random order **do**
  A random order to the rules $\{r_1, \ldots, r_s\}$ is selected
  Let us suppose that the common left-hand side is $u\ [v]_h^\alpha$ and their respective probabilistic constants are $c_{r_1}, \ldots, c_{r_s}$
  The highest number N is computed so that u^N appears in the parent membrane of h and v^N appears in membrane h (only if the charge is α) in the configuration C_t.
  if $N > 0$ **then**
    $d \leftarrow 1$
    for $k \leftarrow 1$ to $s-1$, according to the selected order **do**
      $c_{r_k} \leftarrow \frac{c_{r_k}}{d}$
      $n_{r_k} \leftarrow F(N, c_{r_k})$
      $N \leftarrow N - n_{r_k}$
      $q \leftarrow 1 - c_{r_k}$
      $d \leftarrow d * q$
    end for
    $n_{r_s} \leftarrow N$
    for $k \leftarrow 1$ to s **do**
      $C_t \leftarrow C_t - n_{r_k} * LHS(r_k)$
      $R_{sel} \leftarrow R_{sel} \cup \langle r_k, n_{r_k} \rangle$
    end for
  end if
end for
return R_{sel}

Execution phase

for all $\langle r, n \rangle \in R_{sel}$ **do**
  if $n > 0$ **then**
    $C_t \leftarrow C_t + n * RHS(r)$
    Update charges of membranes from C_t using $RHS(r)$
  end if
end for
$C_{t+1} \leftarrow C_t$
return C_{t+1}

This simulation algorithm is useful for the majority of models, such as [4], [3]. However, it has several disadvantages that restricts the P systems models to be correctly simulated:

- It needs to classify the rules by their left-hand side. The creation of blocks of rules, with the restriction that probabilities have to sum 1, leads to simulate only models with rules distributed in this special way.
- It does not handle rules with intersections on their left-hand side (object competition). Rules with common objects but not the same left-hand side are classified in different blocks, so the common objects will be not distributed since these blocks are maximally executed.
- It does not check the consistency of charges in the selection of rules. As seen in section II, rules are executed in a maximally consistent way. If there are rules changing the charge of a membrane, and others changing to a different one, they cannot be executed in the same step. Fixing one value to the charge, all the rules consistent to it must, if possible, be executed.
- It does not evaluate probabilistic functions related to rules. Only constant probabilities are considered, what is not the case of the new P systems based models

These constraints lead to develop new simulation algorithms, looking for flexibility on the semantics to simulate (i.e. do not restrict blocks of rules) and providing supplementary features for the new requirements of models (maximal consistency and probabilistic functions).

IV. DND-P SIMULATION ALGORITHM

A. Inspiration: Direct Non-deterministic Distribution algorithm (DND)

The algorithm to develop has to solve the restrictions mentioned above: classification of rules, object competition, maximal consistency and calculation of probability functions, and possibly improve the performance. In order to solve the first two points, the algorithm should work with the rules individually without classifications and blocks, and also to assure the object distribution to the rules (according to the probabilities) and maximal execution.

In [8], V. Nguyen et al. introduce an algorithm performing non-deterministic, maximally parallel object distribution for transition P systems and its hardware implementation. Assuming that it is possible to have more than one solution to the object distribution problem, several approaches are also analyzed:

- Indirect approaches: These approaches consider both solutions and non-solutions to the problem in the searching process. For example, the *Incremental Approach* constructs a solution starting from a non-solution.
- Direct approaches: These approaches consider only solutions in the exploration process. For example, the *Direct Non-deterministic Distribution algorithm (DND)* constructs an object distribution in one step, with a possible second step to fix the maximal parallelism. In order to do that, the algorithm executes two phases:
  - Forward phase: it is a loop iterating the rules of a region in a random order, choosing a random value for each and storing it for the next rules that can have intersections.
  - Backward phase: it is a loop iterating the rules of such region in the previous random order, checking if there exists more applications to the rules, and assigning that number to each.

It can be considered that the forward phase implements a non-deterministic object distribution, and the backward phase makes the maximal parallelism, both selecting the rules individually (without using blocks). Therefore, the idea of the

DND algorithm is suitable for solving the restriction of blocks of rules and object competition in probabilistic P systems.

B. Direct Non-deterministic Distribution with probabilities algorithm (DND-P)

In this section we describe the pseudocode of our proposed simulation algorithm for multienvironment functional P systems with active membranes. It is called DNDP, which comes from the inspiration on the algorithm DND [8], and the extension for probabilities.

