Accelerated Simulation of Large Reaction Systems Using a Constraint-Based Algorithm

Paulo E. P. Burke^{1,*} and Luciano da F. Costa²

¹*Bioinformatics Graduate Program, University of São Paulo, São Carlos, SP, Brazil*

²*São Carlos Institute of Physics, University of São Paulo, PO Box 369, São Carlos, 13560-970, SP, Brazil*

**pauloepburke@gmail.com*

Abstract

Simulation of reaction systems has been employed along decades for a better understanding of such systems. However, the ever-growing gathering of biological data implied in larger and more complex models that are computationally challenging for current discrete-stochastic simulation methods. In this work, we propose a constraint-based algorithm to simulate such reaction systems, called the Constraint-Based Simulation Algorithm (CBSA). The main advantage of the proposed method is that it is intrinsically parallelizable, thus being able to be implemented in GP-GPU architectures. We show through examples that our method can provide valid solutions when compared to the well-known Stochastic Simulation Algorithm (SSA). An analysis of computational efficiency showed that the CBSA tend to outperform other considered methods when dealing with a high number of molecules and reaction channels. Therefore, we believe that the proposed method constitutes an interesting alternative when simulating large chemical reaction systems.

1 Introduction

Computational models of biochemical reactions have proven to be useful tools for a better understanding of molecular systems [1, 2], as well as in optimizing experimental research [3, 4]. As a consequence of ever increasing data availability, these models are

significantly growing in size and complexity toward the scale of entire cells [5, 6, 7].

Simulations of such models can be carried out by applying among a choice of many available methods (e.g., Ordinary Differential Equations, Stochastic Simulation Algorithms, and Dynamic Flux Balance Analysis) [8, 9]. However, each simulation method has its own advantages and limitations, which influence their choice concerning specific applications.

Complex reaction systems, such as biochemical interactions in a cellular environment, have some characteristics that are important to consider when modeling and simulating it. One of these characteristics is that complex biochemical systems often involve reactions taking place at significantly different time-scales. Also, the number of copies of molecular species can vary from a single to thousands of molecules [10, 11].

Another important characteristic is that the number of copies of a given molecular species at any time takes a discrete value. However, when the system involves a sufficiently high number of copies for all species involved, it is reasonable to treat this quantity as a continuous variable [12]. With this assumption, the most straightforward approach to simulate such systems is to solve a set of coupled, first-order, ordinary differential equations called Reaction-Rate Equations (RE) [13, 14, 15]. Other approaches have been developed for the case when the rate-constants are not available. It is the case of the Dynamic Flux Balance Analysis [8, 16]. Despite the computa-

tional efficiency of these methods for the case of many molecules, when it comes to smaller numbers, the results can be unsatisfactory in the sense that they will provide real-valued approximations of quantities that are in fact discrete-valued. Also, these methods provide deterministic solutions, while reactions are better described as stochastic processes [17, 18].

A physical consequence of a small number of copies of a given molecule in a biochemical system is that the reactions involving it will have their rate highly influenced by thermal fluctuations and molecular crowding. A better description of these reaction rate fluctuations can be achieved through stochastic formulations of the system. Then, the evolution of molecular counts would be a consequence of the reaction occurrences following a given probability rather than a rate. The single-valued function that gives the probability of finding a particular set of molecular counts at a given time is called the Reaction Master Equation (RME) [19]. Its behavior in time can be formulated as a Markovian random walk in RME space. Although it is often impossible to solve an RME analytically, the most frequently used method being equivalent to numerically solving it is the Stochastic Simulation Algorithm (SSA), also known as the Gillespie Algorithm [20, 21]. Even though SSA can adequately account for the inherent discreteness and stochasticity of molecular interactions, it typically implies a high computational cost. Some faster methods were derived from approximations of SSA, such as the τ -Leaping method [22, 23]. However, in such cases, the efficiency can still be limited when considering bigger, more realistic biochemical models.

So far, we pointed out some important characteristics shared by many complex biochemical systems, such as different scales of reaction rates and molecular concentrations, discrete-valued number of molecules, and random fluctuations in reaction rates. Therefore, an ideal simulation method for such systems should be able to encompass all these features while being computationally efficient. Here we present an alternative method, henceforth called Constraint-Based Simulation Algorithm (CBSA), to simulating reaction systems focused on computational scalability while considering the system discreteness, stochasticity, and possible differences in time-scales. We

use a constraint-based formulation of the models following the well-established methodology applied to Flux Balance Analysis [24]. Then the simulation's computation is entirely performed using vector and sparse-matrix operations, which are highly parallelizable, thus being able to take advantage of GP-GPU architectures.

Although we do not perform a rigorous proof of the method, we show by using theoretical models that the numerical solutions of CBSA can be very close to the ones provided by the Stochastic Simulation Algorithm. Using a model of diffusion in a discrete toroidal-lattice space, we compare the computational efficiency between CBSA and some SSA implementations. The CBSA tends to outperform other algorithms in computational time for the case of large systems as a consequence of its ability to be computed on GPUs.

