Stochastic Approaches in P Systems for Simulating Biological Systems

Paolo Cazzaniga[1], Dario Pescini[1], Francisco J. Romero-Campero[2], Daniela Besozzi[3], Giancarlo Mauri[1]

[1] Università degli Studi di Milano-Bicocca
Dipartimento di Informatica, Sistemistica e Comunicazione
Via Bicocca degli Arcimboldi 8, 20126 Milano, Italy
cazzanipa/pescini/mauri@disco.unimib.it

[2] University of Sevilla
Department of Computer Science and Artificial Intelligence
Avda. Reina Mercedes s/n, 41012 Sevilla, Spain
fran@us.es

[3] Università degli Studi di Milano
Dipartimento di Informatica e Comunicazione
Via Comelico 39, 20135 Milano, Italy
besozzi@dico.unimi.it

Summary. Different stochastic strategies for modeling biological systems with P systems are reviewed in this paper, such as the multi-compartmental approach and dynamical probabilistic P systems. The respective results obtained from the simulations of a test case study (the quorum sensing phenomena in Vibrio Fischeri colonies) are shown, compared and discussed.

1 Introduction

The aims of this work are (*i*) to review some stochastic approaches that were recently defined in the framework of P systems as new tools for the modeling of biological systems, (*ii*) to compare their respective results in simulating the behavior of bacteria colonies, and finally (*iii*) to discuss their advantages, as well as their limits and possible improvements. The models used to describe biological systems are usually huge and complex although few copies of each reactant are present inside them. This is the reason why, in this framework, the classical approach with differential equations is not suitable. This has driven the direction of research to look for experimental evidences of the stochastic noise in these kinds of phenomena, and the importance of this behavior is now clearly stated in the literature, as reported in [4].

Here, we briefly report some notes on a well known and used stochastic algorithm, which is also the underlying structure at the core of the stochastic strategies

reviewed in this paper. Namely, the exact *stochastic simulation algorithm* (SSA), introduced by Gillespie in [3], is applicable to well-stirred chemical reaction systems contained inside a single fixed volume, at a constant temperature. A biochemical system of this type is assumed to contain N molecular species $S_1, \ldots, S_N$ interacting through M reactions channels $R_1, \ldots, R_M$, with reaction constants $c_1, \ldots, c_M$ which only depend on the physical condition of the system (e.g., the temperature) and on properties of the molecules. At time t, $X_i(t)$ molecules of the species S_i are assumed to be present inside the system, for any $i = 1, \ldots, N$.

The SSA algorithm is based on the fundamental hypothesis that $c_\rho \delta t$, for any $\rho = 1, \ldots, M$, is the average probability that a particular reactants combination for reaction R_ρ will react in the next time interval δt. According to the current state of the system, this assumption allows to derive $P(\tau, \rho)d\tau$, that is, the probability determining (1) the next reaction that will occur in the volume V during the next differential time interval $(t + \tau, t + \tau + d\tau)$, and (2) that such reaction will be R_ρ. Thus, the SSA algorithm computes both *which* reaction will be the next one in the system, and *when* it will be actually applied.

The explicit derivation of $P(\tau, \rho) = P_1(\tau)P_2(\rho)$ involves the definition of the propensity function $a_\rho = h_\rho c_\rho$, where h_ρ is the number of distinct reactant molecules combinations and c_ρ is the stochastic rate constant associated to reaction R_ρ. By defining $a_0 = \sum_{\rho=1}^{M} a_\rho$, the explicit expressions for the probabilities are

$$P_1(\tau) = a_0 e^{-a_0 \tau}, \tag{1}$$

$$P_2(\rho) = \frac{a_\rho}{a_0}, \tag{2}$$

which allow to retrieve the information for *when* (τ) and *which one* (ρ) will be the next applied reaction from the probability density function $P(\tau, \rho)$, just by extracting two random numbers r_1 and r_2 from the uniform unit-interval distribution and then taking

$$\tau = \frac{1}{a_0} \ln\left(\frac{1}{r_1}\right) \tag{3}$$

as the time of the next reaction, and choosing the integer corresponding to the next reaction by evaluating

$$\sum_{j=1}^{\rho-1} a_j < r_2 a_0 \leq \sum_{j=1}^{\rho} a_j \ . \tag{4}$$

The SSA algorithm has been widely used as the seed and the main reference model for many stochastic simulations in biochemistry and biology, and it is implemented in many of the most popular cellular simulators within the Systems Biology Markup Language project [13], such as E-Cell, Cellware, Dizzy, Stocks, SBW.

Given this brief review of the Gillespie algorithm, the rest of the paper is structured as follows. In Section 2 we recall some stochastic strategies defined

so far in the area of P systems, namely the multi-compartmental approach and sequential/parallel dynamical probabilistic P systems. Then, in Section 3 we report the simulations on a biological case study, the quorum sensing in Vibrio Fischeri, performed by the different stochastic approaches. We also compare the obtained results in Section 4. We conclude with some final remarks and ideas for future work.

2 Different Strategies for Stochastic Simulations in P Systems

In this section we recall some stochastic strategies, based on SSA, appeared so far in P systems for the modeling of biological or biochemical systems. In Section 3 we will then compare their respective outcomes on a specific biological system, used as a test case study.

2.1 Multi-Compartmental Approach

In this section we briefly summarize the multi-compartmental algorithm, introduced in [1], [2] and lately applied in [7]. The aim of this method is to exploit the P systems topology, namely, the hierarchical structure of its compartments. The core of this approach, based on the Gillespie one, is the creation of an appropriate common list for all the applicable rules occurring in all the membranes, at each time step. The sorting of this list (over time) then allows to choose always the first reaction that has to be applied in the whole system. This approach is intrinsically sequential and inherits the precision of the SSA, but with the additional feature of working with different compartments and moving objects among compartments (by means of the classical communication rules in P systems).

Let us consider a P system $\Pi = (O, Lab, \mu, M_1, M_2, \ldots, M_n, R_1, \ldots, R_n)$, where each membrane initially contains the multiset M_m and is labeled with a symbol in Lab (different membranes can have the same label), and with the set of rules R_m, $1 \leq m \leq n$.

Each membrane m is considered a compartment enclosing a volume, therefore the index ρ of the next rule to be used inside membrane m, and its waiting time τ, will be computed using the classical Gillespie algorithm, which returns the triple (τ_m, ρ_m, m).