Similarly to the previous algorithm (section III), the transitions of the P system are simulated in two phases, selection and execution, in order to synchronize the consumption and creation of objects. However, selection is divided in two micro-phases. The first one calculates a multiset of consistent applicable rules. The second changes the multiplicity of the rules in the previous multiset to assure maximal application, obtaining a multiset of maximally consistent applicable rules.

Before selecting and executing rules, some previous calculations are required. For each environment j, a set of rules (D_j), one per each environment, is constructed by using a random order. It is of the form: rules from R_E applied to the environment, and rules from R_Π whose probabilities are non zero, and that are applied according to the charges of membranes in the corresponding configuration and environment. Finally, a new feature is provided in this initialization procedure, which is that the probabilities of rules are recalculated for each moment t.

In the first selection phase, a multiset of consistent applicable rules, R^1_{sel}, is calculated. For each environment, rules from D_j are processed in a iterative process where the rules are selected a number of times, updating the configuration with the left-hand side. In this loop, two copies of the configuration are needed: one to be updated each time that a rule is selected (removing the left-hand side) and the original one before starting the process. They are called C'_t and C_t respectively. Suppose that the rules in $R^1_{sel} = \{< r_1, n_1 >, < r_2, n_2 > , \ldots, < r_x, n_x >\}$ have been selected before processing the rule r. Then, r is selected if:

- The application of the rule r is consistent with the application of the previously selected rules. Assuming that rule r is of the form $u_r[v_r]^{\alpha_r}_{h_r} \rightarrow u'_r[v'_r]^{\alpha'_r}_{h_r}$, if there exists a selected rule $< r_i, n_i > \in R^1_{sel}, r_i = u_i[v_i]^{\alpha_i}_{h_i} \rightarrow u'_i[v'_i]^{\alpha'_i}_{h_i}$ such that $h_r = h_i$ and $\alpha'_r \neq \alpha'_i$, then the rule is considered non consistent, and so, discarded to be applied.
- The number of applications of r in the auxiliary configuration C'_t is greater than 0. If one rule $< r_i, n_i > \in R^1_{sel}$ has common objects with r in the left-hand side, the total amount of applications in C'_t has been (possibly) decreased n_i times, so the rule r will use the rest of objects in the configuration. In this way, the object competition is solved by serializing the selection of rules. Note that the rules are randomly chosen, so that for each simulation, the first rule trying to get objects will be different.

Once a rule is selected (because it is consistent with the rest of selected rules and it is applicable), a random number of applications is calculated. In this version, this number is generated by using a binomial distribution. However, the function can be changed in order to use other probability distributions without any repercussion to the code. It is considered that the number of times to apply the rule depends on the associated probability and on the initial number of applications in C_t, according to the multinomial distribution. For example, consider a rule r_x with probability p_x, and rule r_y with p_y, both with the same left-hand side, and $p_x + p_y = 1$. Also consider that the total amount of instances of the left-hand side is N. For the rule r_x, the number of applications will be calculated by $B(N, p_x) = N_x$, so that there are $N' = N - N_x$ instances left. If we calculate the number for r_y as $B(N', p_y)$, then it is not a real multinomial distribution. The correct calculation would be $B(N, p_y) = N_y$. When N_y is greater than the amount of objects left ($N_y > N'$), the maximum amount will be directly assigned ($N_y = N'$).

When the binomial random number generator returns 0, the corresponding rule is added to a multiset called R^2_{sel}. This multiset is used to give a new chance to the rules to be selected when applying maximality in the second phase. However, it has to be handled in a special way at this phase. The rule cannot reserve the change of a membrane charge since it has not be actually selected. Hence, R^1_{sel} is the multiset of consistent rules selected a number of times greater than 0, and R^2_{sel} is the multiset of rules selected 0 times.

In the second selection phase, the consistent applicable rules are checked again in order to achieve maximality. Only consistent rules are considered, and they are taken from $R_{sel} = R^1_{sel} \cup R^2_{sel}$. If one rule $r \in R_{sel}$ has a number of applications (N) different to 0 in C'_t, this number will be assigned directly to the rule r. In order to fairly distribute the objects among the rules, they are iterated in order with respect to the probabilities. Moreover, one rule from the multiset R^2_{sel} can be checked, so it is possible that other rule from R^1_{sel}, inconsistent to this one, has been previously selected. In this case, the consistent condition has to be tested again.