This manuscript is organized as follows. First, we introduce the Stochastic Simulation Algorithm, the τ -Leaping method, and the mathematical formulation of constraint-based (CB) models. Next, we discuss the flux estimation on CB models and its extension in order to accommodate a broader range of reactions. In Section 5, we describe the so-proposed Constraint-Based Simulations Algorithm, followed by an example in which we try to demonstrate its correctness in Section 6. Given that computational scalability is one of our main goals, in Section 8, we include a comparison between different implementations of SSA methods and the CBSA regarding their computational efficiency on large chemical systems using a diffusion model. In the last section, we provide some concluding remarks on this work.

2 The Stochastic Simulation Algorithm

Since its publication by Daniel Gillespie in [20], the Stochastic Simulation Algorithm (SSA) has been the standard approach to simulate stochastic systems. In the most used form of this algorithm, called the *Direct Method*, a well-mixed solution where M molecular species can react through R reaction channels is

assumed. Given an initial state X_0 for the number of copies of each molecular species, we define a time-step τ where only one of the R possible reactions can occur. For each reaction channel μ in R , we must calculate its propensity such as:

$$a_\mu = h_\mu c_\mu \quad (1)$$

where h_μ is the number of possible combinations between the reactant molecules of μ in the current state of the system $X(t)$. The c_μ is a reaction constant that can be understood as the probability of that reaction occurs in a period. It is related to the rate constant of the reaction k_μ and the volume of the system V as follows:

$$c_\mu = \frac{k_\mu}{V^{I_\mu-1}} \quad (2)$$

given that I_μ is the number of molecules reacting in r_μ . The simulation step is then performed by selecting a pair of μ and τ such as:

$$\begin{aligned} a_0 &= \sum_{i=1}^R a_i \\ \mu : \sum_{i=0}^{\mu-1} a_i &< r_1 a_0 < \sum_{i=0}^{\mu} a_i \\ \tau &= \frac{\ln(1/r_2)}{a_0} \end{aligned} \quad (3)$$

where r_1 and r_2 are random numbers drawn from a unitary uniform distribution. We then update the system state X by performing the reaction μ and then increase time t by τ . This procedure is repeated until a stopping criterion is reached.

Although SSA has been extensively tested and applied in the simulation of several stochastic systems, it tends to be a computationally expensive algorithm which may become prohibitive for very large systems [18]. There are some optimized formulations and approximation of the SSA aimed at improving computational performance [22]. In the following subsection, we present one of the most used approximation methods, the τ -Leaping method.

2.1 The τ -Leaping Method

The τ -Leaping method, also proposed by D. Gillespie [22], is an approximation of SSA which is considerably faster to compute. In this method, at each iteration, we leap in time by τ so that the changes in the propensity function a will be slight. Then, the number of occurrences k_μ in the time step τ for each reaction channel $\mu \in R$ will be sampled from the Poisson random distribution:

$$P(k; a_\mu, \tau) = \frac{e^{-a_\mu \tau} (a_\mu \tau)^k}{k!} \quad (k = 0, 1, 2, \dots). \quad (4)$$

3 The Constraint-Based Modeling

Let $m = \{m_1, m_2, \dots, m_M\}$ be a set of M molecular species and $r = \{r_1, r_2, \dots, r_R\}$ be a set of R reactions where molecules in m play roles as reactants and/or products. The stoichiometry of each reaction, *i.e.* how much of each molecular species is consumed or produced, can be represented by means of a stoichiometric matrix $S_{M \times R}$. Every entry S_{ij} yields how much of the molecular species i is consumed (negatively signed) or produced (positively signed) by the reaction j . If we consider a vector x of length M , where x_i is the number of m_i molecules, homogeneously distributed in a given volume V , the variation of x along time can be written as:

$$\frac{dx}{dt} = S \cdot v, \quad (5)$$

where v is a vector of fluxes and has length R . The flux v_i can be understood as how many times the reaction i occurs in a given period. The latter will henceforth be considered as 1 second.

Although v is, for now, an unknown variable, its values are constrained by well-known physical properties such as mass balance. Therefore, we can set the following constraints to the solution space:

$$x \geq 0, \quad (6)$$

which assure that there are no negative number of molecules, and

$$Sv + x \geq 0, \quad (7)$$

asserting that a given flux is valid if, and only if, it does not result in any negative number of molecules. An example of a solution space for two parallel reactions is depicted in Figure 1. The allowable space is upper-bounded the line that determines the mass-balance constraint.

Figure 1: Example of a solution space for a system with two reactions. The flux for each reaction is represented in the axes. Both reactions share the same reactant A, thus, the combination of fluxes v_1 and v_2 must satisfy the mass-balance constraint.

