BioSystems 162 (2017) 135–146

Contents lists available at ScienceDirect

BioSystems

journal homepage: www.elsevier.com/locate/biosystems

Slow update stochastic simulation algorithms for modeling complex biochemical networks

Debraj Ghosh, Rajat K. De*

Machine Intelligence Unit, Indian Statistical Institute, Kolkata 700108, India

ARTICLE INFO

Article history:
Received 26 October 2016
Received in revised form 17 June 2017
Accepted 20 October 2017

Keywords:
Gillespie algorithm
B cell receptor signaling network
FcϵRI signaling network
Stochastic modeling

ABSTRACT

The stochastic simulation algorithm (SSA) based modeling is a well recognized approach to predict the stochastic behavior of biological networks. The stochastic simulation of large complex biochemical networks is a challenge as it takes a large amount of time for simulation due to high update cost. In order to reduce the propensity update cost, we proposed two algorithms: slow update exact stochastic simulation algorithm (SUESSA) and slow update exact sorting stochastic simulation algorithm (SUESSSA). We applied cache-based linear search (CBLS) in these two algorithms for improving the search operation for finding reactions to be executed. Data structure used for incorporating CBLS is very simple and the cost of maintaining this during propensity update operation is very low. Hence, time taken during propensity updates, for simulating strongly coupled networks, is very fast; which leads to reduction of total simulation time. SUESSA and SUESSSA are not only restricted to elementary reactions, they support higher order reactions too.

We used linear chain model and colloidal aggregation model to perform a comparative analysis of the performances of our methods with the existing algorithms. We also compared the performances of our methods with the existing ones, for large biochemical networks including B cell receptor and FcϵRI signaling networks.

1. Introduction

The dynamic behavior of large complex biochemical networks introduces difficulties in understanding the state space of entities and their interactions. In order to overcome these difficulties, some mathematical modeling techniques including ordinary differential equations (ODEs) have been introduced. ODEs have extensively been used for capturing the time evolution of a biochemical system. However, ODE based methods do not exhibit the system's stochastic behavior which has been proven to be an important feature depending on the nature of the system. The stochastic behavior can exhibit the mechanism of complex biochemical processes, for example, cell to cell variations of a system depending on the copy numbers of the species in the system (Cao et al., 2004). Stochastic simulation algorithms (SSAs), including direct method (DM) and first reaction method (FRM), have been introduced for capturing the stochastic behavior of a biochemical network (Gillespie, 1976, 1977). In DM and FRM, a biochemical network is assumed to be generated through a well stirred (spatially homogeneous) biochemical system in which each reaction is associated with a parameter termed as propensity. The propensity of each reaction is proportional to the probability of occurrence of that reaction. Based on the parameter propensity, DM and FRM decide which reaction to occur and when to occur. The computational complexities of these two methods are extremely high for simulating large complex biochemical networks. In order to get rid of this issue, several algorithms have been developed.

A priority queue based algorithm, called next reaction method (NRM), has been developed, which is an improved version of FRM (Gibson and Bruck, 2000). In NRM, the computational cost has been reduced by introducing a dependency graph for propensity updates. Another improved version, named as optimized direct method (ODM), has been developed in Cao et al. (2004). ODM sorts the reactions of the network, based on their rates, by performing a pre-simulation. The computational costs towards propensity updates in ODM are less compared to NRM, as NRM needs extra cost of maintaining the priority queue. Sorting direct method (SDM), another version of DM, has been introduced in McCollum et al. (2006). It does not need any pre-simulation, rather it maintains an array holding the reaction indexes and sorts it gradually, based on the reaction rates, during simulation. Some approximation methods, including τ-leaping, R-leaping, K-leap, L-leap,

* Corresponding author.
E-mail address: rajat@isical.ac.in (R.K. De).

https://doi.org/10.1016/j.biosystems.2017.10.011
0303-2647/

slow-scale method, k_α-leaping and implicit τ-leaping methods, have been introduced in Gillespie (2001), Gillespie (2007), Gillespie and Petzold (2003), Cao et al. (2005a,b, 2006), Peng et al. (2007), Auger et al. (2006), Peng and Wang (2007), Cai and Xu (2007), and Rathinam et al. (2003). These approximation methods reduce the duration of the total simulation time by jumping over the sequences of less critical reactions in the reaction firing event.

The methods including ODM and SDM have computational complexities of the order of the number of reactions, and they are basically suitable for modeling weakly coupled networks, e.g., the linear chain model. However, the biochemical networks are mostly complex and strongly coupled in which the number of reactions are much higher than the number of species. Therefore, the methods having computational complexities of the order of the number of reactions take longer simulation times for modeling strongly coupled networks. In order to overcome this issue, Ramaswamy et al. have introduced the partial-propensity direct method (PDM) and sorting partial-propensity direct method (SPDM) in Ramaswamy et al. (2009), which have reduced the computational scaling upto the order of the number of species, and thereby, achieved a significant speed up in execution time for simulating strongly coupled networks. Some attempts on delay stochastic simulation algorithm (DSSA) which is used for describing transcription and translation processes of gene regulatory systems have been made in Barrio et al. (2006, 2013), Leier et al. (2014), and Leier and Marquez-Lago (2015).

A tree-based search algorithm has been introduced in Thanh and Zunino (2014). This algorithm is based on Huffman tree which has minimized the search depth of finding the next reaction to occur. A composition-rejection based algorithm, viz., SSA-CR, has been introduced, for which computational cost is independent of network size (Slepoy et al., 2008). A partial-propensity method combined with composition rejection (PSSA-CR) has been developed in Ramaswamy and Sbalzarini (2010). In weakly coupled networks, SSA-CR and PSSA-CR execute faster than PDM and SPDM, whereas these composition-rejection based methods take longer execution time in the case of strongly coupled networks. An exact rejection based stochastic simulation algorithm (SSA), called RSSA, has been developed in Thanh et al. (2014). RSSA executes the reactions based on pre-computed propensity bounds, and it avoids frequent propensity updates without compromising the exactness of SSA. Some modified versions of RSSA, called RSSA-Binary, RSSA-Lookup and SRSSA, have been introduced in Thanh et al. (2015). A tree based search operation is maintained in RSSA-Binary, whereas RSSA-Lookup is based on look up table search for finding the reaction to be executed next. SRSSA creates several independent trajectories simultaneously in a single execution run. Another approach involving infrequent propensity updates, called lazy updating method, has been introduced, which has been applied to the sorting direct method (Ehlert and Loewe, 2014).

In this paper, we have developed two stochastic simulation algorithms, viz., slow update exact stochastic simulation algorithm (SUESSA) and slow update exact sorting stochastic simulation algorithm (SUESSSA) for modeling strongly coupled networks. SUESSA and SUESSSA do not rely on partial propensities like PDM and SPDM which are limited to elementary reactions, and they can model networks with higher order reactions. The main advantage of these two algorithms is that they are equipped with fast propensity update operations. For this purpose, we have applied propensity bound infrequent propensity update technique for reducing the number of propensity updates. We have used efficient data structures for reducing their propensity update costs. The search operation for finding the next reaction has been improved by applying cache-based linear search technique. We have used linear chain model, colloidal aggregation model and two large biochemical networks, including B cell receptor signaling network and FcϵRI signaling network for comparing the simulation times of our methods with other SSAs.

The organization of the paper is as follows. We discuss our proposed algorithms in Section 2. The comparative performance analysis and validation of our methods with the existing ones have been discussed in Section 3 and it is followed by discussion in Section 4.

2. Methodology

Let us consider a biochemical system of M molecular species $C_1, \ldots, C_M$, with state vector $\boldsymbol{X}(t) = [X_1(t), \ldots, X_M(t)]^T$ representing the number of molecules (populations) of the species at time t. These species form a biochemical network through their conversions/reactions. There are N reactions $R_1, R_2, \ldots, R_N$ in the network. The system is well stirred (spatially homogeneous) in order to focus only on the populations of the chemical species rather than their individual positions in the system. Each reaction R_i is associated with a parameter, called stochastic rate constant k_i and the corresponding stoichiometry. The probability of occurrence of the reaction R_i at time t within a small duration dt is defined by a parameter $a_i(\boldsymbol{X}(t))dt$, where $\boldsymbol{X}(t) = \boldsymbol{x}$ (Cao et al., 2004; McCollum et al., 2006). The parameter a_i is called the propensity of the reaction R_i, which is the product of the substrate populations and the stochastic rate constant.

The system dynamics is characterized by the chemical master equation (CME) (Gillespie, 1976, 1977) which is given by

$$\frac{\partial P(\boldsymbol{x}, t|\boldsymbol{x}_0, t_0)}{\partial t} = \sum_{i=1}^{N} [a_i(\boldsymbol{x} - \boldsymbol{v_i})P(\boldsymbol{x} - \boldsymbol{v_i}, t|\boldsymbol{x}_0, t_0) - a_i(\boldsymbol{x})P(\boldsymbol{x}, t|\boldsymbol{x}_0, t_0)] \quad (1)$$

Here, $\boldsymbol{v_i} = [v_{1i}, \ldots, v_{Mi}]^T$ is the stoichiometry associated with each reaction R_i. That is, $\boldsymbol{v_i}$ is the ith column vector, with the same unit as $\boldsymbol{x}$, of $M \times N$ stoichiometric matrix. $P(\boldsymbol{x}, t|x_0, t_0)$ is the conditional probability that $\boldsymbol{X}(t)$ will be $\boldsymbol{x}$, given the initial species population $\boldsymbol{X}(t_0) = \boldsymbol{x}_0$. It is very difficult to solve the chemical master equation for large networks, and therefore, we use some practical simulation techniques, e.g., stochastic simulation algorithms (SSAs).

