UNIVERSITÀ DEGLI STUDI DI MILANO-BICOCCA
Facoltà di Scienze Matematiche Fisiche e Naturali
Dipartimento di Informatica, Sistemistica e Comunicazione
Dottorato di Ricerca in Informatica, XXIII Ciclo

Membrane Systems and Stochastic Simulation Algorithms for the Modelling of Biological Systems

Ettore Mosca

Supervisors: Prof. Giancarlo Mauri, Dr. Luciano Milanesi
Tutor: Prof. Lucia Pomello
PhD Coordinator: Prof. Stefania Bandini

ANNO ACCADEMICO 2009–2010

To the Universe,
to its laws and wonders,
and to the most precious wonder: life.

Abstract

Membrane Computing is a branch of computer science that was born after the introduction of *Membrane Systems* (or *P systems*) by a seminal paper by Gh. Păun. Membrane systems are computing devices inspired by the structure and functioning of living cells as well as from the way the cells are organized in tissues and higher order structures. The aim of membrane computing is to abstract computing ideas and models imitating these products of natural evolution. A typical membrane system is composed by a number of regions surrounded by membranes; regions contains multisets of objects (molecules) and rules (cellular processes) that specify how objects must be re-written and moved among regions.

In spite of the fact that the initial primary goal of membrane systems concerned computability theory, the properties of membrane systems such as compartmentalisation, modularity, scalability/extensibility, understandability, programmability and discreteness promoted their use for an important task of the current scientific research: the modelling of biological systems (the topic "systems biology, including modelling of complex systems," has now appeared explicitly in the Seventh Framework Programme of the European Community for research, technological development and demonstration activities). To accomplish this task some features of membrane systems (such as nondeterminism and maximal parallelism) have to be mitigated while other properties have to be considered (e.g. description of the time evolution of the modelled system) to ensure the accurateness of the results gained with the models.

Many approaches for the modelling and simulation of biological systems exist and can be classified according to features such as continuous/discrete, deterministic/stochastic, macroscopic/mesoscopic/microscopic, predictive/explorative, quantitative/qualitative and so on. Recently, *stochastic methods* have gained more attention since many biological processes, such

as gene transcription and translation into proteins, are controlled by noisy mechanisms. Considering the branch of modelling focused at the molecular level and dealing with systems of biochemical processes (e.g. a signalling or metabolic pathway inside a living cell), an important class of stochastic simulation methods is the one inspired by the Gillespie's stochastic simulation algorithm (SSA). This method provides exact numerical realisations of the stochastic process defined by the chemical master equation. A series of methods (e.g. next reaction method, tau leaping, next subvolume method) and software (StochKit and MesoRD), belonging to this class, were developed for the modelling and simulation of homogeneous and/or reaction-diffusion (mesoscopic) systems.

A stochastic approach that couples the expressive power of a membrane system (and more precisely of *Dynamical Probabilistic P systems* or DPPs) with a modified version of the *tau* leaping method in order to quantitatively describe the evolution of multi-compartmental systems in time is the *τ-DPP* approach.

Both current membrane systems variants and stochastic methods inspired by the SSA lack the consideration of some properties of living cells, such as the molecular crowding or the presence of membrane potential differences. Thus, the current versions of these formalisms and computational methods do not allow to model and simulate all those biological processes where these features play an essential role.

A common task in the field of stochastic simulations (mainly based on numerical rather than analytical solutions) is the repetition of a large number of simulations. This activity is required, for example, to characterise the dynamics of the modelled system and by some parameter estimation or sensitivity analysis algorithms.

In this thesis we extend the τ-DPP approach taking into account additional properties of living cells in order to expand τ-DPPs modelling (and simulation) capabilities to a broader set of scenarios. Within this scope, we also exploit the main European grid computing platform as a computational platform usable to compute stochastic simulations, developing a framework specific to this purpose, able to manage a large number of simulations of stochastic models.

In our formalism, we considered the explicit modelling of both the objects' (or molecules) and membranes' (or compartments) volume occupation, as mandated by the mutual impenetrability of molecules. As a consequence, the dynamics of the system are affected by the availability of free space. In living cells, for example, molecular crowding has important effects such as anomalous diffusion, variation of reaction rates and spatial segregation, which have significant consequences on the dynamics of cellular processes.

At a theoretical level, we demonstrated that the explicit consideration of the volume occupation of objects and membranes (and their consequences on the system's evolution) does not reduce the computational universality

of membrane systems. We achieved this aim showing that is it possible to simulate a deterministic Turing machine and that the volume required by the membrane systems that carry out this task is a linear function of the space required by the Turing machine.

After this, we presented a novel version of both membrane systems (designated as Sτ-DPPs) and stochastic simulation algorithm (Sτ-DPP algorithm) considering the property of mutual impenetrability of molecules. In addition, we made the communication of objects independent from the system's structure in order to obtain a strong expressive power. After showing that the Sτ-DPP algorithm can accurately reproduce particle diffusion (in a comparison with the heat equation), we presented two test cases to illustrate that Sτ-DPPs can effectively capture some effects of crowding, namely the reduction of particle diffusion rate and the increase of reaction rate, considering a bidimensional discrete space domain. We presented also a test case to illustrate that the strong expressive power of Sτ-DPPs allows the modelling and simulation (by means of the Sτ-DPP algorithm) of processes taking place in structured environments; more precisely, we modelled and simulate the diffusion of molecules enhanced by the presence of a structure resembling the role of a microtubule (a sort of "railway" for intracellular trafficking) in living cells.

Subsequently, we further extended Sτ-DPPs and the respective evolution algorithm to explicitly consider the membrane potential difference and its effect over charged particles and voltage gated channel (VGC, a particular type of membrane protein) state transitions. In fact, the membrane potential difference exhibited by biological membranes plays a crucial role in many cellular processes (e.g. action potential and synaptic signalling cascades). Similarly to what we did for the Sτ-DPPs, we presented the novel version of both the membrane systems (designated as ESτ-DPPs) and the stochastic simulation algorithm (ESτ-DPP algorithm) to capture the additional properties we had considered. In order to describe the probability of charged particle diffusion in a discrete space domain, we defined a propensity function starting from the deterministic and continuous description of charged particle diffusion due to an electric potential gradient. We showed by means of a focused test case that a model for ion diffusion between two regions, in which the number of ions is maintained at two different constant values and where an electric potential difference is available, correctly reaches the expected state as predicted by the Nernst equation. To describe the probability of transition between two VGC states, we derived a propensity function taking into consideration the Boltzmann-Maxwell distribution. We considered a model describing the state transitions of a VGC and we showed that the model predictions are in close agreement with the experimental data collected from literature.

Lastly, we presented the framework to manage a large number of stochastic simulations on a grid computing platform. While creating this framework,

we considered the parameter sweep application (PSA) approach, in which an application is run a large number of times with different parameter values. We ran a set of PSAs concerning the simulations of a stochastic bacterial chemotaxis model and the computation of the difference between the dynamics of one of its components (as a consequence of model parameter variation) compared to a reference dynamics of the same component. We then used this set of PSAs to evaluate the performance of the EGEE project's grid infrastructure (Enabling Grid for the E-sciencE). On the one hand, the EGEE grid proved to be a useful solution for the distribution of PSAs concerning the stochastic simulations of biochemical systems. The platform demonstrated its efficiency in the context of our middle-size test, and considering that the more intensive the computation, the more scalable the infrastructure, grid computing can be a suitable technology for large scale biological models analysis. On the other hand, the use of a distributed file system, the granularity of the jobs and the heterogeneity of the resources can present issues.

In conclusion, in this thesis we extended previous membrane systems variants and stochastic simulation methods for the analysis of biological systems, and exploited grid computing for large scale stochastic simulations. Sτ-DPPs and ESτ-DPPs (and their respective algorithms to calculate the temporal evolution) increase the set of biological systems that can be investigated *in silico* in the context of the stochastic methods inspired by the SSA. In fact, compared to its precursor approach (τ-DPPs), Sτ-DPPs allow the stochastic and discrete analysis of crowded systems, structured geometries, while ESτ-DPPs also take into account some electric properties (membrane electric potential and its consequences), enabling, for example, the modelling of cellular signalling systems influenced by the membrane potential. In future, we plan to improve both the formalisations and the algorithms that we presented in this thesis. For example, Sτ-DPPs can not model and simulate objects bigger than a single compartment, which conversely can be convenient for the analysis of big crowding agents in a tightly discretised space domain; instead, ESτ-DPPs are, for instance, currently limited to the modelling of systems composed by two compartments separated by a boundary that can be assumed to act as a capacitor (e.g biological membranes). Moreover, we plan to optimize the parallel (MPI) implementation of both the Sτ-DPP and ESτ-DPP algorithms, which are presently based on a one-to-one relationship between processes and compartments, a limiting factor for the simulation of discrete spaces composed by a high number of compartments. Lastly, as grid computing demonstrated to be a useful approach to handle a large number of simulations, we plan to develop a solution to handle the simulations required in the context of sensitivity analysis.

Acknowledgements

I would like to thank Professor Giancarlo Mauri who has been my scientific supervisor for this thesis work and hosted me in his bioinformatics group. I also thank Professor Lucia Pomello, for her availability and helpful suggestions.

A special thanks to Dr. Luciano Milanesi, for his scientific and financial support during these three years. I'm also grateful to Dr. Luciano Milanesi and Dr. Ileana Zucchi for allowing me to follow this PhD program in parallel with my activity as a research assistant at ITB-CNR.

I would also like to thank sincerely Paolo Cazzaniga and Dario Pescini, for their helpfulness, precious suggestions and also friendship.

And I'm grateful to colleagues of mine at ITB-CNR, for the many nice hours we spend together every day, and also for their scientific help.

A big thanks to some friends of mine, especially Giacomo (il socio) and Francesco (o' Professo'), for their presence.

Many thanks to all the people who directly or indirectly helped me in this work. A particular thanks to Alessandra, for her work and her precious teaching.

Last but not least, I thank my family for the support during these years, and a special thank to my sister Roberta.

Contents

Introduction

Motivation and aims

Membrane Computing is an area of computer science that was initiated after the introduction of *Membrane Systems* (or *P systems*) with a seminal paper by Gh. Păun [87]. Membrane systems are computing devices inspired to the structure and functioning of living cells as well as from the way the cells are organized in tissues and high order structures, and the aim of membrane computing is to abstract computing ideas and models imitating these products of natural evolution. Hence, membrane computing is part of the broader field of *natural computing*, like genetic algorithms, neural networks and DNA computing. Despite the initial primary goal of membrane systems was devoted to computability theory, the membrane systems domain started to be useful for biological and medical applications [90]. In this context, membrane computing is used to construct models of biological systems and to generate data of interest for the study of the process considered.

The modelling of biological systems is addressed in the multidisciplinary field named *systems biology*. Systems biology concerns the study of biological *systems* adopting a system-level approach [64], and can be rooted in two distinct lines of inquiry in molecular biology: on one hand the formal analysis of molecule systems and on the other hand the evolution of molecular biology [104]. Automated DNA sequencers made possible the genomic revolution (a massive explosion in the amount of biological information concerning genomes) and, subsequently, other types of high-throughput technologies (e.g. microarray platforms, methods for protein-protein interactions) led to the generation of additional "omics"[1] data types. Nowadays, all these technologies allow to collect (approximated) snapshots of the cell state, providing list of parts and interactions among parts. While a list of parts of a (complex) system does not tell how the system actually works (and this can

[1]The term "omics" (usually a suffix) refers to a large scale study of an aspect (specified as a prefix) of living cells: genomics concerns genes, proteomics focuses on proteins, metabolomics studies the metabolites and so forth.

be true even for simple systems), the list of interactions allows, at least, to understand the system structure. However, to gain a deeper understanding of biological systems also their dynamical properties have to be considered. In fact, systems biology approaches are comprehensive quantitative analysis of the manner in which all the components of a biological systems interact functionally over time [1].

In this context, the development of models to represent a given biological system and carry out computer simulations to uncover its (dynamical and structural) properties [63] is a crucial task (the multidisciplinary topic "systems biology, including modelling of complex systems," has now appeared explicitly in the Seventh Framework Programme of the European Community for research, technological development and demonstration activities).In fact biological systems are complex systems [95], and show emergent properties [34] which are hardly understandable without relying on models. Models are essential since allow to investigate the described system in normal and perturbed conditions and by means of *in silico* experiments that lead to useful predictions. Importantly, these investigations if on the one hand reduce expensive and long *in vivo* and/or *in vitro* studies, on the other hand suggest new experiments according to the predictions collected. Clearly, a deeper understanding of biological systems is the fundamental basis for the development of better treatments for diseases.

Considering cellular processes, several modelling approaches have been proposed based on different formalisms. All these approaches can be classified considering the choices facing the experimenter when deciding which strategy or strategies may be most appropriate for a given problem [63]. For example, ordinary differential equations (ODEs) are a continuous and deterministic approach. According to this formalism a pathway of biochemical processes is represented describing the rate of variation of each system variable (the concentration of a molecular species) with an ODE. This approach should be used whenever the well-stirred assumption is reasonable (the compartment in which the processes take place is homogeneous with respect to the molecular species concentration) and the molecular species concentrations are sufficiently high [31].

Recently, the role of noise has been underlined in many biological processes, such as transcription and translation [76, 41], E. coli response to phage λ infectious [7] and nervous system processes [43]. Noise plays a major role when the molecular quantities involved is small [78]. Conversely, when the number of stochastic elements in a system is sufficiently high the randomness is eliminated. However, this assumption may require a reassessment, as small biochemical and electrochemical fluctuations can significantly affect a whole cell response, for example in presence of regulatory mechanisms that are characterized by high gain amplification and positive feedbacks [43]. Two different kind of noise can be identified in biological systems: extrinsic, related to experimental conditions, and intrinsic, due to stochastic events

occurring inside the system itself [41, 93].

As already said, membrane systems have recently found application to the formal description and modelling of biological phenomena. However, some of membrane systems properties (such as nondeterminism and maximal parallelism) have to be mitigated, while a physically correct procedure to calculate the time evolution of the system has to be used. A membrane systems variant that adopts this strategy is called τ-*DPPs* [27], where *dynamical probabilistic p systems* have been coupled with a modified version of the *tau leaping* stochastic simulation method to obtain a quantitative time streamline. This approach has a number of interesting features. As it is stochastic and discrete, it can deal explicitly with noise. Moreover, it is multicompartmental, and thus can be used to study spatially extended systems. The algorithm for the temporal evolution is inspired to one of the most important methods for stochastic simulations, the Gillespie stochastic simulation algorithm (SSA) [53], and, more precisely, to a more efficient implementation, the tau leaping [22]. Therefore, it takes into account the discreteness of the quantity of the molecular species and the inherently random character of the phenomena, is in agreement with theories of thermodynamics and stochastic processes and, lastly, is appropriate for the description of systems characterised by instability [101]. τ-DPPs have been used to study a signal transduction pathway in yeast [28] and other biological processes [25].

As we noticed that nor current membrane systems neither current stochastic simulation methods inspired by the SSA provide an explicit consideration of objects (molecular species) volume and compartments volume we started to work on this topic, Figure 1. The explicit representation of volumes enable the analysis of crowded systems and structured spaces. One of these is the cytoplasm of living cells, a "crowded world" [91], where the volume excluded by macromolecules and other entities such as organelles leads to important effects such as anomalous diffusion, variation of reaction rates and spatial segregation, which have significant consequences on the dynamics of cellular processes [105, 40]. Partly due to crowding and partly an intrinsic properties of living cell structure, the intracellular environment is a structured space. In this environment, cell components such as microtubules (a sort of intracellular "railway") control the trafficking of objects as macromolecules and vesicles.

Another important feature of living cells, but not currently considered by membrane systems and stochastic methods inspired by the SSA, is the membrane electric potential (an electric potential difference between two compartments separated by a cellular membrane, such the extracellular and intracellular environments or, in mithocondria, the intermembrane space and the matrix). There are many examples of biological processes in which living cells take advantage from the presence of a membrane potential [94]. Considering nervous systems, one only needs to think at the action potential and its relation to pre- and post-synaptic signalling cascades: in response to

membrane potential variations, neurotransmitters or sensory stimuli (temperature, mechanical) voltage gated channels (VGCs, membrane proteins that are sensitive to membrane potential difference variation) establish ion fluxes leading to action potentials or the activation of signalling cascades, e.g. in case of the calcium ion (one of the most important second messengers[2]) [12]. Another example is the mitochondrial membrane potential, the alteration of which can be related to cell growth, cell differentiation and cell motility [30], stress [103] and ageing [97].

Another issue related to the modelling of biological systems is the need of repeating a potentially large number of computer simulations. For example, it is important to collect data about the model dynamics according to different parametrisations, and this can be a task included by parameter estimation algorithms and sensitivity analysis methods. A possible solution to handle a large number of independent simulations is the exploitation of *grid computing*, a kind of high-throughput computing, where a combination of computer resources are used to reach a common goal. The difference between the Grid computing and the classic high performance computing, such as a cluster of processors, is that the computational resources shared in a grid tend to be loosely coupled, heterogeneous and geographically dispersed. The term was firstly introduced in the mid '90 to denote a distributed computing infrastructure for advanced science and engineering [44]. The advantage of using a grid approach for large computational challenges relies on the high-end scalability of this technology. As the communication among independent grid jobs is a factor that decreases the grid performance, data parallel applications are the best candidate for grid computing. These applications split the computation of the input data in a series of independent processes and collect the results at the end of the computation.

In this thesis we take into consideration some possible extensions of the current formalisms and computational methods for the modelling and simulation of biological systems, in order to capture a more comprehensive set of biological systems properties. More precisely, we consider τ-DPPs as a starting point for our aim and we study how to extend such approach in order to enable the analysis of crowded systems, structured geometries and membrane potential effects. Moreover, we study a solution to manage a large number of simulations and we consider EGEE project's grid infrastructure (Enabling Grid for the E-sciencE) as it is the main European grid platform.

Overview

The thesis is structured as follows. In the first part (Chapters 1-3), we introduce the preliminary notions that constitute the background of the

[2]Second messengers are components of signal transduction cascades; in this context, they greatly amplify the strength of the signal.

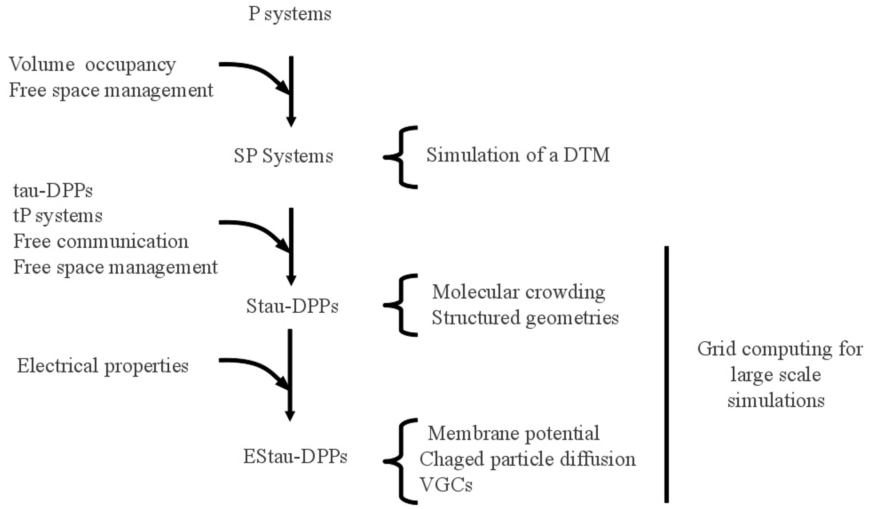


Figure 1: Graphical overview of the work presented in this thesis. tP systems: tissue P systems; DTM: deterministic Turing machine. See the text for the other acronyms.

work. In the second part (Chapters 4-6), we present the results of the work, Figure 1.

Chapter 1 is an introduction to the membrane computing field. We describe the basic version of membrane systems (that was designated as transition P systems), the large panoply of variants and the universality. Chapter 2 deals with the stochastic modelling of biological systems. We present the stochastic simulation algorithm by Gillespie and then its modifications to decrease the computational cost. Then, we consider the approaches for the simulation of spatially extended systems. Lastly, we present τ-DPPs, the main basis of the work of this thesis. In Chapter 3 we introduce the notion of parameter sweep applications (PSAs) and show how to deal with PSAs using grid computing, and more precisely EGEE project grid.

In Chapter 4 we define an extension of τ-DPPs, that we designate as Sτ-DPPs, in which we consider both molecules and compartments that occupy a finite amount of volume and, moreover, we decouple the communication possibilities inside the systems from the system's structure, in order to obtain a strong expressive power. At a theoretical level, we study the consequences of objects' and membranes' volume occupation on the computational universality. We present the modified version of the parallel (MPI) algorithm for the simulation of Sτ-DPPs and then we illustrate with some test cases that Sτ-DPPs can be used to incorporate some effects of molecular crowding and structured spaces on diffusion and reaction rates.

In Chapter 5 we further extend Sτ-DPPs in order to obtain ESτ-DPPs, where we consider some electrical properties. In particular, we include in the formalism object charges and membrane electric potential. We derive

two novel classes of propensity functions to compute the probability that a charged particle will diffuse to a neighbouring region under the force due to an electric potential gradient (class II) and that a voltage gated channel will change its current state as a consequence of an electric potential difference (class III). We present the modified version of the parallel (MPI) algorithm to compute the temporal evolution of ESτ-DPPs, and we illustrate with two test cases the correct functioning of the ESτ-DPP algorithm.

In Chapter 6 we present a framework for the management of PSAs using the EGEE project grid (Enabling grid for E-sciencE). We describe a set of PSAs distributed over the EGEE project grid infrastructure, in which we compare the dynamics of a bacterial chemotaxis model under different parametrisation with a reference dynamics. Moreover, we carry out the grid infrastructure performance analysis highlighting critical factors.

Lastly, in Chapter 7 a discussion about the presented work is proposed. Insights concerning some possible improvements and future directions for research are also briefly described.

Published works

The work presented in this thesis is partly described in the following publications:

- E. Mosca, I. Merelli, P. Cazzaniga, D. Pescini, G. Mauri, L. Milanesi, (2010) Grid computing for parameter sweep applications in systems biology models, *International Journal of High Performance Computing Applications* (submitted)

- E. Mosca, P. Cazzaniga, D. Pescini, G. Mauri, L. Milanesi, (2010) Modelling Spatial Heterogeneity and Macromolecular Crowding with Membrane Systems, *Proceedings of the Eleventh International Conference on Membrane Computing (CMC11)*, M. Gheorghe, T. Hinze, G. Păun, pp305-325. To appear in Lecture Notes in Computer Science.

- P. Cazzaniga, G. Mauri, L. Milanesi, E. Mosca, D. Pescini, (2010), A Novel Variant of P Systems for the Modelling and Simulation of Biochemical Systems, *Membrane Computing, Lecture Notes in Computer Science*, 5957, pp210-226

- E. Mosca, L. Milanesi, (2010), Modelling Biochemical Pathways, *Mathematical Approaches to Polymer Sequence Analysis and Related Problems*, Springer.

- E. Mosca, P. Cazzaniga, I. Merelli, D. Pescini, G. Mauri, L. Milanesi, (2009), Stochastic Simulations on a Grid Framework for Parameter

Sweep Applications in Biological Models, *High Performance Computational Systems Biology, International Workshop on, IEEE Computer Society*, pp. 33-42

- E. Mosca, P. Cazzaniga, D. Pescini, G. Mauri, L. Milanesi, (2008), A stochastic, discrete model for the simulation of the action potential, *SysBioHealth Symposium*, 24-25 November, Bologna, Italy. Oral presentation and Young Investigator Award for the best presentation.

Part I

Prerequisites

1

Membrane computing

Membrane Computing is an area of computer science that was initiated after the introduction of *Membrane Systems* (or *P systems*) with a seminal paper by Gh. Păun [87]. Membrane systems are computing devices inspired to the structure and functioning of living cells as well as from the way the cells are organized in tissues and high order structures, and the aim of membrane computing is to abstract computing ideas and models imitating these products of natural evolution. Hence, membrane computing is part of the broader field of *natural computing*, like genetic algorithms, neural networks and DNA computing.

Shortly, a membrane systems is composed by a membrane structure that defines a series of regions where objects evolve according to given rules. The system evolves by applying the rules in a nondeterministic and maximally parallel manner until any rule can be applied. The result of a computation is composed by the objects placed in a specific region or expelled from the membrane structure. Nowadays, many variants of membrane systems exist, which differ due to properties associated to the various elements of the basic definition.

One of the fundamental ingredients of P systems is the membrane structure, Figure 1.1(a). It is a set of hierarchically arranged membranes contained in a external membrane (or skin membrane). Each membrane is identified with a label and defines a region where objects and one or more membranes can be placed. The whole membrane structure is defined by means of a *rooted tree*, where the root is the skin membrane while the leaves are the elementary membranes Figure 1.1(b). Symbolically, the membrane structure is defined using a string with matching parentheses; considering the membrane systems represented in Figure 1.1 we have:

$$[\,[\,]_2\,[\,]_3\,[\,[\,]_5\,[\,]_6\,]_4\,]_1.$$

Each region (or equivalently membrane, as there is a one-to-one corre-

spondence between membranes and regions) contains *multisets* of objects. A multiset is a generalisation of a set, such that a multiset can contain multiple copies of the same element. A multiset can be represented in many ways and one of these is a string of symbols. For example, the string $w = a^2bc^3$ defines a multiset that contains 2, 1, 3 copies of the objects a, b, c, respectively (all the permutations of w defines the same multiset). The number of times an object appears in a multiset is the *multiplicity* of the object in that multiset.

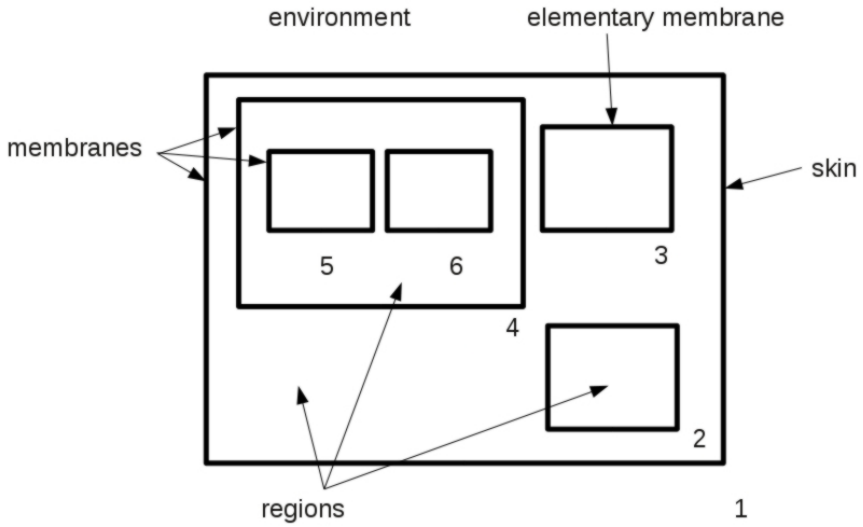
Moreover each region contains a finite set of *evolution rules*. Given an alphabet of objects O, the general form of a rule is $u \to v$, where[1] $u \in O^+$ while $v = v'$ or $v = v'\delta$ where $v \in (O \times \{\text{here}, \text{in}_j, \text{out}\})$ where j runs over the number of membranes; "here" indicates that the multiset v should be placed in the same membrane in which the rule is occurring, "in_j" is used to move the multiset to the membrane labelled j and included in the current membrane, "out" indicates that the multiset must be sent out from the current membrane, and lastly δ specifies the dissolution of the corresponding membrane; whenever a rule of this type occurs, the corresponding membrane and its set of rules disappear (the dissolution of the skin membrane is prohibited), while the objects it contains are assigned to the surrounding membrane. The evolution rules are associated to partial order relations (e.g. $r_1 > r_2$) that define rules priority during rule application. For instance, the rule r_1 belonging to the membrane j,

$$r_1 : a^2bc \to (a, \text{here})(a, \text{in}_2)(c, \text{out})$$

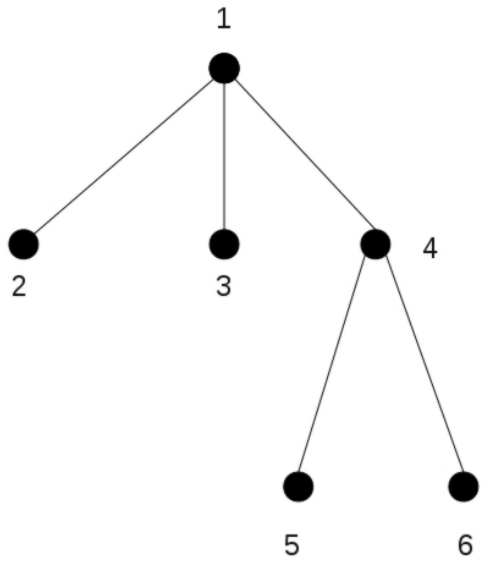
acts on the multiset a^2bc and sends a copy of a in the nested membrane 2 and a copy of c in the membrane surrounding membrane j.

Given a membrane structure, multisets and rules, the system evolves by applying the rules *in the maximally parallel manner, nondeterminstically choosing the rules and the objects*. In other words, all objects to which a rule can be applied must be subject to a rule application. A rule is applicable if there are copies of the objects specified in its left side and if there is not a rule with higher priority in the same membrane that can be applied during the current transition, irrespective of which objects it involves (strong interpretation). In case more than one different rules having the same priority can be applied, the rule to be applied is nondeterministically chosen among them. Rule application takes place in parallel, synchronously in all the membranes: a universal clock is assumed to exist.

[1] By Σ^* we indicate the set of all strings over an alphabet Σ and by Σ^+ we denote the set $\Sigma^* - \lambda$, of all non-empty strings over Σ.



(a)



(b)

Figure 1.1: An example of membrane system with 6 membranes. a) Graphical representation; b) membranes structure defined as a rooted tree.

1.1 Transition P systems

In this Section we describe formally a basic type of membrane systems. In particular we report the definition given in [87], the article in which membrane systems were introduced as *transition P systems*.

Definition 1.1.1. *A transition P system of degree n is the tuple:*

$$\Pi = (V, \mu, w_1, w_2, \dots w_n, (R_1, \rho_1), (R_2, \rho_2), \dots, (R_n, \rho_n), i_O) \tag{1.1}$$

where:

- *V is an alphabet and its elements are called objects;*
- *μ is a membrane structure of degree n, and the regions are labelled in a one-to-one manner with element in a given set Λ*
- *w_i, $i \in \{1, 2, \dots, n\}$, are strings from V^* representing multiset over V associated with the regions $1, 2, \dots, n$ of μ*
- *R_i, $i \in \{1, 2, \dots, n\}$, are finite sets of evolution rules over V associated with the regions $1, 2, \dots, n$ of μ; ρ_i is a partial order relation over R_i, specifying a priority relation among rules of R_i. An evolution rule is a pair (u, v), which is usually written in the form $u \to v$, where u is a string over V and $v = v'$ or $v = v'\delta$, where v' is a string over $(V \times \{here, in_j, out\})$, and δ is a special symbol not in V; the length of u is designated as the radius of the rule $u \to v$*
- *i_0 is a number between 1 and n which specifies the output membrane.*

Definition 1.1.2. *A configuration of a transition P system Π is the tuple*

$$C = (\mu', w'_{i_1}, \dots, w'_{i_k}, (R_{i_1}, \rho_{i_1}), \dots, (R_{i_k}, \rho_{i_k})) \tag{1.2}$$

where μ' is obtained removing from μ all membranes different from $i_1, \dots, i_k$ (of course the skin membrane is not removed), $\{i_1, \dots, i_k\} \subseteq \{1, \dots, n\}$.

Note that the membranes preserve the initial labelling in all subsequent configurations and, in this way, the correspondence between membranes, multisets, and sets of evolution rules is well specified by the subscripts of these elements.

1.2 A large panoply of possible extensions

The versatility and the flexibility of the formalism of membrane computing determined the extension of the variant reported in Definition 1.1.1, and nowadays a high number of P systems variants are available in literature. In the following we mention some of these extensions.

Membrane systems with *symport/antiport* rules consider only communication rules that move objects between membranes, and define a special type of rule (antiport rule) that acts over objects located in two different membranes [86]. Another extension concerns the assignment of electrical polarizations $\{+, -, 0\}$ to objects and/or membranes [88].

An important class of membrane systems is the one of P system with active membranes [89], whose membranes are the main active components, in the sense that they directly mediate the evolution and the communication of objects and, moreover, the membranes can multiply themselves by division. Other P system variants consider further operations over membranes (other than dissolution and division), such as creation, merge, endocytosis/exocytosis[2] [66] and gemmation [17].

Beside cell-like P systems, in which the structure of the membranes is inspired to the one of living cells, tissue P systems were introduced in [75]. Tissue P systems resemble the way in which living cells are organised in a tissue and the membrane structure is defined by a directed graph, where the nodes are the membranes and the objects are communicated along the edges of the graph. Neural-like P systems were introduced in [74], where some ideas from neurobiology are incorporated and subsequently were extended to spiking neural P systems [61]. Moreover, population P systems were introduced in [10] and consider operations such as communication by means of the environment, cell bonding and cell differentiation.

1.3 Universality

According to language and automata theory, a computing model can reach the power of Turing machines as it has two major properties: (i) enough context-sensitivity, in order to send information at any distance in the data structure used, and (ii) erasing capabilities in order to use an arbitrary large workspace [90]. As the biochemical reactions (rules) of the form $u \to v$ (see Def. 1.1.1) ensure context sensitivity and the possibility to throw waste products (objects) in the environment ensure erasing capabilities, living cells seem to satisfy these two requirements.

Consequently, most classes of membrane systems are *computationally universal* and share the same power of Turing machines. Just to mention a couple of cases, this equivalence was demonstrated for multisets re-writing systems with at least two catalists[3] [46] and for symport/antiport P systems

[2]Endocytosis is a process by means of which living cells absorb substances from the extracellular region and is based on the invagination of a membrane leading to a formation of a vesicle. Conversely, esocytosis is the process by means of which the content of a vesicle is expelled and is based on the fusion between the vesicle and the plasma membrane.

[3]A catalyst is a substance that modify (increase or decrease) the rate of a chemical reaction and, conversely from a reactant, is not consumed by the reaction. Hence, in the context of membrane computing, catalysts are objects which are present in the rules but

[4]. The conclusion is that a living cell is a very powerful "computer"[90].

are not modified by the rule application.

2

Stochastic modelling of biological systems

When the intrinsic fluctuations of the chemical system play a major role in the dynamics, as is the case for many systems of interest for biology, a master equation approach is more suitable than the deterministic and continuous approach based on ordinary differential equations [9, 52].

The chemical master equation formulation adopts a mechanistic perspective on the chemical system describing it as a sequence of collision events among molecules. Each of these scattering events can lead either to a new compound (reactive collision) or to an elastic scattering (diffusive collision) which does not alter the chemical species distributions but only the particles speed and direction. Which of the two collisions pathways will be followed by each scattering event is determined by the energy involved in the process: if this energy exceed the Arrhenius threshold (activation energy) then the two molecules will react to form the new compound. The existence of an activation energy imposes that the diffusive events are the most probable ones and, if the environment in which the reactions take place is homogeneous, this picture corresponds to a well stirred reactor and the dynamics can be tracked by means of a stochastic simulation algorithm such as the Gillespie's one [52]. Otherwise, other methods are required to keep track of the position of particles within the system.

In this chapter we review the most important approaches for the modelling of both homogeneous and spatially extended systems, focusing on stochastic approaches; we conclude describing membrane systems as modelling tools and, more precisely, the τ-DPP approach.

2.1 Modelling homogeneous systems

In this Section we consider *well stirred* or spatially homogeneous systems, meaning that the chemical species abundances do not vary with respect to space. This assumption seems to be hardly justified within cells, where there

is a very crowded environment leading to spatial segregation of molecular entities. However, whether the well mixed assumption is a good approximation or not depends on the time scale of the considered biological process. In the case in which the biological process involves a time scale that is greater than the molecules diffusion time scale, the well-stirred approximation is justified. In many cases, such as cell cycle regulation or circadian rhythms, the well-stirred assumption is appropriate and well-stirred models have been successfully used to obtain a deeper understanding of these biological processes [2, 69].

2.1.1 The chemical master equation and the stochastic simulation algorithm

Let us consider a well-stirred system of chemical species $S = \{s_1, \ldots, s_n\}$ interacting by means of chemical reactions $R = \{r_1, \ldots, r_m\}$. According to the classic chemical notation, the general form to describe a chemical reaction is

$$\alpha_1 s_1 + \ldots + \alpha_n s_n \rightarrow \beta_1 s_1 + \ldots + \beta_n s_n \tag{2.1}$$

where the natural numbers $\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_n$ are the stoichiometric coefficients and define the amount of molecules that are involved in the process. Moreover let us consider that system's volume and temperature constant.

Each chemical reaction is characterized by two quantities. The first is the state change vector $\mathbf{v} = (v_{1,j}, \ldots, v_{n,j})^{\mathrm{T}}$, where $v_{i,j}$ is the change (in terms of molecule numbers) in the s_i population due to the reaction r_j; the state change vectors form the stoichiometric matrix: $\mathbf{N} = [\mathbf{v}_1, \ldots, \mathbf{v}_m]$.

The second is the *propensity function* $a(\mathbf{x})$, that is a function such that $a(\mathbf{x})dt$ gives the probability that one reaction of the type r_j occurs in the next infinitesimal time interval $[t, t + \mathrm{d}t]$, given the state $\mathbf{x} = (x_1, \ldots, x_n)^{\mathrm{T}}$, where x_i is the number of molecules of species s_i. The definition of the propensity function is derived by considering the *fundamental hypothesis* of the stochastic formulation of chemical kinetics [52], i.e. the existence of a constant c_j such that the product $c_j \mathrm{d}t$ gives the average probability that a molecule (for uni-molecular reactions) or a randomly chosen combination of molecules (for reaction with more than one reactant) will react in the next infinitesimal time interval $\mathrm{d}t$. According to this argument it is possible to obtain $a_j(\mathbf{x})$ by multiplying c_j with the number of reactants or combination of reactants h_j:

$$a_j = c_j \cdot h_j \tag{2.2}$$

where h_j will be[1]

$$h_j = \begin{cases} x_1 & \text{if } r_j : s_1 \to \dots \\ \frac{1}{2}x_1(x_1 - 1) & \text{if } r_j : s_1 + s_1 \to \dots \\ x_1 x_2 & \text{if } r_j : s_1 + s_2 \to \dots \end{cases} \tag{2.3}$$

where x_1 and x_2 are, respectively, the number of molecules of species s_1 and s_2.