The procedure used in the multi-compartmental approach is the following:

- **Initialization**
  - set time of the simulation $t = 0$;
  - for each membrane m in μ compute a triple (τ_m, ρ, m) using the SSA; construct a list containing all these triples;
  - sort the list of triples (τ_m, ρ, m) according to τ_m (in decreasing order);
- **Iteration**
  - extract the first triple (τ_m, ρ, m) from the list;

- set time of the simulation $t = t + \tau_m$;
- update the waiting time for the rest of the triples in the list by subtracting τ_m;
- apply the rule ρ only once, accordingly changing the number of objects in the membranes affected by the application of the rule;
- for each membrane m' affected by the application of the rule ρ, remove the corresponding triple $(\tau'_{m'}, \rho', m')$ from the list;
- for each membrane m' affected by the application of the rule ρ, run again the Gillespie algorithm for the new context in m' to obtain the triple $(\tau''_{m'}, \rho'', m')$, the next rule ρ'' to be used inside membrane m', and its waiting time $\tau''_{\rho'}$;
- add the new triples $(\tau''_{m'}, \rho'', m')$ in the list, sort this list according to each waiting time and iterate the process.

• **Termination**
  - Terminate the procedure when time t reaches or exceeds a preset maximal time of simulation.

As a general remark about the multi-compartmental approach, we can say that it is useful when one wants to simulate small systems that involve several compartments, each one with a homogeneous distribution of molecules. Exploiting the P systems structure, and possibly defining "virtual" compartments (e.g., regions corresponding to the cellular membrane bilayers) besides the real biological ones, it might be possible to reproduce the previous outlined conditions for many biological processes, thus opening P systems to a vast area of applications.

This approach encapsulates the Gillespie algorithm in each compartment, the resulting description of the dynamics of each membrane is exact and the overall description of the system is accurate.

It is well known that Gillespie algorithm does not scale linearly with the number of objects and the number of rules defined in the system, and this property is obviously inherited also by the multi-compartmental approach. Even a different programming version for the code (i.e., distributed over a set of processors) would not significantly reduce this difficulty, since the structure of the algorithm itself forces to apply one rule at each step. In fact, once that the rule has been fired, the computation takes place only in those membranes affected by the application of this rule. This limits the benefits of a distributed coding only in the initialization stage of the algorithm, where all the membrane compute the propensity function in order to obtain the triples to be inserted in the list.

2.2 Dynamical Probabilistic P Systems

In this section we review the stochastic approach underlying dynamical probabilistic P system (DPP, in short), introduced in [8] and lately developed and applied in [9, 10] (the code is available in the software section of the P systems web page [12]). The aim of this strategy is to capture the major features of a membrane system (compartments, communication, parallelism) and to understand how they could

be exploited in realistic simulations of biological systems. In particular, while the importance and the applicability of a membrane structure has been largely investigated from a practical and a theoretical point of view, it is a matter of fact that the role of maximal parallelism in nature is still not completely understood (and one of the main motivation of this work).

We recently implemented the DPP approach using the MPI (Message Passing Interface) C libraries [11] in order to: (*i*) spread the computation over a cluster of processors to achieve scalability, (*ii*) have a direct mapping of the communications among the membranes, and (*iii*) speed up the computation. As a direct outcome, the performances obtained by DPP with MPI implementation allow to easily and actually execute parallel processes, managing the communication between them, and thus to achieve higher scalability than the multi-compartmental approach.

In the following, we summarize the algorithm used for the evolution of a DPP, referring the reader to [9], where the original version running on a single machine is described. Here we remark that, with respect to the single processor program of DPP, the use of the MPI library affects only the system communications while preserving the rest of the code. In this way each membrane evolves independently from the others for everything but the communication, such that it could reside on an independent process.

Each membrane m in a DPP will be considered as a single volume, and will be simulated using an MPI process. All the membranes of a DPP simultaneously evolve, and require to be synchronized at the end of each step to allow the communication process. Since in each membrane the rules are applied in a maximally parallel way, in each membrane we can divide a step in three stages: (1) the computation of the probability distribution for the rules, (2) the assignment to the rules of all objects which can be modified by rules, (3) the communication and multisets updating.

The corresponding algorithm is:

- **Initialization**
  - create all parallel processes;
  - set the current step $t = 0$;
  - load the set of rules and the initial multisets for each process.

- **Iteration (for each process)**
  - make a snapshot of the current status of the system;
  - calculate the propensity functions of rules with SSA;
  - toss the rules until all the objects are exhausted, and build the relative trace using the snapshot;
  - update the system state according to the generated trace and, for each rule, do what follows:
    - if the rule target is *here*, update the multisets values according to the left and right side of the rule;

    - if the rule target is *in* or *out*, decrease the value of the local multisets according to the left side of the rule, and send the message to the process (i.e., the membrane) indicated by the target of the rule;
    - wait until all other processes have executed one rule;
  - check any incoming messages (i.e., objects from other membranes) and increase the value of the multisets according to the received objects;
  - increment by one the step value.
- **Termination**
  - terminate the simulation when the current step reaches the maximal value previously fixed.

Note that the waiting step in the algorithm is needed to synchronize the execution between the membranes, otherwise a process i could, for instance, check for any received messages before another process j will send a message to i.

An advantage of this approach consists in the fact that, with slight changes in the program code, it is possible to probe various degrees of parallelism. For instance, by forcing the simulator to apply one rule at each step it is possible to obtain a parallel version of the Gillespie algorithm. In this case, if in each membrane we also allow to compute the τ of its rules, then each membrane evolves with its own time stream. One common step corresponds to a potentially different τ in each membrane, thus creating some paradox when objects are to be communicated between two membranes. An example could be a message received in a membrane in a time that corresponds to the past time of the sending membrane. This approach will be referred hereafter as 1 rule dynamical probabilistic P system (1rDPP).

Another way to probe the role of parallelism in this framework is the insertion of rules of the type $A \rightarrow A$, which will compete for the same objects with other rules having A as the left side, without affecting the system evolution, so that the result is a reduced consumption of these objects. Examples of these strategies will be given in Sections 3.3 and 3.4.

Obviously, the main problem that affects a DPP approach is the lack of a defined common time stream for all the membranes, which only allows to derive qualitative descriptions of the modeled phenomena. A deeper investigation of this problem, and of the role of parallelism in nature, is already under development and will appear elsewhere.

The added value of this approach is the independent evolution of the membranes, which is fundamental to lead to a reliable and scalable simulator for large and complex systems, such as biological ones. A pure sequential approach binds the computational efficiency of the simulation to the efficiency of the computer on which the simulation itself is running, while the possibility of sharing the work among multiple processing units allows to scale the simulation with respect to the number of the membranes involved.