SSAs work by performing the following two tasks: (a) finding the index (μ) of the reaction to occur, and (b) finding the time (τ) of occurrence of the reaction having index μ. Let us calculate the total propensity function as

$$a_0(\boldsymbol{x}) = \sum_{i=1}^{N} a_i(\boldsymbol{x}) \quad (2)$$

The time τ of occurrence of the reaction having index μ can be defined by the exponentially distributed random variable and is given by Cao et al. (2004)

$$p(\tau = s) = a_0(\boldsymbol{x}) \exp[-a_0(\boldsymbol{x})s] \quad (3)$$

There is enough justification in literature (Gillespie, 1976, 1977) that $P(\boldsymbol{x}, t|\boldsymbol{x}_0, t_0)$ can equivalently be considered as the probability of occurrence of the reaction having index μ. Thus the problem boils down to finding (searching) the reaction indexes, which will be executed next. This search may be done in various ways. Here, we developed a cache-based linear search technique for finding reactions to be executed. Next we describe our proposed algorithms SUESSA and SUESSSA along with their correctness and computational complexities.

2.1. Cache-based linear search (CBLS)

Simulation of biochemical networks using SSAs is done by executing each reaction of the network based on a stochastic procedure on their propensity values. SSAs perform a search operation for finding each reaction R_μ to be executed. Standard SSAs including DM, ODM and PDM use ordinary linear search for finding R_μ. Apart from ordinary linear search, other searches are also used for finding R_μ, e.g., RSSA-Binary uses binary search for finding R_μ. RSSA-Lookup uses lookup table search for finding R_μ. Here, we developed a cache-based linear search and applied it in the present algorithms (SUESSA and SUESSSA) for finding R_μ. We explain the cache-based linear search technique in details in the next paragraph.

SSAs in ordinary linear search use an array called **PROPENSITY**, where $PROPENSITY_i$ is used for storing the propensity value of ith reaction. The total propensity value of all the reactions is stored in a variable *TOTALPROP*. Then, it generates a uniformly distributed random number (r_1) in the interval (0,1), and obtains the product of r_1 and *TOTALPROP*. Next, it performs a linear search operation which starts from the first element of **PROPENSITY** and keeps on adding each $PROPENSITY_i$ cumulatively to find the reaction having index μ to occur. If the cumulative sum of propensity values is greater than or equal to the product of r_1 and *TOTALPROP*, then the corresponding reaction index is chosen as μ and the reaction is executed. If it is not the end of simulation, the propensity values are updated. Then, r_1 is generated again, and the product of r_1 and *TOTALPROP* is obtained; and the linear search operation is performed which starts from the first element of the array **PROPENSITY**, and so on.

In ordinary linear search, the search procedure always starts from the first element of **PROPENSITY** to obtain μ. In the proposed searching technique, a cache is maintained for holding the cumulative sum of propensity values obtained during simulation. Here, the linear search starts from the first element of **PROPENSITY** to obtain μ for the first time. Once μ is obtained, the value of μ is stored in a variable called *ISAM*, and the cumulative sum of the propensity values which starts from the first reaction up to μth reaction, is stored in a cache represented by a variable *SEL*. To obtain μ for the next time, it is checked whether the value of $r_1 \times TOTALPROP$ is greater than or equal to the value of *SEL*, or not. If the condition is true then the linear search starts from (*ISAM* + 1)th index of **PROPENSITY**, otherwise the search starts from the beginning, and so on. The variable *ISAM* is initially set to zero and *SEL* is set to (*TOTALPROP* + 1) so that the linear search starts from the first element of **PROPENSITY** for obtaining μ at the beginning of the algorithm.

After the execution of μth reaction, its propensity value is subtracted from both the variables *TOTALPROP* and *SEL*. Then the propensity value of μth reaction is recalculated, and is added back to both the variables *TOTALPROP* and *SEL*. The same procedure is followed for all of its dependent reactions (propensity values of these reactions are updated after the execution of the μth reaction). The propensity values of all the dependent reactions of μth reaction are updated in *TOTALPROP*, and the propensity values of only those dependent reactions are updated in *SEL*, whose index values are less than μ. Otherwise, no update procedure of propensity values is required in *SEL*.

Example Let us consider a network consisting of 1000 reactions where the reactions are $R_1, R_2, \ldots, R_{1000}$. Consider a possible trajectory, in terms of sequence of reaction executions after a single simulation run of this network by SSA using ordinary linear search, as $R_{600}(t_1) \rightarrow R_{700}(t_2) \rightarrow R_{400}(t_3) \rightarrow R_{500}(t_4) \rightarrow R_{800}(t_5)$. In order to obtain the reaction index of R_{600} in the case of ordinary linear search, we have to search propensity values from $PROPENSITY_1$ to $PROPENSITY_{600}$, and thereby, we need 600 moves. Similarly, we need 700, 400, 500 and 800 moves respectively, for finding the reaction indexes of R_{700}, R_{400}, R_{500}, and R_{800}. Therefore, the total number of moves required for finding all the reactions in ordinary linear search is (600 + 700 + 400 + 500 + 800) = 3000. Let us consider the same trajectory which is the outcome of a single simulation by any SSA using cache-based linear search. Number of moves needed for finding the reaction index of R_{600} is 600. Unlike ordinary linear search, the number of moves needed for finding the reaction index of R_{700} is 100 as the index of R_{700} is greater than that of the previous reaction and hence, we have to move from (*ISAM* + 1)th or 601th reaction index instead of first reaction index. Similarly, number of moves required for obtaining the reaction indexes of R_{400}, R_{500}, and R_{800} are respectively, 400, 100 and 300. Therefore, the total number of moves required for finding all the reactions in the case of cache-based linear search is (600 + 100 + 400 + 100 + 300) = 1500.

Table 1
D_M values of MODM, MPDM and MSPDM over their earlier counterparts in the case of linear chain model and B cell receptor signaling network.

Methods	D_M (linear chain model)	D_M (B cell receptor signaling)
MODM	33.32%	36.81%
MPDM	33.25%	32.20%
MSPDM	33.24%	26.02%

Computation of total number of moves (TOM)

Let us consider the case of ordinary linear search (OLS) technique. The total number of moves (TOM_{OLS}) required for finding all the reactions to be executed can be defined as

$$TOM_{OLS} = H_{WO}(Sr_1) + H_{WO}(Sr_2) + \cdots + H_{WO}(Sr_{(TT)}) \quad (4)$$

Here, *TT* denotes total number of reactions to be executed. The term $H_{WO}(Sr_1)$ stands for number of moves required to obtain the first reaction to be executed. H_{WO} includes all reactions whose indexes are obtained without caching. In other words, reaction indexes of these reactions are obtained by searching from the first reaction index. Now we consider the case of cache-based linear search (CBLS). The total number of moves (TOM_{CBLS}) required for finding all the reactions to be executed, can be written as

$$TOM_{CBLS} = H_{WO}(Sr_1) + H_{WO}(Sr_2 + \cdots + Sr_{(TT-TTC)}) + H_W(Sr_1 + Sr_2 + \cdots + Sr_{(TTC)}) \quad (5)$$

Here, the first reaction is obtained by ordinary linear search and the corresponding number of moves is denoted by $H_{WO}(Sr_1)$. *TTC* denotes the total number of reactions to be executed, which are obtained through caching. In other words, reaction indexes of these reactions are obtained by searching from (*ISAM* + 1)th reaction index. These reaction index searches are included in H_W. On the other hand, there are ($TT - TTC$) reactions which are obtained without caching. Percentage reduction (D_M) of total number of moves by cache-based linear search over ordinary linear search is given by

$$D_M = \frac{TOM_{OLS} - TOM_{CBLS}}{TOM_{OLS}} \times 100\% \quad (6)$$

A higher value of D_M makes the cache-based linear search more efficient. Here, we provide a comparative study between cache-based linear search and ordinary linear search. We applied cache-based linear search to ODM and named it as Modified ODM (MODM). Similarly, PDM and SPDM were modified to MPDM and MSPDM respectively. The modified versions, including MODM, MPDM and MSPDM are discussed in details in Supplementary material (biosystems_sppl). D_M values of various algorithms including MODM, MPDM and MSPDM over their earlier counterparts is shown in Table 1. We considered 10^7 reaction executions for each method. It is to be mentioned here that each D_M value is the average of 100 independent execution runs for each network. A comparative study of search times and total number of moves required in various algorithms is shown in Fig. 1. It is observed that cache-based