4 Flux Estimation

The flux $v_i(t)$ of a reaction r_i in a given instant t can be understood as how many times the reaction r_i can occur along the next second. Therefore, the flux of a given reaction can be any function $g(x)$ multiplied by the total number of combinations between the reactants. For the sake of simplicity, we will consider $g(x)$ as a constant c_i for each reaction. Thus, following the same reasoning used in SSA to estimate a , we can write $v_i(t + \Delta t)$ as follows:

$$v_i(t + \Delta t) = c_i \prod_{j \forall S_{j,i} < 0} \frac{\binom{x_j(t)}{-S_{j,i}}}{-S_{j,i}!} \Delta t. \quad (8)$$

As we use the stoichiometry matrix to estimate v , the flux will be correct only if all reactants are different from products for all reactions. This is because S cannot yield information about the molecular species

which counts does not change by the end of the reaction, even being necessary in the process. This is the case of catalysts, modifiers, or auto-catalytic reactions (*e.g.* $A \rightarrow A + A$). Thus, to represent such cases, we will define a regulation matrix $R_{N \times R}^*$ where $R_{i,j}^* = \gamma$ if γm_i molecules are needed in reaction r_j . Then, we can rewrite Eq. 8 as:

$$v_i(t + \Delta t) = c_i \prod_{j \forall S_{j,i} < 0} \frac{\binom{x_j(t)}{-S_{j,i}}}{-S_{j,i}!} \prod_{j \forall R_{j,i}^* > 0} \frac{\binom{x_j(t)}{R_{j,i}^*}}{R_{j,i}^*!} \Delta t \quad (9)$$

5 The Constraint-Based Simulation Algorithm – CBSA

In the last sections, we presented the most used algorithm applied to the stochastic simulation of chemical reactions called SSA, a mathematical formalism to describe such reactions and their behavior as a system of constraints, named Constraint-Based Modeling, and an expression to calculate fluxes in each reaction channel. Now, we will use these concepts to formulate a scalable algorithm that aims at simulating chemical reaction systems while considering their inherent discreteness and stochasticity.

Given an initial condition, each iteration of the algorithm is divided into five steps: 1) calculate the flux of each reaction based on the current system state; 2) add noise; 3) check if the calculated fluxes satisfy the mass-balance constraints (Eq. 7); 4) if mass-balance constraint not satisfied, find a valid set of fluxes; and 5) update the molecular counts. Figure 2 illustrates the algorithm scheme. These steps must be repeated until a given stopping criterion is reached, such as a maximum time.

The following paragraphs describe in detail each step of the algorithm.

Initial Parameters The chemical system is defined by the stoichiometric matrix S , which describes the reactants and products of every reaction, and the regulation matrix R^* that indicates other molecules

Figure 2: Flowchart of the Constraint-Based Simulation Algorithm.

necessary to given reactions but without being consumed nor produced in the process. Each reaction must also have an associated constant c , which is the fraction of all combinations between its reactant molecules that would indeed result in a reaction by the time of one second. Also, the initial state of the system x_0 must be provided, defining the initial molecular count for every molecular species. Finally, we need to set a maximum time-step length Δt_{max} .

Adding Noise to Fluxes In order to account for fluctuations which might influence the velocity of reactions, specially in the case of lower molecular counts, we will add noise to the calculated flux on each step using the chemical Langevin Equation [25, 22]. This equation introduces a Gaussian white noise η in the flux v (Eq. 13) as:

$$\begin{aligned} v_i &= v'_i + \eta_i \\ \eta_i &= \sqrt{v'_i} \mathcal{N}(0, 1) \end{aligned} \quad (10)$$

where $\mathcal{N}(0, 1)$ is a random variable drawn from the normal distribution with mean zero, and standard deviation one. The variable v'_i is equal to $v_i + v_{dec_i}$ which will be introduced in the next paragraph.

Discrete Flux Calculation In order to account for the inherent discreteness of the system, the flux v must be always integer. However, Eqs. 9 and 10 can output decimal results. In order to keep $v \in \mathbb{N}$, we will separate the integer from the decimal part and store the later in v_{dec} for the next iteration. Then, v_{dec} and v at each iteration can be calculated using the following operations:

$$\begin{aligned} v_{dec_i} &= v_i - \lfloor v_i \rfloor \\ v_i &= v_i - v_{dec_i} \end{aligned} \quad (11)$$

By storing the decimal part in v_{dec_i} , we assure that reactions in which $v_i < 1$ will occur at a delayed time when $v_i \geq 1$.

Finding a Valid Flux Although each v_i is mostly valid for every single reaction, this might not hold when considering the whole system. To illustrate such a case, consider a system in which a given molecular species m_k is a reactant for many different reactions. Once the estimation of each v_i is independent of the others, the total amount of m_k required for all reactions might exceed the pool x_k and violate the mass-balance constraint from Eq. 7. It may also happen in the case of $c_i \gg 1$. To avoid such condition, if v fails the mass-balance constraint, we may look for the maximal Δt that assure non-negative values on $x(t + \Delta t)$. We do that by multiplying the current Δt by a constant $0 < \alpha < 1$, recalculating the fluxes, updating the decimal part v_{dec} , and testing the mass-balance constraint until a valid solution is found, as shown in Eq. 12.

$$\begin{aligned} \text{while } (x + S \cdot v < 0) : \\ \Delta t &= \alpha \Delta t \end{aligned} \quad (12)$$

In the worst-case scenario, all v_i will reach zero, and another calculation step begins restoring Δt to its initial value.