Using the laws of probability [55], it is possible to deduce a time equation for describing the time evolution of the system's state (given a particular initial condition x_0) :

$$\frac{\mathrm{d}p(\mathbf{x}, t|\mathbf{x}_0, t_0)}{\mathrm{d}t} = \sum_{j=1}^{m} [a_j(\mathbf{x} - \mathbf{v}_j)p(\mathbf{x} - \mathbf{v}_j, t|\mathbf{x}_0, t_0) - a_j(\mathbf{x})p(\mathbf{x}, t|\mathbf{x}_0, t_0)]$$

where $p(\mathbf{x}, t|\mathbf{x}_0, t_0)$ is the conditional probability of the system to be at state $\mathbf{x}$, time t given the initial state $\mathbf{x}_0$ at time t_0. This set of first-order differential equations is the so-called *chemical master equation* (CME). It is easy to see that the CME consists in a number of ordinary differential equations (ODEs) equal to the number of possible states. As the state values are typically unbounded the number of ODEs is infinite. Therefore the analytical solution is possible only in few cases and, unfortunately, also the numerical solutions are usually very computationally intensive.

To overcome this limitation Gillespie's proposed the *stochastic simulation algorithm* (SSA) [53]. This algorithm is a Monte Carlo strategy and provides exact numerical realizations of the stochastic process defined by the CME. Since its introduction this procedure has been representing a reference point for the development of many approaches (such as the Next Reaction Method [51] or the tau-leaping [54], for a review see [71]) and has been implemented in many software tools (such as Copasi[60], CellWare [36] and StochKit [71]). Nowadays, the class of methods inspired to the original SSA is one of the most important in the field of stochastic simulations.

The theoretical basis of the SSA is the function $p(\tau, j|\mathbf{x}, t)$ that is defined so that $p(\tau, j|\mathbf{x}, t)\mathrm{d}\tau$ is the probability, given the state $\mathbf{X}(t) = \mathbf{x}$, that the next reaction in the system will occur in the infinitesimal time $[t + \tau, t + \tau + \mathrm{d}\tau)$ and will be a reaction r_j. The function $p(\tau, j|\mathbf{x}, t)$ is therefore the joint probability density function of the two random variables τ (the time to the next reaction) and j (the next reaction), considering the system in the state $\mathbf{x}$. By applying the laws of probability, it is possible to derive the following analytical expression (details in [52]):

$$p(\tau, j|\mathbf{x}, t) = a_j(\mathbf{x})e^{-a_0(\mathbf{x}\tau)} \tag{2.4}$$

[1]We limit the possible values of h_j to bimolecular reactions as the cases in which more than two molecules take part in a reactive collision to produce one or more products are rare.

where $a_0(\mathbf{x}\tau) = \sum_{k=1}^{M} a_k(\mathbf{x})$. Equation 2.4 is the mathematical basis for the stochastic simulation approach and implies that $p(\tau, j|\mathbf{x}, t)$ can be written as the product

$$p(\tau, j|\mathbf{x}, t) = a_0(\mathbf{x})e^{-a_0(\mathbf{x}\tau)} \cdot \frac{a_j(\mathbf{x})}{a_0(\mathbf{x})} \tag{2.5}$$

between a τ-density function and j-density function. Therefore τ is an exponential random variable with both mean and standard deviation equal to $1/a_0(\mathbf{x})$, and j is a statistical independent integer random variable with point probabilities $a_j(\mathbf{x})/a_0(\mathbf{x})$.

There are several equivalent methods to generate samples of τ and j. The simplest method to achieve this aim is the so called *direct method* in which two random numbers r_1 and r_2 are drawn from the random distribution in the unit interval [0,1] and the values of τ and j are calculated as follows:

$$\tau = \frac{1}{a_0(\mathbf{x})} ln(\frac{1}{r_1}) \tag{2.6}$$

$$j \text{ is the smallest integers satisfying } \sum_{j'}^{j} a_{j'}(\mathbf{x}) > r_2 a_0(\mathbf{x}) \tag{2.7}$$

Considering the direct method, the SSA algorithm can be summarized as follows:

1. initialise the time $t = t_0$ and set the initial amount of molecules to $\mathbf{x} = \mathbf{x_0}$;
2. evaluate all the propensity functions $a_j(\mathbf{x})$ and their sum $a_0(\mathbf{x})$;
3. generate the values τ and j according to Equations 2.6-2.7;
4. update the system: $\mathbf{x} := \mathbf{x} + \mathbf{v}_j$ and $t := t + \tau$;
5. if the termination criteria is satisfied, then end the simulation, else return to step 2.

The main advantages of the use of the SSA rely in the fact that it is logically equivalent to the CME, provides an exact trajectory of the system and considers a step τ which is exact (and not an approximation as for example, for ODE solvers). On the other hand, the main disadvantages of the SSA are related to its computational cost, which depends on the reaction and molecule numbers. As a consequence, the SSA is hardly useful to study complex biological pathways, that are characterised by many different types of molecules and reactions. The presence of this computational issue encouraged the development of different modified versions of the original SSA procedure in order to speed up its computation, such as the first reaction method [53], the next reaction method [51], the optimized direct method [23] and the sorting direct method [77].

However, to achieve a significant speed up, approximations are required over the SSA procedure [71]. In the next subsection we will describe one of the most prominent approximation procedures: the *tau leaping* technique.

2.1.2 The tau leaping simulation method

The tau leaping method was initially introduced in [54] in order to decrease the computation cost of stochastic simulation of biochemical systems. The basic idea of this simulation technique is to fire more than one reaction events after a pre-selected time step τ. Moreover, this quantity must satisfy the *leap condition*, according to which the change in the system's state due to the occurring of the selected reactions must be sufficiently small such that no propensity function will suffer an appreciable change in its value. The gain obtained at the computational level is paid with the introduction of an error on the systems dynamics that is no longer exact, as in the SSA, but approximated. Many versions of the tau-leaping technique have been introduced to improve the initial procedure. Problems related to the generation of negative amount of molecules and to the efficient tau selection have been worked out in the tau-leaping version described in [21] and [22] respectively, and we consider these two versions in the following.

As long as the leaping condition is satisfied, the number of firings of reaction j in the time interval $[t, t+\tau)$ can be approximated with a Poisson random variable $\mathcal{P}(a_j(\mathbf{x}), \tau)$ of mean and variance equal to $a_j(\mathbf{x})\tau$. This argument leads to the basic update formula for the system's state considering the time increment τ:

$$\mathbf{X}(t+\tau) \doteq \mathbf{x} + \sum_{j=1}^{m} \mathbf{v}_j \mathcal{P}_j(a_j(\mathbf{x}), \tau) \tag{2.8}$$

where the values $\mathcal{P}_j(j = 1, \ldots, m)$ are Poisson random numbers with the indicated means. However, to face issues such as the negative amount of molecules and the efficient tau selection, the actual algorithm is more complicated than the generation of Poisson random numbers and the system's state update according to Equation 2.8. Each iterative step of the tau leaping procedure can be divided into six stages:

1. initialisation of the time $t = t_0$ and the initial amount of molecules to $\mathbf{x} = \mathbf{x_0}$;

2. generation of the maximum change of each molecule number that satisfy the leap condition; this step is carried out in order to bound the relative changes of the propensity function within a number $0 \leq \epsilon \leq 1$; a value ϵ_i is calculated for each species s_i considering the highest order of the reactions in which s_i appears as reactant (see [22] for more details);

3. calculation of the mean and variance of the expected changes in the propensity functions; this step involves the identification of the sets of *critical* and *noncritical* reactions, where a critical reaction is characterised by having a positive propensity function and by involving (at least one) reactant which is available in a few molecules and noncritical reactions are all the other;

4. τ value computation using quantities calculated in the previous steps;

5. sampling of the number of firings k_j for each reaction r_j using the Poissonian distributions;

6. system's state update, by replacing $t := t + \tau$ and $\mathbf{x} := \mathbf{x} + \sum_{j=1}^{m} k_j \mathbf{v}_j$.

7. if the termination criteria is satisfied, then end the simulation, else return to step 2.

As already said the tau leaping is faster (but less correct) than any of the SSA versions available. The tau leaping version introduced in [22] requires a computational time proportional to $2m$ (where m is the number of reactions defined in the molecular reacting system), and is more efficient then the original tau leaping version, that is associated to a computational cost proportional to m^2. Considering this evidence, the tau leaping procedure introduced in [22] was used as a reference point for the implementation of the τ-DPP simulation approach [27], which will be described in Section 2.3 and is the starting point for the work described in this thesis.

2.2 Modelling spatially extended systems

Living cells are very far from the homogeneous and diluted compartment that is often used for their modelling. These requirements can be considered satisfied in many cases without taking them explicitly into account; however, there are several processes in which the effects of spatial heterogeneity (due to diffusive processes) and crowding (caused by the presence of big macromolecules and other entities such as organelles [47]) must be considered in order to capture the correct system dynamics [91].

Reaction-diffusion (RD) systems are mathematical models used to describe those chemical systems for which the spatial distribution of chemicals influences the overall dynamics. Their name is self-explanatory concerning the processes used to describe their dynamics: diffusion (the spread of particles due to random motions leading to gradual mixing of matter) and reactions.

Computational approaches aimed at studying spatially heterogeneous systems have to deal with the tabulation of spatial position of particles as a function of time. Several modelling frameworks can be used to analyse such

kind of systems [98]. We report hereafter the classic and most used methods, taking also in consideration the possible explicit or implicit modelling of crowding effects.

2.2.1 Deterministic and continuous approach

The standard approach to describe the dynamics of an RD system exploits a continuous time and space domain description of the system where the mass transport, the chemical kinetics and the conservation laws, together with the boundary conditions, are embedded within the same set of equations that can be solved analytically or numerically.

The two fundamental continuum equations describing diffusion are the *conservation of mass*:

$$\frac{\partial [s](\mathbf{r},t)}{\partial t} = -\nabla \cdot \mathbf{J} \tag{2.9}$$

where[2] $[s]$ is the concentration of species s at location $\mathbf{r}$ at time t and $\mathbf{J}$ is the flux of s; and the *Fick's first law*:

$$\mathbf{J} = -D_s \nabla [s](\mathbf{r},t) \tag{2.10}$$

where D_s is the *diffusion coefficient* of s. Combining Equation 2.9 and Equation 2.10 we obtain the *diffusion equation*, which describes the temporal variation of s concentration in every location $\mathbf{r}$ within a given space domain:

$$\frac{\partial [s](\mathbf{r},t)}{\partial t} = \nabla \cdot (D_s([s],\mathbf{r}) \nabla [s](\mathbf{r},t)) \tag{2.11}$$

where $D_s([s],\mathbf{r})$ is the collective diffusion coefficient for concentration $[s]$ at the spatial coordinates $\mathbf{r}$. In case the molecular species is also involved in one or more reaction processes leading to a rate $R([s](\mathbf{r},t))$ we obtain a reaction-diffusion equation

$$\frac{\partial [s](\mathbf{r},t)}{\partial t} = \nabla \cdot (D_s([s],\mathbf{r}) \nabla [s](\mathbf{r},t)) + R([s](\mathbf{r},t)) \tag{2.12}$$

This deterministic and continuous description represents the classical method used to model RD systems. Equation 2.12 relates the variation of a species concentration at each space coordinate in a given space domain to the variation of the flux and the reaction rate. Crowding effects can be implicitly represented acting on diffusion coefficients (e.g., by lowering their values) and kinetic constants (e.g., by increasing their values). PDEs are usually solved using numerical methods (only in a few cases the analytical solution

[2]The *del* operator represented with the *nabla* symbol ∇ is used to denote (considering the three dimensional Cartesian coordinate system) the *gradient* $\nabla f = (\partial f/\partial x\ \hat{\mathbf{i}}, \partial f/\partial y\ \hat{\mathbf{j}}, \partial f/\partial z\ \hat{\mathbf{k}})$ of a scalar field $f(x,y,z)$, and the *divergence* $\nabla \cdot \mathbf{v} = (\partial v_x/\partial x, \partial v_y/\partial y, \partial v_z/\partial z)$ of a vector field $\mathbf{v} = v_x\hat{\mathbf{i}} + v_y\hat{\mathbf{j}} + v_z\hat{\mathbf{k}}$.

is available); moreover, as the time-step and the sub-volume size (the space domain is usually divided in a number of elements) are reduced, the solutions becomes more accurate while the computational effort increases.

2.2.2 Reaction-diffusion master equation

A natural extension of the chemical master equation to keep track the amount of substance in different location of a space domain Ω consists of dividing Ω into smaller sub-compartments i, each one with a characteristic length l ($V_i = l^d$, being $d \in \{1, 2, 3\}$ the spatial dimensions), such that each of these sub-compartments can be considered homogeneous.

The condition for homogeneity by diffusion is that:

$$t_i \gg \frac{l^2}{2dD_i} \tag{2.13}$$

where t_i is the average time between two reactions involving species s_i and D_i is the diffusion coefficient of s_i. To satisfy Equation 2.13 the reaction rate must be much slower than particle diffusion between sub-compartments. As the right term of Equation 2.13 scales with l, the validity of the above inequality improves as l is reduced. At the same time, l must be much larger than the mean free path. As this mean free path is usually very short in living cells, due to the high concentration of non-reactive molecules, this condition is easily satisfied [67]. Lastly, the compartment size must be much larger than the reaction radii [67].

In this context the system's state changes from $\mathbf{x}$ to $\{\mathbf{x}\} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ where each vector $\mathbf{x}_1$ defines the number of molecules in each of the m compartments. Diffusion is modelled as a memory lacking random walk in a discretized space, where molecules diffuse between neighbouring compartments. The diffusion rate constant for a molecule of type s_i which moves between compartment k and $k+1$ is

$$d_i^{k,k+1} = \frac{D_i}{l^2} \tag{2.14}$$

and enable the connection between the microscopic description of the master equation with the macroscopic Fick's diffusion coefficient.

Considering these arguments it is possible to define a Reaction-Diffusion Master Equation (shortly RDME) [49]. The final form of the RDME is analogous to a semi-discrete form of the reaction-diffusion PDE with the diffusion term discretized using a second-order centered scheme [11]. As in the case of the CME, analytical approaches to RDME are too complicated, especially in the field of the modelling of biological pathways, where we have to deal with many different species and many reactions. An alternative approach relies on the use of a Monte Carlo method to sample trajectories of the Markov process associated to the RDME.

Bernstein showed that SSA can be used to simulate reaction-diffusion systems, even if the space domain is dived into a set of differently sized compartments and a space dependent diffusion coefficient is considered [11]. To model diffusion between three adjacent compartments $k-1, k$ and $k+1$ the author considers the following set of mono-molecular reactions

$$s(k-1) \xrightarrow{d_{k-1,k}} s(k) \tag{2.15}$$

$$s(k) \xrightarrow{d_{k,k-1}} s(k-1) \tag{2.16}$$

$$s(k) \xrightarrow{d_{k,k+1}} s(k+1) \tag{2.17}$$

$$s(k+1) \xrightarrow{d_{k+1,k}} s(k) \tag{2.18}$$

where $a(b) \to a(d)$ indicates that the object a is moved from region b to region d. The quantities $d_{k-1,k}$, $d_{k,k-1}$, $d_{k,k+1}$ and $d_{k+1,k}$ are kinetics constants that must be set as

$$c_{i,j} = \frac{D_{i,j+1}}{l_{i,j}|\tilde{l}_i - \tilde{l}_j|} \tag{2.19}$$

where $D_{i,j}$ is the diffusion coefficient evaluated at the boundary between i and j and values $\tilde{l}_i - \tilde{l}_j$ is the distance between the centres of sub-compartments i and j. Note that Equation 2.19 reduces to Equation 2.14 if i and j have the same length (uniform grid).

An important procedure introduced in this field is the *Next Subvolume Method* [39] which is implemented in MesoRD [59] and SmartCell [5] software. The NSM is an adaptation of the SSA [53] and the next reaction method [51] to compute the dynamics of a reaction-diffusion systems according to the RDME. Given an initial system's state the time for the first event (reaction or diffusion) inside each subvolume is sampled from a respective distribution (that takes into account the rates of reaction and diffusion in the subvolume); these time values are used to sort the subvolume in a priority queue, according to when the events are scheduled to appear. If the first occurring event (top ranking subvolume) is a reaction event, the rates in the respective subvolume must be recalculated and the subvolume placed accordingly in the queue. If the event is a diffusion event, than two subvolumes are involved and hence two elements in the priority queue must be reordered. The NSM requires a computational time which scales logarithmically (rather than linearly) with the number of subvolumes.

To speed up the NSM the binomial tau-leap spatial stochastic simulation algorithm (Bτ-SSSA) has been introduced in [72]. The Bτ-SSSA exploits the binomial tau-leap algorithm [99] and a modified version of the NSM [39]. The comparison of the number of operations required to execute a single iteration show that the Bτ-SSSA has a higher complexity than the NSM [25]. However, as the Bτ-SSSA can execute more than one reaction and/or diffusion event for a given time step τ, the efficiency of the simulations is increased, and in

fact it is reported that Bτ-SSSA is from 2 to 100 times faster than NSM (in the examples provided by the authors) [72].

A limitation of these stochastic methods consists in the fact that the size of chemicals and of the compartments in which reactions take place is not considered during the simulation of the system dynamics. As a consequence, it is not possible to reproduce crowded conditions because volume exclusion due to crowding molecules cannot be represented explicitly when they are depicted as point particles [98].

2.2.3 Other approaches

The *Molecular Dynamics* (MD) approach traces the positions and velocities of all atoms and, therefore, provides detailed trajectories. On the other hand, this method is computationally too expensive to simulate systems formed by a large number of atoms or with time scales above μs. Consequently, MD has only been used in problems involving time-scales of ns and space-scales of tens of nm [98].

The *Brownian dynamics* (BD) is a particle-based stochastic approach used to describe the time and space motion of molecules. As the solvent is treated as a continuum medium and the trajectories of the modelled particles are described by random walks (due to the collisions with the solvent molecules), the computational cost is decrease dramatically compared to MD. In BD, time and space are continuous. Crowded media can be explicitly described since it is possible to represent crowding molecules. However, as the number of particle collisions increases, the BD simulations demands a very high computational cost due to the relatively small time step required to resolve collision events leading to chemical reactions. Examples of methodologies based on BD are the Green's function reaction dynamics algorithm [102], Smoldyn [6] and MCell [96].

The last class of methods we want to cite are those based on *Cellular Automata* (CA). A CA consists of a grid of cells (in any number of dimensions), where each cell has a finite set of states, and evolves according to the neighbours state. CA can be used to simulate RD systems at both microscopic and mesoscopic scales, depending on the number of molecules associated with each cell of the lattice. Crowding can be explicitly represented by considering crowding molecules or fixed barriers. For instance, people of the CyberCell project modelled a virtual cell membrane using discrete automata [18].

2.3 Membrane systems as modelling tools

In this Section, we describe a framework for the modelling of biological systems which is the results of the conjugation of Dynamical Probabilistic P Systems [85] with a modified version of the tau leaping algorithm. Before presenting this approach, introduced in [27] and designated as τ-DPP, we

briefly describe, on one hand, the useful properties of membrane systems as a tool for the modelling of biological systems, and, on the other hand, the ingredients that have to be added/discarded in order to achieve an accurate description of the modelled systems.

Despite the initial primary goals of membrane computing was computability and natural computing in particular, these devices started to be useful for the modelling of biological systems. Just to mention a few, there are applications related to intracellular signalling [92, 48] and metapopulations[3] [16]. These applications are possible due to a series of properties that promote the use of membrane computing as a tool for the modelling of biological systems [90]:

- *compartmentalisation*: compartmentalisation plays an important role in many biological processes, such as the cell cycle regulation [38], and membrane systems, by definition, capture this property;
- *modularity*: a crucial property for the functioning of living systems [58] and an intrinsic feature of membrane systems, as each membrane or sets of membranes can be seen as modules;
- *scalability/extensibility*: membranes and rules can be added to or discarded from an existing system without the need for changing the way of working with the system;
- *understandability*: rules can be used to represent many dynamical processes (such as reaction and diffusion) using the simple notation of chemical reactions;
- *programmability*: it is easy to implement membrane systems into programs and certain languages, such as CLIPS, are perfectly fitted to such a job.
- *discreteness*: a useful feature for modelling many systems (such as reaction-diffusion systems, cell or individual populations), that enables the identification of patterns that can not be accounted using a continuous approximation [100].

On the other hand, some aspects of membrane systems are not adequate to achieve a proper description of the biological reality. The maximal parallelism of rule application, which consists in the execution of all the applicable rules at each step (according to the availability of objects inside the system) must be adjusted, as the biochemical reactions are not synchronised by a global clock, but occur *in time* on the basis of the system state and properties related to the biochemical processes themselves (e.g. the intrinsic catalytic

[3]Metapopulations, or multi-patch systems, are models describing the interactions and the behaviour of populations living in fragmented habitats.

activity of an enzyme). At the same way, the nondeterministic selection of concurrent rules must be handled differently, as at the molecular level, events are not so uniformly accidental. Lastly, the merely topological arrangement of membranes, since also physical dimensions and spatial coordinates respect to a reference system matter.

2.3.1 Dynamical probabilistic P systems

Dynamical Probabilistic P Systems (DPPs) [85] were introduced for the description and analysis of biological or chemical systems. In DPPs, the maximal parallelism has been mitigated by defining a rule application strategy that makes use of probability values associated with the rules. These probability values depend on the multisets of objects and change during the evolution of the system. Hence, DPPs provide a stochastic and discrete description of the system's dynamics, that is, DPPs allow to reproduce the stochastic variations of the elements (e.g. chemical species, individuals) occurring in the system. We recall the formal definition of DPP given in [85].

Definition 2.3.1. *A Dynamical Probabilistic P system (DPP) of degree n is a construct*

$$\Pi = (V, O, \mu, M_0, \ldots, M_{n-1}, R_0, \ldots, R_{n-1}, E, I) \tag{2.20}$$

where:

- *V is the alphabet of the system, $O \subseteq V$ is the set of analysed symbols;*
- *μ is the membrane structure consisting of n membranes labelled with the numbers $0, \ldots, n-1$. The skin membrane is labelled with 0;*
- *M_i, $i = 0, \ldots, n-1$ is the multiset over V initially present inside membrane i;*
- *R_i, $i = 0, \ldots, n-1$ is a finite set of rules associated with membrane i; an evolution rule is of the form $u \xrightarrow{k} v$, where u is a multiset over V, v is a string over $V \times (\{here, out\} \cup \{in_j | 1 \leq j \leq n-1\})$, and $k \in \mathbb{R}^+$ is a constant associated with the rule;*
- *$E = (V_E, M_E, R_E)$ is called the environment, and consists of an alphabet $V_E \subseteq V$, a feeding multiset M_E over V_E, and a finite set of feeding rules R_E of the type $r : u \to (v, in_0)$ for u, v mutlisets over V_E;*
- *$I \subseteq \{0, \ldots, n-1\} \cup \{\infty\}$ is the set of labels of the analysed regions where the label ∞ corresponds to the environment.*

As the DPPs were introduced for the analysis of complex systems, we recall also the definitions of the set of parameters of a DPP and of the concept of family of DPPs. It is in fact important to investigate the relation between the parameters and dynamics of a model representing a complex system.

Definition 2.3.2. *The set of parameters $\mathcal{P}$ of a dynamical probabilistic P system Π consists of*

1. *the multiplicities of all symbols appearing in the multisets $M_0, \ldots, M_{n-1}$ initially present in μ, and of those appearing in the feeding multiset M_E*
2. *the constants associated to all rules in $R_0, \ldots, R_{n-1}$*

Hence, while multiplicities and rule constants are parameters, all the other components appearing in Definition 2.3.1 constitute the main structure of a DPP.

Definition 2.3.3. *Let Π be a DPP and Par be a family of sets of parameters for Π. The* family *of DPPs defined by Π and Par is $\mathcal{F}(\Pi, Par)$ consisting of all DPP's with the main structure of Π and the set of parameters $\mathcal{P} \in Par$; such a DPP is denoted by $(\Pi, \cap P)$.*

Thus, a family of DPPs can be used to analyse the dynamics of a complex system modelled using Π under different parametrisations, $\mathcal{P}_1, \mathcal{P}_2$ and so on, where it holds $\mathcal{P}_1 \neq \mathcal{P}_2$ for a choice of at least one value in $\mathcal{P}_1$ and $\mathcal{P}_2$.

The evolution of a DPP is obtained by means of the simultaneous application of all the applicable rules Hence, the parallelism is maximal and a universal clock is assumed to exist. Objects are assigned to the rules according to probability values, that are calculated considering rule constants and objects availability.

The probability $p_{i,j}$ associated with each rule $r_j : u \xrightarrow{k} v$ (belonging to the set R_i) is calculated starting from the pseudo probability $\tilde{p}_{i,j}(r_j)$:

$$\tilde{p}_{i,j}(r_j) = \begin{cases} 0 & \text{if } M_i(a_h) < \alpha_h \text{ for some } h \in H \\ k \cdot \prod_{h \in H} \frac{M_i(a_h)}{\alpha_h!(M_i(a_h)-\alpha_h)!} & \text{if } M_i(a_h) \geq \alpha_h \text{ for some } h \in H \end{cases} \tag{2.21}$$

where $M_i(a_h)$ is the number of objects of type a_h in the multiset M_i, $H = \{1, \ldots, s\}$ contains the positions of the symbols a_h in the string $u = a_1^{\alpha_1}, \ldots, a_s^{\alpha_s}$. Equation 2.21 tells us that the pseudo-probability of a rule is null if the number of any of the objects it acts on is not sufficient. Otherwise the pseudo-probability is the product of all the possible combinations of the objects the rule acts on, multiplied by the rule constant k. Note the equivalence between Equation 2.21 and the propensity function calculation in the context of the SSA algorithm, Equation 2.2 (that was defined considering only strings u containing at most three objects). In other words, a pseudo-probability corresponds to the possible collisions among reactants in a well-stirred compartment. The pseudo-probability is then normalised considering all the oter rules of same membrane, thus obtaining the rule probability $p(r_j)$.

A simulator to describe the dynamics of DPP was introduced and used to study the Lotka-Volterra system and metapopulations systems in [85].

2.3.2 The τ-DPP approach

To bridge the gap between membrane systems and real world properties, some additional details must be added. An approach that achieves this aim is the τ-DPP, introduced to provide a *quantitative* description of a system dynamics [27], by extending the single-compartment algorithm of tau-leaping [22].

The τ-DPP methodology is based on a class of stochastic membrane systems, DPPs (Section 2.3.1). The system's evolution described by a DPP is only *qualitative*, in the sense that an effective (physical) time streamline cannot be directly associated to the evolution steps of the system. As the τ-DPP approach describes the system's evolution by means of a modified version of the tau-leaping procedure, it overcomes this limitation.

The τ-DPP is a computational method which can be used to describe and perform stochastic simulations of complex biological or chemical systems. The "complexity" of the systems that can be managed by means of τ-DPP, resides both in the number of the (chemical) reactions and of the species involved, and in the topological structure of the system, that can be composed by many membranes (also referred to as compartments). For instance, cellular pathways involving several spatial compartments (as the extracellular ambient, the cytoplasm, the nucleus, etc.), or multicellular systems like bacterial colonies, or multi-patched ecological systems as metapopulations, are all examples of complex systems that could be investigated with τ-DPP.

The correct behaviour of the whole system is achieved by letting all compartments evolve in parallel, and by using the following strategy for the choice of time increments. At each iteration of τ-DPP, we consider the current state of each compartment (determined by the current number of molecules), and we calculate a time increment independently in each compartment (according to the standard tau-leaping algorithm [22]). Then, the smallest time increment is selected and used to evaluate the next-step evolution of the entire system. Since all compartments *locally* evolve according to the same time increment, τ-DPP is able to correctly work out the *global* dynamics of the system. Moreover, by adopting this procedure, the simulated evolutions of all compartments get naturally *synchronized* at the end of each iterative step. The synchronization is also necessary – and exploited together with a parallel update of all compartments – to manage the communication of molecules among compartments (i.e., diffusive events), whenever prescribed by specific (communication) rules.

The system is defined by means of a set of n compartments organised according to the hierarchy specified by the membrane structure. The state of the whole system is characterised by all multisets W_i occurring inside each compartment μ_i $(1 \leq i \leq n)$.

Inside the compartments, the sets of rules $R_1, \ldots, R_n$ are defined along with the sets of stochastic constants $C_1, \ldots, C_n$.

Each compartment μ_i can contain two different kinds of rules, termed *internal* and *communication* rules. An internal rule describes the modification, or evolution, of the objects inside the single compartment where it is applied, while a communication rule sends the objects from the compartment where it is applied to an adjacent compartment (possibly modifying the form of these objects during the communication step).

More precisely, internal rules have the general form $\alpha_1 s_1 + \alpha_2 s_2 + \cdots + \alpha_m s_m \to \beta_1 s_1 + \beta_2 s_2 + \cdots + \beta_m s_m$, where $s_1, \ldots, s_m$ belong to the set of distinct object types Σ, and $\alpha_1, \ldots, \alpha_m, \beta_1, \ldots, \beta_m \in \mathbb{N}$. For instance, $s_1, \ldots, s_m$ can correspond to molecular species, and, in this case, $\alpha_1, \ldots, \alpha_m, \beta_1, \ldots, \beta_m$ represent stoichiometric coefficients. The objects appearing in the left-hand side of the rule are called *reagents*, while the objects on the right-hand side are called *products*. Note that, usually, we will consider the case where (at most) three objects appear in the reagents group. The rational behind this is that we require biochemical reactions to be (at most) of the third-order, since the simultaneous collision and chemical interaction of more than three molecules at a time, has a probability to occur close to zero in real biochemical systems. Moreover, the interaction among more than three molecules can be modelled by using a set of successive reactions with lower order. In what follows, we will refer to rules or reactions without distinction.

When dealing with communication rules inside a compartment, besides defining the sets of reagents and products, it is necessary to specify the target compartment where the products of this rule will be sent[4]. Formally, a communication rule has the form[5] $\alpha_1 s_1 + \alpha_2 s_2 + \cdots + \alpha_m s_m \to (\beta_1 s_1 + \beta_2 s_2 + \cdots + \beta_m s_m, tar)$, where $s_1, \ldots, s_m \in \Sigma$ are distinct object types, $\alpha_1, \ldots, \alpha_m, \beta_1, \ldots, \beta_m \in \mathbb{N}$, and tar represents the compartment where the products of the reaction diffuse.

A complete an extensive description of the τ-DPP algorithm and some applications is available in [27, 25].

In order to correctly describe the behaviour of a system, τ-DPP runs in parallel inside each membrane (or compartment). A modified version of the tau-leaping procedure presented in [22] is exploited to compute the length of the step τ. In this novel version of the simulation algorithm, the least value for the time increment, among those computed inside each compartment, is used to sample the number of reactions to execute (as in the original tau-leaping algorithm). Thanks to this "common" time increment, shared by all compartments, the simulation is synchronized at each step, allowing the correct passage of the molecules involved in communication rules (diffusion between two compartments).

Each step is executed *independently* and *in parallel* within each membrane

[4]This definition can be easily extended in order to assign a different target compartment to each object appearing in the set of products.

[5]The condition that at most three objects appear as reagents is usually required also for communication rules.

or compartment i, $(i = 0, \ldots, n)$ of the system. In the following description, the algorithm execution naturally proceeds according to the order of instructions, when not otherwise specified by means of `goto` commands.

1. load the description of the compartment i, which consists of the initial amount of all object types and the set of rules and their respective constants;
2. `for each` rule r_k $(k = 1, \ldots, l)$: compute the propensity function $a_k := c \cdot h$;
3. evaluate the sum of all the propensity functions in the compartment $a_0 := \sum_{k=1}^{l} a_k$;
4. `If` $a_0 > 0$: `goto` *8*;
5. set $\tilde{\tau}_i := \infty$;
6. wait for the communication of the smallest time increment $\tau_i := \min\{\tilde{\tau}_i, \ldots, \tilde{\tau}_n\}$ among those generated independently inside all compartments, during the current iteration;
7. `goto` *20*;
8. generate the step size τ_i according to the internal state, and select the way to proceed in the current iteration (i.e. SSA-like evolution, tau-leaping evolution with non-critical reactions only, or tau-leaping evolution with non-critical reactions and one critical reaction), using the selection procedure defined in [22];
9. wait for the communication of the smallest time increment $\tau_i := \min\{\tilde{\tau}_i, \ldots, \tilde{\tau}_n\}$ among those generated independently inside all compartments, during the current iteration;
10. `switch` the evolution strategy type:
    - `case` "SSA-like":
      - `if(`$\tilde{\tau}_i > \tau_i$`)`: `goto` *11*;
      - `else`: `goto` *16*;
    - `case` "tau-leaping with one critical reaction plus non-critical reactions":
      - `if(`$\tilde{\tau}_i > \tau_i$`)`: `goto` *19*;
      - `else`: `goto` *18*;
    - `case` "tau-leaping with non-critical reactions only":
      - `goto` *19*;

11. compute $\tilde{\tau}_i := \tilde{\tau}_i - \tau_i$;
12. wait for possible communication of objects from other compartments, by means of communication rules;
13. `If` some object is received: `goto` *2*;
14. set $\tau_i := \tilde{\tau}_i$ for the next iteration;
15. `goto` *8*;
16. use the SSA strategy [53] to extract the rule that will be applied in the current iteration;
17. `goto` *20*;
18. extract the critical rule that will be applied in the current iteration;
19. extract the set of non-critical rules that will be applied in the current iteration;
20. `if` the execution of the selected rules (considering all the compartments) leads to an unfeasible state, namely, there are negative amounts of molecules:
    - reduce τ_{min} by half;
    - send the new value to the other membranes;
    - goto *9*;
21. wait for the possible communication of a τ_{min} reduced by half;
22. `if` a new value of τ_{min} reduced by half is received: `goto` *9*;
23. update the internal state by applying the extracted rules (both internal and communication) to modify the current number of objects;
24. `if` some objects is received from the other compartments: update the internal state modifying the amount of objects;
25. `if` the termination criteria is satisfied: finish; `else`: `goto` *2*.

The algorithm begins with the initialisation of membrane descriptions, which consist in the initial amount of molecules, the rules and the constants. Then, the propensity functions of the rules are calculated using the expression introduced by D.T Gillespie in [53].

If the sum of the propensity functions of the processed compartment is null, no rule can be executed inside this membrane and therefore τ_i is set to a very high value. In this case the compartment waits for the communication

of the global time increment τ_i sent by another membrane, where possibly some rules is executed. If this is the case, the compartment also waits for the possible communication of object from other compartments and then proceed to the next iteration; otherwise, the computation is finished because no rule can be applied in any compartment.

If the sum of the propensity functions of the processed compartment is greater than zero, the local time increment, $\tilde{\tau}_i$, is computed and the evolution strategy is selected using the procedure described in [22].

Once every compartment has computed its $\tilde{\tau}_i$, the smallest one is selected and used to define the evolution of the whole system during the current iteration. Hence, each membrane will not evolve according to its own $\tilde{\tau}_i$, but relying on a global time increment, τ_i; this ensures the systems synchronisation and its evolution along a common time line.

After that every compartment has received the τ_i, the subsequent steps are selected according to it and to the local evolution strategy generated previously.

If the local strategy is SSA and the τ_i is equal to the local $\tilde{\tau}_i$, the τ_i was generated by this membrane and the algorithm proceeds with the extraction of the rule that will be applied. Conversely, if the τ_i is greater than $\tilde{\tau}_i$, the selected time step τ_i is not "long enough" for the application of the rule within this membrane, and hence the rule will not be applied. In this case, if some objects is received from the other membranes the internal state is changed and a new $\tilde{\tau}_i$ will be computed during the next iteration. Otherwise, the next time increment for the membrane is set as $\tilde{\tau}_i := \tilde{\tau} - \tau_i$, because in the case this new value will be the smallest inside the system the compartment will be enabled to apply one rule.

If the local strategy is executing a tau-leaping step with the application of a set of non-critical reactions and one critical reaction, and the local time step is equal to the smallest one, it is possible to execute the critical reaction and the set of non critical reactions extracted from the poissonian distributions (see [22] for details); conversely if the local time increment is greater than the selected time increment, the compartment will execute only non critical reactions.

If the local strategy is executing a tau-leaping step with the application of a set of non-critical reactions the τ_i value is used to sample the number of rules that will be applied using the strategy described in [22].

Subsequently, the algorithm checks whether the execution of the selected rules leads to a unfeasible state. If this is the case, the time step is reduced by half and go back to selection of the new smallest τ_i. This operation can lead to the selection of a different set of rules and is repeated until a valid set of rules is selected.

As the rules selection according to the current τ_i is worked out in each compartment, the internal state is updated. During this step the amount of objects are updated as specified by the selected rules.

The computational cost of the τ-DPP algorithm is $2mn$, where m is the number of the rules and n the number of compartments. In fact, the computational cost of the tau-leaping algorithm, $2m$, must be multiplied by the number of compartments defined in the simulated system. The algorithm is implemented in a parallel (MPI) version in the C language. The parametrisation schema establish a one-to-one relation between processes and membranes.

3

Grid computing for parameter sweep applications

Systems biology models are becoming more and more complex. For instance, the first models describing the cell cycle dynamics (such as the one by Goldbeter [56]) included only a few proteins and biochemical processes, while more recent cell cycle models (e.g. the eukaryotic cell cycle model presented in [32]) describe the dynamics of tens of proteins by means of tens of biochemical processes (a web collection of cell cycle models is presented in [3]). Due to their complexity, these models cannot be solved analytically, and therefore numerical simulations are required to study the models' properties.