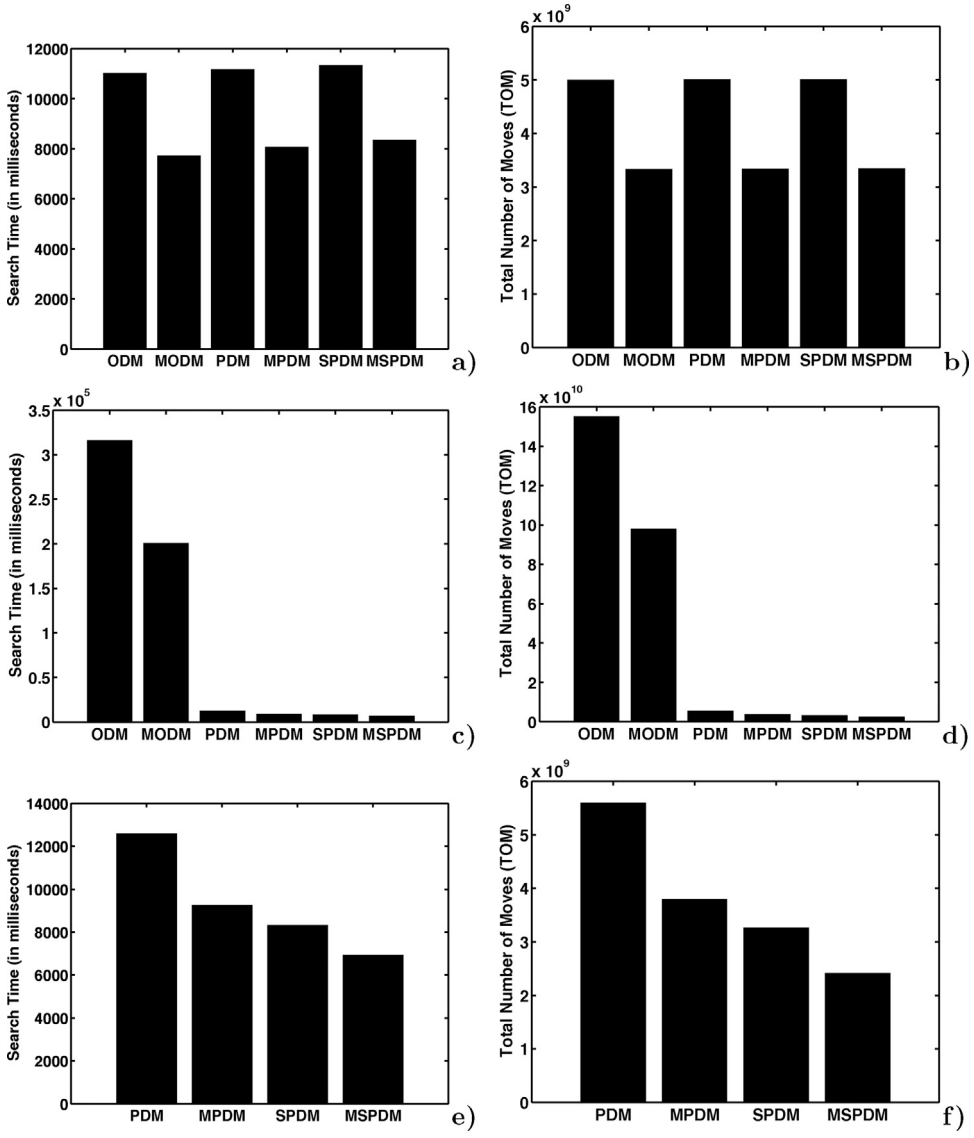

Fig. 1. Comparison of search times and total number of moves (***TOM***) of various SSAs is shown here. (a) Search times for linear chain model; (b) *TOM* for linear chain model; (c) search times for B cell receptor signaling network; (d) *TOM* for B cell receptor signaling network; (e) search times of partial-propensity methods for B cell receptor signaling network; (f) *TOM* of partial-propensity methods for B cell receptor signaling network.

linear search is faster than ordinary linear search in ODM, PDM and SPDM in the case of both linear chain model and B cell receptor signaling network.

2.2. Slow update exact stochastic simulation algorithm (SUESSA)

Algorithm 1 describes slow update exact stochastic simulation algorithm (SUESSA). The necessary data structures are initialized at STEP 1. Here, **x** defines initial populations of species. **Lx** and **Ux** define, respectively, the lower and upper bounds of species populations. Fluctuation level of population x_i of each species C_i, is defined by its lower bound Lx_i and upper bound Ux_i. Propensity update cost can be reduced by changing the fluctuation levels. Propensity update cost becomes low if the fluctuation level of x_i is maintained within +(10% to 20%) for Ux_i and −(10% to 20%) for Lx_i (Thanh et al., 2016). We define an array of arrays $\mathbf{\Gamma}$ for keeping the propensity values of the reactions. Each array $\mathbf{\Gamma}_i$ corresponds to each species C_i, maintains an array of propensity values of those reactions for which C_i is a reactant, given these reactions are not included by any of the previous $\mathbf{\Gamma}_k$ where $(k<i)$. In PDM, a data structure is maintained for keeping groups of partial propensity values of the reactions for each reactant species (Ramaswamy et al., 2009). Unlike PDM, each $\mathbf{\Gamma}_i$ keeps full propensity values of those reactions for which C_i is a reactant. Let us consider a reaction R_j with propensity value a_j having two reactants C_i and C_k. The propensity value a_j will be included in $\Gamma_{i,j}$ if $i<j$ and a_j will not be included by any other element of $\mathbf{\Gamma}$. In $\Gamma_{i,j}$, i and j are called major (group) and minor (element) indexes respectively. The propensity bounds for $\mathbf{\Gamma}$, $\mathbf{\Gamma L}$ (lower) and $\mathbf{\Gamma U}$ (upper) bounds are defined such that $\mathbf{\Gamma L}$ is defined by **Lx** and $\mathbf{\Gamma U}$ is defined by **Ux**. An array $\mathbf{\Omega}$ is defined where each element Ω_i stores the sum of propensity upper bounds of each species C_i. In other words, $\Omega_i = \sum_j \Gamma U_{i,j}$. The total propensity value *TOTALPROP* is defined by the sum of all Ω_i values.

An array of arrays $\mathbf{\Upsilon}$ is defined to store the reaction indexes of the propensity values contained in $\mathbf{\Gamma}$ such that each element $\Upsilon_{i,j}$ stores the reaction index of that reaction whose propensity is stored in $\Gamma_{i,j}$. **E** is an array of arrays where an array $\mathbf{E}_i$ stores the indexes of the reactants and products of the reaction R_i. **G** is an array of arrays which stores the stoichoimetry corresponding to the elements of **E**.

In SUESSA, we apply rejection based sampling and propensity bound infrequent propensity update techniques (Thanh et al., 2014, 2016; Vo, 2013). In rejection based sampling, a candidate reaction is checked for a rejection test once it is chosen. The candidate reaction is selected for execution if it is accepted after the test. A Boolean variable *ACCPT* is used for the rejection based selection procedure, and it is initialized to *FALSE* at STEP 2. The variable *TM* is used for reaction time calculation and it is initialized to 1.

2.2.1. Selection of the reaction index μ

In STEP 3, we choose the candidate reaction index μ by calling a function $SUREACTION_{\mu}()$ which is given in Algorithm 2. Here, we used the **cache-based** reaction search technique. *SEL* is initialized to (*TOTALPROP* + 1) in STEP 1 of Algorithm 1 such that reaction index search always starts from the first reaction index at the beginning of simulation. *SEL*1 is a temporary variable which is used to hold the value of *SEL* during the search procedure. *SEL*1 is initialized to 0. A random number *RN*1 (uniformly distributed in (0,1)) is generated. A variable *SAM* is declared, which holds the product of *TOTALPROP* and *RN*1. The variables *IVAL* and *JVAL* are declared for holding the major (group) and minor (element) indexes respectively of R_{μ}, and they are initialized to 0 at STEP 1 of Algorithm 1. The reaction index search starts by checking whether *SAM* is greater than *SEL* or not. *SEL* is initialized to (*TOTALPROP* + 1) at the beginning of the algorithm, and therefore, *SAM* will be less than *SEL*. Eventually, IF condition is false and the control jumps to else part and *SEL* is initialized to 0. Here, a FOR loop is maintained which iterates upto the number of species (M) and the values of Ω_i is cumulatively added to *SEL* till *SEL* is less than or equal to *SAM*. Otherwise, *SEL* is saved in *SEL*1 and the corresponding Ω_i is subtracted from *SEL* and i is stored in *IVAL* as the major index. Another *FOR* loop is maintained that iterates for all elements j of $\mathbf{\Gamma U}_{\mathrm{IVAL}}$, and $\lambda \mathrm{U}_{\mathrm{IVAL,j}}$ is cumulatively added to *SEL* till *SEL* is less than or equal to *SAM*. If *SEL* is greater than *SAM* then the corresponding value of j is stored in *JVAL* as minor index. We obtain the reaction index μ from $\Upsilon_{\mathrm{IVAL,JVAL}}$. *SEL* and *IVAL* are used each time the reaction index value is searched. In case of the next search for reaction index μ, if *SAM* is greater than *SEL* then the major index value is searched from (*IVAL* + 1)th index instead of the first index value.

2.2.2. Correction mechanism for the acceptance of the reaction index μ

Once we obtain the candidate reaction index μ at STEP 3 of Algorithm 1, we move to STEP 4 for rejection checking. A random variable *RN*2 (uniformly distributed in (0,1)) is generated. If *RN*2 is less than or equal to $\Gamma L_{\mathrm{IVAL,JVAL}}/\Gamma U_{\mathrm{IVAL,JVAL}}$ (propensity lower bound of R_{μ}/propensity upper bound of R_{μ}) then *ACCPT* is *TRUE* (R_{μ} is accepted to be executed) and we move further. Otherwise, actual propensity value of R_{μ} with present state is calculated at $\Gamma_{\mathrm{IVAL,JVAL}}$. Next, it is checked whether *RN*2 is less than or equal to $\Gamma_{\mathrm{IVAL,JVAL}}/\Gamma U_{\mathrm{IVAL,JVAL}}$ (propensity value of R_{μ}/propensity upper bound of R_{μ}). If it is true then *ACCPT* is set to *TRUE* and the candidate reaction is accepted to be executed. Next, another random number *RN*3 (uniformly distributed in (0,1)) is generated which is used for reaction time calculation. Then, it is checked whether *ACCPT* is *TRUE* or not. If *ACCPT* is *TRUE* then reaction occurring time τ is generated and reaction R_{μ} is executed. Otherwise, the control moves to STEP 3 for obtaining another candidate reaction R_{μ}.

2.2.3. Procedure for propensity update

The propensity update operation is performed at STEP 5. Here, $\mathbf{E}_{\mu}$ holds the indexes of all the reactants and products of R_{μ}. A *FOR* loop is maintained which iterates through the indexes of all the reactants and products of R_{μ}. A variable l holds the index value of any species C_l that may be a reactant or a product of R_{μ}. It is checked whether the population x_l of species C_l is within the specified range of its lower Lx_l and upper Ux_l bounds or not. If it is true, no update operation is performed. Otherwise, new lower Lx_l and upper Ux_l bounds are calculated for x_l. Next, Ω_l is subtracted from *TOTALPROP*. If l is less than or equal to major index *IVAL*, Ω_l is subtracted from *SEL*. New values of $\mathbf{\Gamma L}_l$, $\mathbf{\Gamma U}_l$ and Ω_l are obtained. Next, Ω_l is added back to *TOTALPROP*. If l is less than or equal to *IVAL* then, Ω_l is added back to *SEL*. Reaction generation time τ is updated to the variable *TIME* at STEP 6. If *TIME* is less than the maximum simulation time *ENDINGTIME*, control jumps to STEP 2. Otherwise, the algorithm terminates.