Molecular Count Update Having a discrete flux v passing the mass-balance constraint stated in Eq. 7, we simply update the molecular counts by solving Eq. 5 using Euler’s method. The update equation becomes:

$$x(t + \Delta t) = x(t) + S \cdot v \quad (13)$$

6 Biochemical System Simulation

In order to test the correctness of the solutions provided by the CBSA, we use here a reaction system that can demonstrate several of the important characteristics of real biochemical systems, such as different magnitudes of molecular counts and reaction velocities, and regulated reactions while demonstrating the method’s capabilities and limitations. Other examples can be found in the Supplementary Information.

Let us consider the reaction system $\{1\}$ where a Transcription Factor (Tf) triggers the production of a protein P when active. The Transcription Factor Tf is activated and deactivated spontaneously by a slow reaction. The activated Tf triggers a swift production of hundreds of protein P copies, which are spontaneously degraded. With the system starting with only one Tf molecule, it should transit between its activated and inactivated states, triggering bursts on the production of P (Fig. 3a).

The simulation depicted in Figure 3a was performed using the CBSA. The biochemical system is

represented employing the stoichiometric matrix S and the regulation matrix R^* . As S can only represent the reaction’s final stoichiometry without accounting for intermediate molecules, we must represent the catalytic Tf_{active} molecule in the regulation matrix R^* as follows:

$$S = \begin{bmatrix} r_1 & r_2 & r_3 & r_4 \\ -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix} \begin{matrix} Tf_{\text{active}} \\ Tf_{\text{inactive}} \\ P \end{matrix} \quad (14)$$

$$R^* = \begin{bmatrix} r_1 & r_2 & r_3 & r_4 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{matrix} Tf_{\text{active}} \\ Tf_{\text{inactive}} \\ P \end{matrix} \quad (15)$$

where r_1 to r_4 are the respective reaction channels with rate constants c_1 to c_4 .

The times between activation and deactivation of Tf depend solely in the rate constants c_1 and c_2 . Thus, we could estimate these times as $1/c_1$ and $1/c_2$. In the example depicted in Figure 3a we adopted $c_1 = c_2 = 0.05$ making the transition times between activate and inactivate states equals to 20 seconds. However, the system is stochastic and the observed transition times will compose a distribution rather than an exact value. In Figure 3b we can observe the distribution of the switching times between states of Tf with the mean close to the expected value of 20 seconds. Given any equal values for c_1 and c_2 and the initial condition with only one Tf molecule, we can estimate that the mean value for both activated and inactivated states should converge to 0.5 with $t \rightarrow \infty$. This behavior is indeed observed in Figure 3c.

6.1 Validation Respectively to the Stochastic Simulation Algorithm

The SSA is already a well-established method to perform stochastic simulations of chemical systems. Thus, we use it as a gold-standard method in order to evaluate the correctness of the simulations performed using the proposed Constraint-Based Simulation Algorithm.

Regarding the reaction system described in $\{1\}$, we already observed agreement between its simulations

Figure 3: **(a)** Simulation of the chemical system described in {1} using the CBSA with c_1 , c_2 , c_3 , and c_4 equals to 0.05, 0.05, 200, and 0.5 respectively. The initial state of x contains only one $Tf_{inactive}$ molecule. Every time that $Tf_{inactive}$ turns into Tf_{active} , a burst in P is observed. **(b)** With $c_1 = c_2 = 0.05$, the Tf should transit between states every 20 seconds in average. We can see the distribution of switching times with mean 19.09 and standard deviation 17.06 obtained from a simulation of 100,000 seconds. **(c)** The cumulative mean of Tf_{active} , named $\langle Tf_{act} \rangle$, at a given time t corresponds to the mean value of Tf_{active} in the interval $[0, t]$. Once $c_1 = c_2$, they have the same transition probability, thus, $\langle Tf_{act} \rangle$ should converge to 0.5 with $t \rightarrow \infty$. Likewise, the cumulative mean of P should converge to 200 once the number of P molecules peaks at around 400 when Tf is active, and it is approximately for half of the time. The parameter α was fixed as 0.5 for all CBSA simulations.

using CBSA and theoretical mean values expected, as shown in Figure 3. Now, we shall compare these results with trajectories obtained from simulations using the original Stochastic Simulation Algorithm and its approximated form, the τ -leaping. Though all methods are stochastic, it is not necessary to perform several runs of the same experiment to obtain statistics about the model because this is a particular oscillatory system and does not diverge at any point in time. Thus, we can make consistent statistics from the time-series with sufficiently long simulation time.

The first comparison we make regards the distribution of switching times between both states of Tf . In Figure 4a we can see that the distributions for the three considered methods have the same exponential-like shape with similar means and standard deviations. With the SSA and τ -leaping method, we obtained the mean switching times closer to the theoretical value. It is also observed that the CBSA allowed

a lower standard deviation than the other methods. We believe this is a consequence of the constraints imposed on the model after the addition of noise, which can reject nonviable solutions. In this sense, the constraints may introduce a bias in the distribution from which the random numbers are drawn mainly affecting the standard deviation of reaction times.

