# Distributed Simulation of Bio-interpreted (Coloured) Stochastic Petri Nets

Asmaa Badawy
Department of Mathematics and Computer Science,
Faculty of Science, Port Said University,
42521 - Port Said, Egypt
Email: asmaa.magdy931@yahoo.com

Mostafa Herajy
Department of Mathematics and Computer Science,
Faculty of Science, Port Said University,
42521 - Port Said, Egypt
Email: mherajy@sci.psu.edu.eg

Wael Awad
Department of Mathematics and Computer Science,
Faculty of Science, Port Said University
42521 - Port Said, Egypt
Email:waelak71@yahoo.com

Monika Heiner
Computer Science Institute,
Brandenburg University of Technology
Postbox 10 13 44, 03013 Cottbus, Germany
Email:monika.heiner@b-tu.de

***Abstract*—**
**To increase our understanding of the interactions between the different cellular and intercellular processes, it is imperative to construct accurate biological models. An important class of these models are usually used to predict the time evolution of the corresponding chemical species over a specified period of time. Stochastic and coloured stochastic Petri nets can be adopted to construct graphical models that can be executed to simulate the firing of a set of reactions in a well-mixed chemical system. For the purpose of this specific application, (coloured) stochastic Petri nets employ internally the stochastic simulation algorithm (SSA) to perform the simulation. Although the result of the SSA is accurate in comparison with similar numerical simulation approaches (e.g., the systems of ordinary differential equations), it is empirically considered as a computationally expensive algorithm. In this paper, we propose an efficient parallel simulation algorithm for SSA to improve the execution of (coloured) stochastic Petri nets. We illustrate our proposal by applying it to a well-known model in the biological context: reaction diffusion. Furthermore, we evaluate the performance of the parallel algorithm and compare its runtime behaviour with the sequential one.**

***Index Terms*—(Coloured Stochastic Petri Nets; Stochastic Simulation Algorithm; Fine-grained Parallel Simulation)**

## I. Introduction

The traditional approach for studying the time evolution of well-mixed biochemical systems usually follows a deterministic way where a set of ordinary equations is constructed and then numerically solved. However, this deterministic simulation usually fails to reproduce the stochastic and random behaviour of the underlying reaction system, specially when these systems contain species of small number of molecules [1]. The stochastic approach can provide alternative methods to simulate these models more accurately in a way that takes into account discreetness and random fluctuations [2]. It can produce correct numerical pathways of systems that contain extremely small numbers of interacting molecules.

The stochastic simulation algorithm (SSA) is one realisation of the stochastic framework which was developed by Gillespie in [3]. It provides an exact numerical procedure to produce time evaluation for the chemical species interacting in a well-mixed biochemical system. It is able to detect random fluctuations in the biochemical model. Moreover, it can overcome the problem of low number of molecules. However, the stochastic approach has a significant drawback as it often takes a long runtime for the simulation due to the discrete and individual simulation of each reaction [4], [5].

There are different variations of the stochastic simulation algorithm: the direct method, the first reaction method [4] and the next reaction method [6]. The direct method and first reaction method are technically equivalent variations of SSA, while the next reaction method can be considered as an extension of the first reaction method in more efficient way.

Petri nets [7] can be used as a wrapper tool to construct biological models that can later be executed via SSA. The main merits of Petri nets in this context are the graphical depiction of the interacting biological components. Moreover, there are many extensions of Petri nets (e.g., see [8], [9], [10]) that have been developed over the years permitting to model different semantics of biological applications by the help of Petri nets. In this paper, we are interested in a special class of Petri nets, stochastic Petri nets, which can be used to construct graphical models that can be easily simulated via SSA.

Nevertheless, the simulation of stochastic Petri nets by means of SSA can be in many cases prohibitively slow. The mean reason for this limitation is the way in which the SSA executes biological models where each reaction is fired individually. Therefore, the time complexity of the SSA is proportional to the number of reaction firings and not to the the number of reactions or species as the deterministic approach does. For instance, consider a simple model which consists of a very few number of reactions (3-5 reaction), but the number of molecules of the reaction species are abundant. In this case,

there will be a huge number of reaction firing, which will subsequently slow down the simulation.

There are in fact many solutions to this problem which have been proposed in the literature (e.g., [11], [12]), but these solutions are still limited by the capability of the original SSA. In this paper, we propose distributing the simulation of the SSA algorithm on different machines. The message passing interface approach (MPI) can be very helpful in this context to speedup the simulation.