Algorithm 1. Slow Update Exact Stochastic Simulation Algorithm (SUESSA)

STEP 1. (Initialize)
TIME ⟵ 0; M ⟵ No. of species; N ⟵ No. of reactions;
x ⟵ (Initial species population); (**Lx**, **Ux**) ⟵ (Initial lower and upper bounds of **x** respectively)
Γ ⟵ An array of arrays where an element $\Gamma_{i,j}$ stores full propensity of reaction R_j provided C_i is a reactant of R_j;
ΓL ⟵ Lower bound of **Γ**; **ΓU** ⟵ Upper bound of **Γ**; **Ω** ⟵ An array where each element $\Omega_i \longleftarrow \sum_j \Gamma \mathrm{U}_{i,j}$;
TOTALPROP ⟵ $\sum_{i=0}^{M-1} \Omega_i$; **ϒ** ⟵ An array of arrays where each array $\mathbf{\Upsilon}_i$ stores the indexes of those reactions which are associated with $\mathbf{\Gamma}_i$; **E** ⟵ An array of arrays where ith array stores the species involved in ith reaction, **G** ⟵ Stores the corresponding stoichiometry;
IVAL ⟵ 0 (Major index of the reaction to occur), *JVAL* ⟵ 0 (Minor index of the reaction to occur)
SEL ⟵ *TOTALPROP* + 1

STEP 2.
ACCPT ⟵ *FALSE*; *TM* ⟵ 1

STEP 3. (Select the reaction to occur)
μ ⟵ SUREACTION_μ(*IVAL*, *JVAL*, **Ux**, **ΓU**, **Ω**, *SEL*, **ϒ**)

STEP 4.
*RN*2 ⟵ *RAND*()
if (*RN*2 ≤ $\Gamma L_{\mathrm{IVAL,JVAL}}/\Gamma U_{\mathrm{IVAL,JVAL}}$) **then**
  ACCPT ⟵ *TRUE*
else
  $\Gamma_{\mathrm{IVAL,JVAL}}$ ⟵ Calculate the full propensity value of R_{μ} with present state
  if (*RN*2 ≤ $\Gamma_{\mathrm{IVAL,JVAL}}/\Gamma U_{\mathrm{IVAL,JVAL}}$) **then**
    ACCPT ⟵ *TRUE*
  end if
end if
*RN*3 ⟵ *RAND*(); *TM* ⟵ *TM* × *RN*3
if (*ACCPT* ≠ *TRUE*)
  Goto **STEP 3.**
end if
τ ⟵ −*ln*(*TM*)/*TOTALPROP* **(Calculate the time of reaction occurrence)**
For each index k of $\mathbf{E}_{\mu}$, l ⟵ $E_{\mu,k}$ and x_l ⟵ $x_l + G_{\mu,k}$ **(Execute the reaction)**

STEP 5. (**Update Γ**, **Ω** and *TOTALPROP*)
for all k in $\mathbf{E}_{\mu}$ **do**
  l ⟵ $E_{\mu,k}$
  if ($x_l \geq Lx_l$ AND $x_l \leq Ux_l$) **then**
  Continue the loop
end if
Calculate the new lower bound and upper bound for x_l
TOTALPROP ⟵ *TOTALPROP* − Ω_l
if ($l \leq$ *IVAL*) **then**
  SEL ⟵ *SEL* − Ω_l
end if
Ω_l ⟵ 0
for all m in $\mathbf{\Gamma L}_l$ and $\mathbf{\Gamma U}_l$ **do**
  Calculate $\Gamma \mathrm{L}_{l,m}$ and $\Gamma \mathrm{U}_{l,m}$ with present state
  Ω_l ⟵ $\Omega_l + \Gamma \mathrm{U}_{l,m}$
end for
TOTALPROP ⟵ *TOTALPROP* + Ω_l
if ($l \leq$ IVAL) **then**
  SEL ⟵ *SEL* + Ω_l
  end if
end for

STEP 6. Terminate
TIME ⟵ *TIME* + τ
if (*TIME* < *ENDINGTIME*) **then**
  Goto **STEP 2.**
end if

Algorithm 2. Function for the selection of μ

SUREACTION_μ(*IVAL*, *JVAL*, **Ux**, **ΓU**, **Ω**, *SEL*, **ϒ**)
*SEL*1 ⟵ 0;*RN*1 ⟵ *RAND*();*SAM* ⟵ *TOTALPROP* × *RN*1
if (*SAM* > *SEL*) **then**
  for *i* ⟵ *IVAL* + 1 TO *M* **do**
    SEL ⟵ *SEL* + Ω_i
    if (*SEL* > *SAM*) **then**
      *SEL*1 ⟵ *SEL*
      SEL ⟵ *SEL* − Ω_i
      IVAL ⟵ *i* (**Calculate the group index**)
      break
    end if
  end for
  for all *j* in $\mathbf{\Gamma U}_{IVAL}$ **do**
    SEL ⟵ *SEL* + $\Gamma U_{IVAL,j}$
    if (*SEL* > *SAM*) **then**
      JVAL ⟵ *j* (**Calculate the element index**)
      break
    end if
  end for
  SEL ⟵ *SEL*1
else
  SEL ⟵ 0.0
  for *i* ⟵ 1 TO *M* **do**
    SEL ⟵ *SEL* + Ω_i
    if (*SEL* > *SAM*) **then**
      *SEL*1 ⟵ *SEL*
      SEL ⟵ *SEL* − Ω_i
      IVAL ⟵ *i* (**Calculate the group index**)
      break
    end if
end for
for all *j* in $\mathbf{\Gamma U}_{IVAL}$ **do**
  SEL ⟵ *SEL* + $\Gamma U_{IVAL,j}$
    if (*SEL* > *SAM*) **then**
      JVAL ⟵ *j* (**Calculate the element index**)
      break
    end if
  end for
  SEL ⟵ *SEL*1
end if
return ($\Upsilon_{IVAL,JVAL}$) (**Returning the indexes of Γ**)

2.3. Slow update exact sorting stochastic simulation algorithm (SUESSSA)

SUESSSA is the sorting version of SUESSA. The mechanism of obtaining μth reaction in SUESSSA is identical to that in SUESSA and the necessary data structures including **Γ**, **ΓL**, **ΓU**, **Ω**, **ϒ**, **E** and **G** are the same as that in SUESSA except the group index and element index of μth reaction, which are decremented by one, at each time μth reaction occurs. For this purpose, one array **MAJOR** and one array of arrays **MINOR** are used. In **MAJOR**, each element $MAJOR_i$ is initialized such that $MAJOR_i$ contains i where i holds the index value of the ith species. In **MINOR**, each element $MINOR_{i,j}$ corresponds to each element $\Upsilon_{i,j}$ of **ϒ** such that $MINOR_{i,j}$ contains j where j holds the index value of the jth reaction of which C_i is a reactant. Let us consider that reaction R_μ has been chosen to be executed which has been found from $\Upsilon_{i,j}$. Therefore, after the execution of R_μ, $MAJOR_i$ is interchanged with $MAJOR_{i-1}$ and $MINOR_{i,j}$ is interchanged with $MINOR_{i,j-1}$.

2.4. Correctness and complexities of SUESSA and SUESSSA

Number of propensity updates in SUESSA and SUESSSA is very low though the quality of their performances is not compromised. Here we discuss the correctness of the present algorithms. Next, we determine their time complexities.

2.4.1. Correctness

SUESSA and SUESSSA are exact SSAs. Here we provide the notion of correctness of these algorithms. The issue of correctness was dealt in Vo (2013). In the present algorithms (SUESSA and SUESSSA), a candidate reaction R_μ is selected, with probability $\Gamma U_{i,j}/TOTALPROP$, by the group index i and element index j such that $\mu = \Upsilon_{i,j}$. The candidate reaction R_μ is accepted to be executed with a probability $\Gamma_{i,j}/\Gamma U_{i,j}$. Therefore, the candidate reaction R_μ is selected as well as accepted with a probability $P(R_\mu)$, where $P(R_\mu)$ is

$$P(R_\mu) = \frac{\Gamma U_{i,j}}{TOTALPROP} \times \frac{\Gamma_{i,j}}{\Gamma U_{i,j}} = \frac{\Gamma_{i,j}}{TOTALPROP} \quad (7)$$

Let us consider any reaction R_k such that $k = \Upsilon_{l,m}$ is selected as well as accepted with a probability $P(R_k)$. Then, $P(R_k)$ can be written as

$$P(R_k) = \frac{\sum_{l=1}^{M}\sum_{m=1}^{N}\Gamma_{l,m}}{TOTALPROP} = \frac{a_0}{TOTALPROP} \quad (8)$$

Therefore, the probability of reaction R_μ to be selected and accepted provided any candidate reaction is selected and accepted, can be written from the concept of conditional probability as

$$P(R_\mu|R_k) = \frac{\Gamma_{i,j}}{TOTALPROP} \Big/ \frac{a_0}{TOTALPROP} = \frac{\Gamma_{i,j}}{a_0} \quad (9)$$

In any standard SSA, the reactions are executed with the same probability.