A consequence of the lower standard deviation of reaction times can be observed in the time that the system takes to converge the mean number of molecules to the theoretical value. The mean values of both P and Tf_{active} converge faster to the expected values as depicted in Figure 4b. Another limitation of the methods is that it does not necessarily lead to a linear relationship between the time step adopted in the Euler integration and the error obtained when compared to the theoretical value. The main reason for this possible non-linear relationship between time step and error is that the noise η_i added

Figure 4: **(a)** Distribution of Tf switching times for the CBSA, SSA, and τ -leaping methods with a simulation time of 100,000 seconds. Means and standard deviations indicated in the legend. **(b)** The cumulative mean of P and Tf_{active} for the three methods.

to each flux v_i goes with the square-root of the flux. So, when $v_i < 1$, the smaller the value that v_i takes, the greater is the noise η_i introduced. Thus, by reducing the integration step Δt we increase the chance of obtaining a flux $v_i < 1$ and the inverted noise proportion.

Despite the differences in the standard deviation of reaction velocities, all methods agreed regarding the qualitative behavior of the system and molecular counts. Other examples can be found in the Supplementary Information where similar conclusions are obtained. We also discuss other aspects of the algorithm such as the influence of Δt and α in the results, relative errors, and in the number of steps to achieve the simulation time.

7 Accelerated Computing of The CBSA

One of the main constraints of the Stochastic Simulation Algorithm is its computational cost. The algorithm performs only one reaction for each time step and, depending on the size of the system and the rate constants, it may demand a very large number of steps to achieve the desired simulation time. The τ -leaping approximation and some variations of it can reduce the number of steps in some cases, usually being adopted for bigger chemical systems.

Another limitation of SSA is that the way the algorithm is designed makes it difficult to be implemented for parallel environments. Although there are implementations of the SSA to be computed using many computer cores, or even using General-Purpose Graphics Processing Units (GP-GPUs) [26, 27, 28], they can only accelerate replicates of the same model, reducing the time to compute several trajectories, but still limited regarding the size of the chemical system. There are also implementations for computing the SSA in a discretized space [29] where the subsystems in each subspace can be performed in parallel.

The CBSA is almost entirely based on vector and matrix operations, being fast to compute and easy to implement. Also, considering that the matrices S and R^* are often highly sparse, a more sophisticated implementation can take advantage of this feature to further enhance computations regarding both processing time and memory usage. A description of the sparse implementation can be found in the Supplementary Information.

Parallel computing was a design principle when developing the reported method. All operations in the CBSA can be computed in parallel thanks to the relative independence of the operations through the vectors and matrices. Therefore, we could take advantage of the massive computing power of GP-GPUs to accelerate individual simulations of large chemical systems.

8 Computing Time Benchmark

One of the goals of the present work is to develop a simulation method that is computationally scalable to a variety of chemical system sizes. By size, we mean both the total number of molecules and the number of reactions. We shall then compare, for some cases, the CBSA’s computational expense against the SSA and some of its implementations, and also an ODE solver.¹ For the SSA, we use the implementations from three Python libraries: StochPy² [30], STEPS³ [31], and GillesPy2⁴ [32]. The ODE implementation is part of the latter library. We use pyOpenCL⁵ to distribute computations of the CBSA on GPUs.

8.1 Benchmark Model

We use as a benchmark model a diffusion process through a discrete toroidal square-lattice space of length L , as depicted in Fig. 5a. In this model, reactions will represent transitions between neighbor sub-spaces with a diffusion coefficient k_{diff} . Therefore, a system with a space of length L yield $4L^2$ reactions and L^2 molecular species. The system will start with an initial amount of molecules in only one of the sub-spaces and they will diffuse with a rate k_d .

8.2 Overall Performance Comparison

Given that the CBSA is designed to simulate large chemical systems, we first set two distinct scenarios using the diffusion model (a small and a large system) to compare the computational cost of the CBSA with other methods and implementations. The hypothesis is that the larger the system, the more impact the effect of matrix sparseness will have in reducing the CBSA computation time. For the small system case, we considered a system with $L = 2$, 10 initial molecules, and $k_d = 10.0$. For the large

system we set $L = 16$, 10240 initial molecules, and $k_d = 1.0$. In Figure 5b we can see the mean computation time of 10 replicates for each of the considered methods in both small and large system cases. For the small system case, the sequential implementation of CBSA performed similarly to all SSA implementations except for STEPS. It is also observed that the τ -leaping implementations have no improvement of performance for such cases. On the other hand, the GPU implementation of CBSA obtained the worst performance for the small system. The reason for such limited performance is that GPU computations have a computational overhead due to memory transfers between host and device and also a lower clock speed. However, when the system is large enough, the efficiency gained with the computation distributed through the hundreds of GPU’s cores overcomes the memory transfer times making it marginal. This situation can be observed when considering the large system. Besides the higher efficiency of the sequential CBSA compared to the SSA methods, the gain in computation time is even better when using the GPU. For the large system, some methods were not able to conclude due to excessive use of the system’s memory or reached the maximum time established.