In general, these simulations concern the evaluation of the model response as a consequence of variations related to the system structure, the rates associated with each molecular process or the parameters of the models. As the number of these quantities gets higher and higher, and as the models considered are more and more detailed and comprehensive, a large number of simulations is required to explore the spaces formed by their values. This scenario raises issues related to the required computational resources, since in the case of stochastic modelling, the algorithms are more time-consuming than in the deterministic case. Furthermore, due to stochasticity, more than one stochastic simulation is required to characterise the systems dynamics, resulting in a very expensive computation.

A possible solution to cope with a computationally intensive problem is to exploit a distributed approach such as the grid computing, a kind of high-throughput computing, where a combination of computer resources are used to reach a common goal. The difference between the Grid computing and the classic high performance computing, such as a cluster of processors, is that the computational resources shared in a grid tend to be loosely coupled, heterogeneous and geographically dispersed. The term was firstly introduced in the mid '90 to denote a distributed computing infrastructure for advanced science and engineering [44]. Grid belongs to the high throughput computing paradigm and it is characterised by independent and sequential jobs that can

be individually scheduled on many different computing resources. Therefore, grid computing is a useful solution to compute the large number of independent simulations that are required, for instance, during the development and analysis of biochemical models.

The repeated execution of an application using different parameter values is referred to as parameter sweep application (PSA). As each run of a PSA is independent, PSAs fit very well the grid computing approach. In this chapter we illustrate the main European grid infrastructure and how to deal with PSA exploiting this platform.

3.1 Grid computing and the EGEE project grid

An important grid is the one implemented in the context of the project designated as *Enabling Grid for E-sciEnce* (EGEE). The EGEE project grid is wide area grid platform for scientific applications composed of thousands of CPUs, which implements the Virtual Organisation (VO) paradigm [45]. The production framework is a large multi-science grid infrastructure, federating 250 resource centres worldwide, which provides comprehensively 20.000 CPUs and several Petabytes of storage. This infrastructure is used daily by thousands of scientists federated in over 200 VOs.

The EGEE platform uses gLite middleware [68], which was developed through the collaboration of a number of projects, like DataGrid, DataTag, Globus, GriPhyN, and LCG. The gLite distribution is an integrated set of components designed to allow resource sharing and must be installed on a local server, the User Interface (UI) to allow users to manage computations on the EGEE grid. In particular, employing gLite, it is possible to submit grid jobs, monitor their state of advancement, and retrieve the output when the computations are successful or to resubmit them in case of failure. This grid infrastructure is highly scalable and allows computationally intensive challenges to be accomplished, but users must cope with the continuous dynamic reshape of the available resources, which is typical of loosely coupled distributed platforms.

To enable a secure connection to the remote resources, the grid middleware offers a well-established security system. The system relies on the Grid Security Infrastructure, which uses public key cryptography to recognise users. The access to remote clusters is granted by a Personal Certificate encoded in the X.509 format, which accompanies each job to authenticate the user. Moreover, users must be authorised to job submission by a VO, a grid community having similar tasks that vouches for them. For example, in Chapter 6 we will discuss a work we did exploiting the Biomed VO, that shares on average 2000 CPUs and welcomes applications in the bioinformatics field, in medical image processing, and more generally in biomedical data processing.

The resources available on the EGEE project platform are composed of a network of several Computing Elements (CEs), which are gateways for computer clusters where jobs are actually performed and an equal number of Storage Elements (SEs) that implement a distributed filesystem to store temporary files. The computational resources are connected to a Resource Broker (RB) that routes each job on a specific CE, taking into account the directives of the submission script, called JDL (Job Description Language). In detail, the Workload Management System (WMS) is the RB service which schedules jobs by delivering them to the resource that best fits the requirements, balancing the computational load, via a Condor G client [20]. Although this brokering policy is not configurable by the user, it provides high performance: bulk submission enables to send of sets of independent jobs up to a rate of 50Hz for job submission and 0.5Hz for job dispatching to the CEs. Finally, each CE routes the incoming jobs to a batch queue system (PBS or LFS), which hides the farm of Working Nodes (WNs) where computations are effectively performed.

To handle files over the grid, the gLite middleware provides a set of tools to manage data similarly to a distributed filesystem. These tools allow the data to be replicated into different SEs, which can help to reduce the database upload and download time during computations. The RB can redirect the execution of an application to a CE located as near as possible to files being used, hence minimising communication time. Moreover, each CE knows which is the nearest SE to store output data generated during a computation. Although large files should always be managed by using the SEs, both in input and output it is possible to use the SandBox to load and download small files directly to the CE. The main difference is that files transferred using the InputSandBox and the OutputSandBox are temporary stored on the RBs, therefore are managed directly by the middleware (but their size should be less than a few MB, otherwise the RBs will be rapidly stuck) and cannot be reused, while files on the SEs have no limitation in size and availability, but must be handled directly by users.

3.2 Parameter sweep applications and grid computing

A *Parameter sweep application* (PSA) consists in the repeated execution of an application (usually performed a large number of times), where each computation is run using a different parametrisation. Applications formulated by means of PSA contain a large number of independent jobs operating on different data sets in order to explore a wide range of scenarios and parameters. This application is executed by processing N independent instances (the same application, but different input data sets) on M parallel or distributed computational resources (where N is, typically, much larger

than M). Fortunately, this high-throughput parametric computing model is simple, yet powerful enough to formulate distributed applications ranging in many different areas.

Taking into consideration τ-DPPs, the application executed during the PSA consists in the computation of system's evolution, and the parametrisation can be obtained by varying the object types, the rules, the initial distribution of objects inside the membranes, the constants associated with the rules and, finally, it is also possible to vary the values of the parameters of the simulator used to compute the system's evolution.

The definition of the set of parametrisations of a PSA depends on the specific applications and on the data type of the parameters involved. Usually, the parametrisations are defined by considering the Cartesian product of the parameters and by sampling values from the space defined by their ranges of variation. When the number of parameters is high, their values can be sampled by using quasi-random series [82] (or low discrepancy sequences): thus, the values of the parameters are sampled by minimising their discrepancy. The discrepancy of a sequence is a measure of its uniformity and is computed by comparing the actual number of sample points in a given multidimensional space with the number of sample points that "should be" there, assuming a uniform distribution. Therefore, the aim of quasi-random series is to "uniformly" cover the space with "few" samples (i.e., with a lower number of points compared to classic uniform distributions).

The output of each PSA is composed of the set of results generated by all executions of the considered application, each one with a different parametrisation.

In principle, implementing a PSA is not difficult. For instance, the Cartesian product of a range of parameters can be used to create a set of different parametrisations. Then, a number of applications equal to the number of parametrisations created can be distributed over the grid. However in practice, running a high number of jobs on the grid poses some issues. From the computational point of view, the biggest problem of the grid is the dynamic behaviour of the available resources. Due to network and system errors or to the global computational load, the resources available are continuously reshaped, and the rate of failure in computations is quite high. Some solutions, such as Nimrod [19] and APST [24], have been developed to perform PSAs using grid technologies, but they rely on very specific middleware implementations, which are not gLite compliant. Moreover, a grid-inspired solution to distribute stochastic simulations is described in [70], but the approach does not rely on an effective grid implementation. Considering that the EGEE project infrastructure is a standard production environment, it is not possible to customize the middleware implementation. Therefore, a crucial point is the employment of a system that can interact with the grid to manage the whole PSA computation, which should check the consistency of each job in a fault tolerant environment. A system of this

type is will be presented in the next Section.

3.3 Managing jobs on grid: the challenge control system

In the context of the EGEE grid, the *Challenge Control System* (CCS) [79] was developed to completely coordinate a grid computation from a single UI, by submitting and managing the whole set of jobs in which the computation is split dealing with CEs and SEs. In other words, this framework provides automatic management of all the necessary operations to fulfil each single task, since it wraps the grid middleware low level API for file handing (`lcg-*`) and for job submission, status monitoring and output retrieving (`glite-*`) by providing a user-friendly and fault-tolerant environment.

The CCS is highly customisable, thanks to its double layered infrastructure: the first layer was developed to cope with the latest middleware versions of the EGEE infrastructure for managing each single job, while the second level deals with the different requirements of the application in hand by splitting the computation and performing the defined tasks on the remote resources. These layers are interconnected by a MySQL database, designed to collect all the information needed to manage each grid job, which works as the back-end of the system.

The first layer of the CCS works in close connection with the UI and is mainly devoted to the management of each single job, from the definition of the JDL script and its submission, to the retrieval of the output results. This layer employs the network time protocol, using the same servers of the EGEE infrastructure in order to synchronise the time on all the grid components, which is crucial for enabling a correct survey of the computation. The system also makes use of the crond daemon for beating the interval between each round of CCS execution in which, according to the information stored in the MySQL database, tasks such as the submission of new jobs, polling the RB about the status of scheduled jobs, the resubmission of failed jobs and the retrieval of output results are accomplished.

The second layer is the most important one from the application point of view, since it coordinates the job distribution over the grid using a set of scripts. These scripts split the whole computation into specific tasks, coordinate the jobs execution on the remote resources and collect the output results. Moreover, this second layer checks the output consistence of each simulation computed on the grid platform, in connection with the upper layer which monitors each job as an independent unit. This framework has been installed directly on a grid UI of ITB-CNR from where users who have a valid Personal Certificate are authorised to submit jobs through the Biomed VO.

Part II

Results

4

Spatially extended membrane systems

The current versions of membrane systems (including DPPs), consider objects and membranes in an abstract environment where the space occupied by membranes and objects is not defined. In this context, an infinite number of objects can accumulate inside the regions of the systems. Similarly, the volume occupied by reactants and compartments is not taken into account by the current stochastic simulation algorithms (including tau leaping) inspired to the SSA (see Section 2.2 and [98]). Therefore, τ-DPPs (the combination of DPPs with a modified version of the tau leaping procedure) suffer this limitation too.

The explicit consideration of object (reactants) and membrane (compartments) space occupation is a crucial feature in order to use membrane systems for the modelling of real systems in which the *crowding* can have an important impact over the system's dynamics. For example, this is the case of the intracellular environment of living cells [47, 80, 106].

In this chapter, we describe an extension of τ-DPPs, designated as Sτ-DPPs, in order to introduce the concept of *space occupation* due to objects and membranes (also referred to as regions). Moreover, we defined Sτ-DPPs in order to enable the communication of objects among *non adjacent* membranes, as this feature leads to a more powerful modelling formalism.

First, we study the consequences of objects and membranes space occupation at the theoretical level. Subsequently, we present the algorithm for the description of the time evolution of Sτ-DPPs. Then, we describe a series models in order to show that a Sτ-DPP permits the *in silico* investigation of a more extended bunch of systems, respect to its "parent" τ-DPP. In particular, we discuss the use of Sτ-DPPs to model the presence of molecular crowding and structured compartments inside the intracellular environment. Lastly, we conclude the chapter discussing the results.

4.1 Space occupation in membrane systems

In this section we begin with the formal definition of a novel variant of membrane systems where objects and membranes occupy a finite amount of space. In these *Spatially extended P systems*, the application of the evolution rules is restrained from the availability of *free space* inside the membranes, and the evolution of these systems is thus influenced by membranes and objects *sizes*, representing the volume these elements occupy in a space with an arbitrary number of dimensions.

Definition 4.1.1. *A spatially extended P system (shortly, SP system) of degree n is a tuple:*

$$\Pi = (\Sigma, \mu, W, R, V_\mu, V_\Sigma)$$

- $\Sigma = \{s_1, \ldots, s_m\}$ *is the set composed by a number m of symbols, also called objects;*
- μ *is a membrane structure consisting of n membranes labelled with the numbers $1, \ldots, n$;*
- $W = \{w_1, \ldots, w_n\}$, *where w_i is the multiset of objects occurring inside the i^{th} membrane;*
- $R = \{R_1, \ldots, R_n\}$, *where R_i is the finite set rules occurring inside i^{th} membrane; a rule is of the form $r : a \to b$, where a is a multiset over Σ and b is a string over $\Sigma \times \{in_i : 1 \leq i \leq n\}$*
- $V_\mu = \{v_1, \ldots, v_n\}$, *where $v_i \in \mathbb{R}^+$ is the volume of the i^{th} membrane;*
- $V_\Sigma = \{v_{s_1}, \ldots, v_{s_m}\}$, *where $v_{s_j} \in \mathbb{R}^+$, is the volume of object v_{s_j}.*

The definition of the values V_μ and V_Σ leads to the introduction of a further attribute, named *free space* and represented as F_i for the membrane i, that influences the evolution of Π. The quantity F_i indicates the amount of free space left within the membrane labelled i and is defined as:

$$F_i = v_i - \left(\sum_{j=1}^{m} (w_i(s_j) \cdot v_{s_j}) + \sum_{l=1}^{n} \alpha_l \cdot v_l \right) \tag{4.1}$$

where $\alpha_l \in \{0, 1\}$ takes the value 1 if the membrane labelled l is a son of membrane i in the membrane hierarchy, otherwise is 0, and $w_i(s_j)$ denotes the number of occurrences of the symbol s_j in the multiset w_i. Hence, the free space of membrane i is simply defined as the difference between its volume and both (i) the sum of the volumes of the objects it contains and (ii) the sum of the volumes of all the membranes that are both nested in and connected to membrane i as specified by the membrane structure μ.

Since membrane dissolution is not considered, the membrane structure is kept fixed during the system's evolution. Hence a *configuration* C of an SP system Π is described only by the multisets $w_1, \ldots, w_n$ of objects contained in its membranes and a computation step changes the configuration according to the following principles:

- the set of rules selected in order to be applied at each computation step must fulfil $F_i \geq 0$ for $1 \leq i \leq n$ calculated in the configuration potentially reached after their application;
- the rules are applied in a *maximally parallel* way: each object which appears on the left-hand side of applicable rules must be subject to exactly one of them; the only objects which remain unchanged are those associated with no rule, or with inapplicable rules;
- when more than one rule can be applied to an object, the actual rule to be applied is chosen *nondeterministically*; hence, a computation tree can be obtained starting from the initial configuration.

A (halting) computation $\mathcal{C}$ of an SP system Π is a sequence of configurations $(C_0, C_1, \ldots, C_k)$, where C_0 is the initial configuration of Π, every C_{i+1} can be reached from C_i according to the principles just described, and no further configuration can be reached from C_k (i.e., no rule can be applied). The output of Π is constituted by the objects expelled from the skin membrane.

4.2 SP systems and Turing machines

The evolution of SP systems is regulated by the availability of *free space*. In this section, we investigate whether the introduction of objects and membranes volumes affects the computational universality of P systems. In particular, we answer to the questions: do SP systems simulate a Turing Machine? If this is the case, how much amount of volume do SP systems require in order to exploit such task?

To answer to the previous questions, we show that a single tape DTM M having $\Sigma = \{0, 1\}$ as input alphabet and operating in time $f(n)$ and space $g(n)$ can be *efficiently* simulated (with a polynomial slow down) by a semi-uniform family of SP systems $\mathcal{F}(\Pi_M, w) = \{\Pi_{M,w} : w = s_1 s_2 \ldots s_n \in \{0, 1\}^*\}$. Thus, $\Pi_{M,w}$ must use $O(p_f(f(n)))$ time and $O(p_g(g(n))$ space, where p_f and p_g are two polynomial functions. Furthermore, we show that the *volume* occupied by the members of $\mathcal{F}$ is a polynomial p_v of the space required by the DTM and thus, $\Pi_{M,w}$ must use $O(p_v(g(n)))$ volume.

4.2.1 Simulation of a DTM with SP systems

To simulate a DTM with an SP system, the functioning of the DTM must be captured by the SP system without loss of computational power; moreover, for each possible input, the SP system must calculate the same output computed by the DTM on the same input.

We represent the tape by means of the membrane structure, i.e. we establish a one-to-one correspondence between the membranes and the cells of the DTM. To reproduce the linear order of the cells over the DTM's tape, the membranes can be organized in a nested way, where the leftmost cell can be represented by the outer membrane (or equivalently by the inner membrane). Each cell of the DTM contains a symbol belonging to the tape alphabet $\Gamma = \{0, 1, \sqcup\}$, where $\sqcup$ is the blank symbol. Similarly, each membrane will contain an object taken from the same alphabet plus two special objects "yes" and "no" which constitute the two possible answers to a decision problem. Hence, the membrane structure $\mu_{M,w}$ of an SP system $\Pi_{M,w}$ is made of $g(n)$ membranes, as this is the number of cells required by the DTM M to solve a problem of input w with length $|w| = n$.

We represent the current state of the DTM with a single object. We use the SP system rules to re-write this object in order to capture the variation of the DTM's state; the location of this "state" object inside the membrane structure reflects the position of the DTM's tape head.

The initial configuration of $\Pi_{M,w}$ will be:

$$\overbrace{\underbrace{[q_0 s_1\ [s_2\ [\ldots[s_n}_{n}\ [\ldots[}^{g(n)}\ \overbrace{\underbrace{]_i \ldots]_i\ ]_i \ldots]_i}_{n}\ ]_i\ ]_i}^{g(n)} \tag{4.2}$$

where $q_0 \in Q$ is the initial state and $i = \{1, \ldots, g(n)\}$ labels the membranes.

Finally, we use the rules of $\Pi_{M,w}$ to reproduce the DTM's transition function. We need to define a type of rule for each possible value $\{\leftarrow, -, \rightarrow\}$ of d, thus according to the tape head movement. For each transition of the DTM, represented by the generic quintuple $t = (a, q_1, b, q_2, d)$ and denoted by $\delta(a, q_1) \rightarrow (b, q_2, d)$, we have:

1. if $d = \{\leftarrow\}$ then the rule must be

$$[aq_1] \rightarrow q_2[b] \tag{4.3}$$

   the rule rewrites a in b and q_1 in q_2; q_2 is communicated in the membrane surrounding the current one, thus capturing the left movement of the tape head;

2. if $d = \{-\}$ then the rule must be

$$[aq_1] \rightarrow [bq_2] \tag{4.4}$$

the rule rewrites a in b and q_1 in q_2; as the tape head does not move the "state" object is maintained in the current membrane;

3. if $d = \{\rightarrow\}$ then the rule

$$aq_1[] \rightarrow b[q_2] \tag{4.5}$$

rewrites a in b and q_1 in q_2; q_2 is communicated in the membrane nested in the current one, in order to simulate the movement of the tape head on the right of the current cell.

Lastly, the result of the computation is sent out from the membrane structure. If M enters an accepting state q_A, the head/state symbol is replaced by *yes* and expelled:

$$[q_A]_h \rightarrow [\ ]_h \text{ yes} \tag{4.6}$$
$$[\text{yes}]_h \rightarrow [\ ]_h \text{ yes} \tag{4.7}$$
$$[\text{yes}]_{h_0} \rightarrow [\ ]_{h_0} \text{ yes}. \tag{4.8}$$

Similarly, the following three rules occur when M enters a rejecting state q_R:

$$[q_R]_h \rightarrow [\ ]_h \text{ no} \tag{4.9}$$
$$[\text{no}]_h \rightarrow []_h \text{ no} \tag{4.10}$$
$$[\text{no}]_{h_0} \rightarrow [\ ]_{h_0} \text{ no}. \tag{4.11}$$

4.2.2 Computational universality of SP systems

To enunciate the theorem concerning the computational universality of a family of SP systems with efficient volume occupation, we firstly define the DTM variant we consider.

Definition 4.2.1. *A single tape DTM operating in time $f(n)$ and space $g(n)$ is a tuple*

$$M = (Q, \Sigma, \Gamma, \delta, q_0, A, R) \tag{4.12}$$

where:

- *Q is a finite set of states;*
- *$\Sigma = \{0, 1\}$ is the input alphabet;*
- *Γ is the tape alphabet, a finite superset of Σ;*
- *$\delta : \Gamma \times Q \rightarrow \Gamma \times Q \times \{\leftarrow, -, \rightarrow\}$ is the transition function; it assumed that δ is undefined on both accepting and rejecting states;*

- $\{yes\} \subseteq Q$ *is the accepting state;*
- $\{no\} \subseteq Q$ *is the rejecting state;*

To simulate a given DTM M with an SP system we consider the so called *semi-uniform* setting, in which we assume the existence of a DTM M' that, for every pair (M, w) with $w \in \{0,1\}^n$, simulates M acting over w. More precisely:

Definition 4.2.2. *A family of SP systems* $\mathcal{F}(\Pi, w) = \{\Pi_w : w \in \Sigma^*\}$ *is* semi-uniform *when the mapping* $w \mapsto \Pi_w$ *can be computed in polynomial time by a DTM.*

As according to Def. 4.2.2 the mapping has to be computed in polynomial time by a DTM, we have to prove that this operation is possible, but before we have to define formally an SP system that simulates a DTM on the input w.

Definition 4.2.3. $\Pi_{M,w}$ *simulates* M *on the input* w *and is defined as:*

$$\Pi_{M,w} = (\Sigma, \mu, W, R, V_\mu, V_\Sigma)$$

- $\Sigma = \{0, 1, \sqcup, yes, no, q_1, \ldots, q_{|Q|-2}\}$;
- $\mu = \{\mu_0, \ldots, \mu_{g(n)}\}$;
- $W = \{w_1 = \{q_0 s_1\}, w_2 = \{s_2\}, \ldots, w_n = \{s_n\}, w_{n+1} = \{\emptyset\}, \ldots, w_{g(n)} = \{\emptyset\}\}$ *where* $q_0 \in Q$ *represents the initial state of* M;
- $R = \{R_1, \ldots, R_{g(n)}\}$ *where each* R_i *contains*
  1. *a copy of the rules defined in Equations 4.3-4.5 for each possible transition* $t : \Gamma \times Q \times \Gamma \times Q \times \{\leftarrow, -, \rightarrow\}$;
  2. *a copy of the rules defined in Equations 4.6-4.11 for each accepting/rejecting state;*
- $V_\mu = \{v_1, \ldots, v_n\}$;
- $V_\Sigma = \{v_0, v_1, v_\sqcup, v_{yes}, v_{no}, v_{q_1}, \ldots, v_{q_{|Q|-2}}\}$ *where* $v_j \in \mathbb{R}^+$, $j \in \Sigma$ *and* $\hat{v} = max\{v_j\}$.

Theorem 4.2.1. *Let us consider a family of SP systems* $\mathcal{F}(\Pi, w) = \{\Pi_{M,w} : w \in \Sigma^*\}$ *composed by SP systems* $\Pi_{M,w}$ *that simulate a given DTM* M *acting over* w. *This family is semi-uniform.*

Proof. The membrane structure of an SP systems $\Pi_{M,w} \in \mathcal{F}$ consists of $g(n)$ membranes and can be constructed in $O(g(n))$ time steps, as g is time-constructible by hypothesis. The initial configuration of $\Pi_{M,w}$ can be constructed in linear time from w as exactly n symbols have to be placed

inside the outermost membranes $\mu_1, \ldots, \mu_n$. Lastly, the set of rules only depends on M, and not on w. Since $g(n)$ is bounded by a polynomial, the construction of $\Pi_{M,w}$ is semi-uniform. □

While for the usual time and the space complexity definitions we remind the reader to Appendix A, we introduce here the concept of *volume complexity*, a specific feature of SP systems.

Definition 4.2.4. *Let Π be an SP system. The* volume complexity *of Π is the function $p : \mathbb{N} \mapsto \mathbb{R}^+$ where $p(n)$ is the size (volume) of the outermost membrane that Π requires on any input of length n.*

We are ready to enunciate the theorem concerning the simulation of M with the family of SP systems $\mathcal{F}$.

Theorem 4.2.2. *$\mathcal{F} = (\Pi, M, w)$ decides the same language as M and its members $\Pi_{M,w}$ operate in $O(f(n))$ time, $O(g(n))$ space and occupy a volume $O(g(n))$.*

Proof. Each SP system $\Pi_{M,w}$ consists of $g(n)$ membranes and therefore will contain at most $g(n) + 1$ objects; hence, $\Pi_{M,w}$ clearly uses $O(g(n))$ space. Each transition of M on input w is simulated by $\Pi_{M,w}$ in a single step; once that the result object $\{\text{yes}, \text{no}\}$ is produced, at most $g(n)$ steps are required to communicate $\{\text{yes}, \text{no}\}$ to the outermost membrane, plus a further step to exit the outermost membrane. Hence we obtain a total time of $f(n) + g(n)$, which is bounded by $O(f(n))$.

The volume of each membrane has to be sufficient to contain the object representing the DTM cell symbol, the state object and the nested membranes. Therefore, the volume of the membrane representing the rightmost cell, labelled $g(n)$, is equal to $2\hat{v}$. Consequently, the size of the membrane with label $i = g(n) - 1$ is $v_{\mu_i} = v_{\mu_{g(n)}} + 2\hat{v} = 4\hat{v}$. Repeating the operation for all the membranes we get the volume occupied by membrane 1, which is by definition the volume complexity of $\Pi_{M,w}$ and is equal to $v_{\mu_1} = 2\hat{v}g(n)$. Thus, the volume required by $\Pi_{M,w}$ to decide the same languages as M is $O(g(n))$. □

4.3 Sτ-DPPs: the integration of SP systems, tP systems and τ-DPPs

In the previous Section we proved that SP Systems, like many other classes of membrane systems, are computationally universal and this result can be achieved with an efficient volume occupation. In this Section, we present a new variant of membrane systems, designated as *Sτ-DPP*, that integrates the volume occupation introduced in SP systems with other properties taken

from tP systems [73], and τ-DPP [27]. These features have been considered in order to define a flexible and powerful tool for the modelling of biological systems, mainly, but not limited to, at the mesoscopic level, i.e. population of molecules [26].

The membrane structure of Sτ-DPP shares with tP systems the membrane structure, i.e. nodes are arranged in a tissue–like fashion. Moreover, we enabled nodes with a complex internal hierarchy, organised in a tree–like structure. Therefore, the topology of the membranes of an Sτ-DPP is a mixed graph in which the nodes are the membranes, the undirected edges indicate that the two membranes are placed on the same level (as in the first definition of tP systems) and the directed edges denote that the membrane represented by the tail node is nested in the membrane represented by the head node. This strategy for the representation of membrane structure allows to define complex structures in which nodes can be hierarchically organised in a tree-like structure.

As in SP systems, objects and membranes are associated to real numbers that define their volume. In the context of mesoscopic biological systems modelling, these sizes represent the amount of volume occupied by the molecules (objects) and compartments (membranes), respectively, in the three-dimensional space. It follows that the evolution of the system is influenced by the free space available within the compartments: as in SP systems, also in Sτ-DPP rule application must fulfil the free space rule (Equation 4.1).

The evolution of the Sτ-DPP is described by a modified version of the simulation approach used in τ-DPP. Hence, the simulation of an Sτ-DPP is based on a multi-compartmental tau-leaping algorithm which provides a *quantitative* description of system's dynamics. We updated this procedure in order to avoid the unlimited accumulation of objects in a region of finite size due to the mutual impenetrability of molecules.

The communication channels among membranes are represented by a direct graph. The arrows of the edges indicate the direction of the (permitted) flow of objects between different compartments. Note that, this communication graph can contain edges that are not indicated in the graph which describes the topology of the membranes. The meaning of these particular edges is to represent communication channels that connect non adjacent membranes. The explicit representation of the communication channels and the possibility to have communications between non-adjacent membranes extend the modelling power of this membrane systems variant. For example, using these arcs it is possible to create privileged pathways of communication between membranes.

4.3.1 Definition

Formally, Sτ-DPPs are defined as follows.

Definition 4.3.1. *An Sτ-DPP of degree n is a construct:*

$$\Pi = (\Sigma, G_\mu, G_c, C, W, R, V_\mu, V_\Sigma)$$

- $\Sigma = \{s_1, \ldots, s_m\}$ *is a finite set of symbols, also called objects;*
- $G_\mu = (\mu, E, A_\mu)$ *is a mixed graph representing the topological arrangement of the membranes $\mu = \{1, \ldots, n\}$, (E, A_μ) are, respectively, the set of edges and the set of arrows which describe the topology of membranes;*
- $G_c = (\mu, A_c)$ *is a directed graph representing the connections (channels of communication) among the membranes $\mu = \{1, \ldots, n\}$ and A_c is the set of the arrows which describe the available connections;*
- $W = \{w_1, \ldots, w_n\}$, *where w_i is the multisets of objects occurring inside the i^{th} membrane;*
- $\mathcal{C} = \{C_1, \ldots, C_n\}$, *where C_i is the set of stochastic constants $c_{i,j} \in \mathbb{R}^+$ associated to the rules occurring inside the i^{th} membrane;*
- $V_\mu = \{v_1, \ldots, v_n\}$, *where $v_i \in \mathbb{R}^+$ is the volume of the i^{th} membrane;*
- $V_\Sigma = \{v_{s_1}, \ldots, v_{s_m}\}$, *where $v_{s_j} \in \mathbb{R}^+$, is the volume of object v_{s_j}.*
- $R = \{R_1, \ldots, R_n\}$, *where R_i is the set rules occurring inside the i^{th} membrane; an* internal *rule is of the form*

$$\alpha_1 s_1 + \cdots + \alpha_m s_m \xrightarrow{c} \beta_1 s_1 + \ldots + \beta_m s_m \tag{4.13}$$

a communication *rule is of the form*

$$\begin{aligned} \alpha_1 s_1 + \ldots + \alpha_m s_m \xrightarrow{c} (\beta_{1,1} s_1 + \ldots + \beta_{m,1} s_m, in_1) + \ldots + \\ (\beta_{1,n} s_1 + \ldots + \beta_{m,n} s_m, in_n) \end{aligned} \tag{4.14}$$

where the quantities α_i and β_j are natural numbers, c is the stochastic constant and $in_1, \ldots, in_n$ indicate the target membrane to which the object is sent.

The evolution of Sτ-DPPs is computed by a stochastic algorithm (a modified version of the τ-DPP algorithm, see below) that describes the temporal evolution of the system; therefore, we talk about system state at a particular time point, rather than system configuration at a particular evolution step. The *state* of an Sτ-DPP at a given time point t is represented by the multisets of objects contained in its membranes: $W(t) = \{w_1(t), \ldots, w_n(t)\}$.

Note that rules are inspired to the notation used to represent chemical reactions: at the left side of the rule we have reactants, while at the right

part of the rule we have products. Each object is preceded by a number (called stoichiometric coefficient) indicating how many copies of that objects are involved in the rule. Moreover, there are two types of rules: (i) internal rules, in which both reactants and products belong to the same compartment; (ii) communication rules, in which reactants are sent to other compartments as the left side of the rule specifies.

The sets of stochastic constants $C_1, \ldots, C_n$, associated to the sets of rules $R_1, \ldots, R_n$, are required to compute the probabilities of the rule applications (also called propensity functions), along with a combinatorial function depending on the left-hand side of the rule [53].

Different instances of Sτ-DPPs can be grouped into families according to the values assumed by their *parameters*.

Definition 4.3.2. *Let Π be an Sτ-DPP; its set of parameters, named P, consists of:*

1. *the multiplicities of all objects appearing in the multisets $w_1, \ldots, w_n$ initially presents in the membranes $1, \ldots, n$;*
2. *the stochastic constants $C_1, \ldots, C_n$ associated to the rules in $R_1, \ldots, R_n$;*

Now, we can extend Definition 4.3.1 in order to consider a family of Sτ-DPPs where the members differ for the parameters.

Definition 4.3.3. *Let Π be a Sτ-DPP and $\mathcal{P}$ be a family of sets of parameters for Π. The family of Sτ-DPPs defined by Π and $\mathcal{P}$ is $\mathcal{F}(\Pi, \mathcal{P})$ and consists of all Sτ-DPPs with the main structure of Π and the set of parameters $P \in \mathcal{P}$; such a Sτ-DPP is denoted by (Π, P).*

A family $\mathcal{F}(\Pi, \mathcal{P})$ constitutes a general model for the real system of interest (e.g. chemical, biological and metapopulation systems) and, for any choice of the parameters, we can investigate the evolution of the corresponding Sτ-DPP. Understanding how the evolution of a model is affected by its parameters is, in fact, of primary importance in many situations, like in the context of parameter estimation and sensitivity analysis (this topic is addressed in Chapter 6).

4.3.2 Time evolution of Sτ-DPPs

The temporal evolution of Sτ-DPPs is computed by a *modified version* of the τ-DPPs algorithm. We will not report here all the steps of the algorithm, but only the four most important modifications that we have introduced in the procedure described in Section 2.3. We remind the reader that each step is executed *independently* and *in parallel* within each membrane or compartment i, ($i \in \{1, \ldots, n\}$) of the system. In the following description, the order of the instructions, is referred to the algorithm presented in Section 2.3:

1b. calculate the free space at $t_0 = 0$;

$$F_i(t_0) = v_i - \left(\sum_{j=1}^{m} (w_i(s_j, t_0) \cdot v_{s_j}) + \sum_{l=1}^{n} \alpha_l \cdot v_i \right) \tag{4.15}$$

where $w_i(s_j, t_0)$ is the number of occurrences of s_j in the multiset w_i at time t_0;

2 `for each` rule r_k, ($k \in \{1, \ldots, l\}$): compute the propensity function

$$a_k := \begin{cases} c \cdot h; & \text{if } r_k \text{ is a first order reaction;} \\ c/F_i \cdot h & \text{else;} \end{cases} \tag{4.16}$$

20. `if` the execution of the selected rules (considering all the volumes) leads to an unfeasible state, namely (i) there is not enough space either inside the compartment i (for internal rules) or inside the target volumes j, $j \neq i$ (for communication rules) `or` (ii) there are negative amounts of molecules:

    - reduce τ_{min} by half;
    - send the new value to the other membranes;
    - goto *8*;

23b. update the value of the free space F_i;

25. `If` the termination criteria is satisfied, namely (i) the current time exceeds the end time `or` (ii) there is not enough free space in any membrane: finish; `else`: `goto` *2*.

First of all, we added the calculation of the free space during the initialisation of the system, *step 1b*. The second modification concerns the calculation of rule's propensity functions (*step 1b*) and is a consequence of the possible variability of the compartment volume. In fact to model the increase of reaction probability due to molecular crowding, the propensity functions of the internal reactions are computed by also considering the value of the free space of the current compartment. So doing, we can correctly simulate crowded systems: we suppose that while first order reactions (e.g. $a \to b$) are not affected by the value of the free space, in the case of reactions of higher orders, the volume reduction enhances the probability of collisions. Therefore, the propensity functions of second and third order reactions are computed as:

$$a(\mathbf{x}) = \frac{c}{F_i} h \tag{4.17}$$

The third change regards the *step 20* in which the algorithm checks whether the application of the selected rules leads to an unfeasible state; we must

ensure that $F_i \geq 0$ for all the compartments $i \in \{1, \ldots, n\}$. The fourth modification occurs during the system update (*step 23*): we added the calculation of the free space inside every compartment. Lastly (*step 25*), we consider the case in which is not possible to apply any rule satisfying $F_i \geq 0$ as a further termination criteria for the algorithm.

The computational cost of the Sτ-DPP algorithm is still $2mn$, where m is the number of the rules and n the number of compartments. The algorithm is implemented in a parallel (MPI) version in C programming language. The parametrisation schema establishes a one-to-one relation between processes and membranes.

4.4 Modelling spatially heterogeneous systems with Sτ-DPPs

Living cells are very far from the homogeneous compartment that is often used for their modelling. In the case in which the studied biological process involves a time scale that is greater than the molecules diffusion time scale, the well-stirred approximation is justified [31]. If this is not the case, it is important to keep track of the amount of substance in different locations within the volume.

In this section we show how it is possible to exploit the Sτ-DPP[1] to simulate correctly and accurately the dynamics objects which undergo diffusion within different regions of the system.

4.4.1 Handling diffusive events with Sτ-DPP

A natural approach to describe the *space domain* Ω in which diffusion occurs with an Sτ-DPP Π of degree n is to map each compartment (membrane) of Π to a subregion of Ω. These subregions can be arranged according to all the topologies captured by the mixed graph G_μ which defines the connections among the compartments of Π; moreover, as each membrane can occupy an arbitrary amount of volume, subregions can have different sizes.

We describe the movement of particles using mono-molecular reactions and propensity functions in which the stochastic constant is proportional to the diffusion coefficient and in inverse proportion to the square of the compartment size (see Section 2.2). Therefore, to describe the movement of a particle s from a region i to a region j, we use a rule $r_{i,k}$ (belonging to the set of the rules R_i of region i, where k runs from 1 to the cardinality of R_i) of the form

$$s \to (s,\ \mathrm{in}_j) \tag{4.18}$$

[1] As the volume occupied by the molecules and by the compartments is not required to simulate diffusion, this study and its results are also valid for the τ-DPP algorithm.

and to calculate the propensity $a_{i,k}$ of this rule we use

$$a_{i,k} = c_{i,k} \cdot w_i(s,t) = \frac{D_{i\to j}}{d_i(|\tilde{d}_i - \tilde{d}_j|)} \cdot w(s,t) \tag{4.19}$$

where we recall that $D_{i\to j}$ is the diffusion coefficient of s at the boundary between the compartments i and j, d_i is size of region i, $|\tilde{d}_i - \tilde{d}_j|$ is the length between the centers of the regions i and j, respectively, and $w_i(s,t)$ denotes the copy number of s at time t in region i.

Boundary conditions specify the behaviour of the system for extreme values of the independent variables. We focus on boundary conditions for the space coordinate. To explain how it is possible to model these boundary conditions, let us consider a mono-dimensional space domain in the interval $\Omega = [0,1]$, that we divide in a number n of regions $1,\ldots,n$, and let us focus on its left boundary, where the spatial coordinate is $x = 0$; let the first region of the domain have index 1 and let a fictitious region on its left have index 0.

Dirichlet boundary conditions (or first type) define the value the solution (in our case the species number) has to take at the boundary. We model these boundary conditions as shown in [11]. Let us consider the Dirichlet condition

$$[s](x = 0, t) = \alpha \tag{4.20}$$

that sets the value of $[s]$ at the boundary $x = 0$ to α . The flux of particles between regions 0 and 1 will be simulated by the rules $r_{0,k}$ of region 0 and $r_{1,l}$ of region 1 (where k, l are labels for the rules inside the respective sets):

$$r_{0,k}: \; s \xrightarrow{c_{0,k}} (s, \quad \text{in}_1) \tag{4.21}$$

$$r_{1,l}: \; s \xrightarrow{c_{1,l}} (s, \quad (\text{in}_0). \tag{4.22}$$

The two constants $(c_{0,k}, c_{1,l})$ must be defined selecting a reasonable value for d_0, the length of the fictions compartment 0. For $d_0 = d_1$ we obtain [11]:

$$c_{0,k} = \frac{D_{(0\to1)} w_0(s,t)}{d_0 d_1} = \frac{D_{(0\to1)}\alpha}{d_1} \tag{4.23}$$

$$c_{1,l} = \frac{D_{(1\to0)}}{d_1^2}. \tag{4.24}$$

where $\alpha = w_0(s,t)/d_0$ is the desired boundary condition.