2.4.2. Complexity

Here we discuss the computational complexities of the present algorithms (SUESSA and SUESSSA). Let the average length of cache (*SEL*) be *W*. From Eq. (5), there are *TTC* reactions which are searched through caching and (*TT* − *TTC*) reactions which are searched without caching. The search complexity of *TTC* reactions is $O(M-W)$, as searching starts from $(W+1)$th species. On the other hand, the search complexity of (*TT* − *TTC*) reactions is $O(M)$. Therefore, the time complexity of selecting a candidate reaction through searching is $O(M)$, but with a reduced factor due to caching. Next, the candidate reaction passes through a rejection test. The average number of rejection tests performed for the acceptance of a candidate reaction R_μ is $\beta = \Gamma U_{i,j}/\Gamma_{i,j}$, where i and j are major and minor indexes respectively (Thanh et al., 2016). The value of β is very small, and therefore, the total time complexity for the selection procedure is $O(M)$.

During the propensity update operation, the propensity values of the reactions associated with each species C_i are updated, where C_i is either a reactant or a product of the reaction R_j that has been executed. For each C_i, we update the propensity values of those reactions for which C_i is a reactant. Let us consider that each C_i is a reactant to f reactions. The propensity update operation for each species is $O(f)$. The update operation of propensity values corresponding to each species C_i, depends on its fluctuation levels, and will occur if the population of C_i exceeds these levels. Eventually, very few propensity update operations are required, and it is $O(f)$. Overall the time complexity of these algorithms (SUESSA and SUESSSA) is the sum of the time complexities of search and update operations, and that is $O(M+f)$. Here, f is very small due to infrequent propensity update, and therefore, the time complexity can be determined as $O(M)$.

3. Benchmarks

Here we discuss the comparative performance analysis of our methods with the existing ones for various networks, including

colloidal aggregation model, B cell receptor signaling network and FcϵRI signaling network. Comparative analysis in the case of a cyclic linear chain model is provided in Supplementary material (biosystems_sppl). For the comparison purpose, we considered 10^7 reaction executions for each simulation of the methods. For comparing with RSSA-Binary, RSSA-Lookup and RSSA-CR (Thanh et al., 2015, 2016), we considered the versions (implemented using *Java*) available in Thanh et al. (2016). PDM, SPDM, PSSA-CR, SSA-CR and RSSA (Ramaswamy et al., 2009; Slepoy et al., 2008; Ramaswamy and Sbalzarini, 2010; Thanh et al., 2014) were implemented by us using *Java*. The present two algorithms (SUESSA and SUESSSA) were also implemented using *Java*. All the infrequent propensity update methods were simulated with ±10% of the x_i values of each species C_i. Next we discuss the validation of the proposed methods with others. For the purpose of simulation, we used a Z820 workstation with processor: Intel(R) Xeon(R) CPU E5-2650 @2.00 GHz, 128 GB RAM.

3.1. Colloidal aggregation model

Colloidal aggregation model is a strongly coupled network consisting of M species and N reactions, where $N = \left\lfloor \frac{M^2}{2} \right\rfloor$. The network for the colloidal aggregation model Ramaswamy et al. (2009) is given by

$$\begin{aligned} &C_i + C_j \xrightarrow{k_{i,j}} C_{i+j}, \qquad \text{for } i = 1, \ldots, \left\lfloor \frac{M}{2} \right\rfloor; j = i, \ldots, M - i, \\ &C_m \xrightarrow{k_{m,n}} C_n + C_{m+n}, \quad \text{for } m = 1, \ldots, M; \; n = 1, \ldots, \left\lfloor \frac{m}{2} \right\rfloor. \end{aligned} \tag{10}$$

Initial number of molecules (initial population) of each species C_i was set to 10,000. Reaction rate constants $k_{i,j}$ and $k_{m,n}$ were chosen between 0 and 100 randomly. It is to be mentioned that for a particular network, the same set of randomly chosen rate parameters was used in simulating each algorithm. Search time for finding the reactions to be executed using our methods was compared with that of the existing ones in Fig. 2a. Searching in RSSA is the slowest among all as this method is based on linear search technique and its search complexity is $O(N)$. On the other hand, search operation in the slow update methods (SUESSA and SUESSSA) is based on cache-based linear search and it is $O(M)$. In this network, for a size of 1000 (M) species, the number of reactions (N) is equal to 500,000. Searching in SUESSA (SUESSSA) is 823.27 (796.68) times faster than that in RSSA for $M = 1000$. Search times of SUESSA and SUESSSA are less than that of PDM and SPDM as these two partial propensity methods are based on ordinary linear search. Search time in PSSA-CR, SSA-CR, RSSA-Binary and RSSA-CR is little more than that in our methods. RSSA-Lookup is comparable to SUESSA and SUESSSA with respect to search time. SPDM, SSA-CR and RSSA-Lookup were simulated for the colloidal aggregation model only upto $M = 500$ due to longer execution time.

The plot corresponding to propensity update time of various algorithms is given in Fig. 2b. In PDM, SPDM, PSSA-CR and SSA-CR, time taken during propensity updates is huge as the number of propensity updates is the same as the number (10^7) of reaction executions. However, the number of propensity updates in SUESSA, SUESSSA, RSSA, RSSA-Binary, RSSA-Lookup and RSSA-CR is $\sim 6.5 \times 10^5$ due to propensity bound infrequent propensity updates. Propensity update operation in SUESSA (SUESSSA) is, respectively, 192.37 (173.41) and 405.15 (365.27) times faster than that in PDM and PSSA-CR for $M = 1000$. Propensity update operation in SUESSA (SUESSSA) is, respectively, 1145.56 (1114.72) and 984.85 (958.41) times faster than that in SPDM and SSA-CR for $M = 500$. Propensity update operation in SUESSA and SUESSSA is the fastest among all the algorithms considered here as in these two methods, *TOTALPROP* and *SEL* are updated in a straight forward manner. Propensity update operation in SUESSA (SUESSSA) is, respectively, 30.93 (27.89) times faster than that in RSSA-Binary for $M = 1000$, because the propensity update cost becomes high to maintain the binary tree structure used for storing the reaction propensities in RSSA-Binary. Time taken for propensity updates in SUESSA (SUESSSA) is, respectively, 1363.15 (1326.53) times less than that in RSSA-Lookup for $M = 500$ as RSSA-Lookup needs to update the lookup table used for storing propensity values, which contributes to a large portion of its total simulation time. Propensity update time in SUESSA (SUESSSA) is, respectively, 39.19 (35.36) times less than that in RSSA-CR for $M = 1000$. RSSA-CR maintains reaction propensities in groups and during propensity update, it recomputes the reaction propensities for each group G_i corresponding to each species C_i whose population exceeds its fluctuation level. During this time, some propensity values of G_i are moved to other groups if they do not satisfy the condition of being in G_i. Next, group sum is calculated and total propensity value is updated. This entire procedure increases propensity update time of RSSA-CR. Cost of propensity updates in RSSA is almost the same as that in SUESSA and SUESSSA.

Total simulation time of various methods is plotted in Fig. 2c. Total simulation time constitutes (a) search time and (b) propensity update time. SUESSA (SUESSSA) is, respectively, 70.29 (66.58) and 146.53 (138.78) times faster than PDM and PSSA-CR for $M = 1000$. SUESSA (SUESSSA) is, respectively, 399.37 (399.25) and 338.64 (338.53) times faster than SPDM and SSA-CR for $M = 500$. SUESSA (SUESSSA) is 469.62 (444.75) times faster than RSSA for $M = 1000$. Simulation time of RSSA is contributed by its large search time and it is 99.47% of its total simulation time. SUESSA (SUESSSA) is 473.25 (473.03) times faster than RSSA-Lookup for $M = 500$. This is due to the large propensity update time of RSSA-Lookup, which constitutes 99.94% of its total simulation time. Contribution of propensity update time in SUESSA (SUESSSA) is 35.97% (37.83%) of the total simulation time, which is very less compared to the others including RSSA-CR (94.38%), RSSA-Binary (83.65%). SUESSA (SUESSSA) is, respectively, 13.32 (12.61) and 14.97 (14.17) times faster than RSSA-Binary and RSSA-CR for $M = 1000$.

3.2. B cell receptor signaling network

B cell receptor signaling network is responsible for regulating the fates and activities of B cells of immune systems (Barua et al., 2012). This pathway is crucial in the research related to cancer therapy (Seda and Mraz, 2015). It is a strongly coupled network with 1122 species and 24,388 reactions Ramaswamy et al. (2009).

Search time of the algorithms is plotted in Fig. 3a. Searching in RSSA is the slowest among all and it is 100.95 (128.51) times slower than that in SUESSA (SUESSSA). This is because the search complexity of RSSA is $O(N)$ and the number of reactions (N) of this network is large compared to the number of species (M). On the other hand, the slow update methods (SUESSA and SUESSSA) use cache-based linear search, and their computational complexity is O(M). Search operation in SUESSSA is faster compared to that in SUESSA, as SUESSSA sorts its reaction propensities during each reaction execution. SUESSA (SUESSSA) is comparable with PDM (SPDM) with respect to search time. Search operation in PSSA-CR, SSA-CR, RSSA-Binary, RSSA-Lookup and RSSA-CR is faster than that in SUESSA and SUESSSA.