3 The Different Strategies in Practice

In Section 3.1 we briefly describe the quorum sensing phenomena in a Vibrio Fischeri bacteria colony and the corresponding P system based model presented in [2] and [1]. In Sections 3.2, 3.3, 3.4 we apply the strategies previously described on this biological model, used as a test case study.

3.1 Quorum Sensing in Vibrio Fischeri. A Case Study

The marine bacterium Vibrio Fischeri regulates the expression of certain genes in response to population density using a family of transcriptional regulators, in a process called *auto-induction*. In the free living state these bacteria are non luminescent, but they start to produce light when are at high cell density.

The quorum sensing mechanism in Vibrio Fischeri relies on the synthesis, accumulation and subsequent sensing of a signal molecule, which we will call OHHL. When only a small number of bacteria are present, the signal is produced by the bacteria at a low level. OHHL diffuses out of the bacterial cells and into the surrounding environment. At high cell density, the signal accumulates in the area surrounding the bacteria and can also diffuse to the inside of the bacterial cells. The signal is able to interact with the LuxR protein to form the complex LuxR-OHHL, which binds to a region of DNA called the Lux Box causing the transcription of the luminescence genes, a small cluster of 5 genes, luxCDABE. Adjacent to this cluster are two regulatory genes for the transcription of LuxR and OHHL. In this sense OHHL and LuxR are said to be auto-inducers because they activate their own synthesis.

In this section we present the model used in [6] to simulate a colony of bacteria of this type, in order to reproduce the mechanisms at the base of the luminescent behavior observed in nature.

Using the framework of P systems, each bacterium is represented by a membrane, and the skin membrane of the system is intended to enclose the environment (the depth of the system is 2, because each bacteria is an elementary membrane). In this way we can look at the evolution of every single bacterium, as well as at the evolution of the whole system.

The system used for the simulations is

$$\Pi_{Vf}(N) = (O, \{e, b\}, \mu, (w_1, e), (w_2, b), \ldots, (w_{N+1}, b), \mathcal{R}_b, \mathcal{R}_e), \tag{5}$$

where:

1. N represents the number of bacteria in the colony;
2. the alphabet is:

$$O = \{\text{OHHL}, \text{LuxR}, \text{LuxR-OHHL}, \text{LuxBox}, \text{LuxBox-LuxR-OHHL}\};$$

3. the membrane structure is: $\mu = [_e [_b \,]_b \ldots [_b \,]_b]_e$;
4. the initial multisets are: $w_1 = \emptyset,\ w_i = \{\text{LuxBox}\}, \quad 2 \leq i \leq N + 1$;

5. the rules used inside the system are:
   $r_1 : [\ \text{LuxBox}\ ]_b \xrightarrow{c_1} [\ \text{LuxBox}\,,\ \text{OHHL}\ ]_b$
   $r_2 : [\ \text{LuxBox}\ ]_b \xrightarrow{c_2} [\ \text{LuxBox}\,,\ \text{LuxR}\ ]_b$
   $r_3 : [\ \text{LuxR}\,,\ \text{OHHL}\ ]_b \xrightarrow{c_3} [\ \text{LuxR-OHHL}\ ]_b$
   $r_4 : [\ \text{LuxR-OHHL}\ ]_b \xrightarrow{c_4} [\ \text{LuxR}\,,\ \text{OHHL}\ ]_b$
   $r_5 : [\ \text{LuxBox}\,,\ \text{LuxR-OHHL}\ ]_b \xrightarrow{c_5} [\ \text{LuxBox-LuxR-OHHL}\ ]_b$
   $r_6 : [\ \text{LuxBox-LuxR-OHHL}\ ]_b \xrightarrow{c_6} [\ \text{LuxBox}\,,\ \text{LuxR-OHHL}\ ]_b$
   $r_7 : [\ \text{LuxBox-LuxR-OHHL}\ ]_b \xrightarrow{c_7} [\ \text{LuxBox-LuxR-OHHL}\,,\ \text{OHHL}\ ]_b$
   $r_8 : [\ \text{LuxBox-LuxR-OHHL}\ ]_b \xrightarrow{c_8} [\ \text{LuxBox-LuxR-OHHL}\,,\ \text{LuxR}\ ]_b$
   $r_9 : [\ \text{OHHL}\ ]_b \xrightarrow{c_9} \text{OHHL}\ [\ ]_b$
   $r_{10} : [\ \text{OHHL}\ ]_b \xrightarrow{c_{10}} [\ ]_b$
   $r_{11} : [\ \text{LuxR}\ ]_b \xrightarrow{c_{11}} [\ ]_b$
   $r_{12} : [\ \text{LuxR-OHHL}\ ]_b \xrightarrow{c_{12}} [\ ]_b$
   $r_{13} : \text{OHHL}\ [\ ]_b \xrightarrow{c_{13}} [\ \text{OHHL}\ ]_b$
   $r_{14} : [\ \text{OHHL}\ ]_e \xrightarrow{c_{14}} [\ ]_e$

The rules from r_1 to r_{12} are placed inside each bacterium, whereas rules r_{13} and r_{14} are placed inside the environment.

The following set of parameters have been chosen for running simulations: $c_1 = 2,\ c_2 = 2,\ c_3 = 9,\ c_4 = 1,\ c_5 = 10,\ c_6 = 2,\ c_7 = 250,\ c_8 = 200,\ c_9 = 50,\ c_{10} = 30,\ c_{11} = 20,\ c_{12} = 20,\ c_{13} = 0.1 \cdot N,\ c_{14} = 1$. These stochastic rate constants (expressed in hours^{-1}) have been taken or suggested by the literature (see [6] and references therein).

3.2 Multi-Compartmental Approach

In this section we present the results obtained by simulating the Vibrio Fischeri colony model with the multi-compartmental approach.

Two different kinds of membranes are to be initialized: the environment, and the internal membranes corresponding to bacteria. Moreover, all bacteria share the same set of rules.

As previously said, every membrane evolves with the SSA dynamics, apart from the execution of communicating rules, because, in such cases, the rule can affect two membranes (i.e., the environment and one bacterium). It is clear that there are two kinds of communication rules: (1) the environment executes a communicating rule nondeterministically choosing a bacterium of the colony and sending one OHHL signal into it; (2) the bacterium applies a deterministic communicating rule always sending an OHHL signal to the environment.

The multi-compartmental approach is suitable for the simulation of this model because it can trace the behavior of the single bacterium, as well as the behavior of the whole system. In the following, we present the results of the simulations for colonies consisting of 10, 300, 400, 500, and 600 bacteria.