8.3 Parameter Scaling Comparison

To better understand how each variable of the diffusion model affects computation times, we performed simulations varying the square-lattice length L , the initial number of molecules, the diffusion rate k_d , and the total simulation time. Regarding the system size, all the SSA implementations and the sequential CBSA showed an exponential growth in computing time as the value of L increased (Fig. 5c). On the other side, the GPU implementation of CBSA obtained a much lower growth rate in computational time as the size of the system grows, also performing better than all other methods when $L \geq 32$. Similar exponential growth in computation time for all SSA implementations, except the StochPy Tauleap, was obtained when increasing the number of initial molecules in the system (Fig. 5d). However, the computational time is invariant to the number of molecular copies for both CBSA implementations as well

¹All simulations were performed using the same computational architecture: Linux Debian 10 on an Intel i7-5930K, 64GB of DDR4 RAM, and NVidia GTX 1650 4GB.

²StochPy: <http://stochpy.sourceforge.net/>

³StochPy: <http://http://steps.sourceforge.net/>

⁴GillesPy2: <https://pypi.org/project/gillespy2/>

⁵pyOpenCL: <https://pypi.org/project/pyopencl/>

as for the ODE solver.

It is expected in the SSA that the higher the reaction rates are, the more steps it takes to achieve a given simulation time. In Figure 5e we can observe that all SSA implementations present an exponential growth in the computation time as the diffusion constant is increased, with the τ -leaping implementations presenting a lower growth rate. The CBSA implementations only showed an increased computational time for the higher diffusion rate considered. This can happen when several reactions compete for the same reactant and the sum of their fluxes is higher than the available amount of the molecule. Thus, when several fast reactions share the same reactant, the CBSA needs to decrease the time step Δt implicating in a higher number of steps to be computed in order to achieve a given simulation time. The variations in the time steps used in the CBSA is discussed in more detail in the Supplementary Information. For all methods, the computing time has a linear relationship with the total time of the simulation (Fig. 5f).

8.4 Some Observations about the Simulation

It is worth mentioning that one of the reasons for the increased performance of STEPS' implementation in all the above-mentioned analyses is that it uses a limited number of random numbers generated before the actual simulation. One effect of this implementation choice can be observed when analyzing the variance of the molecular counts at the end of the simulations. In Figures 5g and 5h we can see a Principal Component Analysis (PCA) where the final molecular counts of all simulations are projected in a lower dimension space for two different number of initial molecules in the system. Considering that the simulation using ODE can be equivalent in some cases to the mean of several runs of a stochastic method, we can observe that in both PCA plots that the solutions provided by CBSA and SSA are scattered around the single solution provided by the ODE solver, with a lower scattering when the number of molecules in the system is higher. However, in Figure 5g we can observe only 2 out of 10 points simulated by STEPS because they are overlapping.

This result suggests that the limited number of random values has an impact on the stochasticity of the method, mostly when a lower number of molecules in the system is considered. It is also noticed that the solutions provided by the StochPy Tauleap are more scattered in Figure 5h than all other methods, which combined with the no variation in computation time might indicate an inconsistency in the solutions for a high number of molecules in this particular reaction system.

Although it is not the intent of this work to deliver a fully-featured software, both sequential and parallel implementations of CBSA, along with several other examples, are freely available at our GitHub repository⁶. The implementation and examples are also discussed in the Supplementary Information.

⁶CBSA: <https://github.com/pauloburke/CBSA>

Figure 5: Benchmark using a diffusion model through a square-lattice discretized space with periodic boundary conditions. **(a)** Simulation of the system with length $L = 5$ and 100 initial molecules using CBSA. Each plot depicts the distribution of the diffused molecules in the grid for the times $t = \{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$. **(b)** Simulation times in logarithmic scale for a small and a large system. The parallel implementation of the CBSA is denoted as “CBSA OpenCL”. Considering that the CBSA is designed to simulate large systems, we selected two different scenarios where we estimate we could get the worst and the best performance of the algorithm when compared to the other considered methods. The small system has the parameters $L = 2$, 10 initial molecules, and $k_d = 10.0$. The large system has the parameters $L = 16$, 10240 initial molecules, and $k_d = 1.0$. In the case of the large system, the methods marked with a red star were not able to finish the simulations because of system memory shortage (specific to the used machine) or for reaching the maximum computation time of 10 minutes per replicate. All results shown are an average of 10 replicates. In **(c)**, **(d)**, **(e)**, and **(f)** the computing time for varying number of reactions, number of initial molecules, diffusion rate k_d , and total time of simulation are depicted respectively. The fixed parameters are $L = 4$, 100 initial molecules, $k_d = 0.1$, and total simulation time of 300 seconds. In **(g)** and **(h)** we plot the Principal Component Analysis for results obtained from simulations using 100 and 1,000,000 initial molecules in the system. Given that $L = 4$ used PCA to reduce the dimension of the vector of molecule counts x from 16 to 2 in order to be able to visualize the data. The variance explained in each axis is indicated in their respective labels.