Propensity update time in all the algorithms is plotted in Fig. 3b. Number of propensity updates in PDM, SPDM, PSSA-CR, SSA-CR is 10^7. Hence, propensity update time in them is high. Number of propensity updates in SUESSA, SUESSSA, RSSA, RSSA-Binary, RSSA-Lookup and RSSA-CR is around 30,000 due to infrequent propensity update operations. Propensity update time in SUESSA

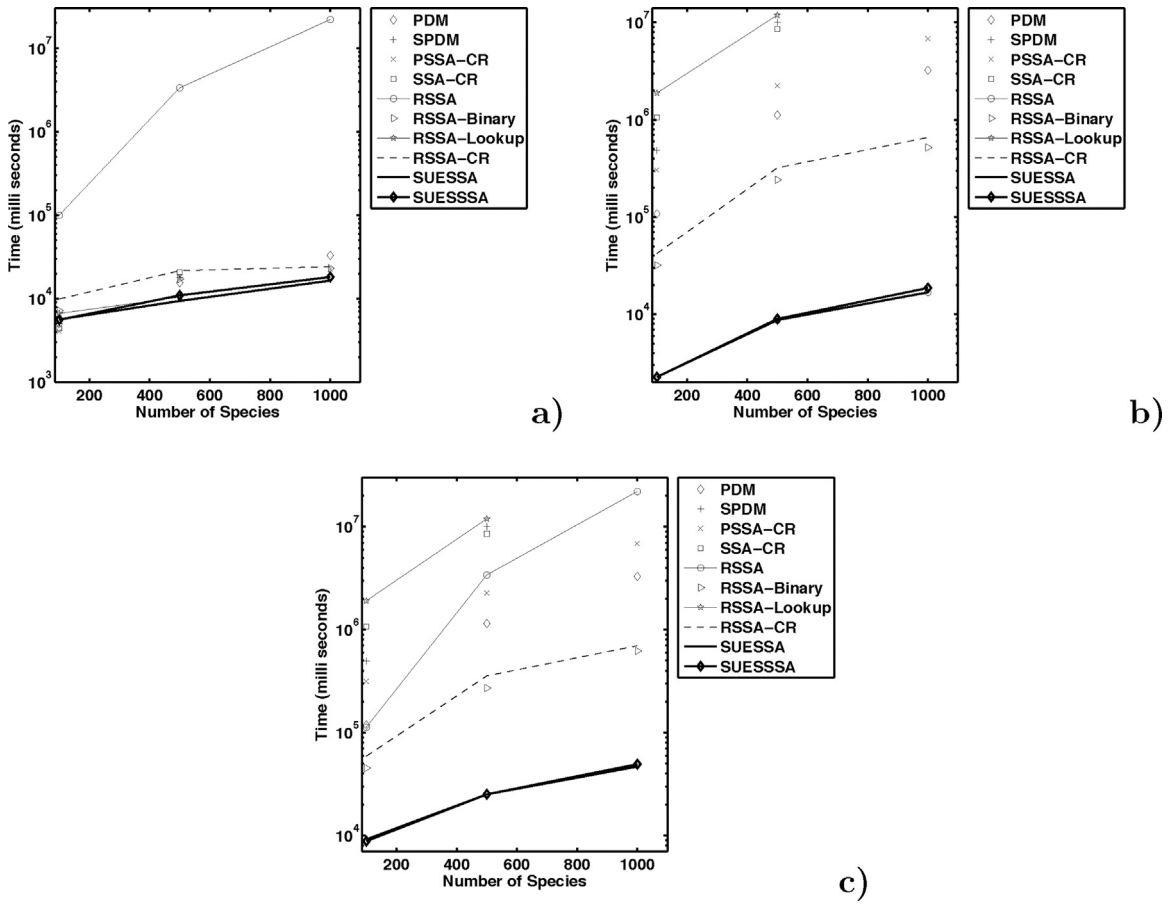

Fig. 2. Performances of various algorithms in the case of colloidal aggregation model. (a) Search times for finding reactions to be executed. (b) Propensity update times of various methods. (c) Total simulation times of various methods. *Y*-axes of all figures were considered in logarithmic scale. It is to be mentioned here that each plot is the average of 100 independent execution runs of the SSAs for each network.

(SUESSSA) is, respectively, 82.92 (81.22), 268.63 (263.14), 220.86 (216.34), 929.49 (910.50) and 264.04 (258.65) times less than that in PDM, SPDM, PSSA-CR, SSA-CR and RSSA-Lookup. Despite infrequent propensity updates, propensity update time in RSSA-Lookup is higher and comparable to the other SSAs which do not use infrequent propensity update technique. This higher update time in RSSA-Lookup is contributed by the cost of updating the lookup table which is used for keeping the propensity values of the reactions. From Fig. 3c, it is observed that propensity update time in RSSA-Binary and RSSA-CR is higher than that in SUESSA and SUESSSA. This is because the cost of updating the tree structure in RSSA-Binary and groups in RSSA-CR are expensive. Propensity update operation in SUESSA (SUESSSA) is, respectively, 7.86 (7.7) and 7.59 (7.44) times faster than that in RSSA-Binary and RSSA-CR. Propensity update time in SUESSA and SUESSSA is comparable to that in RSSA.

The plot of the total simulation time of the SSAs is shown in Fig. 3d. In RSSA, a large portion of the total simulation time is contributed by its searching time. In PDM, SPDM, PSSA-CR, SSA-CR and RSSA-Lookup, a large portion of their total simulation time is contributed by their propensity update time. Hence, their total simulation time is high. SUESSA (SUESSSA) is, respectively, 7.61 (8.72), 23.44 (26.88), 18.06 (20.71), 73.08 (83.82), 87.96 (100.89) and 21.03 (24.12) times faster than PDM, SPDM, PSSA-CR, SSA-CR, RSSA and RSSA-Lookup. Contributions of propensity update time in RSSA-Binary, RSSA-Lookup and RSSA-CR are, respectively, 37.43%, 97.41% and 37.08% of their respective total simulation time. In SUESSA (SUESSSA), propensity update time is 7.08% (8.62%) of the total simulation time. Despite longer search time in SUESSA (SUESSSA), shorter propensity update time make them, respectively, 1.63 (1.87) and 1.59 (1.82) times faster than RSSA-Binary and RSSA-CR (Fig. 3e).

3.3. *FcϵRI signaling network*

FcϵRI signaling triggers allergic reactions in RBL-2H3 cells (Liu et al., 2013). The signaling network is composed of 380 species and 3862 reactions (Thanh et al., 2015). The plot corresponding to search time in various algorithms is given in Fig. 4a. Searching in SUESSA (SUESSSA) is 22.03 (35.97) times faster than that in RSSA. Searching in SUESSA and SUESSSA is faster than that in PDM. In SUESSSA, search time is comparable to that in SPDM. SPDM, PSSA-CR, SSA-CR, RSSA-Binary, RSSA-Lookup and RSSA-CR are little faster with respect to search time, than SUESSA. Searching in SUESSSA is faster than that in SSA-CR and RSSA-CR. RSSA-Binary and RSSA-Lookup are faster than SUESSSA with respect to search time.

Propensity update time of the SSAs is plotted in Fig. 4b. Number of propensity updates in RSSA-Binary, RSSA-Lookup, RSSA-CR, SUESSA and SUESSSA is around 90,000 which is far less than that (10^7) in others. Propensity update operation in SUESSA (SUESSSA) is, respectively, 283.49 (283.89), 345.06 (345.56), 824.12 (825.31) and 1462.90 (1465.01) times faster than that in PDM, SPDM, PSSA-CR and SSA-CR. Similar to the case of B cell receptor signaling, cost of updating the lookup table in RSSA-Lookup is higher in the case of FcϵRI signaling network. Propensity update operation in SUESSA

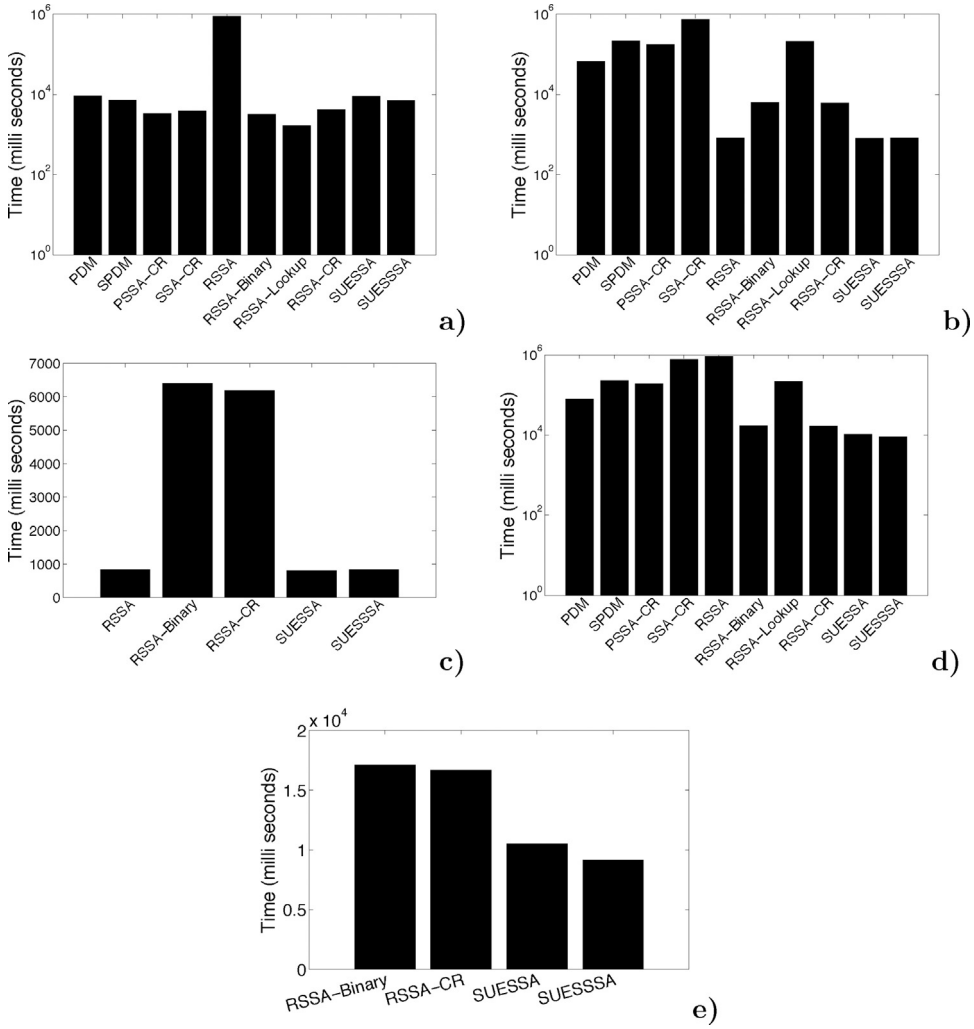

Fig. 3. Performances of various SSAs in the case of B cell receptor signaling network. (a) Search times to find reactions to be executed. The *y*-axis was considered in logarithmic scale. (b) Propensity update times of various methods. Here logarithmic scale was considered for *y*-axis. (c) Propensity update times in the case of RSSA, RSSA-Binary, RSSA-CR, SUESSA and SUESSSA. (d) Total simulation times of various methods. Here we considered logarithmic scale for *y*-axis. (e) Total simulation times in the case of RSSA-Binary, RSSA-CR, SUESSA and SUESSSA. It is to be mentioned that each plot is the average of 100 independent execution runs of the SSAs for each network.