In Figure 1 the behavior of a single bacterium is shown, where the production of light (after about 20 hours of simulation time) is due to the fact that this one

"guessed wrong" the concentration of the population and got up-regulated. But then, after sensing that the signal does not accumulate in the environment, it switched off its systems. Auto-quoration is a rare event but it is still possible due to the stochastic nature of the simulation algorithm, as the figure shows.

In Figure 2 we show the behavior of a colony of 10 bacteria. During the simulation of 30 hours of real life, the signal inside the environment does not exceed 130 copies and there are at most 2 quorate bacteria at the same time.

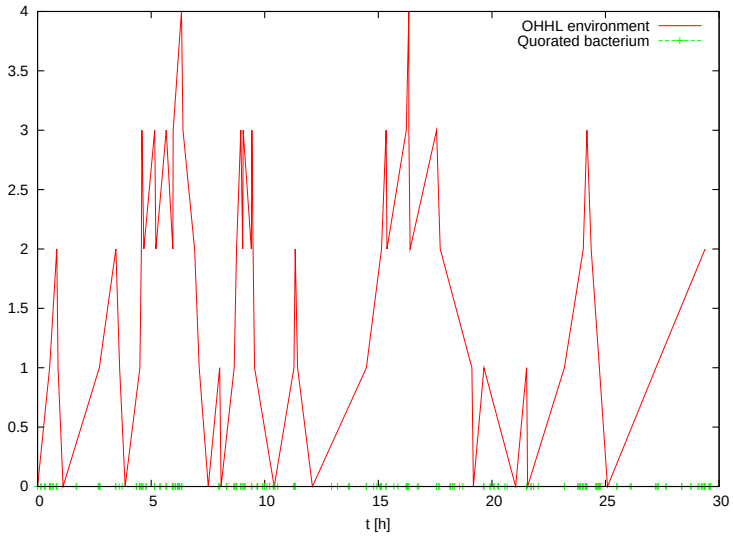

Fig. 1. Single Bacterium Behavior

In Figures 3, 4, 5 and 6 we represent the behaviors of the other colonies (namely, with 300, 400, 500 and 600 bacteria) studied with this approach; it is possible to see that the OHHL signal starts to accumulate at the beginning of the simulation and also the number of quorate bacteria increases until the signal in the environment reaches a steady state.

In Figure 7 the number of OHHL signal in the environment and the number of quorate bacteria of all the simulations are shown. It is possible to see that, as the number of the bacteria of the colony increases, the number of the quorate bacteria increases as well, but the number of OHHL signal inside the environment is about the same for all the simulations. This is due to the fact that in each bacterium the probability to send out a signal does not depend on the whole number N of bacteria in the colony, but at the steady state they will have an internal common average concentration of OHHL, since they share the same constants. On the contrary, in the environment the probability to send a signal inside a bacterium is proportional to N (see the definition of c_{13}). The net effect is that, at the steady state, the ratio between these probabilities is independent from the dimension of the colony.

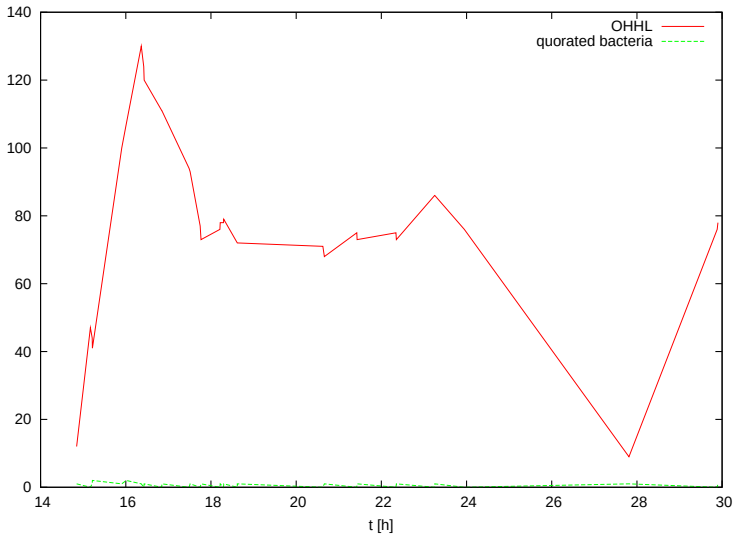

Fig. 2. 10 Bacteria Colony

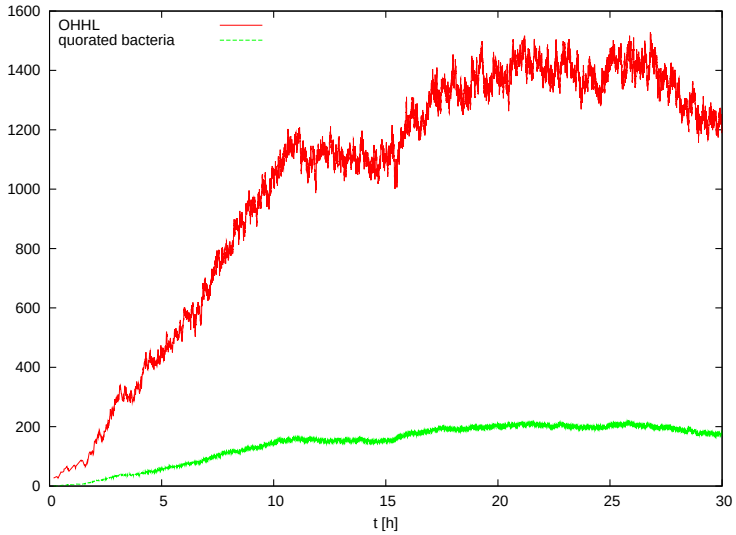

Fig. 3. 300 Bacteria Colony

An advantage of using this approach for simulating biological systems with different volumes is that we can look at the behavior of every single volume, as well as that of the whole system. Moreover, choosing the next applied rule and its waiting time according to the pure SSA, it is possible to trace the simulation time