9 Concluding Remarks

Chemical reaction models are becoming more complex, especially when representing biochemical systems. Such models might currently involve up to thousands of molecular species and reaction channels becoming computationally challenging for current simulation methods. Also, some important characteristics should be appropriately handled on simulations of biochemical systems, such as the discreteness of molecular counts and the stochasticity of reaction occurrences.

Here we report a simulation method called Constraint-Based Simulation Algorithm – CBSA, which employs constraint-based modeling of chemical reactions in order to simulate large systems efficiently. We showed with theoretical examples that it could provide solutions similar to those produced by the most used method, the Stochastic Simulation Algorithm. It also yielded particularly good performance when simulating several system configurations, especially for a high number of molecules and reactions, thus fulfilling one of the initial motivations of obtaining a computationally scalable method.

Besides the benefits of CBSA already discussed in this work, we may point out other possible outcomes from this method. The so employed constraint-based modeling is widely used in the representation of biochemical systems. It should then be straightforward to apply CBSA on the simulation of virtually any model available in databases such as BiGG Models [33], and BioModels [34]. The practical implications of its use on real models should be assessed in future works. Furthermore, considering that the presented method was conceived with large biochemical models in mind, it can be a useful tool given the current advances in whole-cell modeling and simulation.

Acknowledgements

PEPB thanks Bruno Messias and Thaís G. Pedrosa for the insightful discussions regarding this work.

Funding

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. LdaFC also thanks CNPq (Grant No. 307085/2018-0) and FAPESP (Grant No. 015/22308-2) for support.

References

- [1] B. Kholodenko, M. B. Yaffe, and W. Kolch, “Computational Approaches for Analyzing Information Flow in Biological Networks,” *Science Signaling*, vol. 5, pp. re1–re1, apr 2012.
- [2] X. Z. Jiang, M. Feng, K. H. Luo, and Y. Ventikos, “Large-scale molecular dynamics simulation of flow under complex structure of endothelial glycocalyx,” *Computers and Fluids*, vol. 173, pp. 140–146, 2018.
- [3] B. Di Ventura, C. Lemerle, K. Michalodimitrakakis, and L. Serrano, “From in vivo to in silico biology and back,” *Nature*, vol. 443, pp. 527–533, oct 2006.
- [4] G. W. Brodland, “How computational models can help unlock biological systems,” *Seminars in Cell & Developmental Biology*, vol. 47-48, pp. 62–73, dec 2015.
- [5] O. Purcell, B. Jain, J. R. Karr, M. W. Covert, and T. K. Lu, “Towards a whole-cell modeling approach for synthetic biology,” *Chaos*, vol. 23, no. 2, 2013.
- [6] U. Muenzner, E. Klipp, and M. Krantz, “A comprehensive, mechanistically detailed, and executable model of the Cell Division Cycle in *Saccharomyces cerevisiae*,” *Nature communications*, vol. 10, no. 1, p. 1308, 2019.
- [7] D. N. Macklin, T. A. Ahn-Horst, H. Choi, N. A. Ruggero, J. Carrera, J. C. Mason, G. Sun, E. Agmon, M. M. DeFelice, I. Maayan, K. Lane, R. K. Spangler, T. E. Gillies, M. L. Paull, S. Akhter, S. R. Bray, D. S. Weaver, I. M. Keseler, P. D.