Neumann boundary conditions (or second type) define the value of the flux across the boundary. Let us consider the Neumann condition

$$\frac{\partial[s]}{\partial x}(x = 0, t) = \alpha \tag{4.25}$$

that sets the variation of $[s]$ respect to the spatial coordinate across the boundary at a value α. In this case we define the constants $c_{0,k}$ or $c_{1,l}$ of,

respectively, rules $r_{0,k}$ and $r_{1,l}$, as

$$\alpha > 0 : \ c_{0,k} = \frac{\alpha}{w_0(s,t)}, \ \ c_{1,l} = 0 \tag{4.26}$$

$$\alpha < 0 : \ c_{0,k} = 0, \ \ c_{1,l} = \frac{\alpha}{w_1(s,t)} \tag{4.27}$$

$$\alpha = 0 : \ c_{0,k} = c_{1,l} = 0. \tag{4.28}$$

4.4.2 Accuracy of the diffusion described with Sτ-DPPs

Berstein [11] showed that it is possible to simulate mesoscopic RD systems using the Gillespie's algorithm. We adopt the same strategy (essentially, a comparison with a diffusion equation) in order to show that Sτ-DPP can be used to reproduce diffusion introducing a reasonably small error.

4.4.2.1 A popular diffusion equation: the heat equation

The heat equation is a partial differential equation which describes the heat distribution in a region during time, and it is a special case of diffusion equation where the diffusion coefficient D is constant in time and space:

$$\frac{\partial u(\mathbf{x},t)}{\partial t} = D\Delta u(\mathbf{x},t) \tag{4.29}$$

where[2] $u(\mathbf{x},t)$ is the density of the diffusive material in $\mathbf{x}$ at time t.

To test the accuracy of Sτ-DPP in reproducing diffusion, we studied the uni-dimensional diffusion of the molecule s in the region $\Omega \subset \mathbb{R}$:

$$\frac{\partial [w]}{\partial t}(x,t) = D\frac{\partial^2}{\partial x^2}[s](x,t), \ \forall x \in \Omega \tag{4.30}$$

where $[w](x,t)$ indicates the concentration of molecule s in position x at time t. In particular, we considered the region $\Omega = [0,1]$ and the Neumann boundary conditions:

$$\frac{\partial [w]}{\partial x}(0,t) = \frac{\partial [w]}{\partial x}(1,t) = 0, \tag{4.31}$$

indicating that the flux from outside into Ω is null. Considering $D = 1$, an exact solution in the region Ω satisfying Equation 4.31 is:

$$[w](x,t) = w^{\Omega}[1 + \tfrac{1}{2}\mathrm{e}^{-\pi^2\gamma^2 t}\cos(\gamma\pi x)] \tag{4.32}$$

where w^{Ω} is the total number of s molecules inside the system and γ is a non negative integer. For all the simulations that we will discuss in the next following we have considered $\gamma = 3$ and $w^{\Omega} = 500$.

[2] $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$ is the Laplacian operator.

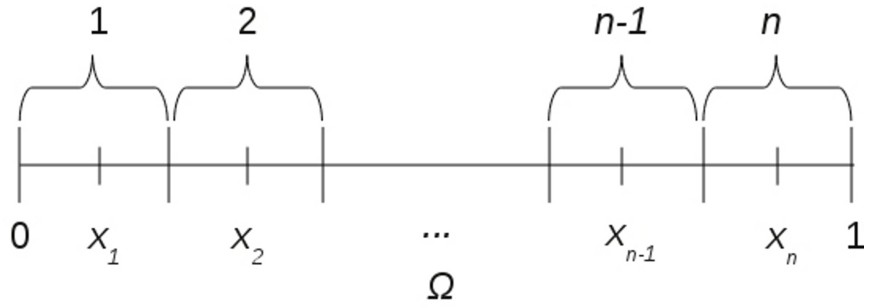


Figure 4.1: Representation of the unidimensional space domain $\Omega = [0, 1]$ and its subdivision in n compartments belonging to the membrane structure μ. We considered locations $x_1, \ldots, x_n$ during the comparison with the solution of Equation 4.32.

4.4.2.2 Comparison between Sτ-DPP and the Heat Equation

In order to compare the simulations performed by using Sτ-DPP with the continuous exact solution of the heat equation (Equation 4.32) we used the following Sτ-DPP:

$$\Pi = (\Sigma, G_\mu, G_c, C, W, R, V_\mu, V_\Sigma)$$

- $\Sigma = \{s\}$;
- $G_\mu = (\{1, \ldots, n\}, E_\mu = \{(k, l) : k \in \{1, \ldots, n-1\}, l = k+1\}, \{\emptyset\})$;
- $G_c = (\{1, \ldots, n\}, A_c = \{(k, l) \cup (l, k), \quad k \in \{1, \ldots, n-1\}, l = k+1\})$;
- $W = \{w_1, \ldots, w_n\}$;
- $\mathcal{C} = \{C_1, \ldots, C_n\}$;
- $V_\mu = \{v_1, \ldots, v_n\}$;
- $V_\Sigma = \{0\}$;
- $R = \{R_1, \ldots, R_n\}$ where the rules in R_i of each compartment i define the diffusion (Equation 4.18) of s from membrane i to the neighbours of i as specified by A_c.

The model is composed by a set of compartments n that are the results of the division of the space domain Ω into n adjacent regions $1, \ldots, n$ (Figure 4.1). Regions $i \in 2, \ldots, n-1$ contain two rules in order to specify the diffusion of s into compartments $i-1$ and $i+1$. Conversely, regions 1 and n have only one rule describing the diffusion towards compartments 2 and $n-1$, respectively, and so doing we model the Neumann boundary conditions

in Equation 4.31. The values of the constants were defined considering $D = 1$ and the distance $1/n$ between two adjacent compartments:

$$c_{i,j} = \frac{1}{1/n^2} = n^2 \tag{4.33}$$

where j runs over the number of rules of each region i. The multisets were initialised to obtain a total number of molecules equal to 500:

$$w_i = s^{(500/n)}. \tag{4.34}$$

A series of issues have to be handled to realize a meaningful comparison between the dynamics of Π and Equation 4.32. First, since Sτ-DPP algorithm is stochastic, it is crucial to consider a high number of simulations in order to obtain a significant comparison with the heat equation. We accomplished this task averaging the results of a sufficiently high number of simulations. Note that, in general, this average is not representative of the final state of a system, like in the case of multistable systems, in which averaging may lead to fictitious states. However, when the average solution converges to the system state – as in the case we are considering here – the deviations from the exact solution can be considered as a type of sampling error and the average solution is a good representative of the system state.

Second, since Sτ-DPP simulator works with molecules, rather than molecule concentration, we must calculate the initial distribution of s molecules $w_1(t_0), \ldots, w_n(t_0)$ (inside regions $1, \ldots, n$) from the solution of Equation 4.32 at time $t = 0$, in order to use this distribution as input for Sτ-DPP. Moreover, we must calculate Sτ-DPP predicted concentrations, $[w_1^*](t), \ldots, [w_n^*](t)$, from the multisets $w_1(t), \ldots, w_n(t)$ at a particular time point t. In both cases we have to consider that Equation 4.32 has been defined over a uni-dimensional space domain. These conversions have been defined according to the following relation between concentration $[w_i]$ and molecule number w_i:

$$[w_i](t) = \frac{w_i(t)}{1/n}, \ \forall i \in \{1, \ldots, n\} \tag{4.35}$$

Note that the solution of Equation 4.32 must be calculated using the appropriate vector $\mathbf{x} = (x_1, \ldots, x_n)$, whose members are located at the middle of each compartment i:

$$x_i = \frac{1}{2n} + (i-1)\frac{1}{n}, \ \forall x_i \in \mathbf{x} \tag{4.36}$$

Third, since the time increments τ are randomly generated, it is very unlikely that the simulator will output a numerical solution exactly at a specific time point t. Therefore, the molecule distribution computed by Sτ-DPP simulator at a particular time point t has been calculated as a linear interpolation of the two numerical solutions at $t_1 < t$ and $t_2 > t$, where t_1

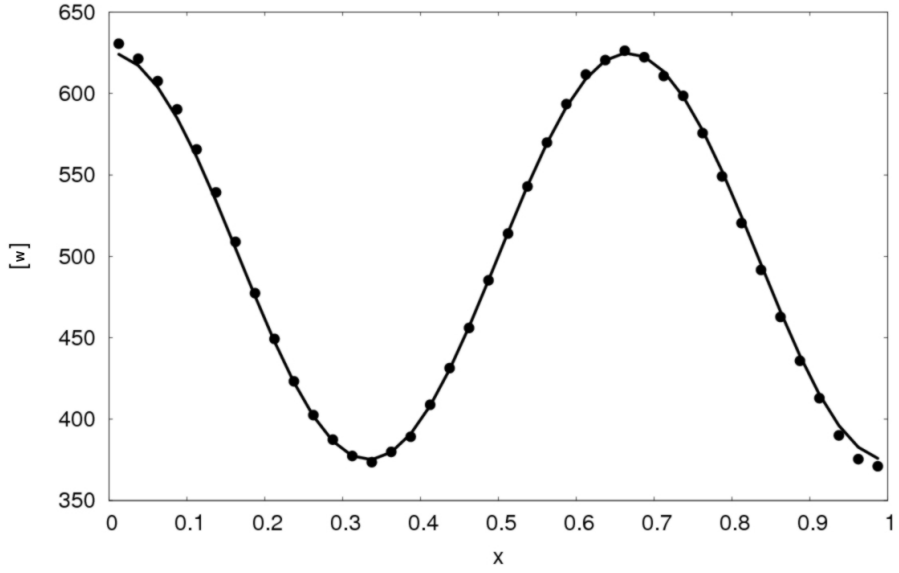


Figure 4.2: The heat equation exact solution (line) and Sτ-DPP average results (dots) at $t = 0.0078034$, $S^{\Omega} = 500$, $N = 10000$, $\gamma = 3$, $v_i = 0.025$.

and t_2 are, respectively, the points computed by the Sτ-DPP algorithm that immediately precede and follow t. An example of comparison between the heat equation and the Sτ-DPP simulations is shown in Figure 4.2, where it is possible to observe the high degree of closeness between the simulations and exact solution.

Quantitatively, the quality of the Sτ-DPP simulation $e = 1, \ldots, N$ has been assessed considering, as in [11], the error due to the difference between the exact solutions $[w_{i,e}]$ and the concentrations computed using the numerical results of Sτ-DPP $[w^*_{i,e}]$:

$$\epsilon_i = \frac{1}{N} \sum_{e=1}^{N} (1 - \frac{[w^*_{i,e}]}{[w_i]}) \tag{4.37}$$

in the compartment i, considering a pool of N simulations ran with the same settings. Note that $\epsilon_i \to 0$ as $[w^*_{i,e}] \to [w_{i,e}]$ (obviously) and in the case in which the distribution of $[w^*_{i,e}]$ is symmetric with respect to the exact value $[w_i]$. The errors ϵ_i have been averaged considering all the elements of Ω:

$$\bar{\epsilon} = \frac{1}{n} \sum_{i=1}^{n} \mid \epsilon_i \mid \tag{4.38}$$

We studied the relation of $\bar{\epsilon}$ with the number of simulations $10 \leq N \leq 10000$ and the number of compartments $10 \leq n \leq 40$ used to discretise the spatial region Ω.

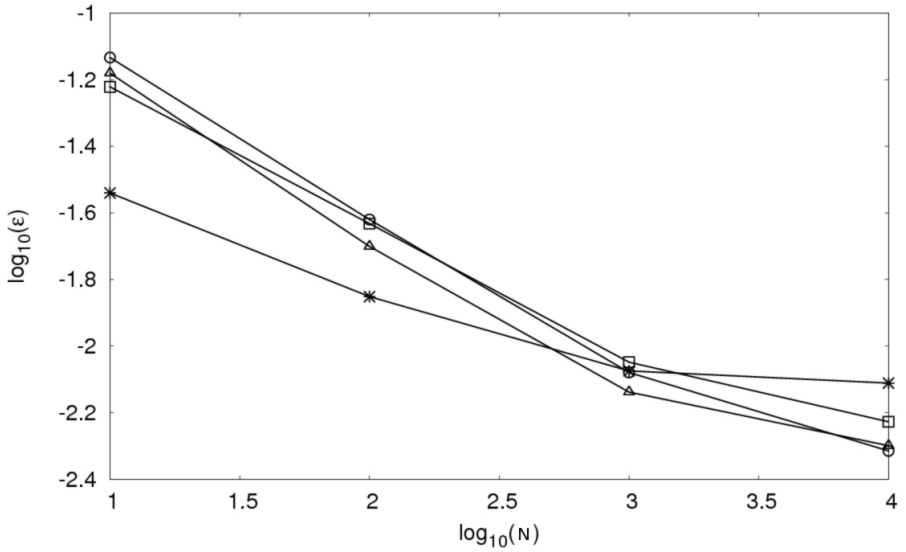


Figure 4.3: Relation between the error ϵ and the number of simulations (N). $v_i = 0.1$ (*), $v_i = 0.05$ (□), $v_i = 0.03\overline{3}$ (△), $v_i = 0.025$ (○), $\gamma = 3$, $t = 0.0078034$.

As the number of simulations increases the sampling error decreases (as shown in Figure 4.3). In particular, we observed a decrease of one order of magnitude passing from 10 to 10^4 simulations in all cases with exception of $n = 10$, where the decrease has been lower. Note that as $N \to 0$ the lowest error is associated to settings with lower n (bigger compartments), while as $N \to \infty$ the lowest error is associated to higher n (smaller compartments). This observation can be attributed to the noise: as $n \to \infty$, the compartments will contain a lower number of molecules (high noise); hence, a higher number of simulations is required to eliminate noise. This phenomenon has been particularly evident in the study presented here due to the relatively low quantity of molecules used, $w^{\Omega} = 500$, with respect to the number of compartments for the discretisation of Ω, $10 \leq n \leq 40$: so doing, we passed from a range of $10 - 100$ molecules/compartment for $n = 10$ to a range of $1 - 10$ molecules/compartment for $n = 40$.

Another source of error is associated with the spatial discretisation, i.e. with the number of membranes in which Ω is divided into. In order to reduce the contribution of the sampling error it is crucial to study the behaviour of the spatial discretisation error with a high N. This error decreases as $v_i \to 0$ (Figure 4.4).

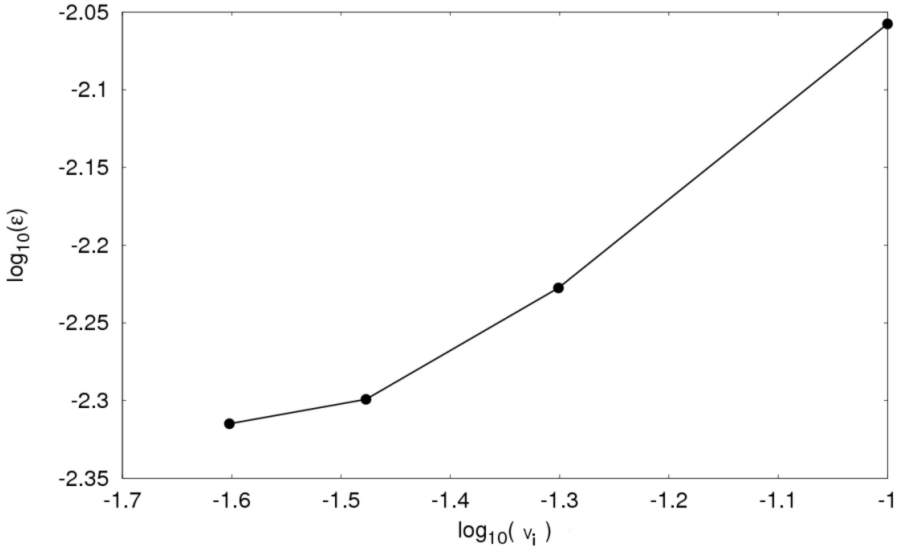


Figure 4.4: Relation between the spatial discretisation error and the compartment size at time $t = 0.0078034$.

4.5 Modelling molecular crowding effects with Sτ-DPPs

The intracellular environment is characterised by the presence of high concentrations of soluble and insoluble macromolecules [47, 80, 106]. This medium is termed "crowded", "confined" or "volume-occupied", rather than "concentrated", because single molecular species may occur at low concentrations, but all species taken together occupy a considerable fraction of the total volume [81].

The term "macromolecular crowding" refers to the non-specific influence of steric repulsions (i.e., a consequence of the mutual impenetrability of molecules due to the Pauli exclusion principle) on molecular processes that occur in highly volume-occupied media [91].

Due to macromolecular crowding, biochemical, biophysical, and physiological processes in living cells may be quite different from those under idealized conditions [105], and order-of-magnitude effects of crowding have been demonstrated by both experimental and theoretical works on a broad range of processes [91]. All these effects are related to variations occurring in macromolecular thermodynamics activities [105] and diffusion [40].

$$\langle r^2 \rangle = 6Dt^{\alpha} \tag{4.39}$$

where, if $\alpha < 1$, the diffusion is called *anomalous subdiffusion*; on the other hand, if $\alpha > 1$ the diffusion is called *anomalous superdiffusion*; if

$\alpha = 1$ the diffusion is normal. Crowding can reduce the rate of diffusion (according to the size of the diffusing molecule and to the degree of volume occupancy) and can lead to anomalous diffusion [8]. Large reductions in solute diffusion are probably indicators of interactions between the solute and cellular components, such as membranes [37]. Therefore, the rates of diffusion-controlled biochemical processes – mainly affected by the diffusion of the reactants – will be reduced in crowded media. The decrease in the diffusion rates due to crowding may also lead to complex phenomena like fractal kinetics (anomalous reaction orders and time-dependent reaction rate coefficients [65]) and spatial segregation of molecules [13].

In this Section, we describe two models as use cases to show how it is possible to capture capture some effects of macromolecular crowding with Sτ-DPPs. The first model concerns particle diffusion, while the second model focuses on reaction rates.

4.5.1 Anomalous diffusion

To model the diffusion of a particle in a crowded environment we considered the following Sτ-DPP of degree 441:

$$\Pi = (\Sigma, G_\mu, G_c, C, W, R, V_\mu, V_\Sigma)$$

- $\Sigma = \{s_1, s_2\}$;
- $G_\mu = (\{1, \ldots, 441\}, E_\mu, \{\emptyset\})$ where the set of edges E_μ defines a 21-by-21 square lattice with the membranes $\{1, \ldots, 441\}$ (Figure 4.5);
- $G_c = (\{1, \ldots, 441\}, A_c)$ where A_c is defined in order to let each membrane $\{1, \ldots, 441\}$ communicate with all its neighbours (Figure 4.5);
- $W = \{w_1, \ldots, w_n\}$;
- $\mathcal{C} = \{C_1, \ldots, C_{441}\}$;
- $V_\mu = \{v_1 = \ldots = v_{441} = 15625\}$;
- $V_\Sigma = \{216, 15625\}$;
- $R = \{R_1, \ldots, R_{441}\}$ where the rules R_i of each membrane i define the diffusion (see Equation 4.18) of s_1 from membrane i to the neighbours of i as specified by A_c.

The system is composed of 441 compartments organised as a 21-by-21 lattice. Each compartment is a cube with a volume equal to 15625nm^3, which reflects a side length of 25nm. We define two objects: s_1 represents a generic

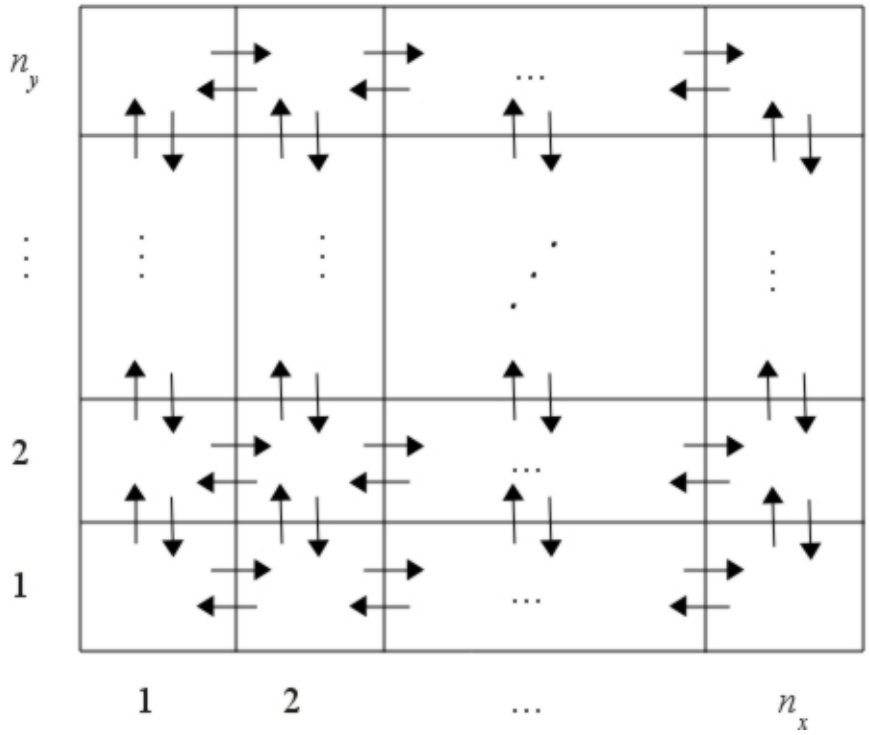


Figure 4.5: Subdivision of a bidimensional space domain in a set of compartments (membranes) according to the topology of an n_x-by-n_y lattice. The arrows indicate the communications allowed between adjacent compartments.

protein of volume 216nm^3 while s_2 is an immobile obstacle of size equal to the size of a single compartment. We set the stochastic constants values to

$$c_{i,j} = 7040\text{s}^{-1} = \frac{4.4\mu\text{m}^2\text{s}^{-1}}{(25 \cdot 10^{-3}\mu\text{m})^2} \tag{4.40}$$

where the diffusion coefficient of $4.4\mu\text{m}^2\text{s}^{-1}$ is a reasonable value for a protein of size 216nm^3 [42]. We modelled macromolecular crowding using a population of s_2 randomly placed within the compartments in order to obtain a volume occupation due to s_2 of approximately $\frac{1}{3}$ of the entire system volume, as this is the typical proportion of volume occupied by macromolecules in a living cell [40].

Before presenting the simulation results we have to introduce some useful concepts. We define the trajectory Υ of s_1 as:

$$\Upsilon = \left\{\mathbf{r}(t_0), \ldots, \mathbf{r}(t_{N_t})\right\} \tag{4.41}$$

where $\mathbf{r}(t) = (x, y)_t$ is the position of the protein s_1 at time t expressed considering the coordinates (x, y) and N_t is the number of time instants of a simulation over the time interval $\{t_0, \ldots, t_{N_t}\}$. The trajectory of s_i is

required to calculate its *mean squared displacement*, a measure of the average distance a molecule travels. It is defined

$$\mathrm{msd}(\tau) = \langle \Delta \mathbf{r}^{\,2}(\tau) \rangle = \left\langle \left(\mathbf{r_j}(t+\tau) - \mathbf{r_j}(t) \right)^2 \right\rangle \tag{4.42}$$

where $\mathbf{r_j}(t+\tau) - \mathbf{r_j}(t)$ is the displacement, the (vector) distance between the initial and final position of a molecule due to its motion during the time interval $(t, t+\tau]$ and $\langle \ldots \rangle$ indicates a time-average over t and/or an ensemble-average over several trajectories. We calculate the displacement as the Euclidean distance between the two positions

$$\mathbf{r_j}(t+\tau) - \mathbf{r_j}(t) = \sqrt{(x_{t+\tau} - x_t)^2 + (y_{t+\tau} - y_t)^2)}. \tag{4.43}$$

We simulated the random walk of one molecule s_1 in the space domain composed by the compartments $\{1, \ldots, n\}$, initialising the simulations with s_1 located at the centre of the grid, i.e. $w_{220}(t_0) = \{s_1^1\}$. We carried out 100 simulations in the time interval $[0, 0.01]$ in diluted conditions and 100 simulations initialising the system with crowding agents s_2.

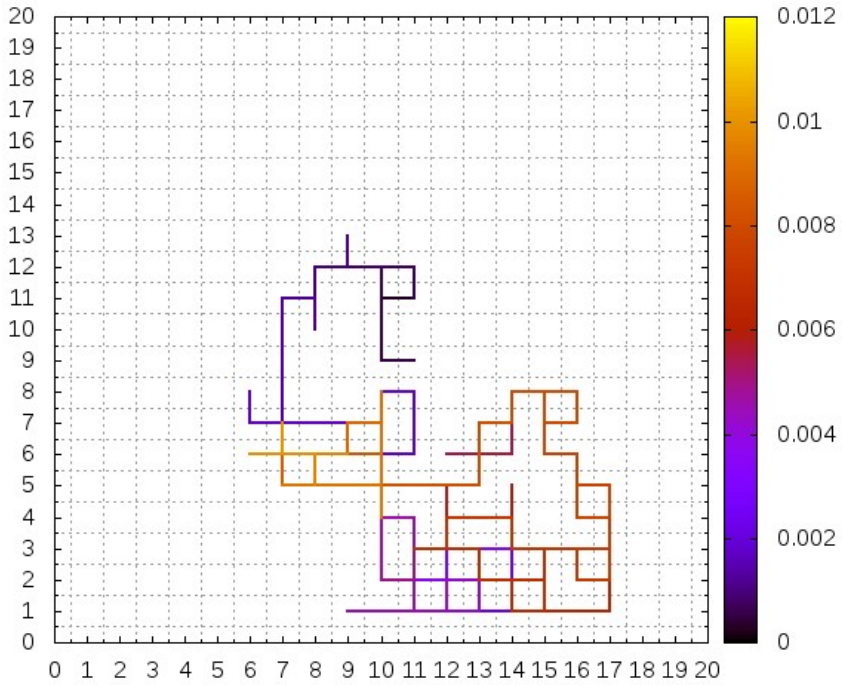
The simulations show a decrease of the particle travelled distance due to the addiction of crowding objects. It is possible to observe this result both considering a single trajectory (Figure 4.6) and the msd values (Figure 4.8). We calculated the msd values dividing each trajectory into a set of displacements, obtained using the same time interval τ. Hence, for a single trajectory, we calculated the msd performing a time-average over the simulated time and considering a specific τ. Then, we calculated the ensemble average considering several trajectories. The longer the trajectory and the smaller the τ, the more displacements can be calculated and, hence, the more accurate will be the estimation of the msd; conversely, as the value of τ increases, the msd values calculated from each single trajectory show a significant statistical uncertainty, Figure 4.7.

4.5.2 Increased reaction probability

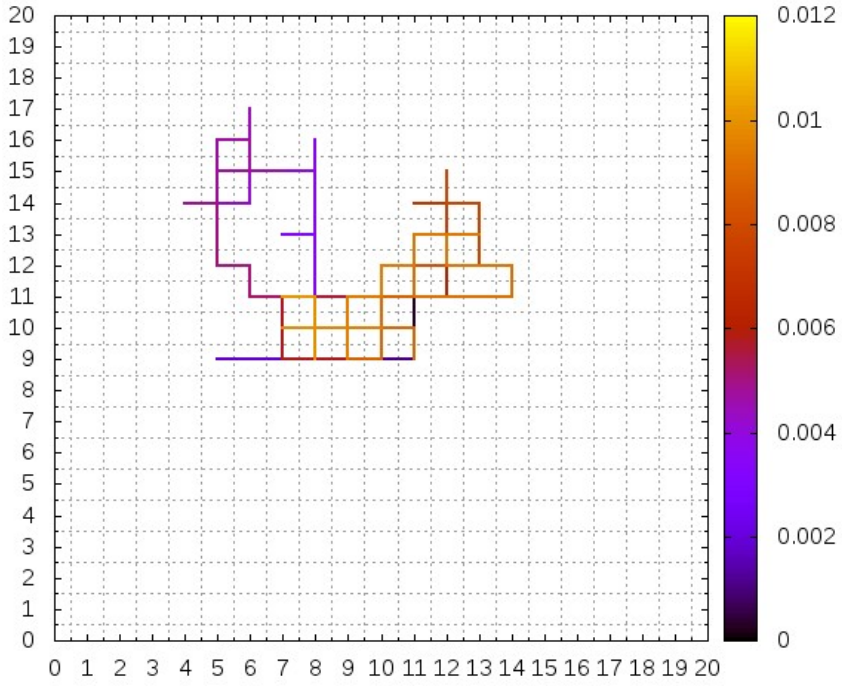
To model the increase of reaction probability due to molecular crowding we considered the following Sτ-DPP:

$$\mathit{\Pi} = (\mathit{\Sigma}, G_\mu, G_c, C, W, R, V_\mu, V_\Sigma)$$

- $\Sigma = \{s_1, s_2, s_3, s_4\}$;
- $G_\mu = (\{1, \ldots, 81\}, E_\mu, \{\emptyset\})$ where the set of edges defines a 9-by-9 square lattice (Figure 4.5);
- $G_c = (\{1, \ldots, 81\}, A_c)$ where A_c is defined in order to let each element communicate with each of its neighbours (Figure 4.5);

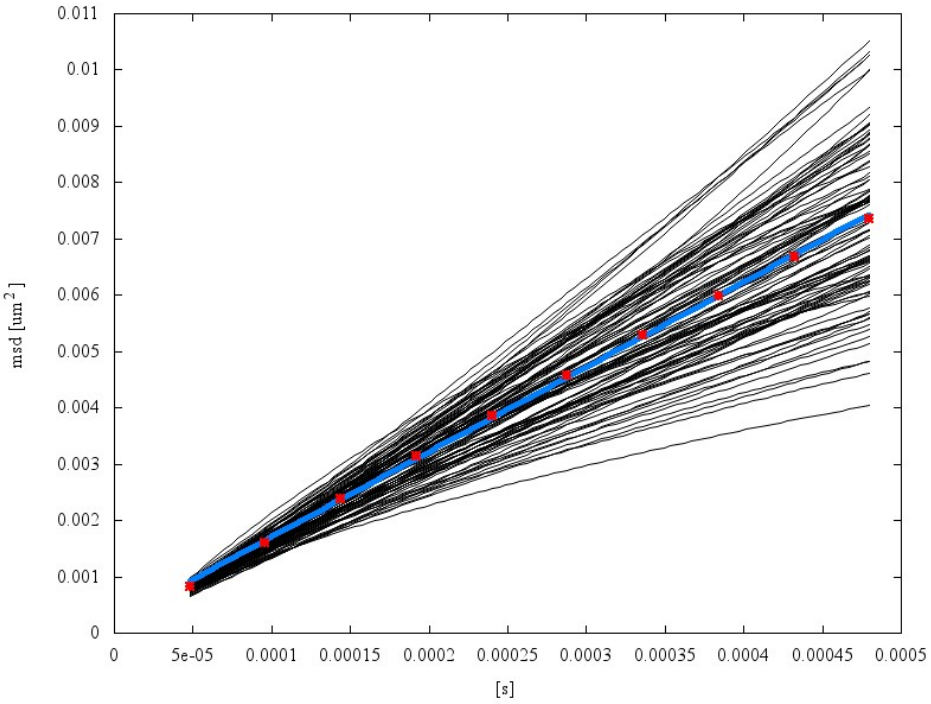


(a)

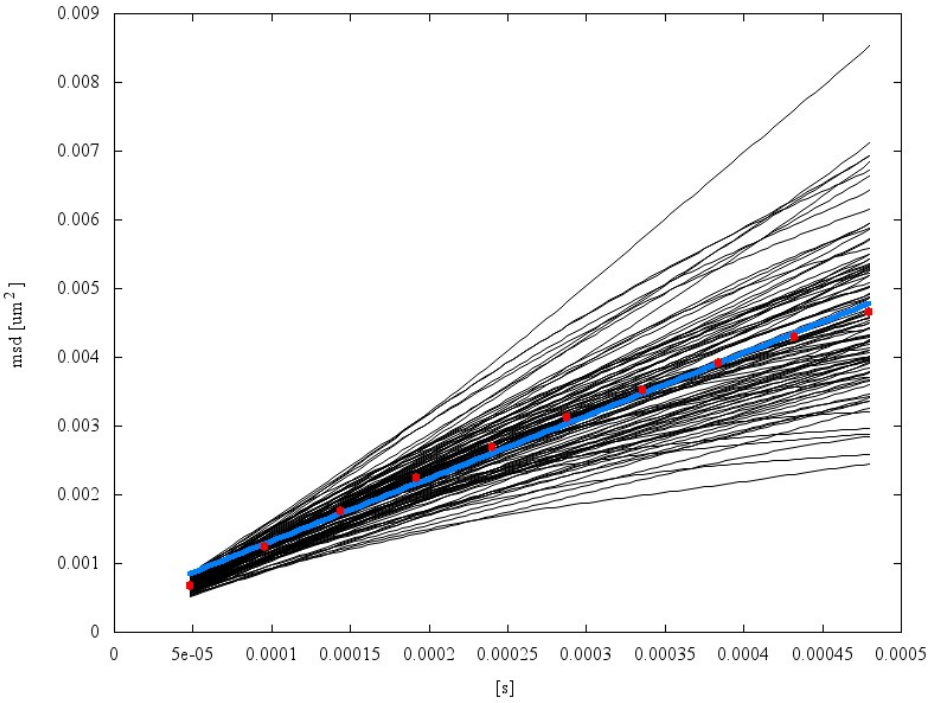


(b)

Figure 4.6: Trajectory of one particle s_1 in the 21x21 lattice in (a) diluted and (b) crowded conditions. In both cases, we placed s_1 at the same initial position (10,10,1). The colour of the trajectory represents the time coordinate value.



(a)



(b)

Figure 4.7: Mean squared displacements of 100 trajectories in (a) diluted and (b) crowded environment. Black curves are smooth fitting of the msd time averages calculated for different τ ; red points represent the msd ensemble averages for each τ value; the blue line is the fitting of the ensemble averages. Vertical axis: msd values; horizontal axis: τ values.

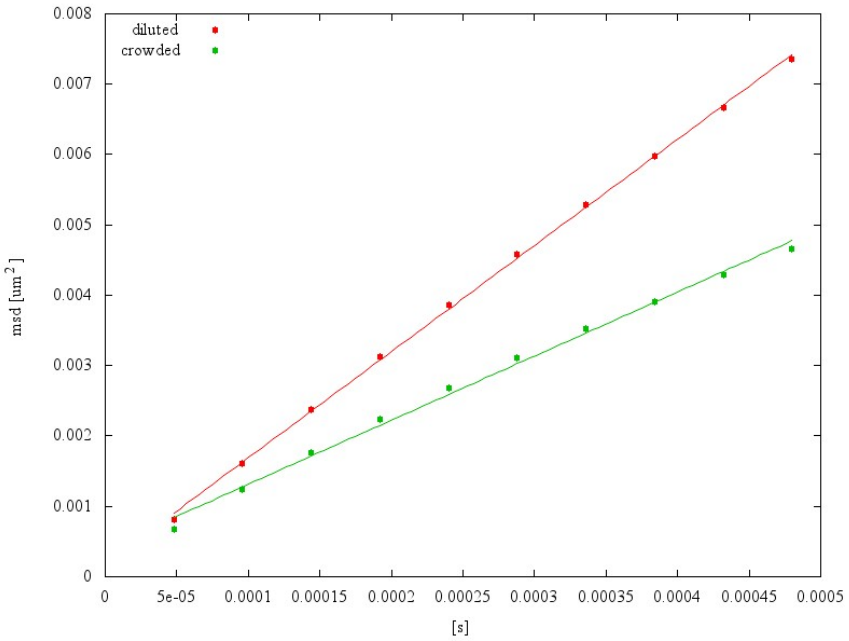


Figure 4.8: Mean squared displacement (points) of the particle s_1 calculated as ensemble and time averages of 100 simulations considering different τ values. Vertical axis: msd values; horizontal axis: τ values.

- $W = \{w_1, \ldots, w_n\}$;
- $\mathcal{C} = \{C_1, \ldots, C_{81}\}$;
- $V_\mu = \{v_1 = \ldots = v_{81} = 1\}$;
- $V_\Sigma = \{0.0001, 0.0001, 0.0002, 0.1\}$;
- $R = \{R_1, \ldots, R_{441}\}$ where each set R_i, $i \in 1, \ldots, n$ contains the two rules $r_{i,1} : s_1 + s_2 \xrightarrow{c_{i,1}} s_3$ and $r_{i,2} : s_3 \xrightarrow{c_{i,2}} s_1 + s_2$ and the diffusion rules $r_{i,k}$ (Equation 4.18) that specify the diffusion of $\{s_1, s_2, s_3\}$ from membrane i to the neighbours of i as specified by A_c;

The system is composed of 81 compartments organised as a 9-by-9-by-1 grid. Each compartment is a cube with a volume equal to 1, which reflects a side length of 0.001. The system contains four types of objects. The species s_1 and s_2 represent two generic proteins of volume 0.0001 which interact by means of a reversible process of association leading to the protein complex s_3

$$r_{i,1} : s_1 + s_2 \xrightarrow{c_{i,1}} s_3 \tag{4.44}$$

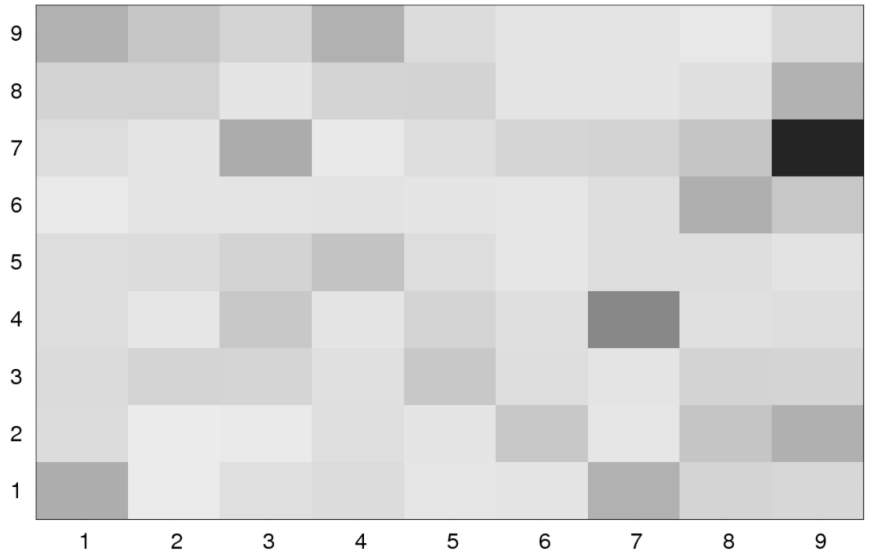
$$r_{i,2} : s_3 \xrightarrow{c_{i,2}} s_1 + s_2 \tag{4.45}$$

where $c_{i,1} = 0.0001$ and $c_{i,2} = 0.001$ are the stochastic constants. The species s_1, s_2 and s_3 can diffuse within the system. We set the stochastic constant associated to the diffusion rules $c_{i,k}$ three orders of magnitude higher than $c_{j,1}$ and $c_{j,2}$ in order to guarantee that diffusion processes are faster than reactions (see Section 2.2). The species s_4 represents a motionless crowding macromolecule with a volume equal to 0.1, a value which is three orders of magnitudes bigger than that of the other species.