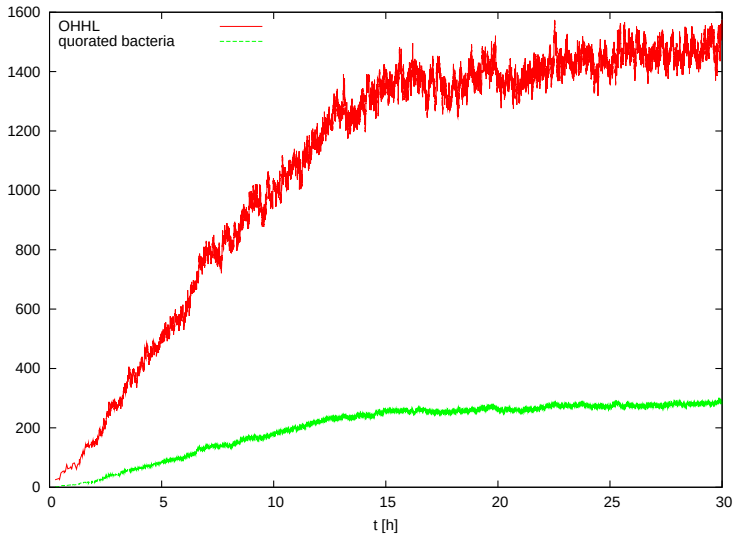

Fig. 4. 400 Bacteria Colony

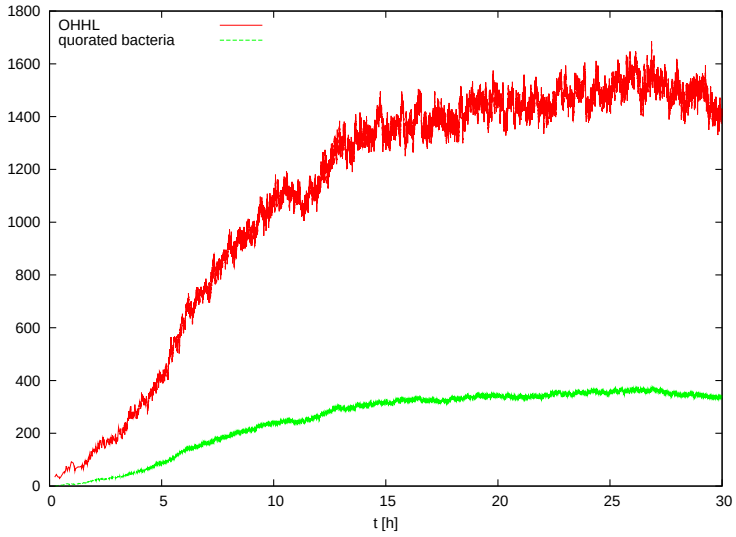

Fig. 5. 500 Bacteria Colony

line. Therefore, we can achieve a quantitative description of the evolution of the system.

Anyway, the main problem of this approach remains at the level of scalability, since incrementing the number of bacteria means to correspondingly increase (in a

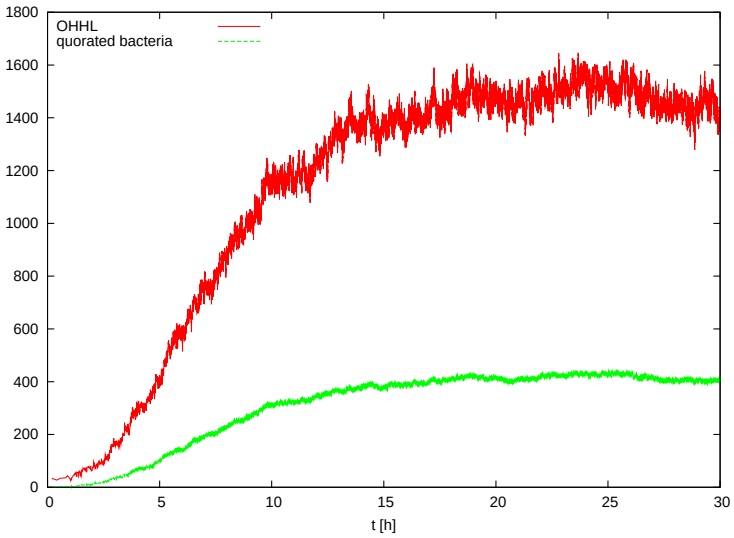

Fig. 6. 600 Bacteria Colony

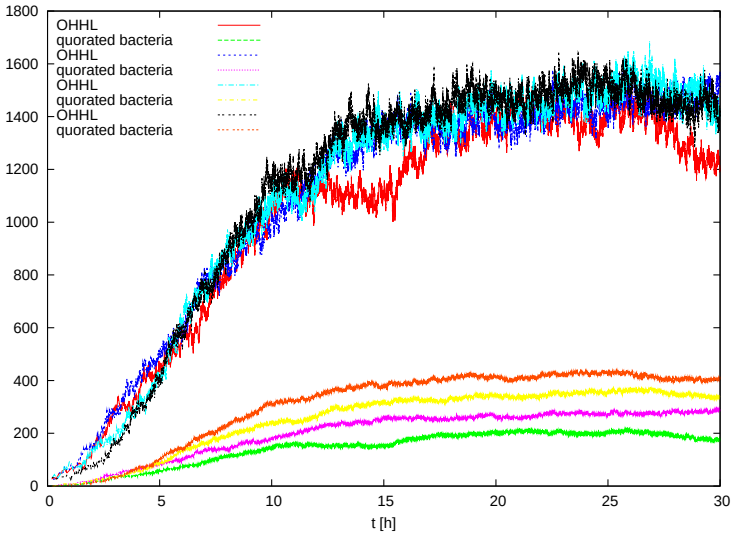

Fig. 7. Comparison between the simulations

non linear mode) the time needed to do the simulations. In Figure 8, the number of bacteria vs the time needed for computations are plotted.

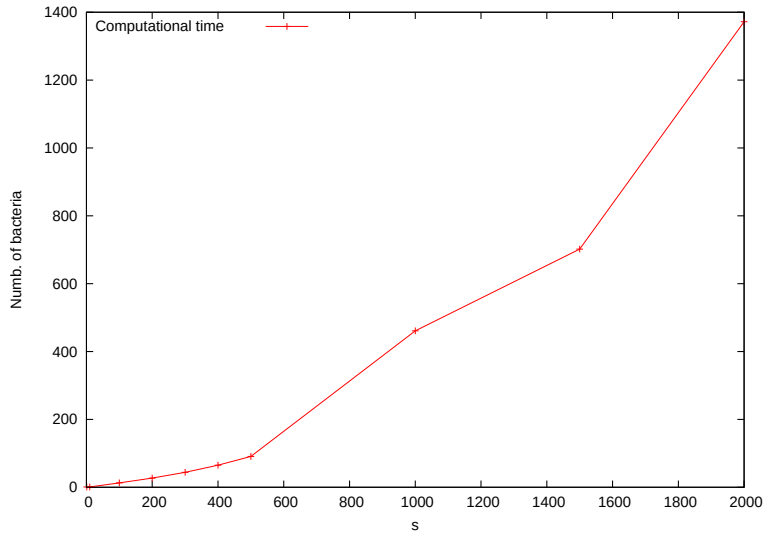

Fig. 8. Non linear time increment

3.3 One Rule DPP

The simulations performed with this approach have been run using the same initial conditions described in Section 3.1. Here the simulation begins initializing $N + 1$ identical processes (where N is the number of bacteria, plus the environment).