Unlike other approaches in the literature which focus on parallelising the different runs of the stochastic simulation, we focus in this paper on parallelising the individual runs of the SSA. This would be useful in simulation cases where the computational bottleneck is related to simulating a single trajectory. Our approach first partitions the system reactions *among all processes* to make each process responsible for a certain group of reactions. Afterwords, all processes take part in selecting the next reaction to fire. The selection process is performed in a distributed manner. The simulation is advanced by firing the selected reaction and asking the other processes to update their states.

This paper is organised as follows: we first provide a brief overview of (coloured) stochastic Petri nets as well as the theoretical background of the SSA algorithm. Next we discuss our idea of simulating stochastic Petri nets in a distributed manner. We evaluate the performance of our proposed algorithm by applying it to a case study from the literature. Finally, we conclude by closing remarks and outlining future work.

## II. Background

This section introduces stochastic and coloured stochastic Petri nets. It also outlines the main steps of the stochastic simulation algorithm.

### A. Stochastic Petri Nets

Petri nets are an efficient mathematical tool that is used to represent and analyse the behaviour of many technical and biological systems. Timed Petri nets are one of the Petri net extensions in which the model is parametrised with time. The time is usually assigned to transitions, although it can also be assigned to places. Stochastic Petri nets are considered as a special type of timed Petri nets in which time is an exponentially distributed random variable [8]. In stochastic Petri nets, the time is updated in a discrete manner and the number of tokens is a discrete quantity [13]. In the biological context, stochastic Petri nets can be employed to offer a graphical representation to reaction networks representing e.g., biological pathways. In this case, SSA might be employed to execute the network semantics.

### B. Coloured Stochastic Petri Nets

Coloured Petri nets [14] have been widely used in modelling and simulating biological systems. The main idea of using Coloured Petri nets is compacting the model size and simplifying the modelling process. The tokens of Coloured Petri nets' places may have colours. They may also have different attributes and different values [13]. Coloured Petri nets reduce the usual size of the model by representing many similar components as one component which leads to less number of places and transitions in the model [8]. Moreover, they can be unfolded into equivalent low-level Petri nets. Furthermore, they can be simulated by one of the stochastic algorithms.

### C. The Stochastic Simulation Algorithm

The stochastic simulation approach has been widely used due to its accuracy. It was introduced originally by Gillespie [4]. It can provide accurate results for models that contain few number of molecules [9]. SSA has many different forms: direct method, first reaction method and next reaction method. Moreover, SSA can generate exact trajectories comparable to those produced by the Chemical Master Equation CME.

More specifically, SSA consists of a few steps. At each step, SSA decides the firing time as well as the index of the next reaction [4]. For example, in the first reaction method, a variant of the SSA algorithm, the time of the next reaction is determined by:

$$\tau_j\prime = -\frac{1}{a_j\prime(X)} \ln r_j\prime, \tag{1}$$

where $a_j\prime(X)$ is the propensity of reaction $R_j$ at the system state X and $r_j$ is a random number generated from the uniform distribution (0,1). According to the first reaction method the next reaction to fire is the one with the minimum $\tau_j\prime$ . After the firing of the selected reaction, the propensities of all affected reactions by such a firing (including the fired reaction) is updated.

The major drawback of the SSA algorithm (and its variant) is the huge processing cycles required to selecting a reaction to fire as well as updating reaction propensities after the firing. In what follow, we present an efficient parallel algorithm to improve the performance of the sequential SSA.

## III. Method

### A. Parallel Stochastic Simulation Algorithm

In this section we introduce the proposed parallel algorithm to speed up the execution of the SSA. Algorithm 2 provides a summary of the parallel simulation procedure while Figure 1 presents a schematic digram of a high level architecture of the proposed idea. The algorithm is designed to operate on a distribute memory system where each processor has its own separate memory and communicate with each other via messages in a master/worker architecture. That is process 0 is assigned to be the master and the rest of processes are treated as workers. In what follows, we discuss the the parallelisation of the first reaction method in more details, but we discuss transition partitioning among available processes first.