We carried out simulations initialising the system with 100 molecules of s_1 and 100 molecules of s_2 in each compartment. Conversely, we chose the number of s_4 in order to occupy 0 (diluted media) or approximately $\frac{1}{3}$ of the total volume (crowded media). In the latter case the molecules s_4 have been randomly distributed among the compartments. For each region, we calculated the time average of the values assumed by the propensity function concerning the process of association of s_1 and s_2 to s_3, Equation 4.44.

In the diluted condition the time average of the reaction propensity values assumes approximately the same value within all the volumes: the mean value of all the averages is equal to 0.9237 and standard deviation is 0.001842 (Figure 4.9(a)). In the crowded system we report an increase of the reaction propensity time averages up to 6 folds compared with the diluted case. As this increase is the consequence of the presence of one or more crowding agents in a volume, we also obtained that the distribution of the mean value of all the averages is heterogeneous, Figure 4.9(b). In this case the mean and the standard deviation are 1.451 and 0.498 respectively. Therefore, the crowded systems is characterized by highly reactive regions in which the production of molecules s_3 is faster, Figure 4.10.

(a)



(b)

Figure 4.9: Average propensity function values of reaction r_1 within the spatial domain in (a) diluted and (b) crowded conditions. The darker the colour the higher the average propensity value.

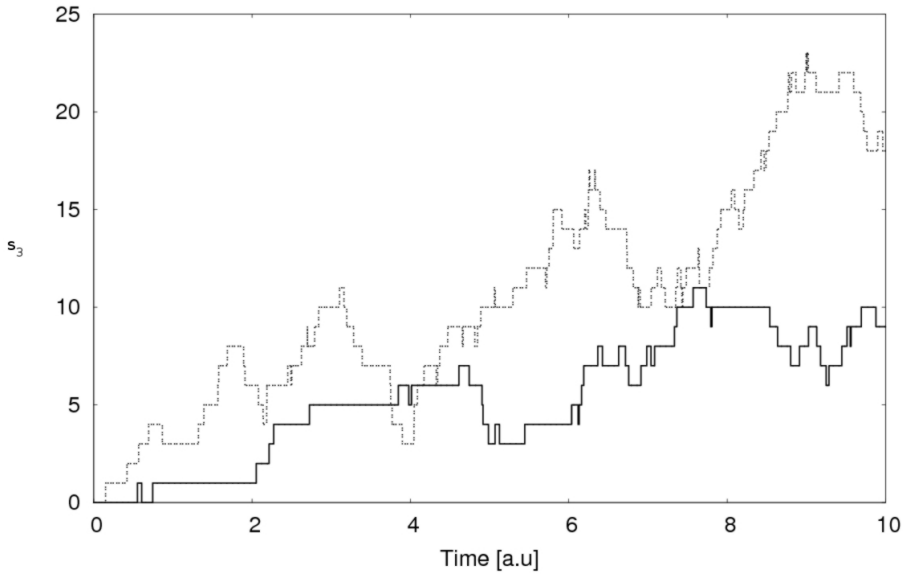


Figure 4.10: Number of s_3 molecules in the volume 79 (row 7 and column 9 of Figure 4.9) in diluted (solid line) and crowded (dashed line) conditions.

4.6 Modelling structured spaces with Sτ-DPPs

In this section we present a model to illustrate the enhanced capabilities of Sτ-DPPs due to the presence of two independent graphs for the definition of object communication and membrane structure. In particular, we show how the *communication between non-adjacent membranes* enables the modelling of preferential communication paths, a solution used by living cells for the regulation of molecules and macromolecules inside the intracellular environment.

In fact, living cells possess specific structures as *microtubules* and the *Golgi apparatus* that play an essential role for the accurate regulation of the movements of molecular species inside the crowded intracellular environment. Microtubules are a component of the cytoscheleon and are involved in many biological processes, including the vesicular and molecule transport, cytokinesis and mitosis. The Golgi apparatus is a stack of flattened compartments where molecules are packaged for delivery to other cell compartments or from secretion from the cell. Despite these two structures accomplish different tasks and have a different structure both of them ensure the correct delivery of different types of entities (from molecules to entire vesicles) to their final destination.

In the following, we will focus on an a qualitative model that capture the role of a microtubule for the transport of cargos inside the intracellular environment. However, with some minor changes concerning the communication possibilities the model can also be used for the Golgi apparatus.

4.6.1 A system with a preferential communication path

To model the movements of objects in a region characterised by the presence of a preferential communication path, we consider the following Sτ-DPP

$$\Pi = (\Sigma, G_\mu, G_c, C, W, R, V_\mu, V_\Sigma)$$

- $\Sigma = \{s_1, s_2\}$;
- $G_\mu = (\{1, \ldots, 8\}\ , \{(1,2),(2,4),\ (4,6),(6,8)\},\ \{(2,3),\ (4,5),\ (6,7)\})$;
- $G_c = (\{1, \ldots, 8\}, A_c)$ where A_c is defined in order to let each element communicate with its neighbours as illustrated in Figure 4.11;
- $W = \{w_1, \ldots, w_8\}$;
- $\mathcal{C} = \{C_1, \ldots, C_8\}$;
- $V_\mu = \{10^3, 10^3, 10^2, 10^3, 10^2, 10^3, 10^2, 10^3\}$;
- $V_\Sigma = \{1, 1\}$;
- $R = \{R_1, \ldots, R_8\}$ where the rules R_i of each membrane i define the diffusion (see Equation 4.18) of s_1 and s_2 to from membrane i to the neighbours of i as specified by A_c.

The model describes the flow of species s_1 and s_2 inside a set of compartments $\{1, \ldots, 8\}$ that contain a preferential communication path represented by $\{3, 5, 7\}$, Figure 4.11(a). Regions 1 and 8 are separated by three intermediate regions, 2, 4 and 6; each one of these compartments contains one further region. Collectively, these nested compartments (3,5 and 7) represent the region surrounding a microtubule, that it is assumed to be placed inside regions $\{3, 5, 7\}$. Even if both species s_1 and s_2 are allowed to enter these regions, particle s_1 moves only towards the region 8, in consequence of the interaction with the microtubule. Therefore, the flow of s_1 through the system is encouraged by means of this added flux, that is due to the presence of a microtubule localised in regions $3, 5, 7$.

We set the volume of the compartments $\{1, 2, 4, 6, 8\}$ to 10^3 while we consider the regions surrounding the microtubule one order of magnitude smaller (10^2). Objects size and all the stochastic constants are set to 1.

We ran a set of simulations to study the dynamics of s_1 and s_2 in relation to their flow towards the region 8. We initialised the simulations considering an equal number of molecules s_1 and s_2 placed in region 1, while we left the other regions empty. The (closed) system reaches a chemical equilibrium characterised by a different spatial distribution of molecules s_1 and s_2. More precisely, the microtubule establishes an increasing gradient of s_1 molecule number from region 1 to region 8; oppositely, the distribution of species

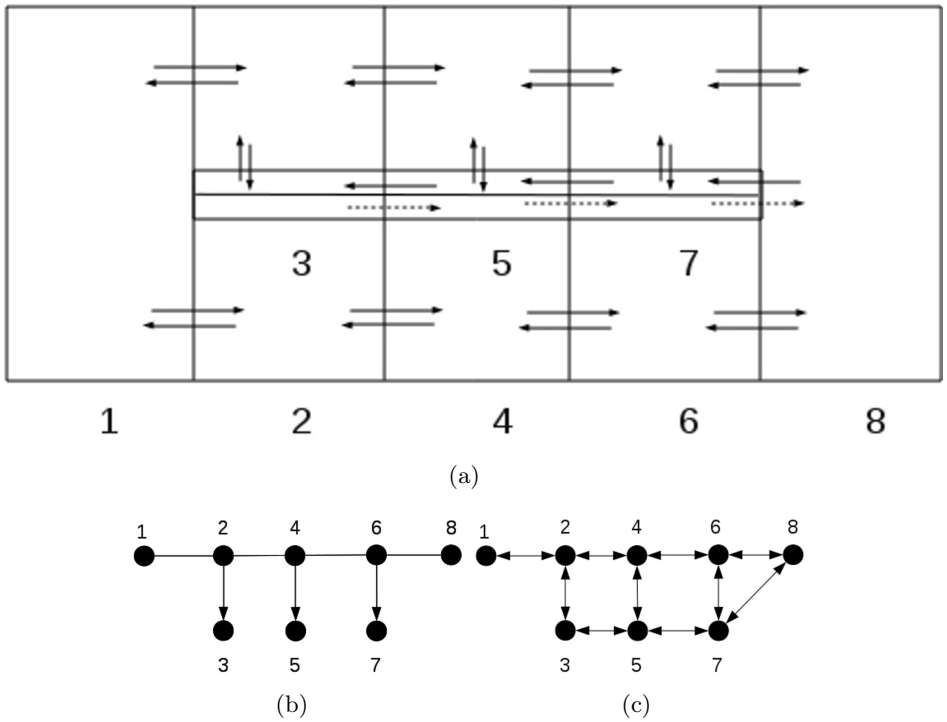
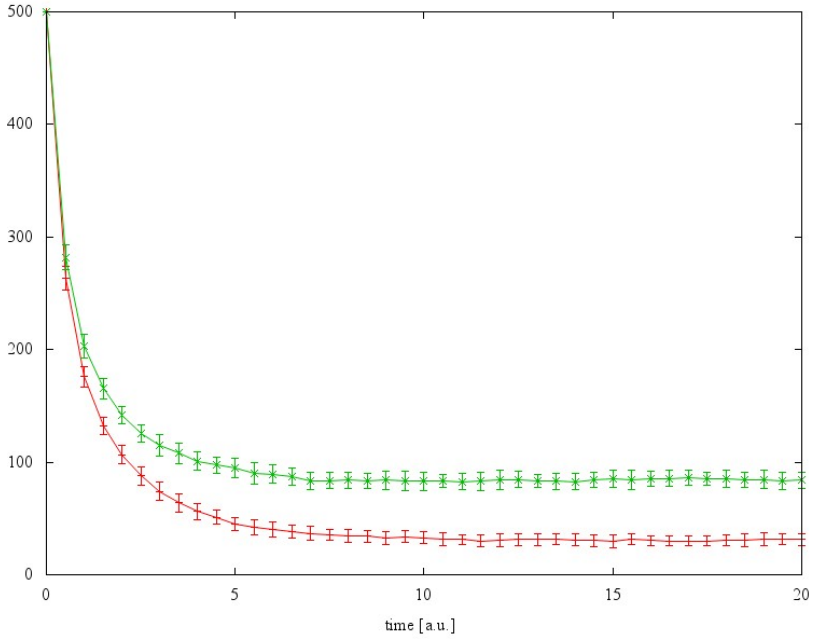
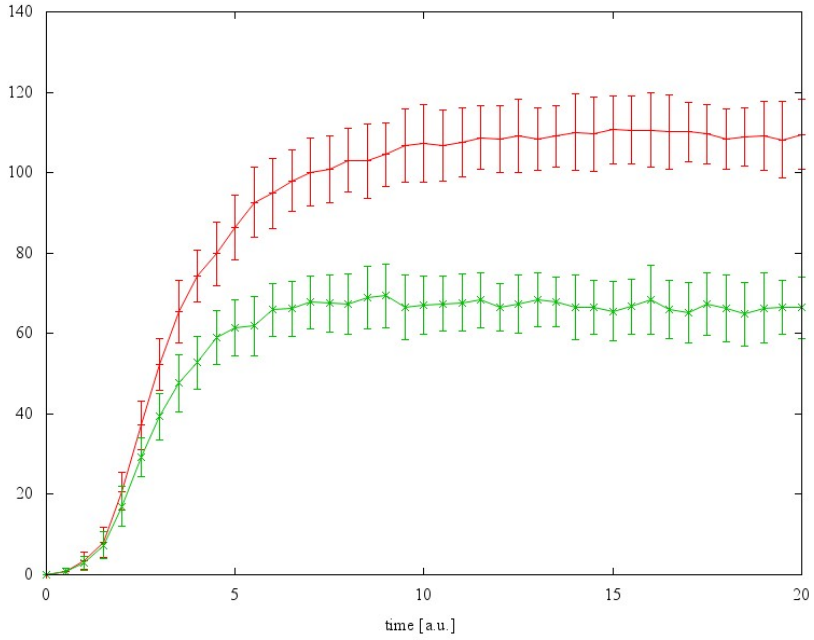


Figure 4.11: A membrane systems of degree 8 with a preferential communication path. a) Graphical representation, in which the arrows indicate the possible movements for all species (black arrows) and for species allowed to use the preferential communication path (dashed arrows); b) Mixed graph that defines the system's structure. c) Directed graph that defines the system's communication possibilities.

s_2 follows a descendant gradient from region 1 to region 8, Figure 4.13. Moreover, species s_1 arrives quickly in region 8, Figure 4.12.



(a)



(b)

Figure 4.12: Average number of molecules s_1 (red) and s_2 (green) during the time interval $[0, 20]$. (a) regions 1, (b) region 8 . We carried out 50 simulations initialising the system with 500 molecules of s_1 and 500 molecules of s_2 all placed in region 1. The values used to calculate averages (points) and standard deviations (bars) were computed interpolating the original trajectory described by the Sτ-DPP algorithm every 0.5 time units.

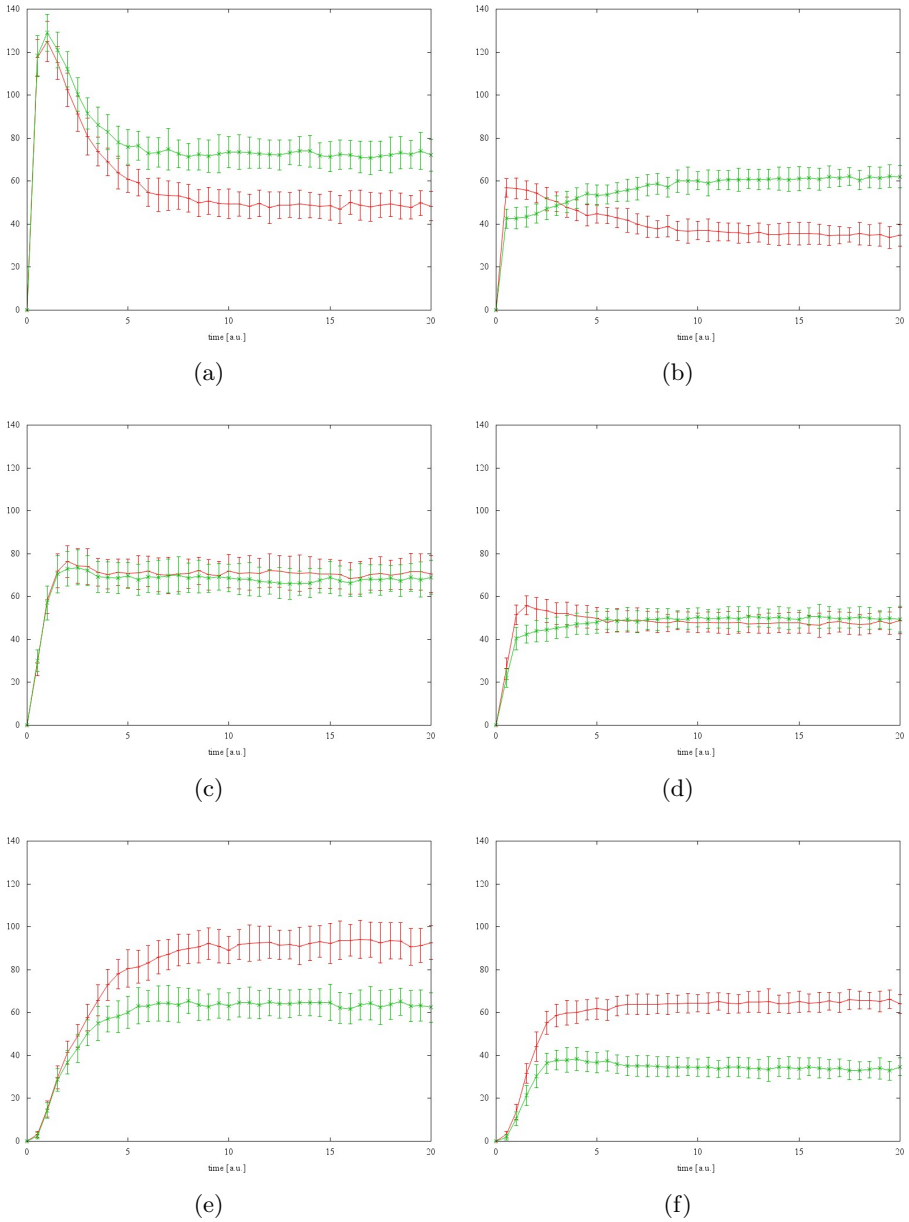


Figure 4.13: Average number of molecules s_1 (red) and s_2 (green) during the time interval $[0, 20]$. Panels (a-f) represent in the lexicographic order results for the regions $1, \ldots, 6$. We carry out 50 simulations initialising the system with 500 molecules of s_1 and 500 molecules of s_2 placed in region 1. The values used to calculate averages (points) and standard deviations (bars) were computed interpolating the original trajectory described by the Sτ-DPP algorithm every 0.5 time units.

4.7 Discussion

In this chapter we introduced SP systems, a novel variant of membrane systems, where objects and membranes occupy a finite amount of volume in a space with an arbitrary number of dimensions. We showed that SP systems are computationally universal (simulate a deterministic Turing machine) and we proved that this task is achieved with a polynomial slowdown, requiring the same space as the Turing machine and occupying an amount of volume which is a linear function of the space required by the Turing machine. SP systems are a computing device closer then classic membrane systems to a living cell, as SP systems consider the mutual impenetrability of objects (as molecules in the real world). The computational universality of SP systems further supports the idea (expressed in [90]) that living cells are a "powerful computer".

Subsequently, we introduced the Sτ-DPP considering properties of SP systems, tP systems and τ-DPP. The novel properties of Sτ-DPP consist in the representation of the membranes structure and the communication within the system with two distinct directed graphs, the possibility to define tissue-like structure where nodes have a complex internal architecture, the association of an amount of occupied volume to objects and membranes, and the consequent handling of the free space during the system evolution with a new version of the τ-DPP simulation technique.

The introduction of the new properties enables the formalism to be used to model a larger number of real systems. We presented three test cases to illustrate the capabilities of Sτ-DPPs.

The other class of applications enabled by the Sτ-DPP approach concerns the modelling of systems where the free space within different regions of the space domain is a critical resource for the system dynamics. We explored this feature modelling the molecular crowding effects over cellular dynamics. As the study of molecular crowding implies the modelling of particles diffusion inside a space domain, we investigated the accuracy of Sτ-DPP in reproducing particle diffusion. We showed that Sτ-DPP can reproduce the diffusion of molecules within a space domain divided into a set of sub-compartments (membranes) according to a defined topology. We tested diffusion using the same strategy followed by Bernstein [11], i.e. we compared the Sτ-DPP simulations with an analytical solution of a PDE for the heat equation (a diffusion equation). Note that for most applications of real interest biological processes analytical solutions are hardly available. The error that we reported is slightly greater than the one found in [11]. This is due to the fact that the τ-leaping algorithm – which stands at the basis of Sτ-DPP – generates an approximated dynamics with respect to the exact solution of the chemical master equation, whereas the Gillespie's algorithm used in [11] is exact. We think that this loss of accuracy (that can be a priori controlled) is well balanced by the increase in performance (tau leaping is faster than SSA)

that enables simulations of more complex systems compared to the SSA. The quantitative characterisation of this discrepancy will be further investigated in future.

Molecular crowding can be explicitly modelled by the Sτ-DPP variant. We studied two quantitative models, concerning particle diffusion and biochemical reactions in a bi-dimensional space domain with a degree of crowding similar to the one observed in real cells ($\frac{1}{3}$ of the total volume). In both case we modelled the crowding using objects with a volume occupation much higher than the one of the molecular species under investigation.

Considering the first model, we showed that Sτ-DPP simulations capture the decrease of the diffusion rate due to the presence of immobile obstacles inside a bidimensional space domain divided in a set of regions. The second model concerns the study the variation of the reaction rate due to the increased recollision probability determined by the presence of crowding objects. The Sτ-DPP simulation technique captures this effect due to the modification of the propensity function calculation of bi- and three-molecular reactions. The simulations showed that the crowded medium is characterised by an heterogeneous distribution of reaction probability within the space domain.

The use of two distinct graphs for describing the membranes structure and the communication within the system provides a formalism with a strong expressive power: indeed, it is possible to have communication channels between membranes that are not adjacent and, conversely, it is possible that adjacent membranes do not communicate. The first possibility allows the creation of preferential paths of communication and we presented a model to demonstrate the use of this feature to reproduce the role of microtubules in living cells.

Considering the results provided in this Chapter, Sτ-DPPs can be used to model and simulate crowded reaction and diffusion (RD) systems. We illustrated that, the current version of Sτ-DPPs capture some of the major effects that a crowded medium determines over RD system dynamics; however, we plan to do an extensive study of this topic in future. Moreover, an interesting direction of investigation, enabled by the possibility of arbitrarily defining the volume occupation of both objects and compartments, and the topology of their communication, consists in the use of Sτ-DPPs to study RD systems dynamics in structured spaces.

5

Electrostatic properties in membrane systems

In this chapter we describe a further extension of the Sτ-DPPs, (designated as ESτ-DPPs) in order to consider two electrical properties, namely the electric charge (for objects) and the electric potential difference (for a pair of membranes). This work is motivated by the crucial role played by the membrane potential difference in regulating many biological processes. The membrane potential, also referred to as transmembrane potential, is an electric potential difference (or voltage difference) between the two compartments separated by a membrane. This voltage difference arises from a different ion concentrations between two compartments sustained by active transport (in this context the word "active" indicates a process sustained by the consumption of energy) of ions realised by specific membrane proteins. Cells use membrane potential (i) as a battery that can be used to activate down stream processes and (ii) to transmit signals. More precisely, we consider two main effects of the membrane potential difference over the intracellular dynamics: (i) diffusion of charged particles and (ii) protein conformational changes.

First, we describe the derivation of two novel classes of propensity functions to model the two effects of the membrane potential delineated above. Then, we define formally ESτ-DPPs and afterwards we describe the algorithm for the temporal evolution of ESτ-DPPs. Subsequently, we study two test cases to show the functioning of the novel types of propensity functions. We conclude discussing the results we have achieved.

5.1 Modelling charged particle diffusion

In the context of living cells, particle diffusion is not only due to a concentration gradient. In fact, the flux of a charged molecule s is driven by both the concentration gradient $\nabla[s]$, where $[s]$ is the concentration of species s,

and the electric field $\mathbf{E} = -\nabla\phi$, where ϕ is the electric potential[1]. In this Section we present a novel form of propensity function in order to describe the diffusion of charged particles. We begin introducing the equations that we will use as a starting point to derive the propensities.

5.1.1 Diffusion due to electrostatic forces

A concentration gradient determines diffusion, generating a flux $\mathbf{J_D}$ according to the Fick's law (see Section 2.2). The electric field establishes an electric force which accelerates the charges species s; however, frictional forces in aqueous solution counteract the electric force and therefore s reaches a drift velocity where frictional and electric forces are equal and opposite. The flux due to drift, also referred to as *electrophoresis*, is:

$$\mathbf{J_e} = -u_s[s]z_s\nabla\phi \tag{5.1}$$

where u_s is the electrical mobility, $[s]$ is the concentration, z is the valence and ϕ is the electric potential. The electrical mobility is related to the diffusion coefficient by the Einstein-Stokes relation:

$$D_s = \frac{u_s k_B T}{q} \tag{5.2}$$

where k_B is Boltzmann's constant, T is temperature and q is the elementary charge. Analogously to the Fick's law, the minus indicates that positively charged species have a positive flux if the potential gradient decreases and *vice versa* negatively charged particles have a positive flux if the potential gradient increases.

The fundamental insight provided by the Nernst-Plank equation, which extends the Fick's law of diffusion for the case where the diffusing particles are also moved with respect to the fluid by electrostatic forces, is that the two fluxes $\mathbf{J} = -D_s\nabla[s]$ and $\mathbf{J_e}$ are additive, and therefore the total flux of s is:

$$\mathbf{J} = \mathbf{J_D} + \mathbf{J_e} = -D_s\nabla[s] - u_s[s]z_s\nabla\phi. \tag{5.3}$$

At steady state condition, when the flux due to drift equals the flux due to the concentration gradient, $\mathbf{J_D} = \mathbf{J_e}$ we obtain:

$$D_s\nabla[s] = u_s[s]z_s\nabla\phi \tag{5.4}$$

Rearranging and integrating this equation it is possible to obtain the *Nernst potential*, or *reversal potential*. To illustrate this result, let us assume that

[1]The *del* operator represented with the *nabla* symbol ∇ is used to denote (considering the three dimensional Cartesian coordinate system) the *gradient* $\nabla f = (\partial f/\partial x\ \hat{\mathbf{i}}, \partial f/\partial y\ \hat{\mathbf{j}}, \partial f/\partial z\ \hat{\mathbf{k}})$ of a scalar field $f(x, y, z)$, and the *divergence* $\nabla \cdot \mathbf{v} = (\partial v_x/\partial x, \partial v_y/\partial y, \partial v_z/\partial z)$ of a vector field $\mathbf{v} = v_x\hat{\mathbf{i}} + v_y\hat{\mathbf{j}} + v_z\hat{\mathbf{k}}$.

$[s]_1$ and $[s]_2$ are the concentration of ion s, respectively, on side 1 and 2 of a permeable membrane. Then, the *Nernst potential* is the electric potential value at which the concentration and voltage gradients have values such that there is no net flow of ions s across the membrane and in this case will be:

$$E_s = \phi_2 - \phi_1 = \frac{RT}{zF} \ln \frac{[s]_1}{[s]_2} \tag{5.5}$$

where $R = k_B N_A$ is the gas constant, $F = qN_A$ is the Faraday constant and N_A is the Avogrado's number.

The Nernst equation is widely used in physiology for finding the electric potential of a cell membrane with respect to one type of ion. The Nernst-Plank equation (combined with the Poisson equation in the Poisson-Nernst-Plank theory) has a wide area of applications in physics, electrochemistry and biophysics. For example, many biological applications are related to the transport of ions through protein channels across membranes, such as [29, 83, 33] and more recently [84].

5.1.2 Propensity functions for charged particle diffusion

Now we show that an appropriate set of uni-molecular rules can be used to approximate the variation of the concentration of a particle s, as a consequence of its diffusion due to a flux $\mathbf{J_e}$ determined by the presence of an electric potential gradient $\nabla\phi$. For the sake of simplicity, we consider the mono-dimensional case $\nabla = \partial/\partial x$. The set of rules that we want to use are:

$$r_{i-1,1} : s \xrightarrow{c_{i-1,1}} (s,\ \mathrm{in}_i) \tag{5.6}$$

$$r_{i,1} : s \xrightarrow{c_{i,1}} (s,\ \mathrm{in}_{i-1}) \tag{5.7}$$

$$r_{i,2} : s \xrightarrow{c_{i,2}} (s,\ \mathrm{in}_{i+1}) \tag{5.8}$$

$$r_{i+1,1} : s \xrightarrow{c_{i+1,1}} (s,\ \mathrm{in}_i) \tag{5.9}$$

where $i-1$, i and $i+1$ are three consecutive regions over x of size d_{i-1},d_i and d_{i+1} respectively, $c_{j,k}$ is the constant labelled k and belonging to a rule of the region j to be used for the respective propensity function $a_{j,k}$ (we recall that the relation between the propensity function a and the stochastic constant c is $a = c \cdot h$, where h is the number of reactant molecules or the number of possible combinations among reactants molecules, see Section 2.2). Let $[s](x,t)$ be the concentration of the molecule s in the space coordinate x at time t. Considering only the flux due to electrostatic forces we have:

$$\frac{\partial [s]}{\partial t} = -\nabla \cdot \mathbf{J_e} = -\nabla \cdot (-u[s]z\nabla\phi) \tag{5.10}$$

Integrating Equation 5.10 over the element i, we obtain:

$$\frac{\partial w_i}{\partial t} = -\int_i \nabla \cdot \mathbf{J_e} = \mathbf{J_e}\left(\tilde{d}_i - \frac{d_i}{2}\right) - \mathbf{J_e}\left(\tilde{d}_i + \frac{d_i}{2}\right) \tag{5.11}$$

where w is the number of molecules s and $\tilde{d}_i$ denotes the centre of the region i. We approximate the gradient of ϕ over the left side of the element i according to the finite volume approximation approach as:

$$\nabla\phi \approx \frac{\phi_i - \phi_{i-1}}{\tilde{d}_i - \tilde{d}_{i-1}} \tag{5.12}$$

Considering the same approximation for the right boundary and using Equations 5.10-5.11, we have:

$$\frac{\partial w_1}{\partial t} = -\phi_i\left(\frac{\lambda_{[i\to(i-1)]}}{|\tilde{d}_i - \tilde{d}_{i-1}|} + \frac{\lambda_{[i\to(i+1)]}}{|\tilde{d}_{i+1} - \tilde{d}_i|}\right) + \phi_{i-1}\left(\frac{\lambda_{[(i-1)\to i]}}{|\tilde{d}_i - \tilde{d}_{i-1}|}\right) + \phi_{i+1}\left(\frac{\lambda_{[(i+1)\to i]}}{|\tilde{d}_{i+1} - \tilde{d}_i|}\right) \tag{5.13}$$

where $\lambda_{(j\to k)} = u_{(j\to k)} z w_{(j\to k)}/d_{(j\to k)}$ and $u_{(j\to k)}$ should be evaluated at the boundary between elements j and k, $[s] = w_{(j\to k)}/d_{(j\to k)}$ is the concentration of molecules calculated considering the size $d_{(j\to k)}$ across compartments j and k. This expression suggests the following form for the propensities:

$$a_{i-1,1} = \left(\frac{\lambda_{[(i-1)\to i]}}{|\tilde{d}_i - \tilde{d}_{i-1}|}\right)\phi_{i-1} \tag{5.14}$$

$$a_{i,1} = \left(\frac{\lambda_{[i\to(i-1)]}}{|\tilde{d}_i - \tilde{d}_{i-1}|}\right)\phi_i \tag{5.15}$$

$$a_{i,2} = \left(\frac{\lambda_{[i\to(i+1)}}{|\tilde{d}_{i+1} - \tilde{d}_i|}\right)\phi_i \tag{5.16}$$

$$a_{i+1,1} = \left(\frac{\lambda_{[(i+1)\to i}}{|\tilde{d}_{i+1} - \tilde{d}_i|}\right)\phi_{i+1} \tag{5.17}$$

Hence, recalling the "classic" propensity function expression ($a = c \cdot h$), we define a propensity function to model the diffusion (due to electrostatic forces) of a charged species s from the compartment j to the compartment k as:

$$a_j = c'_j(\phi_j) \cdot h_j = \frac{u_{(j\to k)} z}{d_{(j\to k)}|\tilde{d}_j - \tilde{d}_k|} \cdot \phi_j \cdot w_{(j\to k)} \tag{5.18}$$

where

$$c'_j(\phi_j) = \frac{u_{(j\to k)} z}{d_{(j\to k)}|\tilde{d}_j - \tilde{d}_k|} \cdot \phi_j \tag{5.19}$$

is an *electric potential dependent parameter* (correctly expressed in [1 / time]) and $h = w_{(j\to k)}$ is the amount of molecules of type s evaluated considering a

region defined across the boundary between the two compartments j and k. With a slight abuse of notation, we identify the constant part $c'_j(\phi_j)$ as c_j:

$$c'_j(\phi_j) = c_j \cdot \phi_j = \frac{u_{(j\to k)} z}{d_{(j\to k)} |\tilde{d}_j - \tilde{d}_k|} \cdot \phi_j \tag{5.20}$$

Moreover, it is possible to simplify Equation 5.18, considering a uniform grid, where $\tilde{d}_j - \tilde{d}_k = d_{(j\to k)} = d$:

$$a_j = c'_j(\phi_j) \cdot h_j = c_j \cdot \phi_j \cdot h_j = \frac{u_{(j\to k)} z}{d^2} \cdot \phi_j \cdot w_{(j\to k)}. \tag{5.21}$$

As the propensity function must assume positive values, the product between the two quantities z and ϕ_j must be positive. As a consequence, we cannot currently take into account positive and negative charged particles in the same system. Conversely, for a given type of charged particle (positive or negative) we can both model negative and positive electric potential differences between two regions, appropriately defining the electric potential in each one of the two regions. If on the one hand, this issue can represent a limit for the application of this propensity for the modelling of complex systems, on the other hand many important biological processes are regulated by a single type of ion (e.g. calcium ion as a second messenger in many signalling cascades) or by combination of ions with the same type of charge (e.g. the action potential in excitable cells is mainly due to sodium and potassium ions).

To distinguish the propensity that we derived in this Section from the "classic version", we designate the propensity for the probability of charged particle diffusion as propensities of *class II*, while we consider the classic propensities as *class I*.

5.1.3 Relation with the Nernst equation

Now let us consider once again the condition depicted in Equation 5.4 which leads to the Nernst equation, Equation 5.5. We show that the propensities we use to model the flux of a charged species considering both a concentration gradient and a electric potential gradient lead to the Nernst equation using an appropriate value for the quantity $w_{(j\to k)}$, the number of molecules defined considering a region across the boundary that separates the two compartments. To do so, let us consider a closed system structured in two compartments, 1 and 2 of the same size $v_1 = v_2 = v$, separated by a membrane and an ion A whose amount in 1 and 2 is respectively A_1 and A_2; moreover, let ϕ_1 and ϕ_2 be the electric potential of the two compartments. To model both the fluxes $\mathbf{J_D}$ and $\mathbf{J_e}$ between the two compartments we use

four communication rules

$$r_{1,1} : s \xrightarrow{c_{1,1}} (s,\ \mathrm{in}_2) \tag{5.22}$$

$$r_{1,2} : s \xrightarrow{c_{1,2}} (s,\ \mathrm{in}_2) \tag{5.23}$$

$$r_{2,1} : s \xrightarrow{c_{2,1}} (s,\ \mathrm{in}_1) \tag{5.24}$$

$$r_{2,1} : s \xrightarrow{c_{2,2}} (s,\ \mathrm{in}_1) \tag{5.25}$$

which are associated to the following propensities

$$a_{1,1} = c_{1,1} A_1 \tag{5.26}$$

$$a_{1,2} = c'_{1,2}(\phi_1)\tilde{A} \tag{5.27}$$

$$a_{2,1} = c_{2,1} A_2 \tag{5.28}$$

$$a_{2,2} = c'_{2,2}(\phi_2)\tilde{A} \tag{5.29}$$

where $r_{j,k}$ and $a_{j,k}$ are, respectively, a rule and a propensity labelled k and belonging to region j, and $\tilde{A}$ is the number of molecules of A calculated considering a region across the boundary between 1 and 2 (the quantity $w_{(j\to k)}$ of Equation 5.21). The flux $\mathbf{J}_D$ is represented by the rules $(r_{1,1}, r_{2,1})$ associated to the propensities $(a_{1,1}, a_{2,1})$ while the flux $\mathbf{J}_D$ is modelled by $(r_{1,2}, r_{2,2})$ and $(a_{1,2}, a_{2,2})$. At steady state the two fluxes are equal $\mathbf{J_D} = \mathbf{J_e}$, and so we obtain

$$c_{1,1}A_1 - c_{2,1}A_2 = c'_{2,2}(\phi_2)\tilde{A} - c'_{1,2}(\phi_1)\tilde{A}. \tag{5.30}$$

Recalling that for diffusion due to a concentration gradient $c = D/d^2$, using the expression for the voltage dependent parameters defined in Equation 5.19, and the Einstein-Stokes relation Equation 5.2 we have:

$$\frac{D_{1,1}}{v^2}A_1 - \frac{D_{2,1}}{v^2}A_2 = \frac{D_{1,2}zq\phi_2}{v^2 z k_B T}\tilde{A} - \frac{D_{1,2}zq\phi_2}{v^2 k_B T}\tilde{A}. \tag{5.31}$$

As the diffusing ion is the same we get $D_{1,1} = D_{1,2}$ and $D_{2,1} = D_{2,2}$; assuming that diffusion coefficient calculate at the boundary is the same in both ways, i.e. $D_{1,1} = D_{1,2} = D_{2,1} = D_{2,2} = D$ we have:

$$\frac{D}{v^2}(A_1 - A_2) = \frac{Dzq\tilde{A}}{v^2 k_B T}(\phi_2 - \phi_1) \tag{5.32}$$

Simplifying and rearranging the expression we have:

$$E_A = \phi_2 - \phi_1 = \frac{k_B T}{zq} \cdot \frac{(A_1 - A_2)}{\tilde{A}} \tag{5.33}$$

We observe that in order to obtain Equation 5.5 we have to define the amount of molecules at the boundary as:

$$\tilde{A} = \frac{A_1 - A_2}{\ln \frac{A_1}{A_2}} \tag{5.34}$$

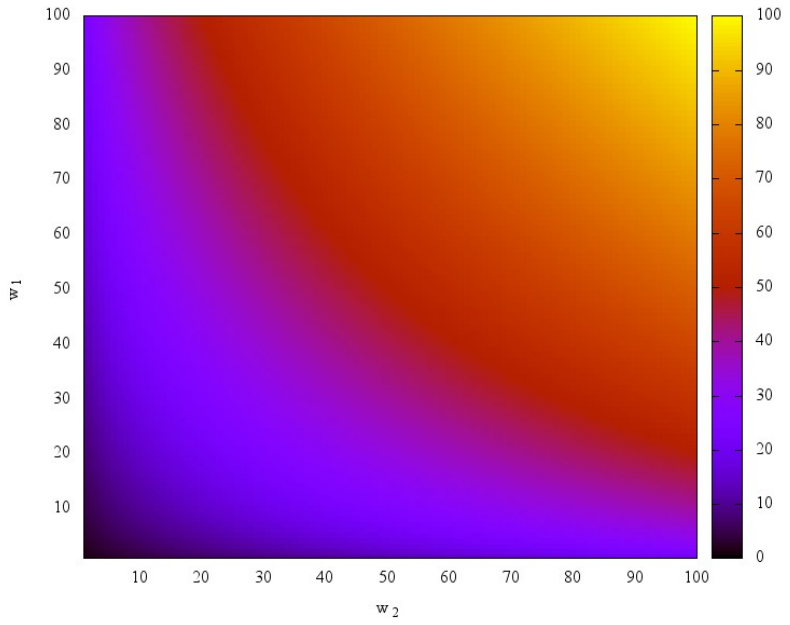


Figure 5.1: Illustration of the $\tilde{w}$ values as calculated by the function expressed in Equation 5.35, for $w_1 = w_2 = \{1, 2, \ldots, 100\}$; $\tilde{w}$ values are represented by colours.