Two colonies consisting of 10 and 300 bacteria were simulated with the DPP approach with the restriction that only one rule per step is allowed.

In Figure 9 the behavior of the simulated colony of 10 bacteria is shown. It is possible to see that the OHHL signal inside the environment starts to accumulate, because the environment is not able to send or to degrade the signal because of the sequential nature of its internal evolution. There is only one bacterium that guessed wrong the colony density and starts to produce light.

The second simulation done with this approach is for a colony of 300 individuals, see Figure 10. The observed behavior is different from the expected one, in the sense that although the OHHL signal is accumulated inside the environment, there is a small number of quorate bacteria (this is easier comprehensible by setting a logarithmic scale to the ordinate axis, as in Figure 11).

This kind of evolution is due to the fact that all the processes evolve in parallel, executing one rule per step. For this reason the environment can send just one OHHL signal per step to a nondeterministically chosen bacterium. This is quite unrealistic, compared with the behavior observed in nature, because in this simulations the environment is not able to send signals to the bacteria in a "faster" way, so the bacteria cannot sense the real density of the colony. In this way, the luminescent state of the colony is not reached although the bacteria would be enough to start to produce light.

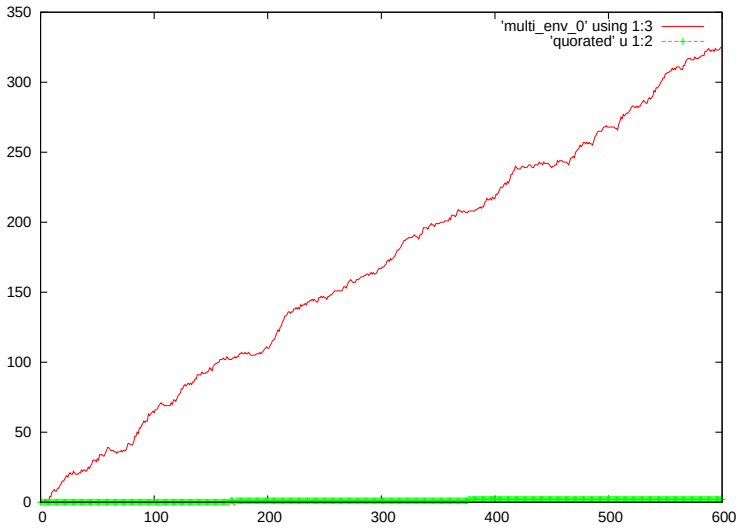

Fig. 9. 10 Bacteria Colony

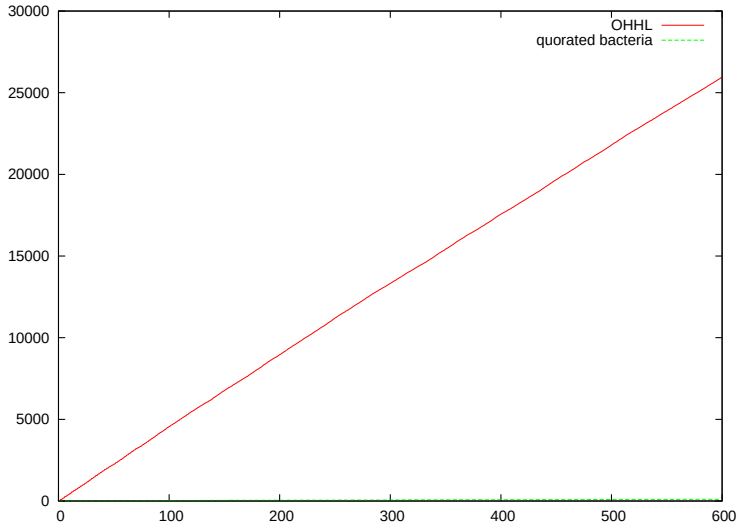

Fig. 10. 300 Bacteria Colony

In the next section we present the behavior of a mixed system in which the environment behaves according to a (parallel) DPP approach, while the single bacteria behave according to the 1 rule DPP approach.

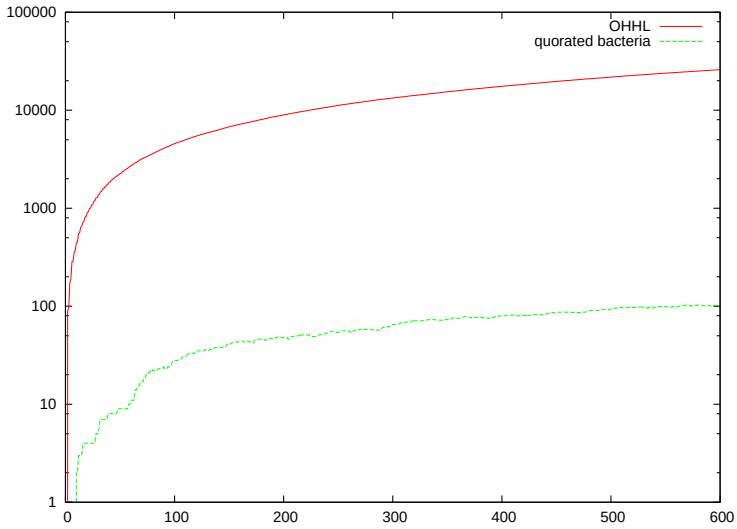

Fig. 11. 300 Bacteria Colony – y log scale

3.4 DPP

In the specific case study of quorum sensing in Vibrio Fischeri, this approach puts the environment in a position to send more than one OHHL signal per step to the bacteria of the colony. Hence, using DPP, a maximal parallelism at the level of the objects consumption inside the environment is introduced. The bacteria of the colony here still evolve in sequential way (one rule each step).

Like the multi-compartmental approach, here we have simulated colonies of 10, 300, 400, 500 and 600 individuals, starting from the same initial conditions.

In Figure 12 the behavior of a colony consisting of 10 bacteria is shown. Using the maximal parallelism inside the environment, the signal does not accumulate, a high number of OHHL is sent inside the bacteria. This is the reason why the bacteria got quorate although the density of the colony is too low to start to produce light.

The evolution of the other colonies are quite similar as shown in Figures 13, 14, 15 and 16, because they follow the same dynamic. At the beginning of the computation, a large number of OHHL signal is sent from the bacteria to the environment and, using the parallel application of the rules, the signal is sent back from the environment to the bacteria.

Thanks to the signal sent from the environment, the bacteria sense that the colony density is enough to start to produce light and an increasing number of them got quorate.