**Transition Partitioning**: As a pre-step of any parallel algorithm, the work should be allocated for each working node. This procedure is usually called partitioning. For the purpose of our problem, the partitioning process tries to divide the set of transitions (reactions) equally between the available

**Algorithm 1** Partitioning reactions among a set of processes

let $p_{id}$ is the rank of the current process;
let $R$ denotes the set of reactions;
let $P$ denotes the set of processes;
**if** $|R| < |P|$ **then**
  Assign all reactions to process zero
  return;
**end if**
**if** $|R| \bmod |P| = 0$ **then**
  $localsize = \frac{|R|}{|P|}$
**else**
  **if** $p_{id} \leq (|R| \bmod |P|)$ **then**
    $localsize = \frac{|R|}{|P|} + 1$
  **else**
    $localsize = \frac{|R|}{|P|}$
  **end if**
**end if**

processors. In what follow we use the term reaction also to mean a Petri net transition.

Algorithm 1 provides a summary of the partitioning procedure of the set of reactions. The algorithm takes as input the set of reactions $R$ as well as the set of available processes, $P$. The set of reactions can be smaller than the number of processes. In this case the algorithm assigns all reactions to the master process resulting in a simulation algorithm equivalent to the sequential one.

Otherwise, there will be two cases: either the number of reactions can be equally divided between the number of processes, or the number of processes cannot be equally divided. In the former case, each process will take $\frac{|R|}{|P|}$ reactions, while on the latter case the first $|R| \bmod |P|$ processes will take $\frac{|R|}{|P|} + 1$ reactions while the remaining processes will take $\frac{|R|}{|P|}$. Please notes, that when a reaction belongs to a process, all substrates and products of this reaction will also belong to the same process. Therefore, if there will be an overlapping between two processes, the overlap reactions as well as their species will need to be updated after each reaction firing in a process.

**Simulation Initialisation**: The initialisation procedure includes the following steps: initialising the system conditions with the system state $X(T_0) = x_0$, i.e. the initial marking which is the number of each molecules of species in the model, initialise the simulation time to 0 (i..e., $T = 0$) and initialise the ranks (i.e., the index) of the processes. Then partition the reactions of the model among processes where each processor is responsible for group of reactions (i.e., call Algorithm 1).

**Choosing the next reaction**: for each reaction of the system we calculate the expected firing time according to Equation (1). Each processor calculates the minimum reaction firing time between the reactions in its range. Therefore we obtain number of minimum values equivalent to the number of processes (local minimum firing times). Afterwards, a reduction process is applied to all the resulted values in order to get the minimum run time value of the overall reactions of the system (global minimum firing time). Finally, the reaction which has the minimum run time value is the next reaction to fire. Therefore, the process which has the minimum $\tau$ will be responsible for the firing and the communication of this information to other processes. After the firing, each process will need to update reaction propensities that have

**Algorithm 2** Parallel Simulation of SSA (First Reaction Method)

**Require:** The set of reactions along their related information

Partition the reaction between processes according to Algorithm 1.
Each process initialises the simulation with the initial system state.
Each process calculates the propensity function of the reactions assigned to it.
**while** $\tau \leq \tau_{end}$ **do**
  All processes draw random numbers from the uniform distribution (0,1) equal to the number of reactions.
  All processes calculate $\tau_j{}'$ according to Equation (1) for each reaction $R_j$.
  Each processor calculates the minimum value $\tau_j{}'$ among all the reactions assigned to it.
  Get the min $\tau_j$ value among all processes.
  The process that has the global min $\tau_j$ fires reaction $j$.
  Communicate this firing with the other processes
  Update reactions propensities of all reactions affected with this firing
**end while**

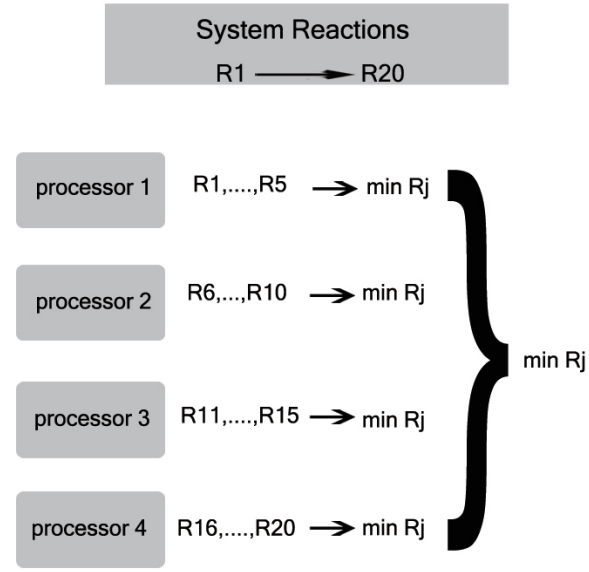



Fig. 1. An illustration of the work of the proposed parallel simulation algorithm.