Finally, execution phase is similar to the binomial algorithm. It will iterate all the rules in R^1_{sel} (maximal applicable consistent rules), and it will add the right-hand side of them to the configuration C'_t. At the end of the process, C'_t is actually the next configuration: the left-hand side of rules has been removed in the first and second selection phases, and the right-hand side of rules is added in the execution stage.

Next, the pseudocode for the DNDP algorithm is depicted.

Input: A multienvironment functional P system with active membranes of degree (q, m, n) with $q \geq 1,\ m \geq 1,\ n \geq 1$, taking T time units, $T \geq 1$, and a natural number $K \geq 1$.

1: **for** $t \leftarrow 0$ to $T - 1$ **do**
2: $C_t \leftarrow$ configuration of the system at the moment t
3: $C'_t \leftarrow C_t$
4: Initialization
5: First selection phase. It generates a multiset of *consistent* applicable rules.

Second selection phase. It generates a multiset of *maximally consistent* applicable rules.
Execution of selected rules.
$C_{t+1} \leftarrow C'_t$
end for

Initialization

$R_\Pi \leftarrow$ ordered set of rules of Π
for $j \leftarrow 1$ to m **do**
  $R_{E,j} \leftarrow$ ordered set of rules from R_E related to the environment j
  $A_j \leftarrow$ ordered set of rules from $R_{E,j}$ whose probability at the moment t is > 0
  $M_j \leftarrow$ ordered set of pairs $\langle label, charge \rangle$ for all the membranes from C_t contained in the environment j
  $B_j \leftarrow \emptyset$
  for each $\langle h, \alpha \rangle \in M_j$ (following the considered order) **do**
    $B_j \leftarrow B_j \cup$ ordered set of rules $u[v]_h^\alpha \leftarrow u'[v']_h^\beta$ from R_Π whose probability at the moment t is greater than 0 for the environment j
  end for
end for

First selection phase (*consistency*)

for $j \leftarrow 1$ to m **do**
  $R^1_{sel,j} \leftarrow$ *the empty multiset*
  $R^2_{sel,j} \leftarrow$ *the empty multiset*
  $D_j \leftarrow A_j \cup B_j$ with a *random order*
  for each $r \in D_j$ (following the considered order) **do**
    if r is *consistent* with the rules in $R^1_{sel,j}$ **then**
      $N' \leftarrow \max\{$number of times that r is applicable to $C'_t\}$
      if $N' > 0$ **then**
        if $p_{r,j}(t) = 1$ **then**
          $n \leftarrow N'$
        else
          $N \leftarrow \max\{$number of times that r is applicable to $C_t\}$
          $n \leftarrow F_b(N, p_{r,j}(t))$
          if $n > N'$ **then**
            $n \leftarrow N'$
          end if
        end if
        if $n > 0$ **then**
          $C'_t \leftarrow C'_t - n \cdot LHS(r)$
          $R^1_{sel,j} \leftarrow R^1_{sel,j} \cup \{< r, n >\}$
        else
          $R^2_{sel,j} \leftarrow R^2_{sel,j} \cup \{< r, n >\}$
        end if
      end if
    end if
  end for
end for

Second phase of rules selection (*maximality*)

for $j \leftarrow 1$ to m **do**
  $R_{sel,j} \leftarrow R^1_{sel,j} + R^2_{sel,j}$ *with an order by the rule probabilities, from highest to lowest*
  for each $< r, n > \in R_{sel,j}$ (following the selected order) **do**
    if $n > 0 \vee (r$ is *consistent* with the rules in $R^1_{sel,j})$ **then**
      $N' \leftarrow \max\{$number of times that r is applicable to $C'_t\}$
      if $N' > 0$ **then**
        $R^1_{sel,j} \leftarrow R^1_{sel,j} \cup \{< r, N' >\}$
        $C'_t \leftarrow C'_t - N' \cdot LHS(r)$
      end if
    end if
  end for
end for

Execution of selected rules

for each $< r, n > \in R^1_{sel,j}$ **do**
  $C'_t \leftarrow C'_t + n \cdot RHS(r)$
  Update the electrical charges of C'_t according to $RHS(r)$
end for

C. A parallel implementation

In this section we provide an implementation to the pseudocode following the imperative programming paradigm. We have included this implementation in the pLinguaCore library in order to evaluate it.

The key aspect of this implementation is the provided parallel solution. Considering that in the left-hand side of rules, only one environment is affected, the selection of rules can be executed in parallel for each environment, without compromising the concurrency. However, the selection of rules have to be synchronized, so that the execution phase cannot start before finishing the selection in every environment.