(SUESSSA) is 251.33 (251.69) times faster than that in RSSA-Lookup. During the propensity update operation, maintenance of the tree structure in RSSA-Binary and groups in RSSA-CR increase their corresponding propensity update costs to a higher value. Propensity update operation in SUESSA (SUESSSA) is 6.04 (6.05) and 6.27 (6.28) times faster than that in RSSA-Binary and RSSA-CR respectively (Fig. 4c). Propensity update time in SUESSA and SUESSSA is comparable to that in RSSA.

The plot of total simulation time of various algorithms is given in Fig. 4d. Total simulation time of RSSA is slow due to its slow search time. Total simulation time of PDM, SPDM, PSSA-CR, SSA-CR and RSSA-Lookup is also slow due to their slow propensity update time. SUESSA (SUESSSA) is, respectively, 28.58 (34.64), 33.85 (41.01), 79.74 (96.64), 141.11 (171.03), 18.03 (21.86) and 24.98 (30.28) times faster than PDM, SPDM, PSSA-CR, SSA-CR, RSSA and RSSA-Lookup. SUESSA and SUESSSA are the fastest among all the SSAs due to their fastest propensity update time which is 8.9% and 11.54% of their total simulation time respectively. Contribution of propensity update time in RSSA-Binary and RSSA-CR is 34.8% and 31.37% of their total simulation time respectively. SUESSA (SUESSSA) is 1.58 (1.92) and 1.91 (2.31) times faster than RSSA-Binary and RSSA-CR respectively (Fig. 4e).

3.4. Validation

Due to lack of access to experimental data, we considered the existing methods for validation. We assumed that the existing methods resulted in biologically relevant solutions. We showed the final population of any 10 species of each network after 10^7 reaction executions by the methods, including ODM, SDM, PDM, SPDM, SUESSA and SUESSSA, in case of (a) colloidal aggregation model in Table 2, (b) B cell receptor signaling network in Table 3, (c) FcϵRI signaling network in Table 4 and (d) linear chain model in Table 5. Here ODM, SDM, PDM and SPDM were considered as the benchmarks. We observed that after the simulation by SUESSA and SUESSSA in each network, the final population of each species smoothly follows that after the simulation of the benchmarked methods. Due to lack of space we included the results of validation for 10 species only.

4. Discussion

In this article we developed two algorithms, viz., SUESSA and SUESSSA. In both these algorithms, propensity update operations are very fast. Number of propensity updates in them is much less due to the usage of propensity bound infrequent propensity updates. During propensity update operation, propensity value of

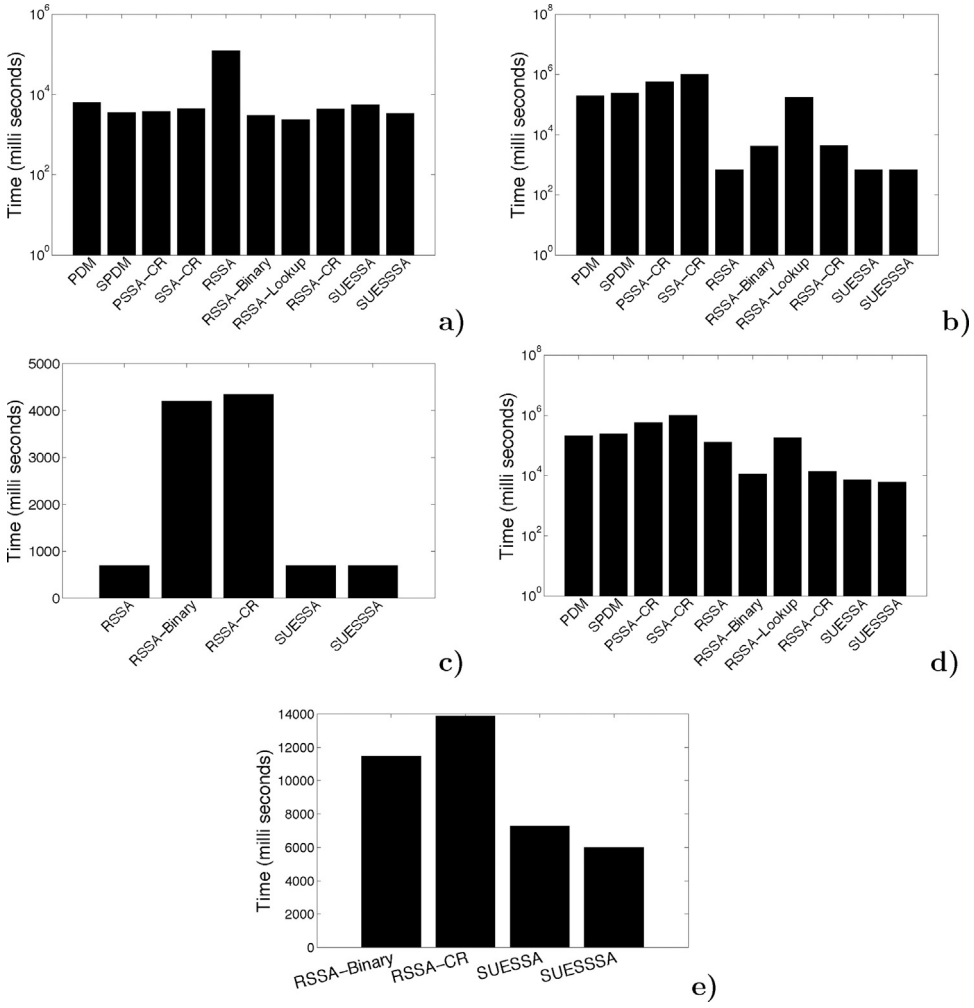

Fig. 4. Performances of various SSAs in the case of FcϵRI signaling model. (a) Search times to find the reactions to be executed next. The *y*-axis was considered in logarithmic scale. (b) Propensity update times of various methods. Here, logarithmic scale was considered for *y*-axis. (c) Propensity update times in case of RSSA, RSSA-Binary, RSSA-CR, SUESSA and SUESSSA. (d) Total simulation times of various methods. Here, we considered logarithmic scale for *y*-axis. (e) Total simulation times in the case of RSSA-Binary, RSSA-CR, SUESSA and SUESSSA. It is to be mentioned that each plot is the average of 100 independent execution runs of the SSAs for each network.

Table 2
The final population (number of molecules) of 50th, 100th, 150th, . . ., 500th species after 10^7 reaction executions by ODM, SDM, PDM, SPDM, SUESSA and SUESSSA in colloidal aggregation model.

Species	ODM	SDM	PDM	SPDM	SUESSA	SUESSSA
50th	2034	2021	2032	2012	2035	2028
100th	8900	8956	8987	8922	8913	8943
150th	21,098	20,097	20,013	21,089	22,003	21,045
200th	31,595	31,606	31,578	31,546	31,614	31,589
250th	53,156	53,109	53,203	53,089	53,134	53,178
300th	62,035	62,022	62,034	62,014	62,035	62,029
350th	84,034	84,221	84,332	84,012	84,135	84,082
400th	102,045	102,027	102,035	102,099	102,055	102,063
450th	130,039	130,123	130,232	130,013	130,049	130,076
500th	203,412	202,145	203,267	201,278	203,590	202,842

Table 3
The final population (number of molecules) of 100th, 200th, . . ., 1000th species after 10^7 reaction executions by ODM, SDM, PDM, SPDM, SUESSA and SUESSSA in B cell receptor signaling network.

Species	ODM	SDM	PDM	SPDM	SUESSA	SUESSSA
100th	190	203	197	210	205	187
200th	3783	3894	3945	3595	3724	3767
300th	1045	1120	1094	987	1023	1048
400th	856	879	897	843	812	833
500th	315	310	320	308	313	317
600th	4203	4202	4234	4298	4197	4205
700th	840	842	867	913	847	867
800th	789	734	767	709	765	761
900th	13,033	13,012	13,023	13,001	13,024	13,007
1000th	10,590	10,784	10,678	10,456	10,593	10,587

reaction R_μ and its corresponding dependent reactions are recalculated and updated to *TOTALPROP* and *SEL* in an efficient way. On the other hand, in PDM, SPDM, PSSA-CR and SSA-CR, the number of propensity updates is as high as the number of reaction firings, and hence the cost of propensity updates is high. RSSA-Binary, RSSA-Lookup and RSSA-CR use complex data structures for fast search operations. In the case of simulating strongly coupled networks, cost of propensity updates in them is high in spite of using propensity bound infrequent propensity updates as cost of updating their respective complex data structures is high. RSSA-Binary and RSSA-Lookup need to update tree structure and lookup table respectively during propensity update operations, which result in expensive propensity update costs. RSSA-CR needs to update group (used for keeping propensity values of the reactions) sums and move propensity values between groups when needed.

Unlike PDM and SPDM, SUESSA and SUESSSA can model networks with higher order reactions (more than second order reactions). The search procedures of the algorithms, including

Table 4
The final population (number of molecules) of 50th, 75th, 100th, 125th, 150th, 175th, 200th, 250th, 300th and 380th species after 10^7 reaction executions by ODM, SDM, PDM, SPDM, SUESSA and SUESSSA in Fc∈RI signaling network.