Note that this value is undefined whenever $A_1 = A_2$; we solve this issue using in this case $\tilde{A} = A_1 = A_2$. Generalising, we define the amount of molecules around the boundary between two compartments 1 and 2 as

$$\tilde{w} = \begin{cases} w_1 & \text{if } (w_1 = w_2) \\ \frac{w_1 - w_2}{\ln(w_1/w_2)} & \text{if } (w_1 \neq w_2) \end{cases} \tag{5.35}$$

An example of the values calculated by Equation 5.35 is illustrated in Figure 5.1. Note that the definition of $\tilde{w}$ specified by Equation 5.35, further than ensuring the equality with the Nernst equation, generates values of $\tilde{w}$ that are enclosed between the minimum and the maximum of the pair of values (w_1, w_2):

$$\min(w1, w2) \leq \tilde{w} \leq \max(w1, w2). \tag{5.36}$$

A test case that makes use of the propensity function defined in this Section will be discussed in Section 5.4.

5.2 Modelling voltage gated channel state transitions

Living cells express a class of proteins, *ion channels*, which allow the flux of ions through the cell membrane. There are several types of ion channels which

are classified according to ion selectivity and gating type [94]. Considering the gating, ion channels can be ligand-gated or voltage-gated. Ligand-gated channels open and close due to conformational changes caused by the physical interaction with a ligand, such as an ion or a neurotransmitter. Voltage-gated channels (VGCs) open and close due to conformational changes caused by membrane voltage variations.

The transitions between open and closed states of VGCs involve the movements of charged components of the channel in response to the variation of the membrane potential. Taking into account thermodynamics arguments, it is likely that these movements are rate-limited by energy-barriers [35]. To derive an expression for the probability that a transition takes place, let us consider a membrane characterized by an electric potential difference $\Delta\phi$. This requires an energy $b\Delta\phi$, where the constant b captures both the amount of charged being moved and the distance covered. The probability that thermal fluctuations provide energy enough to surmount the energy barrier $b\Delta\phi$ is proportional to the Boltzmann factor $\exp\left(\frac{-b\Delta\phi}{k_B T}\right)$ [35], where T is the temperature and k_B is the Boltzmann constant. According to this argument, we describe the transition of a VGC between the two states s_1 and s_2 with a uni-molecular rule $r : s_1 \to s_2$ and the propensity function

$$a = c''(\Delta\phi) = c \cdot \exp\left(\frac{-b\Delta\phi}{k_B T}\right) \cdot w \tag{5.37}$$

where the constant c is multiplied by the dimensionless Boltzmann factor and w is the number of VGCs. Note that a is correctly expressed in [1/time] and, according to the sign of b, the propensity function can be an increasing or decreasing function of the membrane potential difference $\Delta\phi$ (see Figure 5.4). We designate this further type of propensities as propensities of *class III*.

5.3 ESτ-DPPs: integration of electrical properties in Sτ-DPPs

In the previous Sections we introduced two novel propensity functions to model (i) diffusion of charged objects and (ii) state transitions of VGCs. In this Section we describe the ESτ-DPPs, a further extension of Sτ-DPP in order to include processes (i) and (ii) in the formalism and to describe their time evolution by means of a modified version of the Sτ-DPP algorithm. Currently, ESτ-DPPs can model only systems with two compartments.

We added five components to the formal definition of Sτ-DPP (see Definition 4.3.1). Obviously, we added charges to objects and electric potentials to membranes: thus, an electric potential difference can be calculated considering the pair of membranes. As ESτ-DPPs consider three types of propensity functions, classes I (Equation 2.2), class II (Equation 5.18) and class III (Equation 5.37), we added a label to each rule to map the rule to the correct

propensity class. Moreover, we added a set for the values of the quantity b, that is required by the propensities of class III. Finally, we consider a value, the capacitance, associated to the pair of membranes: this further ingredient is required for the correct calculation of the temporal evolution of a ESτ-DPPs. In fact, we assume that the two modelled regions are separated by a membrane that acts as a *capacitor*. This assumption is common in computational neuroscience and is based on the evidence that living cells membranes are constituted by a lipid bilayer which is essentially impermeable to most charged molecules [35]. This property causes the membrane to act as a capacitor which separates charges located in the two regions divided by the membrane.

5.3.1 Definition

Formally, ESτ-DPPs are defined as follows.

Definition 5.3.1. *An ESτ-DPP of degree 2 is a construct:*

$$\mathit{\Pi} = (\Sigma, G_\mu, G_c, \mathcal{C}, \tilde{\mathcal{C}}, W, R, V_\mu, V_\Sigma, \Phi, Q, B, \gamma)$$

- $\Sigma = \{s_1, \ldots, s_m\}$ *is a finite of symbols, also called objects;*
- $G_\mu = (\mu, E, A_\mu)$ *is a mixed graph representing the topological arrangement of the membranes* $\mu = \{1, 2\}$, (E, A_μ) *are, respectively, the set of edges and the set of arrows which describe the topology of membranes;*
- $G_c = (\mu, A_c)$ *is a directed graph representing the connections (channels of communication) among the membranes* $\mu = \{1, 2\}$ *and* A_c *is the set of the arrows which describes the available connections;*
- $W = \{w_1, w_2\}$, *where* w_i *is the multisets of objects occurring inside the* i^{th} *membrane;*
- $\mathcal{C} = \{C_1, C_2\}$, *where* C_i *is the set of stochastic constants* $c_{i,j} \in \mathbb{R}^+$ *associated to the rules occurring inside the* i^{th} *membrane;*
- $\tilde{\mathcal{C}} = \{\tilde{C}_1, \tilde{C}_2\}$, *where* $\tilde{C}_i$ *is the set of rule classes* $\tilde{c}_{i,k} \in \{I, II, III\}$;
- $V_\mu = \{v_1, v_2\}$, *where* $v_i \in \mathbb{R}^+$ *is the volume of the* i^{th} *membrane;*
- $V_\Sigma = \{v_{s_1}, \ldots, v_{s_m}\}$, *where* $v_{s_j} \in \mathbb{R}^+$, *is the volume of object* v_{s_j}.
- $R = \{R_1, R_2\}$, *where* R_i *is the finite set of rules occurring inside membrane* i; *an* internal *rule is of the form*

$$\alpha_1 s_1 + \cdots + \alpha_m s_m \xrightarrow{\tilde{c}, c} \beta_1 s_1 + \ldots + \beta_m s_m \tag{5.38}$$

a communication *rule is of the form*

$$\alpha_1 s_1 + \ldots + \alpha_m s_m \xrightarrow{\tilde{c},c} (\beta_{1,1} s_1 + \ldots + \beta_{m,1} s_m, in_1) + \ldots + (\beta_{1,n} s_1 + \ldots + \beta_{m,n} s_m, in_n) \tag{5.39}$$

where the quantities α_i and β_j are natural numbers, $\tilde{c}$ is the rule class, c is the stochastic constant and $in_1, \ldots, in_n$ indicate the target membrane to which the object have to be sent;

- $\Phi = \{\phi_1, \phi_2\}$ *are the membrane electric potentials, where* $\phi_i \in \mathbb{R}$, $i \in \{1, 2\}$*;*
- $Q = \{q_1, \ldots, q_m\}$ *is the set of object charges, where* $q_i \in \mathbb{Q}$, $i \in \{1, \ldots, n\}$*;*
- $B = \{B_1, B_2\}$*, where* B_i *is the set of constants* $b_{i,j} \in \mathbb{R}$ *which describe the amount of charge that has to be moved and the distance these charges have to cover during a voltage induced state transition;*
- $\gamma \in \mathbb{R}^+$ *is the capacitance for the pair of regions* $(1, 2)$*;*

5.3.2 Time evolution of ESτ-DPPs

The temporal evolution of an ESτ-DPP is computed by a modified version of the Sτ-DPP algorithm and describes the evolution of a system with two compartments. In particular, we have introduced the following modifications in the procedure described in Section 4.3.2:

1c. calculate initial total charge $\tilde{Q}$ inside the membrane at $t_0 = 0$

$$\tilde{Q}_i(t_0) = \sum_{j=1}^{m} (w_i(s_j, t_0) \cdot q_j) \tag{5.40}$$

where we recall that $w_i(s_j, t_0)$ indicates the copy number of s_j in the multiset w_i at time step t_0.

2. `for each` rule r_k ($k \in \{1, \ldots, l\}$: `switch` $\tilde{c}_k$`:`

- `case I:` $a_k := \begin{cases} c \cdot h; & \text{if } r_k \text{ is a first order reaction;} \\ c/F_i \cdot h & \text{else;} \end{cases}$
- `case II:` $a_k := c'(\phi) \cdot h$;
- `case III:` $a_k := c''(\Delta\phi) \cdot h$;

23c. calculate the current charge $\tilde{Q}_i(t+\tau)$ inside the membrane

$$\tilde{Q}_i(t+\tau) = \sum_{j=1}^{m} (w_i(s_j, t+\tau) \cdot q_j) \tag{5.41}$$

and update the electric potential

$$\phi_i(t+\tau) = \phi_i(t) + \frac{\tilde{Q}_i(t+\tau) - \tilde{Q}_i(t)}{\gamma} \tag{5.42}$$

During the initialisation we added the calculation of the total charge inside the compartment $\tilde{Q}$. This operation is required for the subsequent internal state update. The second modification concerns the application of the correct propensity function definition according to the rule class. After rule application, the electric potential of the compartment is updated using an expression which assumes that the membrane that separates regions 1 and 2 acts as a capacitor. Therefore it is possible to apply the standard equation for a capacitor $\tilde{Q} = C\Delta\phi$ to define the relation between the amount of separated charge $\tilde{Q}$ and the membrane potential difference $\Delta\phi$, where C is the capacitance. We calculate the variation of the membrane potential from the previous value and considering the total net charge that crosses the membrane, namely, counting the number of charged particles moved in both directions, at each step of length τ.

The computational cost of the ESτ-DPP algorithm is $2mn$, where m is the number of the rules and n the number of compartments. The algorithm is implemented in a parallel (MPI) version in C programming language. The parametrisation schema establishes a one-to-one relation between processes and membranes.

5.4 Test cases for ESτ-DPPs

In this section we describe two use cases, the first devoted to show the use of propensity of class I, while the second for propensity of class III. More precisely, the first concerns a model to describe the movements of charges due to the presence of both a concentration gradient and an electric potential gradient between two regions. The second deals with the state transitions of VGCs in response to variations in the electric potential across two compartments.

5.4.1 Nernst potential

We consider a model that captures the essential dynamics of the events behind the establishment of the Nernst potential for the sodium ion (Na). Living cells evolved in order to maintain a stable concentration gradient

of sodium and other ions (potassium, calcium and chloride) between the extracellular and the intracellular regions. Although these ions can use protein channels to flow between the two compartments, the concentration gradients are sustained by passive redistribution due to impermeable ions and active transport by means of specific membrane proteins. If on the one hand, these concentration gradients establish a diffusion "force", on the other hand, the membrane potential counteracts it. Using the Nernst equation it is possible to calculate the electric potential difference required to sustain (separate) a given concentration gradient.

In this use case we show that ESτ-DPPs can be used to model these dynamics. More precisely, we consider a fixed concentration gradient of an ion between two compartments separated by a membrane (we assume the existence of a number of other processes that maintain this gradient). We show that, according to the initial membrane potential difference, the propensity functions of class I and II are correctly activated and the system reaches the condition specified by the Nernst potential.

To achieve this aim we use the following ESτ-DPP:

$$\mathit{\Pi} = (\Sigma, G_\mu, G_c, \mathcal{C}, \tilde{\mathcal{C}}, W, R, V_\mu, V_\Sigma, V, Q, B, \Gamma)$$

- $\Sigma = \{\mathrm{Na}^+\}$;
- $G_\mu = (\{1,2\}, \{(1,2)\}, \{\varnothing\})$;
- $G_c = (\{1,2\}, \{(1,2),(2,1)\})$;
- $\mathcal{C} = \{\{c_{1,1}, c_{1,2}\}, \{c_{2,1}, c_{2,1}\}\}$;
- $\tilde{\mathcal{C}} = \{\mathrm{I}, \mathrm{II}, \mathrm{I}, \mathrm{II}\}$;
- $W = \{w_1, w_2\}$;
- $R = \{R_1 = \{r_{1,1}, r_{1,2}\}, R_2 = \{r_{2,1}, r_{2,2}\}\}$;
- $V_\mu = \{v_1, v_2\}$;
- $V_\Sigma = \{v_{Na^+}\}$;
- $\Phi = \{\phi_1, \phi_2\}$;
- $Q = \{1.6 \cdot 10^{-19}\}$;
- $B = \{\{\varnothing\}\}$;
- $\gamma = 10^{-9}$

Table 5.1: List of the rules and constants used to model the Na^+ flow between the two regions 1 and 2. We omitted classes and constants over the arrows as we have reported their values in the respective columns.

rule	class	constant
$r_{1,1} : Na^+ \to (Na^+, in_2)$	I	11.6
$r_{1,2} : Na^+ \to (Na^+, in_2)$	II	433.84
$r_{2,1} : Na^+ \to (Na^+, in_1)$	I	11.6
$r_{2,2} : Na^+ \to (Na^+, in_1)$	II	433.84

The system is composed by two membranes (regions), where the communication is enabled in both ways, from 1 to 2 and *vice versa*. Only one type of object is available inside the system and represents the sodium ion (Na^+): thus, it carries a positive charge of $1.6 \cdot 10^{-19}$C. Four rules define the diffusion of Na^+between 1 and 2 (Table 5.1); in particular, each compartment has two rules, one belonging to class I and one belonging to class II. We set the capacitance for the pair to 10^{-9}F [35]. Compartment 1 can be thought as (a portion of) the extracellular space around a cell, that is represented by compartment 2. We consider Na^+ concentration outside and inside this "neuron" of respectively 150mM and 15mM [94], which leads to a ratio w_1/w_2 equal to 10 (assuming that the volumes of the two compartments are equal). The explicit consideration of the volume occupied by the two regions and the volume occupied by Na^+ are not relevant for this model, and thus we omit these values. We considered the diffusion coefficient of the sodium ion in rat brain ($1.16 \cdot 10^{-3}$ mm^2/s) described in [57], which is a satisfiable approximation for our test case. Therefore we obtained the values for the constants

$$c_{1,1} = c_{2,1} = \frac{D_{\mathrm{Na}}}{d^2} = 11.6\mathrm{s}^{-1} \tag{5.43}$$

$$c_{1,2} = c_{2,2} = \frac{D_{\mathrm{Na}}}{d^2} \cdot \frac{q_1}{k_B T} = 433.84\mathrm{s}^{-1}\mathrm{V}^{-1} \tag{5.44}$$

where we used $d = 10\mu$m.

Considering Equation 5.5 we calculate the expected equilibrium potential for Na^+ for the concentrations 150mM and 15mM: $E_{Na^+} = 61.5$mV. We ran a series of simulations considering a series of different values for the initial electric potential difference between the two regions. During the simulations we kept the electric potential of the extracellular compartment fixed (this assumption is common in computational neuroscience, as the extracellular space is conventionally considered the reference for the calculation of the membrane potential [94]) and thus only ϕ_2 was variable. The simulator correctly updates the membrane electric potential ϕ_2 as a consequence of the net diffusion of Na^+ ions Figure 5.2. More precisely, when $\Delta\phi < E_{Na^+}$,

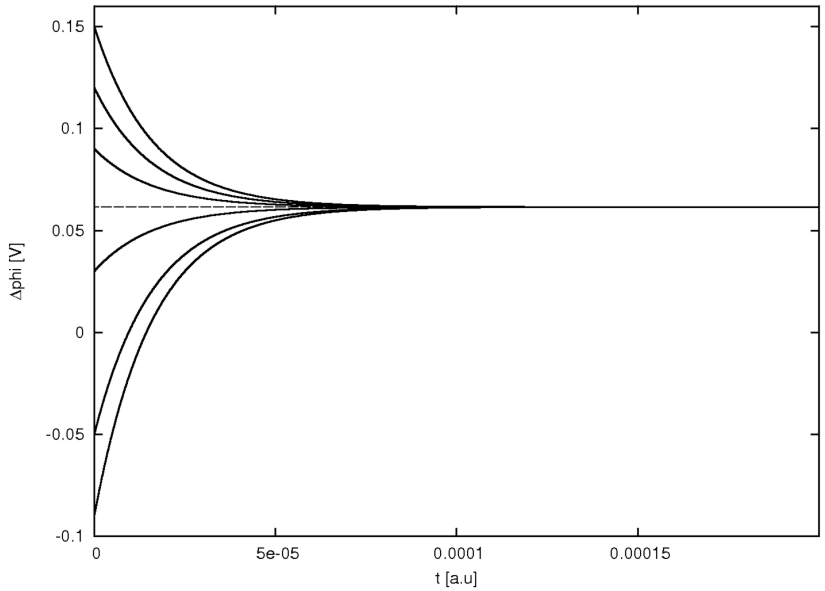


Figure 5.2: Reproduction of the Nernst equilibrium potential with an ESτ-DPP. Far from the Nernst equilibrium, if $\Delta\phi < E_{\mathrm{Na}^+}$ (simulations with initial $\Delta\phi = \{-0.09, -0.05, 0.03\}$) the electric potential difference $\phi_2 - \phi_1$ increases as a consequence of the net flow of Na^+ from region 1 to region 2 until the Nernst equilibrium potential is reached; conversely, if $\Delta\phi > E_{\mathrm{Na}^+}$ (simulations with initial $\Delta\phi = \{0.09, 0.120, 0.150\}$) the electric potential difference $\phi_2 - \phi_1$ decreases as a consequence of the net flow of Na^+ from region 2 to region 1 until the Nernst equilibrium potential is reached.

ϕ_2 increases due to the diffusion of the Na^+ ions from region 1 to region 2. Note that this situation reflects the flux of Na^+ ions from the extracellular compartment to the cell when the Na^+ VGCs open during the depolarisation phase of the action potential[2]. Conversely, when $\Delta\phi > E_{\mathrm{Na}^+}$, ϕ_2 decreases due to the diffusion of the Na^+ ions from region 2 to region 1. In both cases, the system correctly settles in a state in which $\Delta\phi = E_{\mathrm{Na}^+}$.

5.4.2 Voltage gated channels dynamics

In this test case we consider the gating of VGCs as a function of the membrane potential. Actually, the gating mechanism involves complex changes in the conformational structure of the channels, which are membrane proteins with a quaternary structure resulting from the interactions among more than one subunit (i.e. polypeptide chain). Models describing the transitions between conformational states of the channels involve many states and transitions, because these models describe complex molecules. Here we consider a simple

[2]The action potential is the variation of the membrane potential difference from the resting value to a positive value (depolarisation) and then from this value to the resting value (repolarisation). Nervous system cells use action potential to transmit information.

version of a sodium VGC (described in [35]) which captures the essential properties of the channel, in order to show the use of the rules associated to the propensities of class III (Equation 5.37).

States and transitions of the sodium VGC model we consider are depicted in Figure 5.3. The model includes 5 states and 10 transitions. The opening of the channel is due to transitions that are increasing function of the voltage and, conversely, the closing is determined by transitions that have the opposite voltage dependence. In this situation we expect an increase of the number of open channels as the voltage is increased. However, other than closed and open states the sodium VGC model we consider has a state called *inactive*. As the exit from this state is controlled by a decreasing function of the voltage, the number of inactive channels will increase as the voltage increases. Hence, the sodium VGC is characterised by a transient response during membrane *depolarisation* (an increase of the membrane potential) due to a transient increase of open channels determined by the smaller and smaller probability of exiting the inactive state.

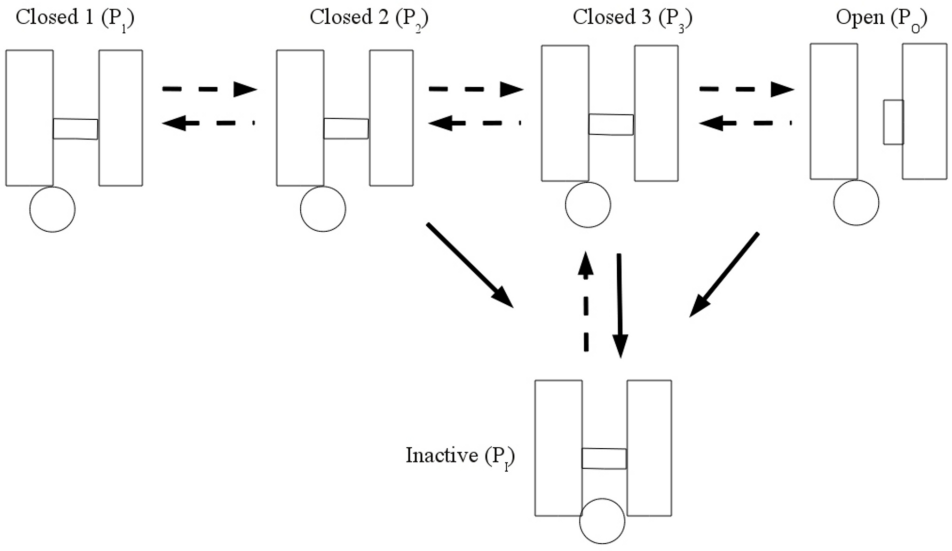


Figure 5.3: States and transitions of a Na^+VGC (adapted from [35]). Arrows line style reflects the voltage dependence; continuous arrows: voltage independent transitions; dashed arrows: voltage dependent transitions.

To model this system we use the following ESτ-DPP:

$$\mathit{\Pi} = (\Sigma, G_\mu, G_c, \mathcal{C}, \tilde{\mathcal{C}}, W, R, V_\mu, V_\Sigma, V, Q, B, \Gamma)$$

- $\Sigma = \{\mathrm{P_1}, \mathrm{P_2}, \mathrm{P_3}, \mathrm{P_O}, \mathrm{P_I}\}$;
- $G_\mu = (\{1,2\}, \{(1,2)\}, \{\varnothing\})$;
- $G_c = (\{\varnothing\}, \{\varnothing\})$;

Table 5.2: List of the rules and constants used to model the state transitions of the Na^+ VGC. Model and constants values are adapted from [35]. We omitted classes and constants over the arrows as we have reported their values in the respective columns.

R_2	class	C_2	B_2
$r_{2,1} : \mathrm{P}_1 \to \mathrm{P}_2$	III	3666	-1.068
$r_{2,2} : \mathrm{P}_2 \to \mathrm{P}_1$	III	56	0.3337
$r_{2,3} : \mathrm{P}_2 \to \mathrm{P}_3$	III	3666	-1.068
$r_{2,4} : \mathrm{P}_2 \to \mathrm{P}_I$	I	1	0
$r_{2,5} : \mathrm{P}_3 \to \mathrm{P}_2$	III	56	0.3337
$r_{2,6} : \mathrm{P}_3 \to \mathrm{P}_O$	III	3666	-1.068
$r_{2,7} : \mathrm{P}_3 \to \mathrm{P}_I$	I	1	0
$r_{2,8} : \mathrm{P}_O \to \mathrm{P}_3$	III	56	0.3337
$r_{2,9} : \mathrm{P}_O \to \mathrm{P}_I$	I	1	0
$r_{2,10} : \mathrm{P}_I \to \mathrm{P}_3$	III	2	0.748

- $\mathcal{C} = \{\{\varnothing\}, C_2 = \{c_{2,1}, \ldots, c_{2,10}\}\}$;
- $\tilde{\mathcal{C}} = \{\{\varnothing\}, \{\tilde{c}_{2,1}, \ldots, \tilde{c}_{2,10}\}$;
- $W = \{w_1, w_2\}$;
- $R = \{\{\varnothing\}, \{r_{2,1}, \ldots, r_{2,10}\}\}$;
- $V_\mu = \{1, 1\}$;
- $V_\Sigma = \{v_1 = v_2 = \ldots = v_5 = 1 \cdot 10^{-6}\}$;
- $\Phi = \{\phi_1, \phi_2\}$;
- $Q = \{\{\varnothing\}\}$;
- $B = \{\{\varnothing\}, B_2 = \{b_{2,1}, \ldots, b_{2,10}\}\}$;
- $\gamma = 1$.

The system is composed by two adjacent compartments. The first is empty and is required only to obtain the electric potential difference $\phi_2 - \phi_1$. The second compartment contains five different object, namely P_1, P_2, P_3, P_O, P_I that correspond to "closed 1", "closed 2", "closed 3", "open" and "inactive" states, respectively. Region 2 includes 10 rules (Table 5.2) to model the transitions between pairs of states as illustrated in Figure 5.3. Rules that specify the transition from P_2, P_3 and P_O to P_I are not sensitive to electric potential differences (class I), while all the other rules are associated to voltage dependent propensities of class III.

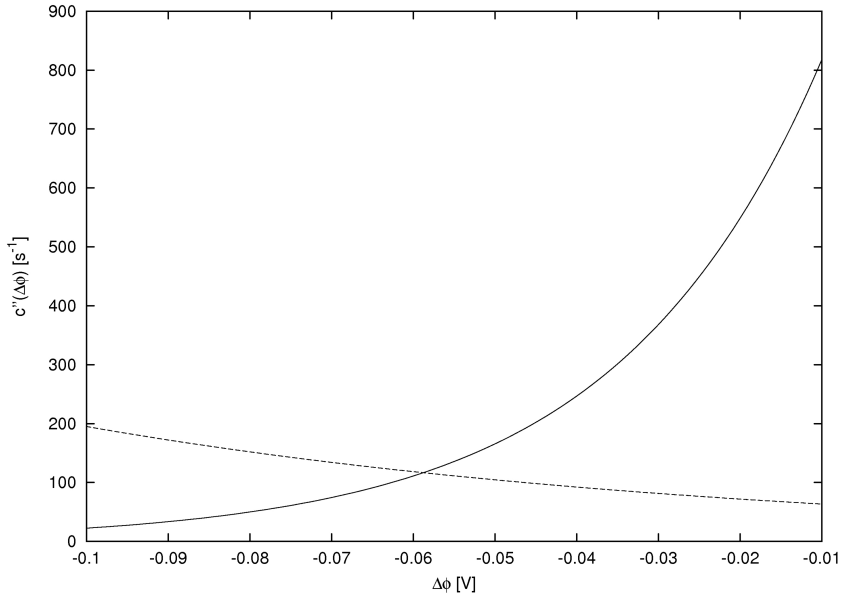


Figure 5.4: Values assumed by the voltage difference dependent parameter $c''(\Delta\phi)$ of class III propensities (Equation 5.37). Opening (continuous line): $c = 1220$, $b = -1.0680$; closing (dashed line): $c = 56$, $b = 0.337$.

The value of the constants of B_2 and in C_2 (Table 5.2) were adapted from [35], in order to let rule for opening $\{r_{2,1}, r_{2,3}, r_{2,6}\}$ transitions be increasing functions of the voltage difference and, conversely, rule for closing $\{r_{2,2}, r_{2,5}, r_{2,8}\}$ and de-inactivation $r_{2,10}$ propensities be decreasing function of the membrane potential difference, Figure 5.4; rules $\{r_{2,4}, r_{2,7}, r_{2,9}\}$ are not sensitive to the electric potential differences.

Simulations of the model initialised with 10^4 channels in the state P_1 show that the amount of channels in the others states increases transiently and then reaches an equilibrium, Figure 5.5. As the value of $\Delta\phi = \phi_1 - \phi_2$ increases the equilibrium is characterised by a higher and higher fraction of channels in the inactive state, from 0.61% at $\Delta\phi = -0.170$ to 99.65% at $\Delta\phi = -0.010$, Figure 5.6. The variation of the fraction of inactive channels in response to the value of the membrane potential difference is in close agreement with the experimental measurements presented in [62], concerning a cardiac sodium channel (Figure 5.7).

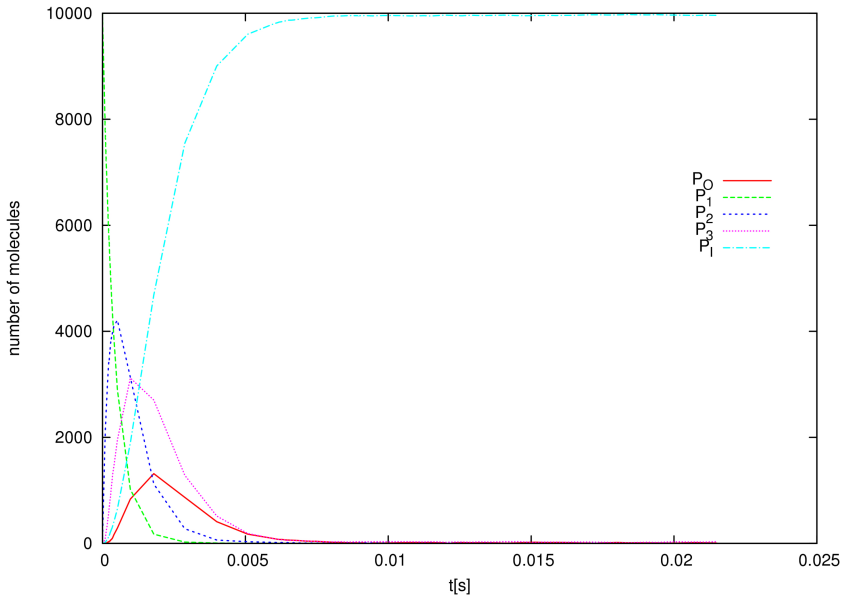


Figure 5.5: Dynamics of the Na^+ VGC states at $\Delta\phi = -0.010$V.

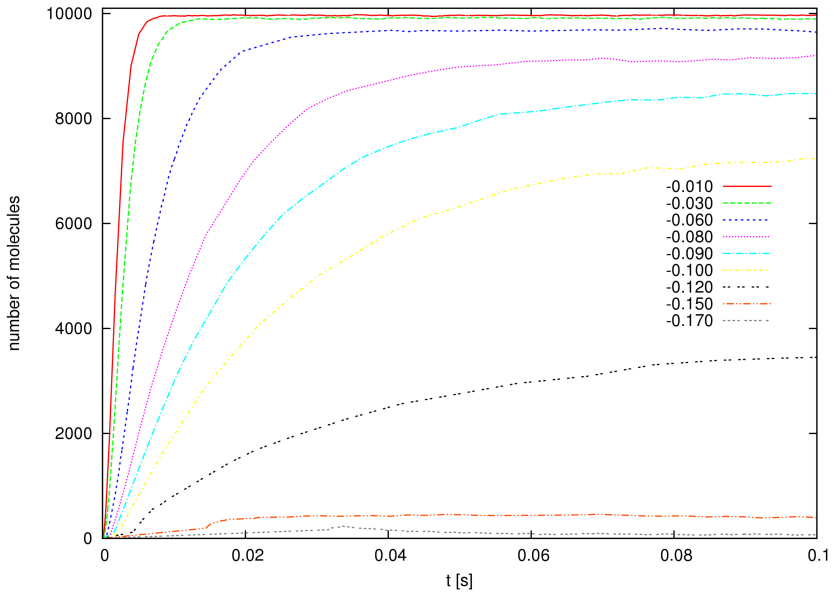


Figure 5.6: Dynamics of the inactive channels P_I in response to different values of $\Delta\phi$.

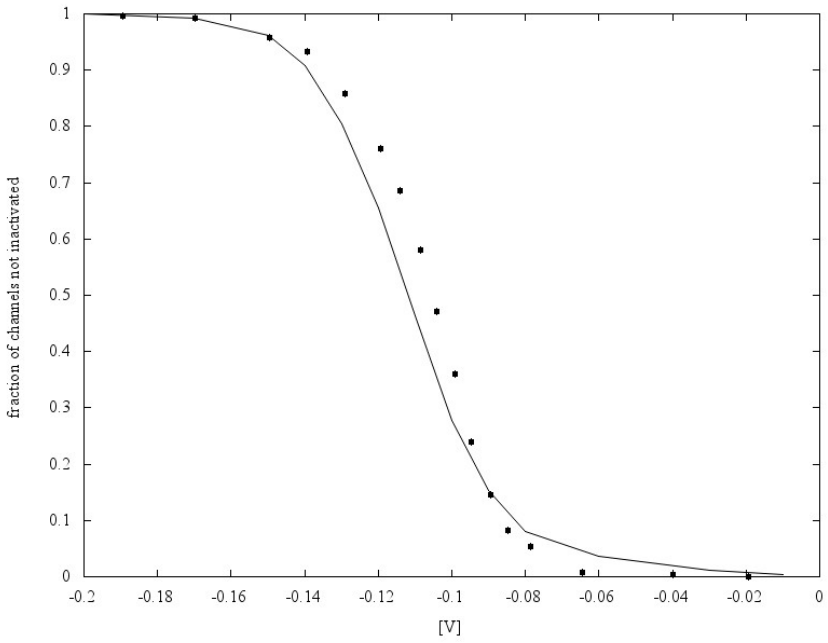


Figure 5.7: Fraction of channels not inactivated under different electric potential difference conditions. Points are adapted from data presented in [62] and concerning a cardiac sodium channel.

5.5 Discussion

In this chapter we have introduced a further extension of Sτ-DPPs (designated as ESτ-DPPs) in which we considered objects associated to a value representing their electrical charge and membranes (compartments) associated to a value representing the electric potential. To describe the time evolution of ESτ-DPPs we have introduced two novel classes of propensity functions derived from physical arguments. Therefore, the current version of the ESτ-DPP simulation technique includes three classes of propensity functions (Table 5.3): class I is based on the mass action law and is used to model reactions and particle diffusion; class II is derived from the equation for the drift velocity and is required to model particle diffusion due to an electric potential gradient; lastly, class III is derived from Boltzmann-Maxwell distribution and concerns VGCs state transitions due to an electric potential difference.

The ESτ-DPP algorithm updates the membrane electric potential at each step of the simulation assuming that the two modelled regions are separated by a semipermeable membrane such that it is possible to use the standard equation for the capacitor.

We discussed two test cases to show the correct functioning of the new propensities and of the ESτ-DPP algorithm. The first test case (related

Table 5.3: The three classes of propensity functions used by the ESτ-DPP algorithm and type of processes which they model. See the text for details.

class	definition	processes modelled
I	$c \cdot h$	mass action process particle diffusion (concentration)
II	$c'(\phi) \cdot h = c \cdot \phi \cdot h$	particle diffusion (electric potential)
III	$c''(\Delta\phi) \cdot h = c \cdot \exp\left(\frac{-b\Delta\phi}{k_B T}\right) \cdot h$	state transitions (electric potential difference)

to propensities of class II) concerns the reproduction of the dynamics of an ion between two regions in which both a molecule number gradient and an electric potential gradient is present. The system correctly reaches the electric potential difference required to sustain the molecule number gradient. The second test case (related to propensities of class III) deals with state transitions of a (generic) Na^+ VGC. The response of the VGC to different value of the electric potential difference is in close agreement respect to the experimental data that we collected from literature.

Powered by the new propensities, ESτ-DPPs enables the modelling of biochemical processes taking into account the effects of electric potential over molecule diffusion and VGC state transitions. As a consequence, the modelling capabilities of ESτ-DPPs cover a wider ensemble of biological situations compared to its precursor.

There are many examples of biological processes in which living cells take advantage from the presence of a membrane potential. Considering nervous systems, one only needs to think at the action potential and its relation to pre- and post- synaptic signalling cascades: in response to membrane potential variations, neurotransmitters or sensory stimuli (temperature, mechanical), VGCs establish ion fluxes leading to action potentials or the activation of signalling cascades, e.g. in case of the calcium ion (one of the most important second messengers[3]) [12]. Another example is the mitochondrial membrane potential, the alteration of which can be related to cell growth, cell differentiation and cell motility [30], stress [103] and aging [97].

Currently, ESτ-DPPs consider only systems composed by two compartments and manage the diffusion of only one type of charge (either positive or negative). These two issues will be further investigated in future in order to obtain a more comprehensive modelling approach.

[3]Second messengers are components of signal transduction cascades; in this context, they greatly amplify the strength of the signal.