First, the set of rules is processed. For each transition and for each environment, a list of applicable rules to the configuration is constructed, according only to the membrane charge in the left-hand side. When these lists are completed, they are used to fill an array (one per environment), called D_j, of triples $< rule, probability, number-applications >$. At this point, the probabilities of rules are calculated using the associated function, and stored in the triple with the corresponding rule. The construction of this array, assigning a random order, corresponds to the set of rules D_j defined in the pseudocode.

Furthermore, a boolean array B, with one entry per membrane, is used to check the consistency of rules in a efficient way. In the pseudocode, the consistency condition is assured by testing each rule with the rest of the previously selected. In this implementation, each time that a rule $r = u[v]_h^\alpha \rightarrow u'[v']_h^{\alpha'}$ is selected a number of times greater than 0, the value in B for the active membrane h is set to true ($B[h] \leftarrow true$). Moreover, the charge is changed in C'_t. Then, if an inconsistent rule is checked, it can be seen by B that a rule has reserved the change of the membrane charge, that is stored in C'_t.

The first selection phase works directly with the array D_j, so it is a loop using a shuffle iterator (an index that takes random

values between 0 and max_k). Like in the pseudocode, after checking the consistency condition, the number of applications is calculated. If it is non-zero, the binomial distribution is applied, getting a number n. When the rule is processed, n is stored in the corresponding triple, which is changed by the last one (max_k) in order to do not select again. In the case that the rule is inconsistent or not applicable, it has to be discarded. That is, the rule is changed by the last one (max_k) as before, but also changed with the last selected rule (max_{D_j}), avoiding to be processed in the second phase.

At the end of the first selection phase, D_j corresponds to the consistent applicable rules, both with zero and non-zero selected numbers (R_{sel}). The second selection phase is similar to the one depicted in the pseudocode. It iterates the rules in D_j, according to the order given by probabilities, to fix the number of applications of each one by checking the consistency and maximality. The algorithm ends with the execution phase, which adds the right-hand side of rules to C'_t.

Next we show the algorithm that simulates a computation of a functional multienvironment probabilistic P system with active membranes of grade (q, m), with $q \geq 1$, $m \geq 1$, using T time units. $(\Gamma, \Sigma, G, R_E, \Pi, \{f_{r,j} : r \in R_\Pi, 1 \leq j \leq m\}, \{\mathcal{M}_{ij} : 0 \leq i \leq q-1,\ 1 \leq j \leq m\})$
The algorithm must receive the following input parameters:

- The initial configuration of the system, C_0.
- The sets of rules R_E y R_Π.
- The values q, m y T.
- The set of functions $\{f_{r,j} : r \in R_\Pi, 1 \leq j \leq m\}$.

The following computable functions are considered for random number generation:

- $F_b(N, p)$ is a function that returns a random natural number $n \in \{0, \ldots, N\}$, according to the binomial distribution $B(N, p)$, where $N \in \{0, \ldots, 2^{64}\}$ and $p \in \{\mathbb{R} \cap [0, 1]\}$. Real numbers are encoded in floating point (float) with a precision of 32 bits.
- $F_u(N)$ is a function that returns a random natural number $n \in \{0, \ldots, N-1\}$, according to the uniform distribution $U(N)$, where $N \in \{1, \ldots, 2^{64}\}$.

Procedure DNDP (C_0, R_E, R_Π, q, m, T, K, $\{f_{r,j} : r \in R_\Pi, 1 \leq j \leq m\}$)

1: $\{L_{R,E,j}, L_{R_\Pi,j,h,\alpha} : 1 \leq j \leq m,\ 0 \leq h \leq q-1,\ \alpha \in \{-, 0, +\}\}$ ← Initialization(R_E, R_Π, q, m)
2: **for** $t \leftarrow 0$ to $T-1$ **do**
3: $C'_t \leftarrow C_t$
4: **for** $j \leftarrow 1$ to m **do**
5: Throw new thread Selection-execution(j, t, C'_t, C_t, $L_{R,E,j}$, $\{L_{R_\Pi,j,h,\alpha}\ 0 \leq h \leq q-1, \alpha \in \{-, 0, +\}\}$, q, m, K, $\{f_{r,j} : r \in R_\Pi\}$)
6: **end for**
7: Barrier-synchronization {All threads wait until everyone reaches this point}
8: $C_{t+1} \leftarrow C'_t$
9: **print** C_{t+1}
10: **end for**