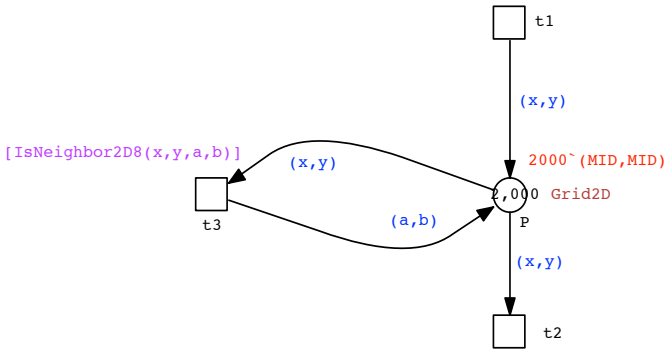



Fig. 2. Modelling the system diffusion dynamics using colour Petri nets in two dimension space

been affected with the reaction execution. For example, if a species is shared among two reactions as a substrate, the firing of one of the two reactions will require the update of the other one. This steps are repeated until the end simulation time is reached.

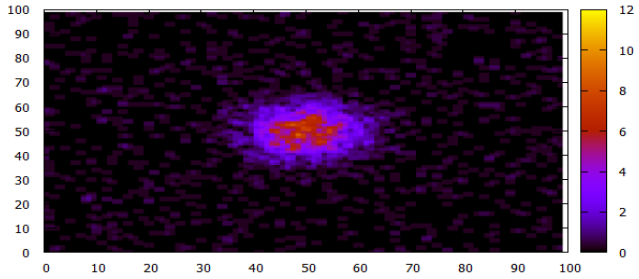
The proposed algorithm requires that the master processor which has the rank 0 is responsible for reading the configuration file as well as the distribution of the net information to worker nodes.

## IV. Performance Evaluation

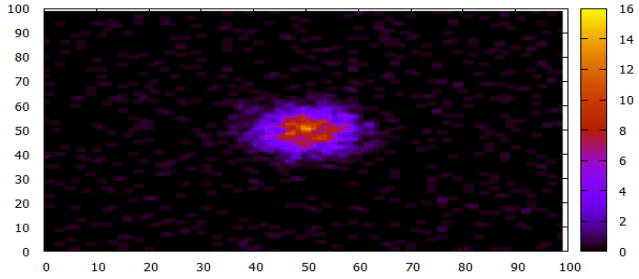
In this section we use a case study to measure the performance of the proposed parallel simulation algorithm for bio-interpreted coloured stochastic Petri nets. We adopt a coloured Petri net model from [15]. The model represents reaction diffusion in two dimensional space. This model is constructed by creating a 2D grid which contains reversible reactions. Initially, only the place in the grid center has initial marking and the other places start with initial marking equal to zero. Later when the model is executed, the marking at the middle of the grid starts to diffuse in the four directions. Figure 2 presents the coloured stochastic diffusion model that will be used in this paper. More details for the model construction as well as the model parameters can be found in [15].

This model is ideal for our purpose, since the net size can be increased/decreased easily by increasing/decreasing a few parameters. Therefor we can design different experiments using the same model. First to make sure the parallel simulation functions correctly, we have executed the model using the serial as well as the parallel simulation. Figure 3 provides the simulation results for the serial implementation (a) and parallel simulation (b). We can notice that the two results are similar except for a minor variation due to randomness in simulating single runs.

Once the obtained result from the parallel simulation is accurate, we can use it to measure the performance of the parallel simulation. We experimented with the parallel SSA algorithm using model size of 5x5, 10x10, 25x25, 50x50 and 100x100. Figure 4 compares the runtime behaviour of these five experiments. We used a PC of 8 GB RAM 8, and core i7 processor.



(a)



(b)

Fig. 3. A snapshot of the simulation result of the coloured model in Figure 2 with a grid size of 100X100 using (a) serial implementation and (b) parallel implementation. The two shots are close to each other but not identical due to the randomness.