The problem of this approach is that we have just a qualitative description of the evolution. We can only observe the cardinality of the multisets at each step, but the time evolution of the system cannot be traced (just the internal

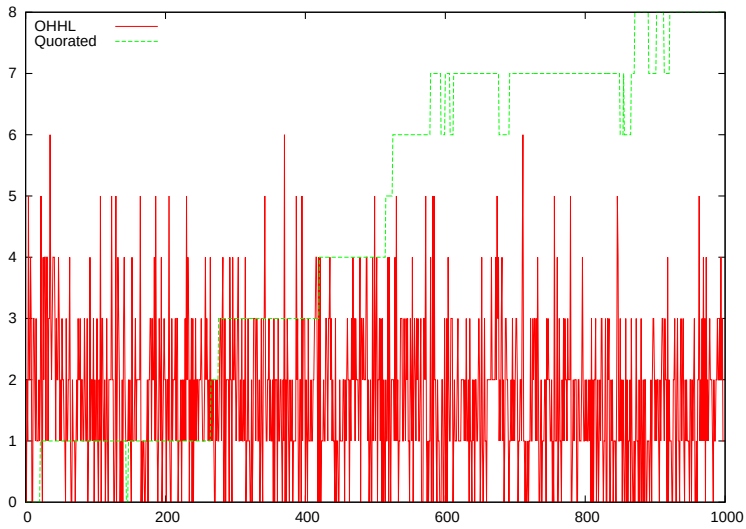

Fig. 12. 10 Bacteria Colony

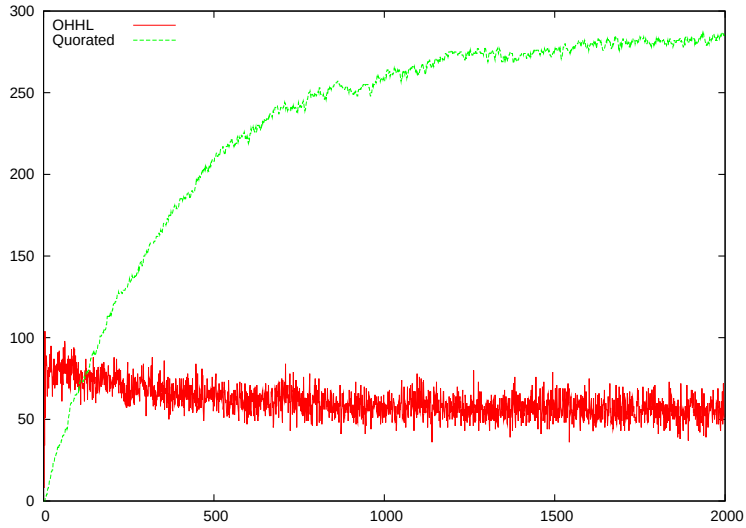

Fig. 13. 300 Bacteria Colony

time of every membrane could be). However, using this approach we can share out the load of work between different nodes of a cluster. In this way, it is possible to simulate larger colonies in a smaller time, compared with the multi-compartmental approach.

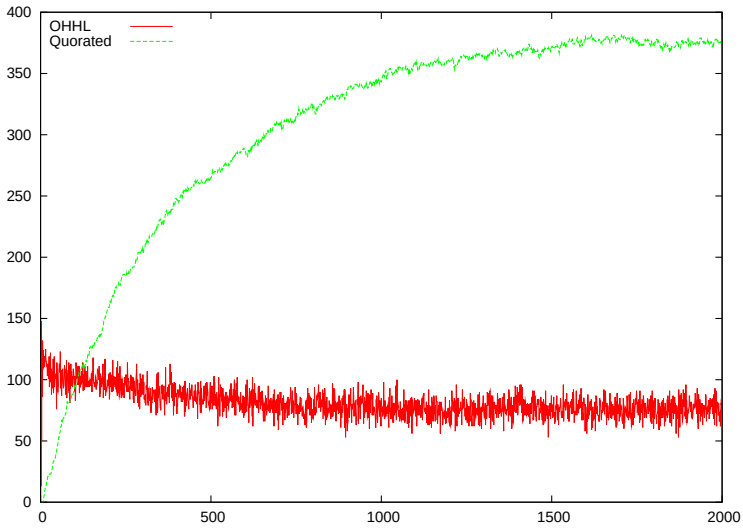

Fig. 14. 400 Bacteria Colony

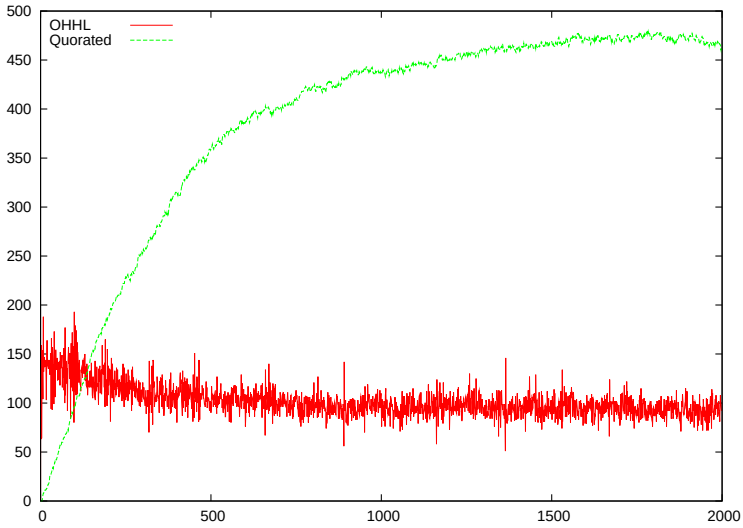

Fig. 15. 500 Bacteria Colony

4 Conclusion and Future Work

In this work we have shown that the stochastic strategies here presented are able to reproduce the bacterial mechanism of quorum sensing, which describes how each bacterium senses the presence of the other members of the colony.