Thread Selection-execution (j, t, C'_t, C_t, $L_{R,E,j}$, $\{L_{R_\Pi,j,h,\alpha}\ 0 \leq h \leq q-1, \alpha \in \{-, 0, +\}\}$, q, m, K, $\{f_{r,j} : r \in R_\Pi\}$)

1: D_j, max_{D_j}, B ← Initialize-selection-phase(j, t, C'_t, C_t, $L_{R,E,j}$, $\{L_{R_\Pi,j,h,\alpha}\ 0 \leq h \leq q-1, \alpha \in \{-, 0, +\}\}$, q, m, K, $\{f_{r,j} : r \in R_\Pi\}$)
2: D_j, max_{D_j}, B ← Selection-first-phase(j, D_j, max_{D_j}, B, K, C'_t, C_t)
3: D_j, max_{D_j}, B ← Selection-second-phase(j, D_j, max_{D_j}, B, C'_t)
4: Barrier-synchronization {
All threads wait until everyone reaches this point}
5: Execution(j,D_j,max_{D_j},C'_t)

Function Selection-first-phase (j, D_j, max_{D_j}, B, K, C'_t, C_t)

1: $max_k \leftarrow maxD_j$
2: **while** $max_k > 0$ **do**
3: $i \leftarrow F_u(max_k)$
4: $< r, p, n > \leftarrow D_j[i]$
5: **if** Is-consistent(r,j,B_j,C'_t) **then**
6: $N' \leftarrow$ Count-applications(r,j,C'_t)
7: **if** $N' > 0$ **then**
8: **if** $p = 1$ **then**
9: $n_i \leftarrow N'$
10: **else**
11: $N \leftarrow$ Count-applications(r,j,C_t)
12: $n_i \leftarrow F_b(N, p)$
13: **if** $n_i > N'$ **then**
14: $n_i \leftarrow N'$
15: **end if**
16: **end if**
17: **if** $n_i > 0$ **then**
18: Remove-left-hand-rule-objects(r,j,n_i,C'_t)
19: Update-charge(r,j,B,C'_t)
20: $n \leftarrow n + n_i$
21: **end if**
22: Swap(D_j,i,$max_k - 1$)
23: $max_k \leftarrow max_k - 1$
24: **else**
25: Swap(D_j,i,$max_k - 1$)
26: Swap(D_j,$max_k - 1$,$max_{d_j} - 1$)
27: $max_k \leftarrow max_k - 1$
28: $max_{D_j} \leftarrow max_{D_j} - 1$
29: **end if**
30: **else**
31: Swap(D_j,i,$max_k - 1$)
32: Swap(D_j,$max_k - 1$,$max_{d_j} - 1$)
33: $max_k \leftarrow max_k - 1$
34: $max_{D_j} \leftarrow max_{D_j} - 1$
35: **end if**
36: **end while**
37: **return** D_j, max_{D_j}, B

Function Selection-second-phase (j, D_j, max_{D_j}, B, C'_t)

Sort triples of the form $< r, p, n >$ from and to D_j, from 0 to max_{D_j}, in decrement order by p.
for $i \leftarrow 0$ to max_{D_j} **do**
  $< r, p, n > \leftarrow D_j[i]$
  if $n > 0 \vee$ Is-consistent(r,j,B_j,C'_t) **then**
    $N' \leftarrow$ Count-applications(r,j,C'_t)
    if $N' > 0$ **then**
      $n \leftarrow n + N'$
      Remove-left-hand-rule-objects(r,j,N',C'_t)
      if $n = 0$ **then**
        Update-charge(r,j,B,C'_t)
      end if
    end if
  end if
end for
return D_j, max_{D_j}, B

Procedure Execution (j, D_j, max_{D_j}, C'_t)

for $i \leftarrow 0$ to max_{D_j} **do**
  $< r, p, n > \leftarrow D_j[i]$
  if $n > 0$ **then**
    Add-left-hand-rule-objects(r,j,n,C'_t)
  end if
end for