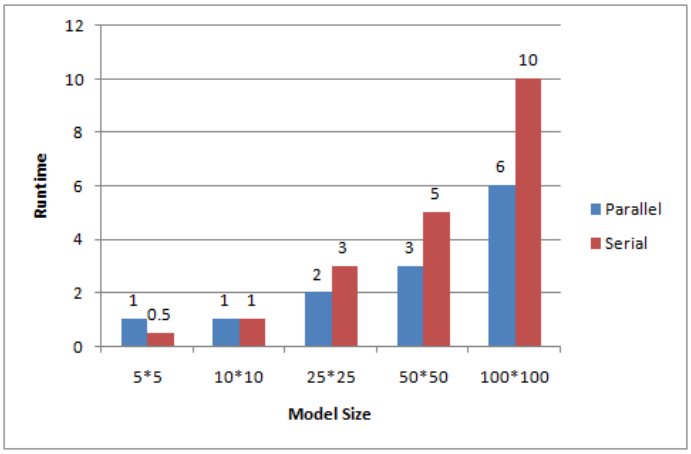



Fig. 4. A relationship between the sequential and the parallel runtime with respect to increasing the system size using 2 processes

According to the results presented in Figure 4, the parallel algorithm takes more times than the serial one for small models. This is not a surprise in fact, since for smaller models the additional overhead due to the communication will be much higher than the saved time due to the algorithm steps. As the model size is increased, the save in terms of runtime is also increased. This can clearly be shown for the model size of 100x100 gird.

### A. Conclusions

In this paper we have proposed a parallelisation approach to enhance the performance of the stochastic simulation and introduced some optimisations to reduce the overload on the working nodes. This algorithm is implemented by standard C++ language and the parallelism is carried out using MPICH as a method to prove our theory.

Although our implementation resulted in an improvement in terms of the runtime, we are not satisfied with the current performance of the current implementation. One direction to improve the performance of the current parallel algorithm is to reduce the random number generation that takes place at each single iteration. Therefore, we are currently experimenting with the next reaction method to reduce the number of times random numbers are generated and therefore improving the runtime performance.

## REFERENCES


[1] H. Mcadams and A. Arkin, "It's a noisy business!" *Trends in Genetics*, vol. 15, no. 2, pp. 65–69, 1999.

[2] Y. Cao, H. Li, and L. Petzold, "Efficient formulation of the stochastic simulation algorithm for chemically reacting systems," *The journal of chemical physics*, vol. 121, no. 9, pp. 4059–4067, 2004.

[3] D. T. Gillespie, "A general method for numerically simulating the stochastic time evolution of coupled chemical reactions," *Journal of computational physics*, vol. 22, no. 4, pp. 403–434, 1976.

[4] ——, "Stochastic simulation of chemical kinetics," *Annu. Rev. Phys. Chem.*, vol. 58, pp. 35–55, 2007.

[5] M. Herajy, "Semantics-based parallelization for the stochastic simulation of complex cell cycle regulations," in *Biomedical Engineering Conference (CIBEC), 2016 8th Cairo International*. IEEE, 2016, pp. 110–113.

[6] M. A. Gibson and J. Bruck, "Efficient exact stochastic simulation of chemical systems with many species and many channels," *The journal of physical chemistry A*, vol. 104, no. 9, pp. 1876–1889, 2000.

[7] T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, Apr. 1989.

[8] R. David and H. Alla, "Discrete, continuous, and hybrid Petri nets," *IEEE Control Systems*, vol. 28, no. 3, pp. 81–84, 2008.

[9] M. Herajy, "Computational steering of multi-scale biochemical networks," Ph.D. dissertation, Cottbus, Brandenburgische Technische Universität Cottbus, Diss., 2013, 2013.

[10] A. A. Tovchigrechko, "Efficient symbolic analysis of bounded Petri nets using interval decision diagrams." Ph.D. dissertation, Brandenburg University of Technology, 2008.

[11] D. D. Jenkins and G. D. Peterson, "Gpu accelerated stochastic simulation," in *Symposium on Application Accelerators in High Performance Computing (SAAHPC)*, 2010.

[12] D. D. Jenkins, "Accelerating the stochastic simulation algorithm using emerging architectures," 2009.

[13] A. Meister, "Timenet-examples of stochastic coloured petri nets."

[14] K. Jensen, "Coloured Petri nets and the invariant-method," *Theoretical Computer Science*, vol. 14, no. 3, pp. 317–336, 1981.

[15] F. Liu, M.-A. Blätke, M. Heiner, and M. Yang, "Modelling and simulating reaction–diffusion systems using coloured Petri nets," *Computers in Biology and Medicine*, vol. 53, pp. 297 – 308, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0010482514001693