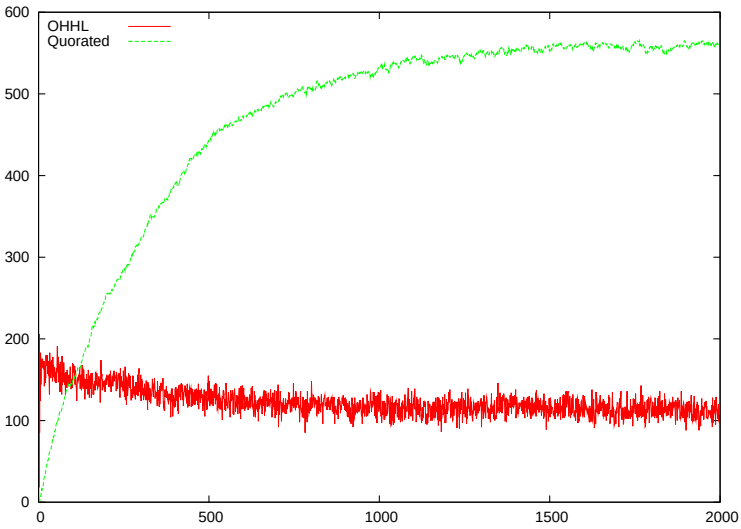

Fig. 16. 600 Bacteria Colony

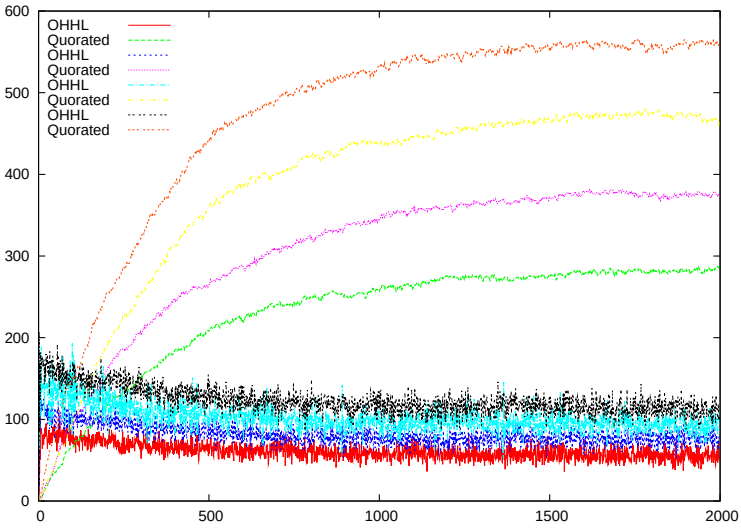

Fig. 17. Comparison between the simulations

The main difference between the results obtained using the multi-mompartmental and the DPP approaches regards the simulation of the 10 bacteria colony. In the first case (Figure 2), the simulated colony behaves in the right way, that is, at most 2 bacteria got quorate (because they guessed wrong the size of the colony) and the OHHL signal does not accumulate in the environment. While

using the DPPs (Figure 12), we obtained the luminescent behavior although the colony concentration is not enough to start to produce light, and the OHHL signal does not accumulate inside the environment, because the signal is immediately redistributed from the environment to the bacteria in a single parallel step.

Looking at the simulations for the larger colonies performed with both approaches, it is possible to see that the number of quorate bacteria of the colonies increases as the cardinality of the colonies increases, as expected. Using the multi-compartmental approach we can observe that the OHHL signal starts to accumulate inside the environment, and then it reaches a steady state value common to all the simulated colonies. The number of quorate bacteria follows the same dynamic of the signal.

With the DPPs, the OHHL signal and the concentration of quorate bacteria do not share the same behavior. A smaller "constant" (average) amount of OHHL signal is present inside the environment in all the colonies, because of the aforementioned parallel redistribution of the signal, while the number of quorate bacteria increases at the beginning of the evolution and then reaches a steady state.

At the moment the great advantage in using the Multi-Compartmental strategy is the defined time stream for the whole simulated system, which allows a quantitative description of the model evolution. The DPP strategy is still at a qualitative description of the system evolution, though the parallel design of this strategy leave room for large improvement (it is currently under deep development) and it is very promising for its computational performances and its scalability. For instance, the introduction of mute rules in the DPPs (i.e., $A \rightarrow A$) allows to simulate different "levels" of parallelism. The 1 rule DPP and the DPP represent the sequential and the maximal parallel approaches, respectively, though between them there exist different levels of parallelism that could be closer to reality and thus it is worthwhile to investigate. In the case study of Vibrio Fischeri, one possible improvement for the DPP model would be to add a mute rule like OHHL $\rightarrow$ (OHHL, *here*) inside the environment, by means of which we can simulate the different levels of parallelism just by changing the rate constant of that rule. Moreover, adding mute rules for every symbol of the alphabet in every membrane, it is easy to simulate the whole system exploiting the parallel processes of the DPPs for every compartment.

References

1. F. Bernardini: *Membrane Systems for Molecular Computing and Biological Modelling.* Doctoral Thesis, The University of Sheffield, Department of Computer Science, November 2005.
2. F. Bernardini, M. Gheorghe, N. Krasnogor, R.C. Muniyandi, M.J. Pérez-Jiménez, F.J. Romero-Campero: On P systems as a modelling tool for biological systems. In *Pre-Proceedings of the 6th International Workshop on Membrane Computing (WMC6)* (R. Freund, G. Lojka, M. Oswald, Gh. Păun, eds.), TU Vienna, July 2005, 193–213.

3. D.T. Gillespie: Exact stochastic simulation of coupled chemical reactions. *Journ. Phys. Chem.*, 81 (1977), 2340–2361.
4. T.C. Meng, S. Somani, P. Dhar: Modelling and simulation of biological systems with stochasticity. *In Silico Biology*, 4 (2004).
5. Gh. Păun: *Membrane Computing. An Introduction.* Springer-Verlag, Berlin, 2002.
6. M.J. Pérez-Jiménez, F.J. Romero-Campero: Modelling Vibrio Fischeri's behaviour using P systems. *Systems Biology Workshop*, ECAL 2005, Sept. 2005.
7. M.J. Pérez-Jiménez, F.J. Romero-Campero: P systems, a new computationl modelling tool for Systems Biology. *Transactions on Computational Systems Biology*, to appear.
8. D. Pescini, D. Besozzi, C. Zandron, G. Mauri: Analysis and simulation of dynamics in probabilistic P systems. In *Proceedings of DNA11 - 11th International Meeting on DNA Computing* (N. Pierce, A. Carbone, eds.), LNCS, to appear.
9. D. Pescini, D. Besozzi, G. Mauri, C. Zandron: Dynamical probabilistic P systems. *International Journal of Foundations of Computer Science*, 17, 1 (2006), 183–204.
10. D. Pescini, D. Besozzi, G. Mauri: Investigating local evolutions in dynamical probabilistic P systems. *Proceedings of SYNASC 2005–TAPS (Workshop on Theory and Applications of P Systems)*, IEEE Computer Press, to appear.
11. The MPI standard website `http://www.mcs.anl.gov/mpi/index.html`.
12. The P systems web page `http://psystems.disco.unimib.it/`.
13. The Systems Biology Markup Language web page `http://www.sbml.org`.