Function Initialization (R_E, R_Π, q, m)

for $j \leftarrow 1$ to m **do**
  The set of rules of the form $(x)_{e_j} \xrightarrow{p_{(x,j,j_1,\ldots,j_s)}} (x_1)_{e_{j_1}}, \ldots, (x_s)_{e_{j_s}} \subseteq R_E$ is ordered lexicographically with respect of $x, x_1, \ldots, x_s$.
  Be $L_{R_E,j}$ the list of rules with the previous order.
end for
for $h \leftarrow 0$ to $q - 1$ **do**
  for all $\alpha \in \{-, 0, +\}$ **do**
    The set of rules of the form $u[v]_h^\alpha \rightarrow u'[v']_h^{\alpha'} \subseteq R_{\Pi_j}$ is ordered lexicographically with respect of u, v, u', v'.
    Be $L_{R_\Pi,j,h,\alpha}$ the list of rules with the previous order.
  end for
end for
return $\{L_{R,E,j}, L_{R_\Pi,j,h,\alpha}\ 0 \leq j \leq m-1, 0 \leq h \leq q-1, \alpha \in \{-,0,+\}\}$

Function Initialize-selection-phase (j, t, C'_t, C_t, $L_{R,E,j}$, $\{L_{R_\Pi,j,h,\alpha}\ 0 \leq h \leq q-1, \alpha \in \{-,0,+\}\}$, q, m, K, $\{f_{r,j} : r \in R_\Pi\}$)

$D_j \leftarrow$ New array of triples $< rule - id, float, integer >$, representing $< rule, probability, selection-number >$ respectively, of size: $\sum_{h \leftarrow 0}^{q-1} maximum_{\alpha \in \{-,0,+\}} \{Length(L_{R_\Pi,j,h,\alpha})\} + Length(L_{R_{E,j}})$
$max_{D_j} \leftarrow 0$
for $h \leftarrow 0$ to $q - 1$ **do**
  $\alpha \leftarrow$ charge of membrane h in environment j, from C_t
  for all $r \in L_{R_\Pi,j,h,\alpha}$ **do**
    $p \leftarrow f_{r,j}(t)$
    if $p > 0$ **then**
      $D_j[max_{D_j}] \leftarrow < r, p, 0 >$
      $max_{D_j} \leftarrow max_{D_j} + 1$
    end if
  end for
  $B_j[h] \leftarrow false$
end for
for all $r \in L_{R_{E,j}}$ **do**
  $p \leftarrow p_{(x,j,j_1,\ldots,j_s)}(t)$
  if $p > 0$ **then**
    $D_j[max_{D_j}] \leftarrow < r, p, 0 >$
    $max_{D_j} \leftarrow max_{D_j} + 1$
  end if
end for
return max_{D_j}, D_j, B_j

Function Is-consistent (r, j, B, C)

$check \leftarrow true$
if r is of the form $u[v]_h^\alpha \rightarrow u'[v']_h^{\alpha'}$ **then**
  $\beta \leftarrow$ charge of membrane h in $\Pi_j \in C$
  if $B[h] = true \wedge \alpha' \neq \beta$ **then**
    $check \leftarrow false$
  end if
end if
return $check$

Function Count-applications (r, j, C)

if r is of the form $(x)_{e_j} \xrightarrow{p_{(x,j,j_1,\ldots,j_s)}} (x_1)_{e_{j_1}}, \ldots, (x_s)_{e_{j_s}}$ **then**
  $n \leftarrow$ multiplicity of object x in environment $e_j \in C$
else if r is of the form $u[v]_h^\alpha \rightarrow u'[v']_h^{\alpha'}$ **then**
  $n \leftarrow$ minimum value such that u^n appears in the multiset of the upper compartment of h in $\Pi_j \in C$ and v^n appears in the multiset of h in $\Pi_j \in C$
end if
return n

Procedure Remove-left-hand-rule-objects (r, j, n, C)

if r is of the form $(x)_{e_j} \xrightarrow{p_{(x,j,j_1,\ldots,j_s)}} (x_1)_{e_{j_1}}, \ldots, (x_s)_{e_{j_s}}$ **then**
  Remove the multiset x^n from environment $e_j \in C$
else if r is of the form $u[v]_h^\alpha \rightarrow u'[v']_h^{\alpha'}$ **then**
  Remove the multiset u^n from the multiset of the parent membrane of h in $\Pi_j \in C$
  Remove the multiset v^n from the multiset of membrane h in $\Pi_j \in C$
end if

Procedure Update-charge (r, j, C)

1: **if** r is of the form $u[v]_h^{\alpha} \rightarrow u'[v']_h^{\alpha'}$ **then**
2: Assign α' to the electrical charge of membrane h in $\Pi_j \in C$
3: **end if**

Procedure Swap (D, i, j)

1: $aux \leftarrow D[i]$
2: $D[i] \leftarrow D[j]$
3: $D[j] \leftarrow aux$

Procedure Add-left-hand-rule-objects (r, j, n, C)