Species	ODM	SDM	PDM	SPDM	SUESSA	SUESSSA
50th	304	390	297	310	305	310
75th	783	894	745	795	784	797
100th	1145	1125	1194	1087	1123	1148
125th	556	579	597	573	562	573
150th	3215	3210	3220	3208	3213	3217
175th	403	402	434	498	407	405
200th	1845	1840	1867	1913	1847	1867
250th	12,689	12,634	12,667	12,609	12,665	12,661
300th	133	112	123	101	124	107
380th	20,690	20,684	20,678	20,456	20,693	20,687

Table 5
The final population (number of molecules) of 100th, 200th, . . ., 1000th species after 10^7 reaction executions by ODM, SDM, PDM, SPDM, SUESSA and SUESSSA in linear chain model.

Species	ODM	SDM	PDM	SPDM	SUESSA	SUESSSA
100th	30	23	28	31	29	32
200th	83	89	95	97	81	86
300th	145	135	197	185	165	153
400th	74	79	57	63	71	73
500th	321	313	345	329	323	317
600th	603	602	594	578	607	605
700th	325	367	347	304	311	327
800th	121	129	133	139	119	123
900th	1003	983	1087	1047	1019	1023
1000th	10,340	10,254	10,323	10,456	10,398	10,329

PDM, SPDM and RSSA take longer execution time. Search operation in RSSA is of the order of reactions. Due to this reason, the cost of search operation in RSSA becomes very high in simulating strongly coupled networks. SUESSA and SUESSSA involve comparatively faster search operations than PDM, SPDM and RSSA, due to the usage of cache-based linear search technique. However, search operation in PSSA-CR, SSA-CR, RSSA-Binary, RSSA-Lookup and RSSA-CR is faster than that in SUESSA and SUESSSA, due to their respective data structures.

In SUESSA, the search process is order dependent. In this case, if the propensity values at the beginning of the Propensity array are large compared to that at the end then search operation will be fast. On the other hand, if the propensity values at the beginning of the propensity array are small compared to that at the end then search operation will be slow. In SUESSSA, the search process is also order dependent. Here, the indexes of the larger propensity values in the propensity array moves gradually towards the beginning of this array and indexes of the smaller propensity values move towards the end during the simulation process. This makes the search process faster.

It is observed in SSAs with fast search operations that their propensity updates become slow in simulating strongly coupled networks. The present algorithms are equipped with fast update operations, although their search techniques are not much efficient. The data structure of **cache-based** linear search is simpler compared to that of the search operations in SSA-CR, PSSA-CR, RSSA-Binary, RSSA-Lookup and RSSA-CR. Now the point is that in order to reduce the search operation cost, we need complex data structures as applied in SSA-CR, PSSA-CR, RSSA-Binary, RSSA-Lookup and RSSA-CR. These methods need much higher propensity update operation cost particularly in simulating strongly coupled networks including colloidal aggregation network, B cell receptor signaling and Fceri signaling networks. In the strongly coupled networks, the total simulation cost is dominated by propensity update cost rather than search operation cost, and therefore, using complex data structure the total simulation cost will be higher. Linear chain network is a hypothetical network whereas, most of the real biological systems are strongly coupled networks. In SUESSA and SUESSSA, we focused more on reducing propensity update cost. SSAs with fast search as well as update operations will be developed as possible future work.

Data availability

The executable files along with some networks can be downloaded from http://www.isical.ac.in/∼rajat/.

Acknowledgements

Mr. Debraj Ghosh, one of the authors, gratefully acknowledges CSIR, India for providing him a Senior Research Fellowship (9/93 (0150)/2013, EMR-I). Both authors have no funding. The first author contributed to the conception and design of this study. Both authors drafted the manuscript.

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, at https://doi.org/10.1016/j.biosystems.2017.10.011.

References

Auger, A., Chatelain, P., Koumoutsakos, P., 2006. R-leaping: accelerating the stochastic simulation algorithm by reaction leaps. J. Chem. Phys. 125 (8), 084103.
Barrio, M., Burrage, K., Leier, A., Tian, T., 2006. Oscillatory regulation of Hes1: discrete stochastic delay modelling and simulation. PLoS Comput. Biol. 2 (9), e117.
Barrio, M., Leier, A., Marquez-Lago, T.T., 2013. Reduction of chemical reaction networks through delay distributions. J. Chem. Phys. 138 (10), 104114.
Barua, D., Hlavacek, W.S., Lipniacki, T., 2012. A computational model for early events in b cell antigen receptor signaling: analysis of the roles of Lyn and Fyn. J. Immunol. 189 (2), 646–658.
Cai, X., Xu, Z., 2007. K-leap method for accelerating stochastic simulation of coupled chemical reactions. J. Chem. Phys. 126 (7), 74102.
Cao, Y., Li, H., Petzold, L., 2004. Efficient formulation of the stochastic simulation algorithm for chemically reacting systems. J. Chem. Phys. 121 (9), 4059–4067.
Cao, Y., Gillespie, D.T., Petzold, L.R., 2005a. Avoiding negative populations in explicit Poisson tau-leaping. J. Chem. Phys. 123 (5), 054104.
Cao, Y., Gillespie, D.T., Petzold, L.R., 2005b. The slow-scale stochastic simulation algorithm. J. Chem. Phys. 122 (1), 014116.
Cao, Y., Gillespie, D.T., Petzold, L.R., 2006. Efficient step size selection for the tau-leaping simulation method. J. Chem. Phys. 124 (4), 044109.
Ehlert, K., Loewe, L., 2014. Lazy updating of hubs can enable more realistic models by speeding up stochastic simulations. J. Chem. Phys. 141 (20), 204109.
Gibson, M.A., Bruck, J., 2000. Efficient exact stochastic simulation of chemical systems with many species and many channels. J. Phys. Chem. A 104 (9), 1876–1889.
Gillespie, D.T., Petzold, L.R., 2003. Improved leap-size selection for accelerated stochastic simulation. J. Chem. Phys. 119 (16), 8229–8234.
Gillespie, D.T., 1976. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. J. Comput. Phys. 22 (4), 403–434.
Gillespie, D.T., 1977. Exact stochastic simulation of coupled chemical reactions. J. Phys. Chem. 81 (25), 2340–2361.
Gillespie, D.T., 2001. Approximate accelerated stochastic simulation of chemically reacting systems. J. Chem. Phys. 115 (4), 1716–1733.
Gillespie, D.T., 2007. Stochastic simulation of chemical kinetics. Annu. Rev. Phys. Chem. 58, 35–55.
Leier, A., Marquez-Lago, T.T., 2015. Delay chemical master equation: direct and closed-form solutions. In: Proc. R. Soc. A, Vol. 471, The Royal Society, p. 20150049.
Leier, A., Barrio, M., Marquez-Lago, T.T., 2014. Exact model reduction with delays: closed-form distributions and extensions to fully bi-directional monomolecular reactions. J. R. Soc. Interface 11 (95), 20140108.
Liu, Y., Barua, D., Liu, P., Wilson, B.S., Oliver, J.M., Hlavacek, W.S., Singh, A.K., 2013. Single-cell measurements of ige-mediated fcεri signaling using an integrated microfluidic platform. PLOS ONE 8 (3), e60159.
McCollum, J.M., Peterson, G.D., Cox, C.D., Simpson, M.L., Samatova, N.F., 2006. The sorting direct method for stochastic simulation of biochemical systems with varying reaction execution behavior. Comput. Biol. Chem. 30 (1), 39–49.
Peng, X.-j., Wang, Y.-f., 2007. L-leap: accelerating the stochastic simulation of chemically reacting systems. Appl. Math. Mech. 28 (10), 1361–1371.

Peng, X., Zhou, W., Wang, Y., 2007. Efficient binomial leap method for simulating chemical kinetics. J. Chem. Phys. 126 (22), 224109.
Ramaswamy, R., Sbalzarini, I.F., 2010. A partial-propensity variant of the composition-rejection stochastic simulation algorithm for chemical reaction networks. J. Chem. Phys. 132 (4), 044102.
Ramaswamy, R., González-Segredo, N., Sbalzarini, I.F., 2009. A new class of highly efficient exact stochastic simulation algorithms for chemical reaction networks. J. Chem. Phys. 130 (24), 244104.
Rathinam, M., Petzold, L.R., Cao, Y., Gillespie, D.T., 2003. Stiffness in stochastic chemically reacting systems: the implicit tau-leaping method. J. Chem. Phys. 119 (24), 12784–12794.
Seda, V., Mraz, M., 2015. B-cell receptor signalling and its crosstalk with other pathways in normal and malignant cells. Eur. J. Haematol. 94 (3), 193–205.
Slepoy, A., Thompson, A.P., Plimpton, S.J., 2008. A constant-time kinetic Monte Carlo algorithm for simulation of large biochemical reaction networks. J. Chem. Phys. 128 (20), 205101.
Thanh, V.H., Zunino, R., 2014. Adaptive tree-based search for stochastic simulation algorithm. Int. J. Comput. Biol. Drug Des. 7 (4), 341–357.
Thanh, V.H., Priami, C., Zunino, R., 2014. Efficient rejection-based simulation of biochemical reactions with stochastic noise and delays. J. Chem. Phys. 141 (13), 134116.
Thanh, V.H., Zunino, R., Priami, C., 2015. On the rejection-based algorithm for simulation and analysis of large-scale reaction networks. J. Chem. Phys. 142 (24), 244106.
Thanh, V.H., Zunino, R., Priami, C., 2016. Efficient constant-time complexity algorithm for stochastic simulation of large reaction networks. IEEE/ACM Trans. Comput. Biol. Bioinform. PP (99), http://dx.doi.org/10.1109/TCBB.2016.2530066, 1-1.
Vo, H.T., 2013. On efficient algorithms for stochastic simulation of biochemical reaction systems (Ph.D. thesis). University of Trento.