6

Grid computing for large scale simulations

Grid computing is a useful solution to compute a large number of independent simulations that can be required during the development and analysis of a kinetic model. The advantage of using a grid approach for large computational challenges relies on the high-end scalability of this technology. In fact, if grid jobs are completely independent, then the theoretical scalability of the system is linear. This is not true for real computations, due to the time needed for scheduling jobs, for transferring data and for resubmitting failed jobs. As the communication among independent grid jobs is a factor that decreases grid performance, *data parallel* applications are the best candidate for grid computing. These applications split the computation of the input data in a series of independent processes and collect the results at the end of the computation.

In this chapter we study the use of grid for large scale simulations computed with τ-DPP. In particular, we exploit the EGEE project grid by means of the BioMed Virtual Organisation (see Chapter 3). As each simulation is independent, grid represent a reliable solution to deal with this computational load. We point out critical factors, bottlenecks and scalability considering the results obtained from the simulations of a bacterial chemotaxis model. The results obtained from this analysis represent a useful benchmark for the future implementation of more sophisticated approaches for the analysis of kinetic models (such as sensitivity analysis and parameter estimation), that require several simulations of the system.

6.1 Distribution of PSAs on the EGEE project grid

A crucial factor in performing a grid computation is the identification of a suitable strategy for splitting it into a set of grid jobs, which means defining the granularity of the computation. Jobs lasting more than 45 minutes are

defined long jobs, compared to middle jobs (between 5 and 45 minutes) and short jobs (less than 5 minutes).

However, the identification of the best granularity for a given application is not a trivial task. The computation of long jobs on the grid may cause significant data loss in the case of system failure or problems in the data transfer. On the other hand, the execution of a large number of short jobs raises the total latency time in the batch queues, affecting the global performance of the system. Moreover, the size of the output results should be considered, due to the impact of the transfer time on the total computation efficiency. Middle and long jobs are generally considered the most suitable to exploit grid computing because they represent a good trade-off between grid latency and failure problems [50].

For this reason we performed different PSAs, by varying the number of jobs and the number of simulations performed in each job, and by altering the computation time using different strategies of parameter selection and consequently the size of the output files. To enable the execution of a large number of τ-DPP based simulations over the grid platform, the CCS [79] has been adapted to satisfy the application requirements. While the lower layer was updated to be compliant with the latest release of the EGEE grid middleware, the upper layer was customised for these specific applications. Two scripts have been modified to manage the input and output of the PSAs computations: the first concerns the input management, while the second coordinates the job execution on the remote resource.

The first script splits the computation into grid jobs according to the desired granularity. This script populates a directory in which all the required files for each single job are temporary collected for submission (the τ-DPP program and a folder tree containing the previously generated input parameter files for the simulations) and creates the JDL (job description language) script which describes the job. Then, it calls the lower layer for the real submission to the grid, specifying some important parameters such as the maximum number of resubmissions in case of failure (which overcomes most of the problems due to the dynamic reshape of the grid facilities), the maximum time for the job to be queued on a grid cluster before its deletion and resubmission (to avoid over-crowded computational resources), and the output directory on the local server where results will be collected.

The second script of the CCS which must be customised, is the one actually executed on the remote clusters. This script manages the input files, unpacking the input from the InputSandBox, defines the operation pipeline to be carried out on the remote resources (in this case it performs many τ-DPP executions according to the specified input granularity), rebuilds the output directory in a structure which allows easy evaluation of the results, packs files to be retrieved and transfers the output results. Following the definition of PSA given in Section 3, the output of a general purpose PSA dealing with stochastic simulations is constituted by the set of calculated

dynamics . In this case, storage elements (SEs) must be used to archive the numerical results before downloading them to the user interface (UI). This approach is essential when the complete numerical results need to be retrieved for further investigations of the system's dynamics. However, preliminary analysis of grid performance suggested that the output data size have a significant effect on both the computation overhead and the success rate of the grid infrastructure.

Hence, in order to test the grid infrastructure in a wider range of conditions, we developed another approach in which the analysis of the dynamics is done just after the simulations. In this case the output is significantly reduced: instead of the complete time series for each molecular species, a scalar value that summarises the analysis is retrieved immediately through the OutputSandBox, without using the SEs. Clearly, there is a loss of information in evaluating the system behaviour directly on the grid platform, but an improvement of the performance is expected.

In other words, we developed two different implementations. The first one, designated as implementation A, in which the whole dynamics is retrieved. The second one, implementation B, in which the dynamics are analysed within the relative remote resource and only the result of the analysis is retrieved.

6.1.1 Analysis of the model dynamics

The results of the stochastic simulations performed during a PSA can be analysed in order to obtain some information about the biochemical system under investigation. Here, we consider a function f that measures the "difference" between a given experimental outcome, which represents the observed behaviour of the system, with the dynamics obtained by means of a simulation with a particular set of parameters. In this case, the aim of this analysis is to quantify the variation of the system's behaviour as a consequence of the variation of the stochastic constant values.

Working in the field of stochastic simulation, the definition of the function f used during our analysis is based on the idea of comparing the target dynamics (TD) with the estimated dynamics (ED), which are generated by using the τ-DPP stochastic simulation algorithm. Therefore, we have to manage some troublesome properties, hereby discussed, that are inherent to stochastic simulations.

First of all, the time instants (except the initial one) of the ED series will be distinct from those sampled in the TD series, since the two methods use different time samplings and, above all, each simulation performed by using a stochastic algorithm generates a different time series (hence, it does not correspond to the constant-step temporal sampling of TD). In addition, the time interval between any couple of consecutive time instants of the ED series will generally be distinct from every other time interval in the same series.

Therefore, the comparison between the two time series is achieved by first computing a linear interpolation between points of the TD in correspondence to the points of the ED, and then by summing up the areas of the trapezoids defined by the points of the two time series. The function f is defined as follows:

$$f = \sum_{i,l} \frac{1}{2}(|x'_l(\tau_i) - x_l(\tau_i)| + |x'_l(\tau_{i+1}) - x_l(\tau_{i+1})|)(\tau_{i+1} - \tau_i) \qquad (6.1)$$

where $x(\tau_i)$ are points of one time series (Estimated Dynamics, ED) and $x'(\tau_i)$ are the corresponding points obtained by a linear interpolation between two consecutive points $x(t_j)$ and $x(t_{j+1})$ of the other time series (Target Dynamics, TD); l runs over the set of chemical species while i runs over the set of points of the ED. An extensive description of the function used here to evaluate the distance between two dynamics can be found in [15].

6.2 PSAs of a bacterial chemotaxis model

In this section we present a case study that can be considered a good test to benchmark the grid infrastructure for future and more complex analysis of biological and chemical systems. As a matter of fact, the relatively large number of chemical reactions and molecular species, and the average time required to perform a single stochastic simulation are all factors that makes the bacterial chemotaxis model considered here a suitable test case to prove the effectiveness of the grid infrastructure exploited to execute a PSA.

Chemotaxis allows bacterial cells to move in biased directions, in response to concentration gradients of attractant or repellent ligands occurring in their surrounding environment. This behaviour depends on the frequency at which bacteria switch between clockwise and counter-clockwise rotation of the flagella. If the ligand concentration remains constant over time, the switching frequency is reset to the prestimulus level: this is an adaptation to the ligand concentration change by returning to random walk motions. The bacterial chemotaxis model we consider for this use case was presented in [14] and is composed of 59 biochemical processes (see Appendix B).

6.2.1 PSAs settings and results

We have performed four PSAs (in the following called PSA1, PSA2, PSA3 and PSA4) that have been defined by creating a number of parameterisations in which the stochastic constant values of the bacterial chemotaxis model have been modified following different strategies. In particular, the results of the PSAs have been analysed by using a distance function f which compares the dynamics computed during the stochastic simulations with a reference dynamics, namely a known behaviour of the bacterial chemotaxis model.

More precisely, we considered the temporal evolution of CheY (precisely, of its phosphorylated state CheYp). These analyses allowed us to test PSA performance over grid in two general purpose scenarios: the first, in which the whole set of dynamics is fully retrieved and is therefore available for further analysis (implementation A); the second, in which the analysis of the dynamics is performed remotely and only the results of the computation of the function f on the obtained dynamics are retrieved (implementation B).

To keep the expected CPU time t_e^{cpu} constant, in each PSA we assigned the same number of simulations to each job. Conversely, with the aim of experimenting the grid using different settings, the number of jobs and simulations per job have been modified across the four PSAs. Therefore, PSAs differ from each others both for the number of simulations per job and for the number of grid jobs.

These settings have been selected such that $0.5\text{h} \leq t_e^{cpu} \leq 3.5\text{h}$, which is the typical time of middle and long jobs, the most appropriate for a grid computation. The t_e^{cpu} was estimated computing a single job on an Intel Xeon 2.5GHz, 10GB RAM.

In PSA1 and PSA2, only one element of the stochastic constants set C has been varied in every parameterisation p_i. In PSA1, for each parameter c_j, 10 simulations have been run using 10 values of c_j linearly distributed within the interval $[0.5\bar{c_j}, 1.5\bar{c_j}]$. A total of 590 simulations have been distributed on grid, organised in 59 jobs, each one composed of 10 simulations. Every simulation has been performed with a relatively long time length, 10 time units, in order to check for potential late effects on the system's dynamics. The computation of a single job, relative to a c_j , had $t_e^{cpu} \approx 45'$ and was associated to a data volume of about 70MB (leading to a total data volume of 4GB).

The results of PSA1 denoted that there were three most influential parameters, c_8, c_{37} and c_{49}, that affected the system's dynamics. The variation of the other stochastic constants had negligible effects on the systems behaviour; therefore, for the subsequent PSA analysis we decided to extend the range of variation to all the stochastic constants and to have a finer grain sampling.

PSA2 has been composed of 5900 instances, organised in 59 jobs in which each parameter c_j has been varied 100 times (logarithmically distributed) considering the interval $[10^{-1}\bar{c_j}, 10^{1}\bar{c_j}]$. Every job had $t_e^{cpu} \approx 90'$ and was associated to a data volume of about 25MB. The lower data volume with respect to PSA1, despite the higher number of instances, was obtained by reducing the length of the single simulations down to 2 time units. Understandably, this gave an output of about 1.5 GB.

The results of PSA2 confirmed those obtained from PSA1. As depicted in Figure 6.1, parameters c_8, c_{37}, c_{49} highly influence the systems dynamics, while the other stochastic constants have little effect on the system's behaviour. These results confirm that the crucial points of the model, consid-

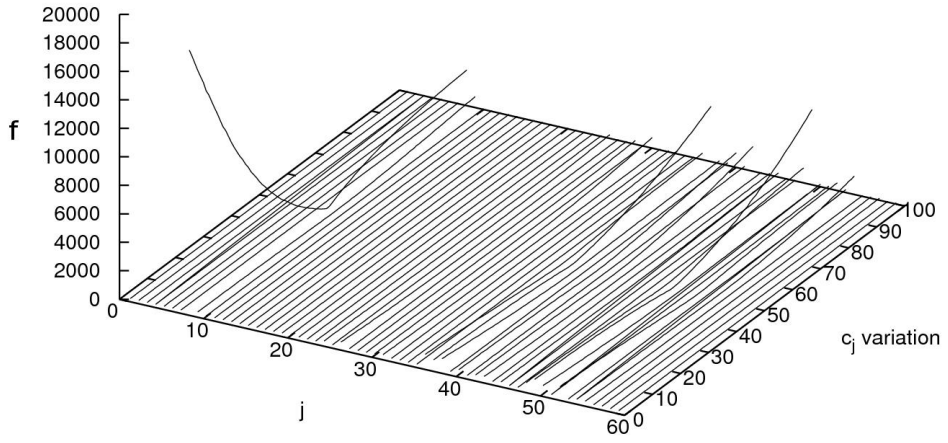


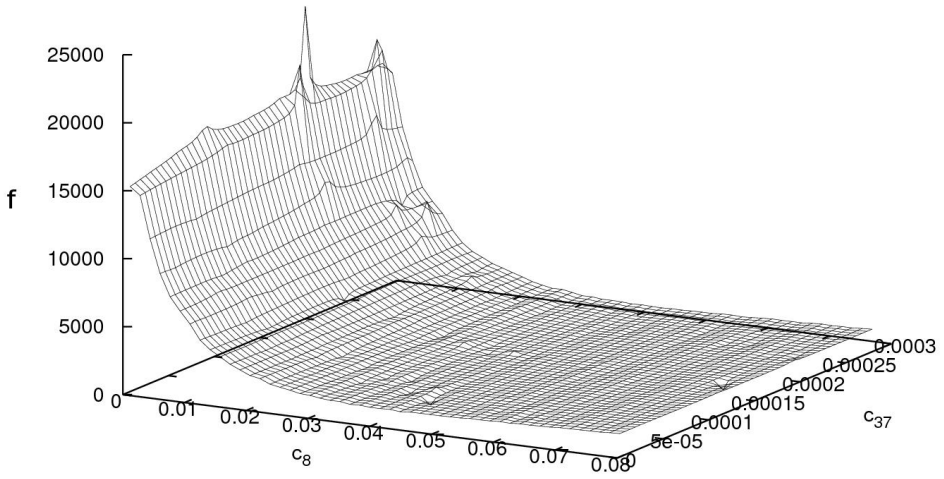
Figure 6.1: Values of function f obtained varying a single parameter for each parametrisation in PSA2. The x-axis represents the indexes j of the 59 stochastic constants c_j , while on the vertical axis there are the relative intensities of the variation with respect to c_j.

ering these ranges for the parameters, are represented by reactions involving the methyl-accepting transmembrane protein in the highly methylated state.

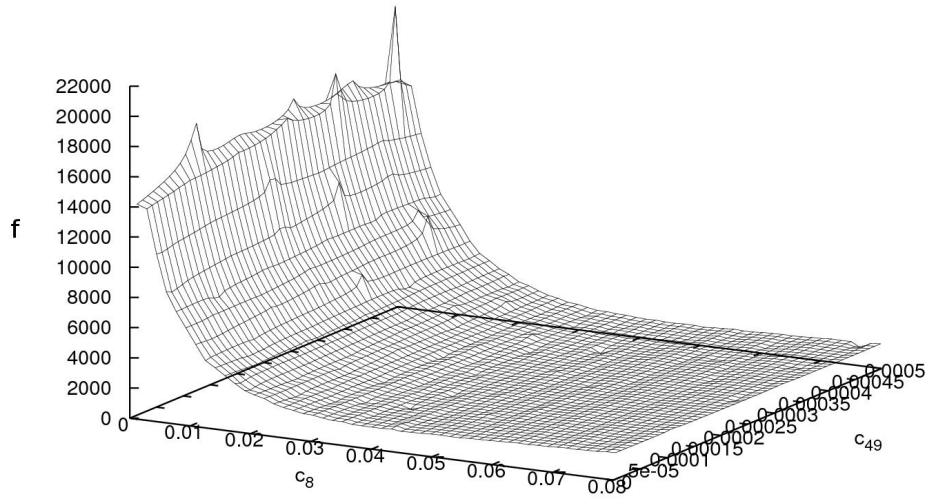
PSA3 and PSA4 have been composed of 10000 simulations each: 100 jobs, of 100 simulations each, have been distributed on the grid. The samples have been obtained using a quasi-random number generator within the interval $[10^{-1}\bar{c_j}, 10^{1}\bar{c_j}]$. Each simulation was performed with a time length of 10 time units.

In PSA3, every job had of about 230 minutes and produced an output of 188MB in size (for a total of 19 GB), while PSA4 was characterised by jobs with of about 30 minutes and produced 12MB in size (for a total of 1.2 GB). The differences in space occupation and computation time are related to the different number of free parameters: 59 in PSA3 and 3 in PSA4. Moreover, these results show that when varying 59 parameters at the same time, the chances of achieving values that lead to faster dynamics are higher. Using a stochastic algorithm this results in a higher number of steps, and hence in greater occupation of resources and a larger output size. The results of PSA3 have shown that, even though the parameterisation is obtained quasi-randomly sampling all the 59 parameters, the most influent parameter is c8. This is clear when comparing Figure 6.2a and Figure 6.2b where we show the landscape of the function computed on the obtained system's dynamics of the 10000 simulations with respect to parameters (c_8, c_{37}) and (c_8, c_{49}), respectively, with Figure 6.3, where the values of the function are plotted versus parameters c37, c49. The effect of parameter c8 induces an ordering of the function values. In particular, as the parameter value decreases, the values of the function increase.

In PSA4, we performed 10000 simulations where the parameterisations



(a)



(b)

Figure 6.2: Function f values landscape of PSA3 related to parameters a) c_8, c_{37} and b) c_8, c_{49}.

have been defined by varying only the three most influent parameters of the system. The aim of PSA4 was to investigate the 3-dimensional space delimited by parameters c_8, c_{37} and c_{49}. The results obtained highlight the influence of parameter c_8. The plateau of the values of the function have a complex and noisy shape where particular combinations of parameter values lead to dynamics that are far from the reference one Figure 6.3.

6.2.2 Performances

In this subsection we present the results related to the performance obtained by executing the PSAs with the two implementations A and B.

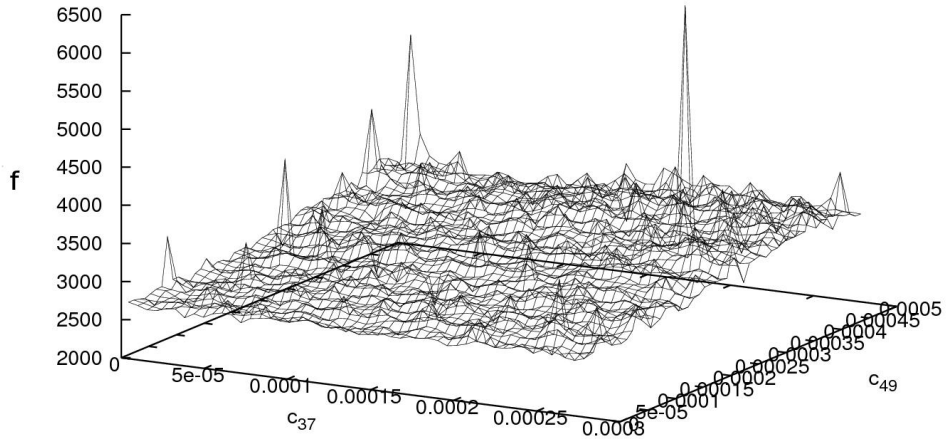
6.2.2.1 Implementation A

As already said, implementation A is characterised by the analysis of the system's dynamics on the UI after the computation of all the simulations on the grid platform. The four PSAs have comprehensively an estimated computational time of 24 days employing a single CPU, while over the grid the full computation lasted 2 days. A commonly used metric of the parallelisation gain during the grid computation is the *crunching factor*, which is defined as the ratio between the total expected CPU time over a single CPU and the duration of the experiment, that is the time needed to accomplish the longest job. Hence, by formalizing the crunching factor as

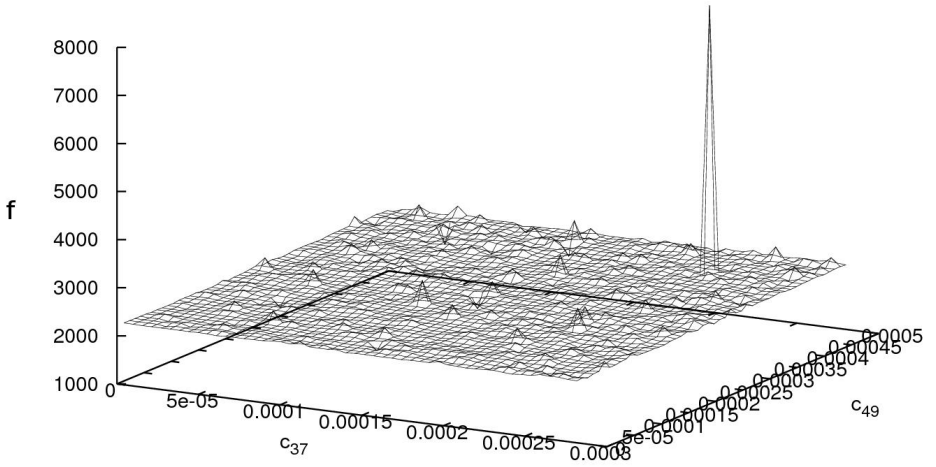
$$C_f = \sum_{i=1}^{N} \sum_{j=1}^{n_i} \frac{t_{e,i,j}^{cpu}}{\max(t_{g,i,j})} \tag{6.2}$$

where $t_{g,i,j}$ is the total grid time spent to accomplish the job j, N is the number of PSAs and n_i is the number of jobs per PSA; we achieved comprehensively a result of $C_f = 12$. C_f basically represents the average number of CPUs used simultaneously along the whole computation, taking into account the longest job. Investigating the issues that lowered C_f, we discovered that the number of processors used concurrently was significantly higher, with a peak parallelism of 76 CPUs one hour after PSA3 submissions, when almost all the jobs were started on the computing elements (CEs), Figure 6.4. However, due to unforeseen failure, the scalability was considerably reduced.

Considering the four PSAs, a total of 318 jobs have been submitted to the EGEE grid infrastructure: there were about 67% of the jobs reported as successfully finished according to the status logged in the resource broker (RB), at the first submission. However, the ratio went down to 57% after checking the existence of the output. Generally, among the most frequently reported problems there are: faults in the RB scheduling because resources with the required characteristics were not available; faults in the jobs management by the CEs queue, due to overload problems incorrectly reported



(a)



(b)

Figure 6.3: Function f values landscape of a) PSA3 related to parameters c_{37}, c_{49} and b) PSA4 related to parameters c_{37}, c_{49}.

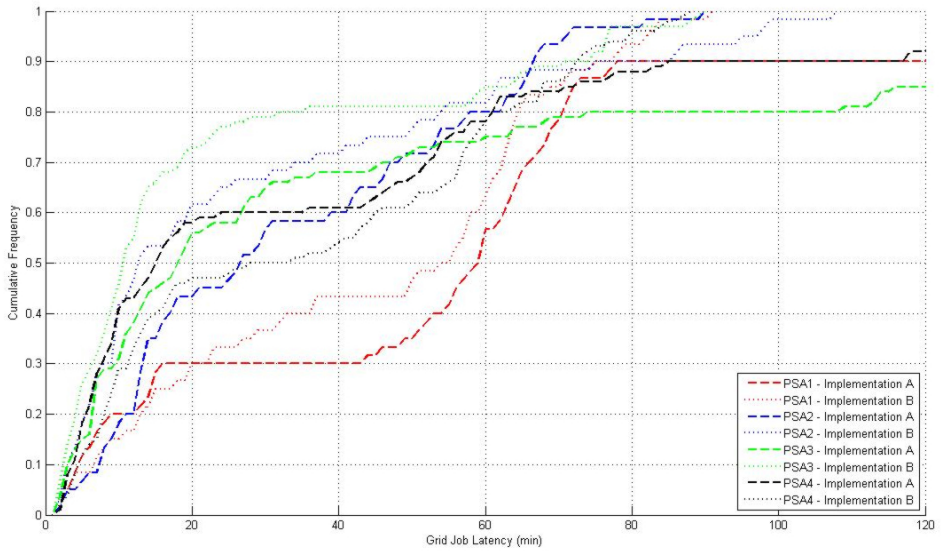


Figure 6.4: Cumulative frequency of grid jobs latency.

to the grid; and problems with the SEs files holding. In our tests the main cause of job failure seemed to be the data transfer between the CEs and the SEs and in a few cases, the unexpected termination of the jobs due to errors on the CEs.

6.2.2.2 Implementation B

In this implementation the function f for the analysis of the system's dynamics is computed directly on the grid infrastructure, avoiding the full download of the output data describing the entire system's dynamics. Computing this function during the simulations, the output was reduced to less than 1 MB for each job, which can be stored using the OutputSandBox. Implementation B has a success rate of 78%, according to the RB, which leads to roughly 75% of results correctly retrieved. Also in this case, the expected computational time was about 24 days on a single CPU because the computation of the function that analyses the system's dynamics is very fast. However, using this second implementation, the whole computational time took only 30 hours, which corresponds to $C_f = 20$. Similarly to implementation A, C_f is lower than the peak number of processors used concurrently, which reached 81 CPUs when almost all the jobs started to be computed by the CEs (Figure 6.4).

6.2.2.3 Comparison

Comparing the two implementations, the EGEE grid infrastructure's potential can be investigated with the aim of avoiding the bottleneck represented by the fragility of the distributed filesystem, and at the same time exploiting

Table 6.1: Percentage variation of the average grid job CPU time ($\bar{t}_g^{cpu}$) respect to the expected CPU time (t_e^{cpu}): $(\bar{t}_g^{cpu} - t_e^{cpu})/t_e^{cpu} \cdot 100$, and average grid job CPU time, in minutes.

	Percentage variation		$\bar{t}_g^{cpu}$ [min].	
PSA	A	B	A	B
1	48	52	67	68
2	32	35	118	122
3	16	16	266	268
4	36	38	41	41

the power of its computational resources as much as possible.

By analysing the components of the total grid job time, three contributes can be found $t_g = t_g^l + t_g^{cpu} + t_g^t$:

- t_g^l, the job latency time: after the jobs submission, RBs route them to the best available CEs; here, jobs enter in a queue and wait to be executed; this waiting time is t_g^l and includes the time required to upload the input file, the time needed to route the job by the RB, and the time spent in the grid clusters queue;
- t_g^{cpu}: the effective CPU time on grid facility
- t_g^t: the time required to download the output from the CE to the SE.

The job latency time was almost similar for all the PSA in both implementations, Figure 6.4: in particular, 80' had to be passed in order to observe the execution of almost all the jobs. The average CPU time of a grid job $\bar{t}_g^{cpu}$ was always worse than t_e^{cpu} (Table 6.1). This result was expected considering both the computer used for the simulations done to calculate , which is equipped with one of the latest processors and the resource sharing on the remote facilities (memory and disks). However, considering that the EGEE grid offers a large number of working nodes (WNs), the fact that the expected average performance is "only" from 16% to 52% slower than an Intel Xeon 2.5GHz with 10GB RAM is not a bad scenario.

In detail, both t_g and t_g^{cpu} show large deviations Figure 6.5 that are caused by the heterogeneity of the computational resources. Moreover, Figure 6.5 shows that implementation A is characterised by a large number of jobs with high that affected the performance of the PSAs. Since both and of the four PSAs are similar in the two implementations (Supplementary material S3), t_g^t was the factor which had the largest impact on the performance of the system.

This trend is confirmed by the overhead ratio, defined as

$$O_r = \frac{(t_g - t_g^{cpu})}{t_g^{cpu}} \tag{6.3}$$

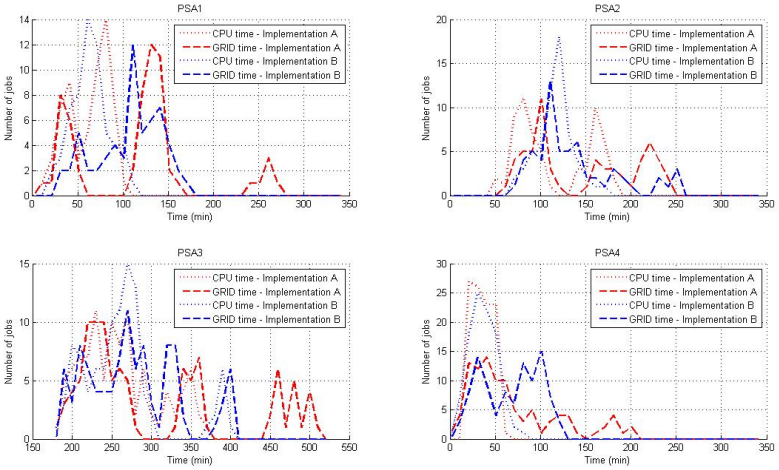


Figure 6.5: Distribution of the jobs of each PSA in relation to their CPU time over grid (dotted lines) and total grid time (dashed lines).

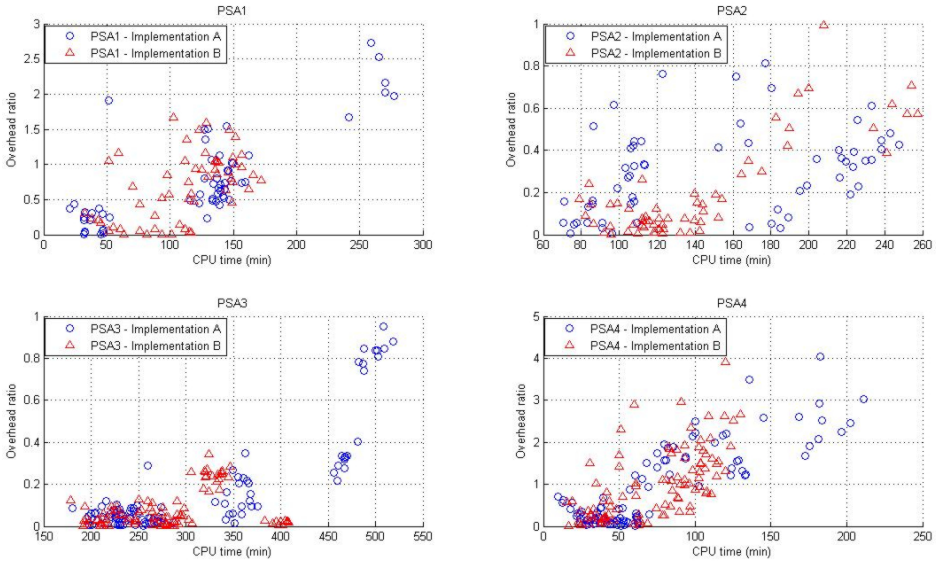


Figure 6.6: Overhead ratio values with respect to the grid job CPU time.

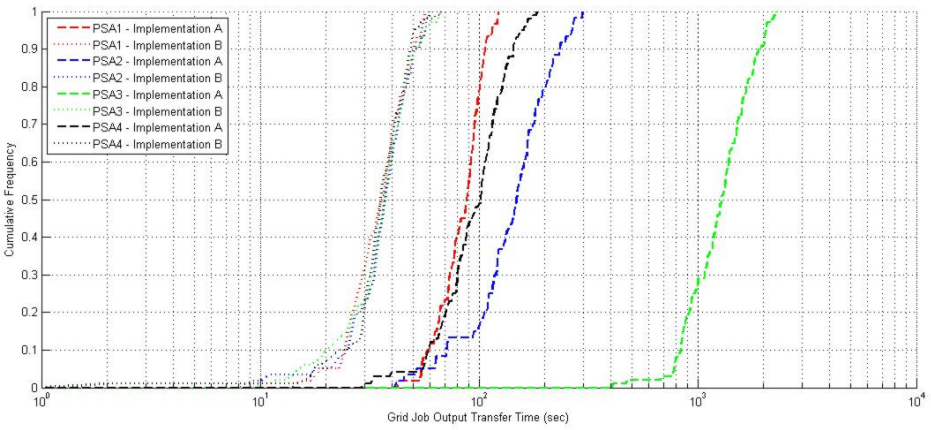


Figure 6.7: Cumulative frequency of files transfer time between grid and the UI.

which is shown in Figure 6.6. The overhead is an indicator of how much time is spent "on the grid" in relation to the actual t_g^{cpu}. The distribution of the O_r values underlines the performance enhancement achieved with implementation B, which shows a higher density of low values than implementation A. Furthermore, the best overhead was obtained during PSA3, indicating that the best granularity, according to our study, was associated with = 230' and equal to 266' and 281', respectively, in implementation A and B. In other words, the use of the grid is justified when the computational time is considerably long, since for short jobs the t_g^l can be very large with respect to the actual t_g^{cpu}. Therefore, the empirical idea we exposed about the relevance of the job granularity is confirmed by the achieved results: while there is a considerable failure rate which suggests that short tasks are performed more efficiently, the job duration should be carefully considered in the trade off with the large overhead of short jobs which do not fully exploit the grid's potential.

When a job computation is completed, the output must be retrieved on the UI. In implementation A, the time needed for this task, t_g^{ui}, corresponds to the file transfer time from the SEs to the UI, while in the implementation B, t_g^{ui} is the time required to get the OutputSandBox back from the RB. Clearly, the latter is considerably shorter than that needed to transfer the full system's dynamics from the grid distributed filesystem: the download took approximately 100 seconds with implementation A while only 30 seconds with the implementation B (Figure 6.7).

The values assumed by t_g^{ui} have not been included in the previous performance analysis because there is a random noise caused by the time between each polling of the RB by the CCS. In particular, this time interval has been set to 10 minutes in order to avoid a request overload on the grid infrastructure. The reason is that the output is retrieved on the UI only when its

status is reported as successfully completed, and a direct interrogation of the RB is needed to be informed of this status. In other words, the time elapsed for the effective presence of the output results on the UI has an uncertain component, due to the interval between two consecutive interrogations of the RB. Hence, the time we can effectively measure is the time needed for the mere transfer from the grid infrastructure to our UI.

Due to the authentication on the grid facilities and to the interrogation of the distributed filesystem, t_g^{ui} has a large overhead which is considerably high for small files but decreases while dealing with large data. For example, files of a few MB had a transfer rate of about 33KB/sec, files of tens of MB had transfer rate of about 90KB/sec and files larger than 100MB had a transfer rate of about 140 KB/sec.

In implementation B, the job success rate increased by about 18%, from 57% to 75%. Moreover, the resubmission ratio decreased from a peak of 5 times per job in implementation A to a maximum of 3 times in implementation B. The remaining faults can hardly be eliminated, because they are due to unrecoverable hardware problems (such as hard disk burns, RAM and motherboard failures or power supply discontinuities) and network access disruption or missconfiguration, which are factors that can lead to many job failures.

6.3 Discussion

PSA allows to repeat a large number of simulations of a biochemical system, each one with a different parametrisation, including variations - in the context of stochastic modelling - of the molecular species, chemical reactions, initial molecular quantities and stochastic constant values. Due to the independence of each instance of this particular kind of PSA and the large number of simulations to be run, grid computing offers a well-suited solution to this problem.

In this study we reported our experience with the distribution of stochastic simulations over the EGEE project grid. We focused on proving the effectiveness of the PSA approach and we tested the EGEE grid infrastructure to show its performance and its bottlenecks for this application, providing useful insights for future development. In particular, we adapted a fault-tolerant framework to handle the complete distribution process of running PSAs over the EGEE grid. We developed two implementations: one in which the simulations dynamics are retrieved, and the other in which the dynamics are analysed remotely. As a case study, we ran four PSAs for both implementations in which we varied the stochastic constant values of a bacterial chemotaxis model and analysed its behaviour with respect to reference dynamics.

The EGEE grid proved to be a useful solution for the distribution of

PSAs concerning the stochastic simulations of biochemical systems. This platform demonstrated its efficiency in the context of our middle-size test, and considering that the more intensive the computation is, the more scalable is the infrastructure, grid computing can be a suitable technology for large scale biological model analysis. However, due to the high failure rate, a complete submission and monitoring environment, like the CCS (introduced in [79] and customized for the management of the PSAs that we run), should be set up in order to appropriately manage the volume of data.

Moreover, in our experience the granularity of the submitted jobs and the use of SEs for managing files are elements to carefully consider when using the grid infrastructure. We obtained the lowest overhead with jobs of around 230mins and a 15% decrease of failed jobs avoiding the use of SEs. As expected, the grid resources revealed their heterogeneity, which suggests running preliminary tests to trace the behaviour of different grid sites, preventing the use of computation resources that are particularly inefficient and promoting those with suitable performances.

The results achieved in this study encourage the use of our framework in the context of approaches such as sensitivity analysis and parameter estimation, which require a large number of simulations in order to calculate the model response. Moreover, our framework can be used to evaluate the performances of the simulator according to its initial settings, in order to find its optimal parametrisation according to the model under investigation. With minor modifications the framework can be used also to handle parallel implementation of the simulation engine. In this case, the single grid job will be computed on more than one WN, but a test should be performed to determine the impact of the eventual increase of the grid job latency time, due to the need for more than one WN per grid job.

7

Conclusion and Future Developments

The work presented in this thesis has been motivated by the need to extend the current formalisms and computational methods for the modelling and simulation of biological systems, in order to capture a more comprehensive set of biological systems properties. Moreover we wanted to study a solution to manage a large number of simulations.

We took into account membrane systems since they represent an expressive formalism to model living systems [90] and stochastic methods (and more precisely those inspired by the Gillespie's stochastic simulation algorithm (SSA) [53]) for the simulation of the time evolution of a given system of biochemical processes, since noise plays an important role in many biological processes [41, 76, 43]. As τ-DPPs [27] are an approach that associates a membrane systems variant (Dynamical Probabilistic P systems) with a stochastic simulation algorithm (a modified version of the tau leaping algorithm, a computationally efficient approximation of the SSA), we took τ-DPPs into consideration as a basis of this thesis work. Concerning the management of a large number of (independent) simulations we exploited the grid computing as this platform represent a good solution for data parallel applications, in which the computation of input data is split in a number of independent processes and then the results are collected. More precisely, we used the EGEE (Enabling Grid for E-sciencE) project grid infrastructure as it is the main European grid computing platform.

We noticed that both current membrane systems and the most important stochastic simulation methods inspired by the SSA lack the explicit consideration of both molecules' and compartments' volume occupancy. The modelling of this feature enables the simulation of an important property of living cells, molecular crowding and, more precisely, the effects that molecular crowding has over the dynamics of processes that take place in the cytoplasm [98].

We studied from a theoretical point of view the consequences of objects' and membranes' volume occupation on the computational universality of

membrane systems. To achieve this goal we introduced the *Spatially extended P systems* (shortly, SP systems), where both the objects and regions occupy a finite amount of volume (or, equally, the objects and regions have a given size). As a consequence of the definitions of volumes, the dynamics of the system are affected by the availability of free space inside the regions. Therefore, unlike current membrane systems, only a finite number of objects can occupy a given region. We found that SP systems simulate efficiently a Turing Machine both considering the time and space required by the system. Moreover, we demonstrated that to achieve this result SP systems show an efficient volume occupation. More precisely, the volume required is a linear function of the space required by the Turing Machine. This result supports the idea that living cells can be powerful computing devices.

Then, we presented *Sτ-DPPs*, the integration of SP systems, tissue P systems and τ-DPPs. Sτ-DPPs permit cell-like and tissue like (as tissue P systems) membrane structures, extend the modelling capabilities of τ-DPPs by means of the introduction of both objects' and compartments' volume occupation (as SP systems) and the possibility of communication between non-adjacent regions. In order to describe the time evolution of Sτ-DPPs we modified the τ-DPP algorithm. These additional features allow Sτ-DPPs to be used for the modelling of crowded systems and structured geometries.