1: **if** r is of the form $(x)_{e_j} \xrightarrow{p_{(x,j,j_1,\ldots,j_s)}} (x_1)_{e_{j_1}}, \ldots, (x_s)_{e_{j_s}}$ **then**
2: Add the multisets $x_1^n, \ldots, x_s^n$ to the environments $e_{j_1}, \ldots, e_{j_s} \in C$, respectively.
3: **else if** r is of the form $u[v]_h^{\alpha} \rightarrow u'[v']_h^{\alpha'}$ **then**
4: Add the multiset u'^n to the multiset of the parent membrane of h in $\Pi_j \in C$
5: Add the multiset v'^n from the multiset of membrane h in $\Pi_j \in C$
6: **end if**

V. Performance Evaluation

In order to test the performance of our parallel implementation, we have created four simple P systems with the same skeleton, that has no biological meaning.

The four test P systems are multienvironment functional-probabilistic P systems with active membranes of degrees $(2,2)$, $(4,2)$, $(8,2)$ and $(16,2)$ (2 membranes per P system, and only varying the number of environments per each) with electrical charges $(0,+,-)$, of the following form:

$$\Pi = (G, \Gamma, \mu, R, T, \{f_r : \ r \in R\}, \mathcal{M}_1, \mathcal{M}_2)$$

, where:

- G is a empty graph because $R_E = \phi$.
- $\Gamma = \{a, b, x, y, z, r, f_i, g_i : 1 \leq i \leq 1000\}$
- $\mu = [[]_2]_1$ is the membrane structure.
  - $\mathcal{M}_1 = \{a^{2^{32}}\} \cup \{f_i, g_i : 1 \leq i \leq 1000\}$
  - $\mathcal{M}_2 = \{b^{2^{32}}\}$
- The rules R to apply are:

$$r_1 \equiv a, f_i[b]_2^0 \xrightarrow{0.8} a, f_i[b,x]_2^0, 1 \leq i \leq 1000$$

$$r_2 \equiv a, f_i[b]_2^0 \xrightarrow{0.2} a, f_i[b,y]_2^0, 1 \leq i \leq 1000$$

$$r_3 \equiv a, g_i[b]_2^0 \xrightarrow{0.9} a, g_i[b,z]_2^0, 1 \leq i \leq 1000$$

$$r_4 \equiv a, g_i[b]_2^0 \xrightarrow{0.1} a, g_i[b,r]_2^0, 1 \leq i \leq 1000$$

These four P systems have been simulated using PLingua-Core with the three available implementations written in Java: binomial block based (binomial in short), DNDP sequential and DNDP parallel algorithms. The simulations have been executed in a computer with an Intel core2 Quad Q9550 system running at 2.83GHz with 4GB of main memory, and using Ubuntu Linux Server 8.04 with the Java Virtual Machine 1.6.0.

Figure 1 shows the simulation time for the three implementations simulating the example depicted above. It can be seen that the DNDP sequential code is a little bit faster than the binomial one. As the complexity of the P system increments (in this example, the number of environments), the DNDP reaches better performance. In this experiment, we report up to 1,07x of speedup for the P system with 16 environments.

Furthermore, the parallel implementation of the DNDP algorithm has a better overall performance than the other two. Using a 4 cores based computer, the Java Virtual Machine distributes the threads (assigned to each environment of the system) among them. In this way, for the P system with 16 environments, we report a 1,72x of speedup with respect to the DNDP sequential, and 1,84x with the binomial.

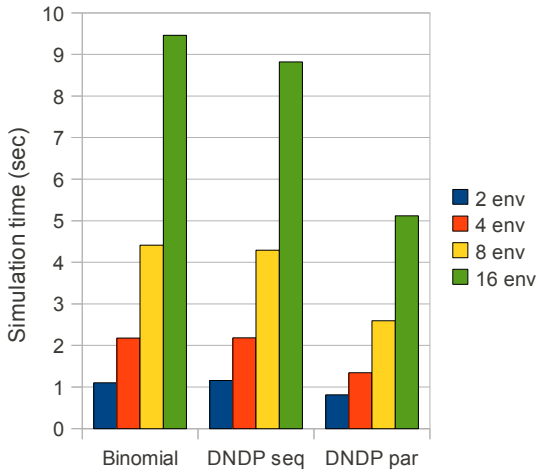

Fig. 1. Simulation time of the three different implementations in a 4 cores based computer

VI. Conclusions and Future Work

In this paper we have introduced a novel algorithm for multienvironment functional probabilistic P systems with active membranes. The new applications of this model to the field of Population (Ecosystems) Dynamics leads to search for new and efficient algorithms.