- Karp, J. H. Morrison, and M. W. Covert, “Simultaneous cross-evaluation of heterogeneous E. coli datasets via mechanistic simulation,” *Science*, vol. 369, no. 6502, 2020.
- [8] R. Mahadevan, J. S. Edwards, and F. J. Doyle, “Dynamic Flux Balance Analysis of diauxic growth in Escherichia coli,” *Biophysical Journal*, vol. 83, no. 3, pp. 1331–1340, 2002.
- [9] S. K. Hahl and A. Kremling, “A comparison of deterministic and stochastic modeling approaches for biochemical reaction systems: On fixed points, means, and modes,” *Frontiers in Genetics*, vol. 7, no. AUG, 2016.
- [10] W. Gottstein, B. G. Olivier, F. J. Bruggeman, and B. Teusink, “Constraint-based stoichiometric modelling from single organisms to microbial communities,” *Journal of The Royal Society Interface*, vol. 13, no. 124, p. 20160627, 2016.
- [11] M. Slak Rupnik, M. Marhl, M. Gosak, J. Dolenšek, R. Markovič, M. Perc, and A. Stožer, “Network science of biological systems at different scales: A review,” *Physics of Life Reviews*, vol. 24, pp. 118–135, 2017.
- [12] T. G. Kurtz, “The relationship between stochastic and deterministic models for chemical reactions,” *The Journal of Chemical Physics*, vol. 57, no. 7, pp. 2976–2978, 1972.
- [13] A. Fernández-Ramos, J. A. Miller, S. J. Klippenstein, and D. G. Truhlar, “Modeling the Kinetics of Bimolecular Reactions,” *Chemical Reviews*, vol. 106, pp. 4518–4584, nov 2006.
- [14] N. Ishii, Y. Suga, A. Hagiya, H. Watanabe, H. Mori, M. Yoshino, and M. Tomita, “Dynamic simulation of an in vitro multi-enzyme system,” *FEBS Letters*, vol. 581, no. 3, pp. 413–420, 2007.
- [15] K. Peskov, E. Mogilevskaya, and O. Demin, “Kinetic modelling of central carbon metabolism in Escherichia coli,” *FEBS Journal*, vol. 279, pp. 3374–3385, sep 2012.
- [16] J. Min Lee, E. P. Gianchandani, J. A. Eddy, and J. A. Papin, “Dynamic analysis of integrated signaling, metabolic, and regulatory networks,” *PLoS Computational Biology*, vol. 4, no. 5, 2008.
- [17] D. A. McQuarrie, “Stochastic approach to chemical kinetics,” *Journal of Applied Probability*, vol. 4, pp. 413–478, dec 1967.
- [18] D. T. D. T. Gillespie, “Stochastic Simulation of Chemical Kinetics,” *Annual Review of Physical Chemistry*, vol. 58, pp. 35–55, jan 2007.
- [19] D. T. Gillespie, “A rigorous derivation of the chemical master equation,” *Physica A: Statistical Mechanics and its Applications*, vol. 188, pp. 404–425, sep 1992.
- [20] D. T. Gillespie, “A general method for numerically simulating the stochastic time evolution of coupled chemical reactions,” *Journal of Computational Physics*, vol. 22, no. 4, pp. 403–434, 1976.
- [21] D. T. Gillespie, “Exact stochastic simulation of coupled chemical reactions,” *Journal of Physical Chemistry*, vol. 81, no. 25, pp. 2340–2361, 1977.
- [22] D. T. Gillespie, “Approximate accelerated stochastic simulation of chemically reacting systems,” *Journal of Chemical Physics*, vol. 115, no. 4, pp. 1716–1733, 2001.
- [23] Y. Cao, D. T. Gillespie, and L. R. Petzold, “Efficient step size selection for the tau-leaping simulation method,” *Journal of Chemical Physics*, vol. 124, no. 4, 2006.
- [24] J. D. Orth, I. Thiele, and B. O. Ø. Palsen, “What is Flux Balance Analysis ?,” *Nature Biotechnology*, vol. 28, no. 3, pp. 245–248, 2010.
- [25] D. T. Gillespie, “The chemical Langevin equation,” *The Journal of Chemical Physics*, vol. 113, pp. 297–306, jul 2000.
- [26] I. Komarov and R. M. D’Souza, “Accelerating the Gillespie Exact Stochastic Simulation Algorithm Using Hybrid Parallel Execution on

- Graphics Processing Units,” *PLoS ONE*, vol. 7, no. 11, pp. 1–9, 2012.
- [27] M. S. Nobile, P. Cazzaniga, D. Besozzi, D. Pescini, and G. Mauri, “cuTauLeaping: A GPU-Powered Tau-Leaping Stochastic Simulator for Massive Parallel Analyses of Biological Systems,” *PLoS ONE*, vol. 9, p. e91963, mar 2014.
- [28] K. Sumiyoshi, K. Hirata, N. Hiroi, and A. Funahashi, “Acceleration of discrete stochastic biochemical simulation using GPGPU,” *Frontiers in Physiology*, vol. 6, no. FEB, pp. 1–10, 2015.
- [29] W. Chen and E. de Schutter, “Parallel STEPS: Large scale stochastic spatial reaction-diffusion simulation with high performance computers,” *Frontiers in Neuroinformatics*, vol. 11, no. February, pp. 1–15, 2017.
- [30] T. R. Maarleveld, B. G. Olivier, and F. J. Bruggeman, “StochPy: A comprehensive, user-friendly tool for simulating stochastic biological processes,” *PLoS ONE*, 2013.
- [31] I. Hepburn, W. Chen, S. Wils, and E. De Schutter, “STEPS: Efficient simulation of stochastic reaction-diffusion models in realistic morphologies,” *BMC Systems Biology*, vol. 6, 2012.
- [32] J. H. Abel, B. Drawert, A. Hellander, and L. R. Petzold, “GillesPy: A Python Package for Stochastic Model Building and Simulation,” *IEEE Life Sciences Letters*, vol. 2, no. 3, pp. 35–38, 2017.
- [33] P. Miller, N. E. Lewis, A. Dräger, J. A. Lerman, Z. A. King, S. Federowicz, B. O. Palsson, J. Lu, and A. Ebrahim, “BiGG Models: A platform for integrating, standardizing and sharing genome-scale models,” *Nucleic Acids Research*, vol. 44, no. D1, pp. D515–D522, 2015.
- [34] V. Chelliah, N. Juty, I. Ajmera, R. Ali, M. Dumousseau, M. Glont, M. Hucka, G. Jalowicki, S. Keating, V. Knight-Schrijver, A. Lloret-Villas, K. N. Natarajan, J. B. Pettit, N. Rodriguez, M. Schubert, S. M. Wimalaratne, Y. Zhao, H. Hermjakob, N. Le Novère, and C. Laibe, “BioModels: Ten-year anniversary,” *Nucleic Acids Research*, vol. 43, no. D1, pp. D542–D548, 2015.