We showed that Sτ-DPPs can be used to model and simulate the diffusion of molecules introducing a reasonably small error, calculated in comparison with an analytic solution of the heat equation. The error was slightly higher compared to the one made with SSA in the same context, but we think that the lower computational cost of Sτ-DPPs (due to the use of the tau leaping) compared to SSA justifies the slightly higher error. Moreover, we presented two models as test cases to illustrate how Sτ-DPPs can be used to study crowding effects on intracellular dynamics: the first model was devoted to show that it is possible to reproduce a decrease of the diffusion rate of a particle in a crowded environment; with the second model we captured the heterogeneous probability of reaction in the different regions of a crowded medium. A third test case concerned a structured geometry; more precisely, we illustrated that Sτ-DPPs can be used to analyse the diffusion of two molecular species through a series of regions characterised by the presence of a preferential communication path representing a microtubule (a sort of "railway" for intracellular trafficking of objects, such as macromolecules and vesicles).

Subsequently, we proceeded considering electrical properties of living cells. We took into consideration the membrane potential difference, a property of living cells' membranes exploited as a battery to activate downstream processes and to transmit information. In particular, we extended Sτ-DPPs to capture the effect of membrane potential difference on charged species diffusion and voltage gated channels (VGCs) state transitions. We designated *ESτ-DPPs* this novel version of membrane systems.

To reproduce the two consequences of membrane potential difference that we have just mentioned, we added a number of features (such as object charges, membrane electric potentials and capacitances) to the previous formalism, and we defined two novel propensity functions in addition to the one used by the Sτ-DPP algorithm. We introduced a propensity function (designated as propensity functions of class II) in order to obtain the probability of diffusion of a charged species due to the presence of an electric potential gradient and we derived its expression taking into account the equations defining the conservation of mass and the flux due to an electric potential gradient. We introduced the other propensity function (designated as propensity functions of class III) to obtain the probability of state transition of a VGC, and we derived its expression from the Boltzmann-Maxwell distribution.

In order to describe the time evolution of ESτ-DPPs, we modified the Sτ-DPP algorithm. To update the membrane potential difference due to a flux of ions between two regions we assumed that the membrane separating these two regions behaves as a capacitor (a reasonable assumption for biological membranes). Mainly for this assumption the current version of ESτ-DPPs can model only systems composed by two compartments, such as the extracellular space and the intracellular environment, or the matrix and the intermembrane space of a mithocondrion. Powered by the new propensities, ESτ-DPPs enables the modelling of biochemical processes taking into account the effects of electric potential over molecule diffusion and VGC state transitions. Coupling these two mechanism it is possible to model a number of biological processes in which living cells take advantage from the presence of a membrane potential. Considering nervous systems, one only needs to think at the action potential and the complex pre- and post-synaptic signalling cascades; another example is the mitochondrial membrane potential, the alteration of which can be related to cell growth, cell differentiation and cell motility [30], stress [103] and aging [97].

To demonstrate the effective functioning of the ESτ-DPP algorithm, we considered two test cases. First, we showed that a model for ion diffusion between two regions, in which the number of ions is maintained at two different constant values and where an electric potential difference is available, correctly reaches the expected state as predicted by the Nernst equation. Second, we modelled the state transitions of a VGC and we obtained that the fraction of not-inactivated channels was in close agreement with experimental data collected from literature.

In parallel to the development of Sτ-DPPs and ESτ-DPPs, we studied grid computing to manage a large number of independent simulations. In fact, different approaches for the analysis of a model require the repeated simulation of the model with a different set of parameters, such as parameter estimation (e.g. in the case of genetic algorithms) and sensitivity analysis. The execution of the same application with different parametrisations is designated as parameter sweep application (PSA). As PSAs can be treated as

data parallel applications, grid computing is a good solution. We performed a set of PSAs of a bacterial chemotaxis model exploiting the EGEE project grid infrastructure, one of the biggest of its kind. During the study we compared two possible scenarios, an implementation associated to a heavier output and an implementation characterised by a higher computational cost. Despite the grid infrastructure has a series of critical factors (such as resource heterogeneity, significant failure due to storage elements) our study encourages the exploitation of this high-throughput solution for more sophisticated applications, such as sensitivity analysis. In fact, there are a series of expedients to deal with such issues (such as the use of a monitoring and re-submission tool and pilot jobs to test the current state of the grid) and the overall performance of grid during our middle-size test is appreciable.

In conclusion, the membrane systems variants Sτ-DPPs and ESτ-DPPs (equipped with stochastic algorithms for the computation of the temporal evolution) developed in this thesis work increase the set of biological systems that can be investigated *in silico*. The additional features we have developed did not affect the computational cost, since both Sτ-DPP and ESτ-DPP algorithms have the same computational cost ($2mn$) of the τ-DPP algorithm (where m is the number of compartments while n is the number of rules).

It is also important to recall that while Sτ-DPPs an ESτ-DPPs are a powerful modelling framework, they imply a series of assumptions (such as well-stirred compartments, division of the space domain in a set of homogeneous subregions in order to simulate diffusion, membranes considered as capacitors) that have to be carefully considered for the correct interpretation of the results. Whether the assumptions of a model are reasonable or not depend on the real systems that we want to model and from the aims of the study, and hence, the task of the proper creation of a model and of its correct parametrisation is obviously left to the modeller.

In future, we plan to improve both the formalisations and the algorithms that we presented in this thesis. For example, Sτ-DPPs can not model and simulate objects bigger than a single compartment, which conversely can be convenient for the analysis of big crowding agents in a tightly discretised space domain; instead, ESτ-DPPs are currently limited to systems made by two compartments and only one type of charge within the same system. If on the one hand this issue can represent a limit for the application of ESτ-DPPs for the modelling of biological systems, on the other hand, many important biological processes are regulated by a single type of ion (e.g. calcium ion as a second messenger in many signalling cascades) or by combination of ions with the same type of charge (e.g. the action potential in excitable cells is mainly due to sodium and potassium ions). Lastly, we plan to optimize the parallel (MPI) implementation of both the Sτ-DPP and ESτ-DPP algorithms, which are currently limited to a one-to-one relationship between processes and compartments, a limiting factor for the simulation of discrete spaces composed by a high number of compartments. Lastly, as grid computing

demonstrated to be a useful approach to handle a large number of simulations, we plan to develop a solution to manage the simulations required in the context of sensitivity analysis.

A

Theory of Computation: useful definitions

In this Appendix we report some definitions concerning the Turing Machine and P systems in relation to the theoretical study presented in Chapter 4.

A.1 Turing Machine

Turing machines were proposed in 1936 By Alan Turing. A Turing machine is an accurate model of general computer and can do everything that a real computer can do. Nevertheless, even a Turing machine can not solve certain problems, and these problems are currently considered beyond the theoretical limits of computation.

The single tape DTM is the more simple model of TM. In fact, despite there are several types of TMs, such as multi-tape TMs or nondeterminstic TMs, they have the same power, i.e. they recognise the same class of languages.

A TM operates on an infinite tape (unlimited memory). Initially the tape contains only the input string and it is blank everywhere else. A TM has a tape head which reads and writes symbol over the tape and moves around it. Hence, the TM can store and read an unlimited amount of information. The machine computes until it reaches an accepting or rejecting state. Otherwise, it will never halt.

Definition A.1.1. *A single tape DTM is a tuple*

$$M = (Q, \Sigma, \Gamma, \delta, q_0, A, R) \tag{A.1}$$

where:

- Q *is a finite set of states;*
- Σ *is finite input alphabet;*

- Γ *is the tape alphabet, a finite superset of* Σ*;*
- $\delta : \Gamma \times Q \to \Gamma \times Q \times \{\leftarrow, -, \rightarrow\}$ *is the transition function; it assumed that* δ *is undefined on both accepting and rejecting states;*
- $A \subseteq Q$ *is the set of accepting states;*
- $R \subseteq Q$ *is the set of rejecting states.*

The machine is initialised with an input string $w = s_1 s_2 \dots s_n \in \Sigma^*$ placed at the leftmost cell of the tape, while the rest of the tape contains blank symbols. The computation starts as the tape head reads the leftmost symbol over the tape. As Σ does not contain the blank symbol, the first blank symbol marks the end of the input. The computation proceeds with the execution of the transition function. If M ever tries to move to the left off the first cell (at the left-hand end of the tape), the tape head stays in the same location. The computation ends only if the current state belongs to A or R.

Definition A.1.2. *Let* M *be a DTM. The* time complexity *(or running time) of* M *is the function* $f : \mathbb{N} \mapsto \mathbb{N}$ *where* $f(n)$ *is the maximum number of steps that* M *uses on any input of length* n*.*

Definition A.1.3. *Let* M *be a DTM. The* space complexity *of* M *is the function* $g : \mathbb{N} \mapsto \mathbb{N}$ *where* $g(n)$ *is the maximum number of tape cells that* M *scans on any input of length* n*.*

A.2 P systems

Definition A.2.1. *A recogniser P system* Π *has an alphabet composed by the two objects yes and no, used to communicate acceptance and rejection respectively; every computation of* Π *is halting and one object between yes and no is sent out from the skin membrane in each computation. If all computations starting from the initial configuration agree on the result, then* Π *is confluent; otherwise it is non-confluent; the global result is acceptance if an accepting computation exists.*

Definition A.2.2. *Let* Π *be an P system and the size* $|C|$ *of a configuration* C *of* Π *be the sum of the membranes and the total number of objects these membranes contain. The* space complexity *of* Π *is the function* $g : \mathbb{N} \mapsto \mathbb{N}$ *where* $g(n)$ *is the size of the largest configuration that* Π *requires on any input of length* n*.*

Definition A.2.3. *Let* Π *be an P system. The* time complexity *of* Π *is the function* $f : \mathbb{N} \mapsto \mathbb{N}$ *where* $f(n)$ *is the maximum number of steps that* Π

uses on any input of length n. Let the size $|C|$ of a configuration C of Π be the sum of the membranes and the total number of objects these membranes contain. The space complexity *of Π is the function $g : \mathbb{N} \mapsto \mathbb{N}$ where $g(n)$ is the size of the largest configuration that Π requires on any input of length n.*

Definition A.2.4. *A function $f : \mathbb{N} \mapsto \mathbb{N}$ is said to be time-constructable iff the mapping $1^n \mapsto 1^{f(n)}$, i.e. from the unary representation of n to the unary representation of $f(n)$, can be computed by a DTM in $O(f(n))$ time. The function is space-constructable iff the mapping $1^n \mapsto 1^{f(n)}$ can be computed by a DTM in $O(f(n))$ space.*

B

Bacterial Chemotaxis Model

In this Appendix we describe the bacaterial chemotaxis model introduced in [14], that we used for PSAs described in Chapter 6.

The chemotaxis signal transduction pathway is composed of two main parts: (1) a signal sensor module, constituted by a receptor complex involving a (methyl-accepting) transmembrane protein (MCP), an adaptor protein (CheW) and a transmitter kinase protein (CheA); (2) a response regulator module, constituted by a methylesterase (CheB), a methyltransferase (CheR), the response regulator protein (CheY) and a CheY-regulator protein (CheZ).

The chemical reactions describing the Bacterial Chemotaxis system are listed in Table B.1.The reactions describe the following molecular interactions: (1) formation of the receptor complex 2MCP::2CheW::2CheA (reaction 1-4); (2) binding and unbinding of ligand molecules to the receptor complex in different methylation states (reactions 28-37); (3) methylation and demethylation of the receptor in presence/absence of ligand molecules (reactions 5-12 and 38-45); (4) autophosphorylation of CheA in different methylation states of MCP, in presence/absence of ligand molecules (reactions 13-17 and 46-49); (5) phosphotransfer to CheY and CheB in different methylation states of MCP, in presence/absence of ligand molecules (reactions 18-27 and 50-57); (6) dephosphorylation of CheYp and CheBp (reactions 58-59).

Each reaction is characterised by a stochastic constant needed to reconstruct the temporal evolution of the species that accounts for the following biological features: the binding affinity of the ligand is directly proportional to the methylation state of the receptor; the ligand-receptor binding reactions occur at a faster rate compared to phosphorylation and methylation/demethylation reactions; the methylation and demethylation activities of CheR and CheBp are, respectively, inversely and directly proportional to the methylation state of the receptor.

The initial conditions we used to calculate the temporal evolution of the protein CheYp (designated as reference condition, TD, see Chapter 6) are

reported in Table B.2, while the stochastic constants set used for the 59 reactions is: $C = \{c_1 = 0.1, c_2 = 0.01, c_3 = 0.1, c_4 = 0.02, c_5 = 5.0 \cdot 10^{-7}, c_6 = 5.0 \cdot 10^{-4}, c_7 = 2.0 \cdot 10^{-4}, c_8 = 0.0080, c_9 = 1.0, c_{10} = 0.6, c_{11} = 15.0, c_{12} = 0.35, c_{13} = 5.0 \cdot 10^{-7}, c_{14} = 5.0 \cdot 10^{-4}, c_{15} = 2.0 \cdot 10^{-4}, c_{16} = 6.0 \cdot 10^{-4}, c_{17} = 0.325, c_{18} = 7.0 \cdot 10^{-6}, c_{19} = 0.0035, c_{20} = 0.0014, c_{21} = 0.0044, c_{22} = 0.8, c_{23} = 15.0, c_{24} = 0.325, c_{25} = 7.0 \cdot 10^{-6}, c_{26} = 0.0035, c_{27} = 0.0014, c_{28} = 0.0044, c_{29} = 0.29, c_{30} = 2.8 \cdot 10^{-5}, c_{31} = 0.014, c_{32} = 0.0056, c_{33} = 0.0175, c_{34} = 1.0, c_{35} = 15.0, c_{36} = 0.29, c_{37} = 2.8 \cdot 10^{-5}, c_{38} = 0.014, c_{39} = 0.0056, c_{40} = 0.0175, c_{41} = 0.165, c_{42} = 5.0 \cdot 10^{-5}, c_{43} = 0.025, c_{44} = 0.01, c_{45} = 0.0306, c_{46} = 1.2, c_{47} = 15.0, c_{48} = 0.165, c_{49} = 5.0 \cdot 10^{-5}, c_{50} = 0.03, c_{51} = 0.0112, c_{52} = 0.0343, c_{53} = 0.05, c_{54} = 6.8 \cdot 10^{-5}, c_{55} = 0.0336, c_{56} = 0.0135, c_{57} = 0.035, c_{58} = 1.4, c_{59} = 15.0\}$.

Table B.1: Bacterial Chemotaxis reactions.

	Reagents	Produtcs
1	2MCP^m+2CheW	2MCP^m::2CheW
2	2MCP^m::2CheW	2MCP^m+2CheW
3	2MCP^m::2CheW+2CheA	2MCP^m::2CheW::2CheA
4	2MCP^m::2CheW::2CheA	2MCP^m::2CheW+2CheA
5-8	2MCP^m::2CheW::2CheA+CheR	2MCP^m+1::2CheW::2CheA+CheR
9-12	2MCP^m::2CheW::2CheA+CheBp	2MCP^{m}1::2CheW::2CheA+CheBp
13-17	2MCP^m::2CheW::2CheA+ATP	2MCP^m::2CheW::2CheAp
18-22	2MCP^m::2CheW::2CheAp+CheY	2MCP^m::2CheW::2CheA+CheYp
23-27	2MCP^m::2CheW::2CheAp+CheB	2MCP^m::2CheW::2CheA+CheBp
28-32	lig + 2MCP^m::2CheW::2CheA	lig::2MCP^m::2CheW::2CheA
33-37	lig::2MCP^m::2CheW::2CheA	lig + 2MCP^m::2CheW::2CheA
38-41	lig::2MCP^m::2CheW::2CheA+CheR	lig::2MCP^{m+1}::2CheW::2CheA+CheR
42-45	lig::2MCP^m::2CheW::2CheA+CheBp	lig::2MCP^{m1}::2CheW::2CheA+CheBp
46-49	lig::2MCP^m::2CheW::2CheA+ATP	lig::2MCP^m::2CheW::2CheAp
50-53	lig::2MCP^m::2CheW::2CheAp+CheY	lig::2MCP^m::2CheW::2CheA+CheYp
54-57	lig::2MCP^m::2CheW::2CheAp+CheB	lig::2MCP^m::2CheW::2CheA+CheBp
58	CheYp+CheZ	CheY+CheZ
59	CheBp	CheB

Table B.2: Molecular species and initial copy number.

Species	Initial copy number
2MCP	4000 dimers
2CheW	4000 dimers
2CheA	4000 dimers
CheY	17000 monomers
CheZ	12000 monomers
CheR	200 monomers
CheB	1700 monomers
ATP	$1.2 \cdot 10^6$ molecules

References

[1] Aderem, A. Systems biology: its practice and challenges. *Cell 121*, 4 (May 2005), 511–513.

[2] Alfieri, R., Merelli, I., Mosca, E., and Milanesi, L. A data integration approach for cell cycle analysis oriented to model simulation in systems biology. *BMC Syst Biol 1* (2007), 35.

[3] Alfieri, R., Merelli, I., Mosca, E., and Milanesi, L. The cell cycle db: a systems biology approach to cell cycle analysis. *Nucleic Acids Res 36*, Database issue (Jan 2008), D641–D645.

[4] Alhazov, A., Margenstern, M., Rogozhin, V., Rogozhin, Y., and Verlan, S. *Membrane Computing. International Workshop WMC5.* Springer, Berlin, 2004, ch. Communicative P systems with minimal cooperation, p. 162178.

[5] Ander, M., Beltrao, P., Ventura, B. D., Ferkinghoff-Borg, J., Foglierini, M., Kaplan, A., Lemerle, C., Toms-Oliveira, I., and Serrano, L. Smartcell, a framework to simulate cellular processes that combines stochastic approximation with diffusion and localisation: analysis of simple networks. *Syst. Biol. (Stevenage) 1*, 1 (Jun 2004), 129–138.

[6] Andrews, S. S., Addy, N. J., Brent, R., and Arkin, A. P. Detailed simulations of cell biology with smoldyn 2.1. *PLoS Comput. Biol. 6*, 3 (2010), e1000705.

[7] Arkin, A., Ross, J., and McAdams, H. H. Stochastic kinetic analysis of developmental pathway bifurcation in phage lambda-infected escherichia coli cells. *Genetics 149*, 4 (Aug 1998), 1633–1648.

[8] Banks, D. S., and Fradin, C. Anomalous diffusion of proteins due to molecular crowding. *Biophys. J. 89*, 5 (Nov 2005), 2960–2971.

[9] Baras, F., and Mansour, M. Reaction-diffusion master equation: A comparison with microscopic simulations. *Phys. Rev. E 54*, 6 (1996), 6139–6148.

[10] Bernardini, F., and Gheorghe, M. Population p systems. *Journal of Universal Computer Science 10* (2005), 509–539.

[11] Bernstein, D. Simulating mesoscopic reaction-diffusion systems using the Gillespie algorithm. *Phys. Rev. E Stat. Nonlin. Soft. Matter. Phys. 71*, 4 Pt 1 (Apr 2005), 041103.

[12] Berridge, M. Cell signalling biology.

[13] Berry, H. Monte carlo simulations of enzyme reactions in two dimensions: fractal kinetics and spatial segregation. *Biophys. J. 83*, 4 (Oct 2002), 1891–1901.

[14] Besozzi, D., Cazzaniga, P., Dugo, M., Pescini, D., and Mauri., G. A study on the combined interplay between stochastic fluctuations and the number of flagella in bacterial chemotaxis. In *Proceedings of CompMod2009 - 2nd International Workshop on Computational Models for Cell Processes* (2009).

[15] Besozzi, D., Cazzaniga, P., Mauri, G., Pescini, D., and Vanneschi, L. A comparison of genetic algorithms and particle swarm optimization for parameter estimation in stochastic biochemical systems. In *EvoBIO '09: Proceedings of the 7th European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics* (Berlin, Heidelberg, 2009), Springer-Verlag, pp. 116–127.

[16] Besozzi, D., Cazzaniga, P., Pescini, D., and Mauri, G. Modelling metapopulations with stochastic membrane systems. *Biosystems 91*, 3 (Mar 2008), 499–514.

[17] Besozzi, D., Zandron, C., Mauri, G., and Sabadini, N. P systems with gemmation of mobile membranes. In *Proceedings of the ICTCS 2001* (2001), A. Restivo, S. D. Rocca, and L. Roversi, Eds., LNCS, Springer-Verlag, Berlin, p. 136153.

[18] Broderick, G., Ru'aini, M., Chan, E., and Ellison, M. J. A life-like virtual cell membrane using discrete automata. *In Silico Biol. 5*, 2 (2005), 163–178.

[19] Buyya, R., Abramson, D., and Giddy, J. Nimrod/g: an architecture for a resource management and scheduling system in a global computational grid. In *Proc. Fourth International Conference/Exhibition on High Performance Computing in the Asia-Pacific Region* (2000), vol. 1, p. 283289.

[20] Campana, S., Rebatto, D., and Sciab, A. Experience with the glite workload management system in atlas monte carlo production on lcg. *Journal of Physics: Conference Series 119* (2009).

[21] Cao, Y., Gillespie, D. T., and Petzold, L. R. Avoiding negative populations in explicit poisson tau-leaping. *J Chem Phys 123*, 5 (Aug 2005), 054104.

[22] Cao, Y., Gillespie, D. T., and Petzold, L. R. Efficient step size selection for the tau-leaping simulation method. *J Chem Phys 124*, 4 (Jan 2006), 044109.

[23] Cao, Y., Li, H., and Petzold, L. Efficient formulation of the stochastic simulation algorithm for chemically reacting systems. *J Chem Phys 121*, 9 (Sep 2004), 4059–4067.

[24] Casanova, H., Obertelli, G., Berman, F., and Wolski, R. The apples parameter sweep template: user-level middleware for the grid. In *Supercomputing '00: Proceedings of the 2000 ACM/IEEE conference on Supercomputing (CDROM)* (Washington, DC, USA, 2000), IEEE Computer Society, p. 60.

[25] Cazzaniga, P. *Stochastic algorithms for biochemical processes*. PhD thesis, Università degli Studi di Milano-Bicocca, 2010.

[26] Cazzaniga, P., Mauri, G., Milanesi, L., Mosca, E., and Pescini, D. *Membrane Computing, Lecture Notes Computer Science*, vol. 5957. Springer Berlin / Heidelberg, 2010, ch. A novel variant of tissue P Systems for the modelling of biochemical systems, pp. 210–216.

[27] Cazzaniga, P., Pescini, D., Besozzi, D., and Mauri, G. *Membrane Computing, Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2007, ch. Tau Leaping Stochastic Simulation Method in P Systems, pp. 298–313.

[28] Cazzaniga, P., Pescini, D., Besozzi, D., Mauri, G., Colombo, S., and Martegani, E. Modeling and stochastic simulation of the ras/camp/pka pathway in the yeast saccharomyces cerevisiae evidences a key regulatory function for intracellular guanine nucleotides pools. *J Biotechnol 133*, 3 (Feb 2008), 377–385.

[29] Chen, D., Lear, J., and Eisenberg, B. Permeation through an open channel: Poisson-nernst-planck theory of a synthetic ionic channel. *Biophys J 72*, 1 (Jan 1997), 97–116.

[30] Chen, L. B. Mitochondrial membrane potential in living cells. *Annu Rev Cell Biol 4* (1988), 155–181.

[31] Conrad, E. D., and Tyson, J. J. *System modelling in cellular biology: from concepts to nuts and bolt*. MIT, 2006, ch. Modelling Molecular Interaction Networks with Nonlinear Ordinary Differential Equations, pp. 97–125.

[32] Csiksz-Nagy, A., Battogtokh, D., Chen, K. C., Novk, B., and Tyson, J. J. Analysis of a generic model of eukaryotic cell-cycle regulation. *Biophys J 90*, 12 (Jun 2006), 4361–4379.

[33] Crdenas, A. E., Coalson, R. D., and Kurnikova, M. G. Three-dimensional poisson-nernst-planck theory studies: influence of membrane electrostatics on gramicidin a channel conductance. *Biophys J 79*, 1 (Jul 2000), 80–93.

[34] Davies, P. C. W. Emergent biological principles and the computational properties of the universe. *Complexity 10* (2004), 11–15.

[35] Dayan, P., and Abbott, L. *Theoretical Neuroscience. Computational and Mathematical Modeling of Neural Systems*. The MIT Press, 2005.

[36] Dhar, P., Meng, T. C., Somani, S., Ye, L., Sairam, A., Chitre, M., Hao, Z., and Sakharkar, K. Cellware–a multi-algorithmic software for computational systems biology. *Bioinformatics 20*, 8 (May 2004), 1319–1321.

[37] Dix, J. A., and Verkman, A. S. Crowding effects on diffusion in solutions and cells. *Annu. Rev. Biophys. 37* (2008), 247–263.

[38] Edgar, B. A., Kiehle, C. P., and Schubiger, G. Cell cycle control by the nucleo-cytoplasmic ratio in early drosophila development. *Cell 44*, 2 (Jan 1986), 365–372.

[39] Elf, J., and Ehrenberg, M. Spontaneous separation of bi-stable biochemical systems into spatial domains of opposite phases. *Syst. Biol. (Stevenage) 1*, 2 (Dec 2004), 230–236.

[40] Ellis, R. J., and Minton, A. P. Cell biology: join the crowd. *Nature 425*, 6953 (Sep 2003), 27–28.

[41] Elowitz, M. B., Levine, A. J., Siggia, E. D., and Swain, P. S. Stochastic gene expression in a single cell. *Science 297*, 5584 (Aug 2002), 1183–1186.

[42] Elowitz, M. B., Surette, M. G., Wolf, P. E., Stock, J. B., and Leibler, S. Protein mobility in the cytoplasm of escherichia coli. *J Bacteriol 181*, 1 (Jan 1999), 197–203.

[43] Faisal, A. A., Selen, L. P. J., and Wolpert, D. M. Noise in the nervous system. *Nat Rev Neurosci 9*, 4 (Apr 2008), 292–303.

[44] Foster, I., and Kesselman, C., Eds. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 1999.

[45] Foster, I., Kesselman, C., and Tuecke, S. The anatomy of the grid: Enabling scalable virtual organizations. *International Journal of High Performance Computing Applications 15* (2001), 200–222.

[46] Freund, R., Kari, L., Oswald, M., and Sosik, P. Computationally universal p systems without priorities: two catalysts are sufficient. *Theoretical Computer Science 300* (2005), 251–266.

[47] Fulton, A. B. How crowded is the cytoplasm? *Cell 30*, 2 (Sep 1982), 345–347.

[48] G. Ciobanu, G. H. Păun, M. P.-J., Ed. *Applications of Membrane Computing*. Springer, Berlin, 2005.

[49] Gardiner, C., McNeil, C., Walls, D., and Matheson, I. Correlations in stochastic theories of chemical reactions. *Journal of Statisti. Phys., 14* (1976), 307–331.

[50] Germain Renaud, C., Loomis, C., Texier, R., and Osorio, A. Grid Scheduling for Interactive Analysis. In *HealthGrid 2006 Challenges and Opportunities of Health Grids* (Valencia/Spain Spain, 06 2006), vol. 120 of *Studies in Health Technology and Informatics*, IOS Press, pp. 25–33.

[51] Gibson, M., and Bruck, J. Efficient exact stochastic simulation of chemical systems with many species and many channels. *Journ. Phys. Chem. A 104*, 9 (2000), 1876–1889.

[52] Gillespie, D. T. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *J. Comput. Phys. 22*, 4 (1976), 403 – –434.

[53] Gillespie, D. T. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry 81* (1977), 2340–2361.

[54] Gillespie, D. T. Approximate accelerated stochastic simulation of chemically reacting systems. *The Journal of Chemical Physics 115* (2001), 1716–1733.

[55] Gillespie, D. T., and Petzold, L. R. *System Modeling in Cellular Biology, from concepts to nuts and bolts*. The MIT Press, 2006, ch. Numerical Simulation for Biochemical Kinetics, pp. 331–353.

[56] Goldbeter, A. A minimal cascade model for the mitotic oscillator involving cyclin and cdc2 kinase. *Proc Natl Acad Sci U S A 88*, 20 (Oct 1991), 9107–9111.

[57] Goodman, J. A., Bretthorst, G. L., Kroenke, C. D., Ackerman, J. J. H., and Neil, J. J. Estimating the sodium ion diffusion coefficient in rat brain. In *BAYESIAN INFERENCE AND MAXIMUM ENTROPY METHODS IN SCIENCE AND ENGINEERING: 23rd International Workshop on Bayesian Inference and Maximum Entropy Methods in Science and Engineering* (2004).

[58] Hartwell, L. H., Hopfield, J. J., Leibler, S., and Murray, A. W. From molecular to modular cell biology. *Nature 402*, 6761 Suppl (Dec 1999), C47–C52.

[59] Hattne, J., Fange, D., and Elf, J. Stochastic reaction-diffusion simulation with mesord. *Bioinformatics 21*, 12 (Jun 2005), 2923–2924.

[60] Hoops, S., Sahle, S., Gauges, R., Lee, C., Pahle, J., Simus, N., Singhal, M., Xu, L., Mendes, P., and Kummer, U. Copasi–a complex pathway simulator. *Bioinformatics 22*, 24 (Dec 2006), 3067–3074.

[61] Ionescu, M., Păun, G., and Yokomori, T. Spiking neural p systems. *Fundamenta Informaticae 71* (2006), 279–308.

[62] Irvine, L. A., Jafri, M. S., and Winslow, R. L. Cardiac sodium channel markov model with temperature dependence and recovery from inactivation. *Biophys J 76*, 4 (Apr 1999), 1868–1885.

[63] Kell, D. B., and Knowles, J. D. *System modelling in cellular biology: from concepts to nuts and bolt*. MIT, 2006, ch. The role of Modeling in Systems Biology, pp. 3–19.

[64] Kitano, H. Systems biology: a brief overview. *Science 295*, 5560 (Mar 2002), 1662–1664.

[65] Kopelman, R. Fractal reaction kinetics. *Science 241*, 4873 (Sep 1988), 1620–1626.

[66] Krishna, S., and Păun, G. P systems with mobile membranes. *Natural Computing 4* (2005), 255–274.

[67] Kruse, K., and Elf, J. *System Modeling in Cellular Biology: FROM CONCEPTS TO NUTS AND BOLTS*. The MIT Press, 2006, ch. Kinetics in Spatially Extended Systems.

[68] Laure, E., Fisher, S., and Frohner, A. Programming the grid with glite. computational methods in science and technology. 33–45.

[69] Leloup, J.-C., and Goldbeter, A. Modeling the circadian clock: from molecular mechanism to physiological disorders. *Bioessays 30*, 6 (Jun 2008), 590–600.

[70] Leye, S., Himmelspach, J., Jeschke, M., Ewald, R., and Uhrmacher, A. M. A grid-inspired mechanism for coarse-grained experiment execution. In *DS-RT '08: Proceedings of the 2008 12th IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications* (Washington, DC, USA, 2008), IEEE Computer Society, pp. 7–16.

[71] Li, H., Cao, Y., Petzold, L. R., and Gillespie, D. T. Algorithms and software for stochastic simulation of biochemical reacting systems. *Biotechnol Prog 24*, 1 (2008), 56–61.

[72] Marquez-Lago, T. T., and Burrage, K. Binomial tau-leap spatial stochastic simulation algorithm for applications in chemical kinetics. *J Chem Phys 127*, 10 (Sep 2007), 104101.

[73] Martín-Vide, C., Pazos, J., Păun, G., and Rodríguez-Patón, A. *Computing and Combinatorics, Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2002, ch. A New Class of Symbolic Abstract Neural Nets: Tissue P Systems, pp. 573–679.

[74] Martín-Vide, C., Pazos, J., Păun, G., and Rodríguez-Patón, A. A new class of symbolic abstract neural nets: Tissue p systems. In *COCOON '02: Proceedings of the 8th Annual International Conference on Computing and Combinatorics* (London, UK, 2002), Springer-Verlag, pp. 290–299.

[75] Martín-Vide, C., Păun, G., Pazos, J., and Rodríguez-Patón, A. Tissue p systems. *Theoretical Computer Science 296*, 2 (2003), 295–326.

[76] McAdams, H. H., and Arkin, A. Stochastic mechanisms in gene expression. *Proc Natl Acad Sci U S A 94*, 3 (Feb 1997), 814–819.

[77] McCollum, J. M., Peterson, G. D., Cox, C. D., Simpson, M. L., and Samatova, N. F. The sorting direct method for stochastic simulation of biochemical systems with varying reaction execution behavior. *Comput Biol Chem 30*, 1 (Feb 2006), 39–49.

[78] Meng, T. C., Somani, S., and Dhar, P. Modeling and simulation of biological systems with stochasticity. *In Silico Biol 4*, 3 (2004), 293–309.

[79] Milanesi, L., Merelli, I., and Trombetti, G. *Functional Genomics Applications in GRID.* IGI Global, 2009, ch. Handbook of Research on Computational Grid Technologies for Life Sciences, Biomedicine, and Healthcare, pp. 149–167.

[80] Minton, A. P. The effect of volume occupancy upon the thermodynamic activity of proteins: some biochemical consequences. *Mol. Cell. Biochem. 55*, 2 (1983), 119–140.

[81] Minton, A. P. The influence of macromolecular crowding and macromolecular confinement on biochemical reactions in physiological media. *J. Biol. Chem. 276*, 14 (Apr 2001), 10577–10580.

[82] Niederreiter, H. *Random number generation and quasi-Monte Carlo methods.* Society for Industrial and Applied Mathematics. Philadelphia, PA, USA, 1992.

[83] Nonner, W., Chen, D. P., and Eisenberg, B. Progress and prospects in permeation. *J Gen Physiol 113*, 6 (Jun 1999), 773–782.

[84] Pabst, M., Wrobel, G., Ingebrandt, S., Sommerhage, F., and Offenhusser, A. Solution of the poisson-nernst-planck equations in the cell-substrate interface. *Eur Phys J E Soft Matter 24*, 1 (Sep 2007), 1–8.

[85] Pescini, D., Besozzi, D., Mauri, G., and Zandron, C. Dynamical probabilistic p systems. *International Journal of Foundations of Computer Science 17* (2006), 183 – 204.

[86] Păun, A., and Păun, G. The power of communication: P systems with symport/antiport. *New Generation Computing 20* (2002), 295–306.

[87] Păun, G. Computing with membranes. *J. Comput. Syst. Sci. 61*, 1 (2000), 108–143.

[88] Păun, G. Computing with membranes - a variant: P systems with polarized membranes. *Intern. J. of Foundations of Computer Science 11* (2000), 167–182.

[89] Păun, G. P systems with active membranes: attacking np-complete problems. *J. Autom. Lang. Comb. 6*, 1 (2001), 75–90.

[90] Păun, G., and Pérez-Jiménez, M. J. Membrane computing: brief introduction, recent results and applications. *Biosystems 85*, 1 (Jul 2006), 11–22.

[91] Rivas, G., Ferrone, F., and Herzfeld, J. Life in a crowded world. *EMBO Rep. 5*, 1 (Jan 2004), 23–27.

[92] Romero-Campero, F. J., and Prez-Jimnez, M. J. Modelling gene expression control using p systems: The lac operon, a case study. *Biosystems 91*, 3 (Mar 2008), 438–457.

[93] Rosenfeld, N., Young, J. W., Alon, U., Swain, P. S., and Elowitz, M. B. Gene regulation at the single-cell level. *Science 307*, 5717 (Mar 2005), 1962–1965.

[94] Smith, C. U. M. *Elements of Molecular Neurobiology.* 2002.

[95] Stelling, J., Sauer, U., III, F. J. D., and Doyle, J. *System modeling in cellular biology: from concepts to nuts and bolts.* MIT, 2006, ch. Complexity and Robustenss of Cellular Network, pp. 19–41.

[96] Stiles, J. R., and Bartol, T. M. *Computational Neuroscience: Realistic Modeling for Experimentalists.* CRC Press, 2001, ch. Monte Carlo Methods for Simulating Realistic Synaptic Microphysiology Using MCell, pp. 87–127.

[97] Sugrue, M. M., and Tatton, W. G. Mitochondrial membrane potential in aging cells. *Biol Signals Recept 10*, 3-4 (2001), 176–188.

[98] Takahashi, K., Arjunan, S. N. V., and Tomita, M. Space in systems biology of signaling pathways–towards intracellular molecular crowding in silico. *FEBS Lett. 579*, 8 (Mar 2005), 1783–1788.

[99] Tian, T., and Burrage, K. Binomial leap methods for simulating stochastic chemical kinetics. *J Chem Phys 121*, 21 (Dec 2004), 10356–10364.

[100] Togashi, Y., and Kaneko, K. Molecular discreteness in reaction-diffusion systems yields steady states not seen in the continuum limit. *Phys Rev E Stat Nonlin Soft Matter Phys 70*, 2 Pt 1 (Aug 2004), 020901.

[101] Turner, T. E., Schnell, S., and Burrage, K. Stochastic approaches for modelling in vivo reactions. *Comput Biol Chem 28*, 3 (Jul 2004), 165–178.

[102] van Zon, J. S., and ten Wolde, P. R. Green's-function reaction dynamics: a particle-based approach for simulating biochemical networks in time and space. *J. Chem. Phys. 123*, 23 (Dec 2005), 234910.

[103] Vayssier-Taussat, M., Kreps, S. E., Adrie, C., Dall'Ava, J., Christiani, D., and Polla, B. S. Mitochondrial membrane potential: a novel biomarker of oxidative environmental stress. *Environ Health Perspect 110*, 3 (Mar 2002), 301–305.

[104] Westerhoff, H. V., and Palsson, B. O. The evolution of molecular biology into systems biology. *Nat Biotechnol 22*, 10 (Oct 2004), 1249–1252.

[105] Zimmerman, S. B., and Minton, A. P. Macromolecular crowding: biochemical, biophysical, and physiological consequences. *Annu. Rev. Biophys. Biomol. Struct. 22* (1993), 27–65.

[106] Zimmerman, S. B., and Trach, S. O. Estimation of macromolecule concentrations and excluded volume effects for the cytoplasm of escherichia coli. *J. Mol. Biol. 222*, 3 (Dec 1991), 599–620.