We have presented the previous algorithm, called binomial block based algorithm, and studied the deficiencies in the simulation. Then, we have based on the idea presented by V. Nguyen et al. [8] in order to improve it. Finally, we have developed an algorithm, called DNDP, performing object dis-

tribution, maximal consistency and calculation of probability functions. A parallel implementation is also provided.

The algorithm DNDP is divided in two phases, one for selection and other for execution of rules. The first phase is also divided in other two micro phases, a first selection phase which generates a consistent set of applicable rules, and a second selection phase that changes the previous set for maximal application.

We show that this algorithm solvents the restrictions presented in the previous one, and also achieves better performance thanks to the parallel implementation. However, the algorithm needs to be more flexible. If the model to be simulated does not have normalized probabilities for object competition, the simulator would need to perform the normalization automatically.

Moreover, the models are increasing in complexity, so that efficient and parallel implementations have to be developed. We will study to use parallel architectures, such as GPGPUs and CUDA [5], to speedup the simulation.

ACKNOWLEDGEMENT

The authors acknowledge the support of the project TIN2009–13192 of the Ministerio de Educación y Ciencia of Spain, cofinanced by FEDER funds, and the support of the "Proyecto de Excelencia con Investigador de Reconocida Valía" of the Junta de Andalucía under grant P08-TIC04200.

REFERENCES

[1] D. Besozzi, P. Cazzaniga, D. Pescini, G. Mauri. Modelling metapopulations with stochastic membrane systems, Biosystems, 91 (2008), 499–514.

[2] L. Bianco, V. Manca, L. Marchetti, M. Petterlini. Psim: a simulator for biomolecular dynamics based on P systems, IEEE Congress on Evolutionary Computation (2007), 883–887

[3] M. Cardona, M.A. Colomer, A. Margalida, A. Palau, I. Pérez-Hurtado, M.J. Pérez-Jiménez, D. Sanuy. A computational modeling for real ecosystems based on P systems, Natural Computing online version (DOI: 10.1007/s11047-010-9191-3).

[4] M. Cardona, M.A. Colomer, A. Margalida, I. Pérez-Hurtado, M.J. Pérez-Jiménez, D. Sanuy. A P system based model of an ecosystem of some scavenger birds, Membrane Computing, 10th International Workshop, LNCS 5957 (2010), 182–195.

[5] J.M. Cecilia, J.M. García, G.D. Guerrero, M.A. Martínez-del-Amor, I. Pérez-Hurtado, M.J. Pérez-Jiménez. Simulation of P systems with active membranes on CUDA, Briefings in Bioinformatics, 11, 3 (2010), 313–322

[6] S. Cheruku, A. Păun, F.J. Romero-Campero, M.J. Pérez-Jiménez, O.H. Ibarra. Simulating FAS-induced apoptosis by using P systems, Progress in Natural Science, 17, 4 (2007), 424–431.

[7] M. García-Quismondo, R. Gutiérrez-Escudero, I. Pérez-Hurtado, M.J. Pérez-Jiménez, Agustín Riscos-Núñez. An overview of P-Lingua 2.0, Membrane Computing, 10th International Workshop, LNCS 5957 (2010), 264–288.

[8] V. Nguyen, D. Kearney, G. Gioiosa. An algorithm for non-deterministic object distribution in P systems and its implementation in hardware, Membrane Computing, 9th International Workshop, LNCS 5391 (2009), 325–354.

[9] G. Păun. Computing with membranes. Journal of Computer and System Sciences, 61, 1 (2000), pp. 108–143, and Turku Center for Computer Science-TUCS Report No 208.

[10] Gh. Păun, F.J. Romero-Campero. Membrane Computing as a Modeling Framework. Cellular Systems Case Studies, Formal Methods for Computational Systems Biology, Lecture Notes in Computer Science, 5016 (2008), 168–214.

[11] M.J. Pérez-Jiménez, F.J. Romero-Campero. P systems, a new computational modelling tool for systems biology, Transactions on Computational Systems Biology VI, LNCS 4220 (2006), 176–97.

[12] G. Terrazas, N. Krasnogor, M. Gheorghe, F. Bernardini, S. Diggle, M. Cámara. Environment aware P-System model of quorum sensing, New Computational Paradigms, LNCS 3526 (2005), 473–485.