This is an electronic reprint of the original article.
This reprint may differ from the original in pagination and typographic detail.

Thanh, Vo Hong

A Critical Comparison of Rejection-Based Algorithms for Simulation of Large Biochemical Reaction Networks

Published in:
Bulletin of Mathematical Biology

DOI:
[10.1007/s11538-018-0462-y](https://doi.org/10.1007/s11538-018-0462-y)

Published: 01/08/2019

Document Version
Peer reviewed version

Please cite the original version:
Thanh, V. H. (2019). A Critical Comparison of Rejection-Based Algorithms for Simulation of Large Biochemical Reaction Networks. *Bulletin of Mathematical Biology*, 81(8), 3053-3073. <https://doi.org/10.1007/s11538-018-0462-y>

This material is protected by copyright and other intellectual property rights, and duplication or sale of all or part of any of the repository collections is not permitted, except that material may be duplicated by you for your research use or educational purposes in electronic or print form. You must obtain permission for any other use. Electronic or print copies may not be offered, whether for sale or otherwise to anyone who is not an authorised user.

A Critical Comparison of Rejection-based Algorithms for Simulation of Large Biochemical Reaction Networks

Vo Hong Thanh

Received: date / Accepted: date

Abstract The rejection-based simulation technique has been applying to improve the computational efficiency of the stochastic simulation algorithm (SSA) in simulating large reaction networks, which are required for a thorough understanding of biological systems. We compare two recently proposed simulation methods, namely the composition-rejection algorithm (SSA-CR) and the rejection-based SSA (RSSA), aiming for this purpose. We discuss the right interpretation of the rejection-based technique used in these algorithms in order to make an informed choice when dealing with different aspects of biochemical networks. We provide the theoretical analysis as well as the detailed runtime comparison of these algorithms on concrete biological models. We highlight important factors that are omitted in previous analysis of these algorithms. The numerical comparison shows that for reaction networks where the search cost is expensive then SSA-CR is more efficient, and for reaction networks where the update cost is dominant, often the case in practice, then RSSA should be the choice.

Keywords Computational biology · Stochastic simulation · Rejection-based simulation technique

1 Introduction

The stochastic simulation algorithm (SSA) [1,2] has been widely used for studying dynamical behavior of biological networks. It takes intrinsic noise due to the discreteness in populations of species and the randomness in reactions between these species as an inherent part of the simulation. A quantitative

Vo Hong Thanh

Department of Computer Science, Aalto University, Finland and

The Microsoft Research - University of Trento Centre for Computational and Systems Biology (COSBI), Italy.

E-mail: thanh.vo@aalto.fi

understanding of the stochastic noise can help to unravel complicated switching behavior in biological responses [3–5], leading to significant phenotypic variants [6].

In SSA, the state of the system is modeled as a vector of population of each molecular species and the interactions of species are encoded as chemical reactions between these species. Each reaction is assigned a probability to fire that is proportional to a *propensity*, which depends on the system state and the reaction kinetics. Although the probability distribution of the state of the biochemical network can be fully expressed by the chemical master equation (CME) [7], a direct analytical/numerical solution to CME is often difficult to obtain due to the exponential increase of the number of differential equations to describe transitions between system states and SSA is often used for exactly realizing temporal dynamics of the reaction network. SSA is an exact simulation algorithm in the sense that it selects a reaction firing and moves the system to a new state according to a probability distribution derived under the same hypothesis as CME. Several extensions of SSA for incorporating environmental effects such as biochemical reactions with time-dependent rates [8–10], reactions with delay times [11, 8, 12, 22], are also introduced to extend its applicability.

The core of SSA is a discrete event simulation strategy where a reaction is randomly selected to fire in each simulation loop. The direct method (DM) [1, 2] is commonly used to implement this selection step. It selects the next reaction with a probability proportional to the reaction propensity. The firing time is generated by sampling an exponential distribution. The simulation then updates the state as well as the system time according to the selected reaction. It also updates the propensities of affected reactions due to the changes in the system state by the reaction firing. Several efficient formulations to accelerate performance of SSA has been introduced including the next reaction method (NRM) [14], sorting DM [15, 16], tree-based search DM [17–19], partial-propensity DM [20] as well as approximate algorithms [21–24].

Efficient exact stochastic simulation is an important tool to have an exact quantitative description of biochemical reaction networks; however, the performance of the basic SSA approach is often prohibitively expensive in simulating large reaction models. The rejection-based technique [25, 26] provides a promising framework for improving computational efficiency of SSA when dealing with different aspects of large biochemical reaction networks. The composition-rejection method (SSA-CR) [27, 28] and the rejection-based SSA (RSSA) [12, 29], in particular, are the two main general algorithms introducing recently for this purpose. Improvements and implementations of these algorithms also have been introduced recently [30–36]. These algorithms focus on different computational bottlenecks of SSA to improve the simulation performance. SSA-CR employs the rejection-based technique for improving the selection of next reaction firings, while RSSA uses such technique to reduce the average number of propensity updates during the simulation.

In this paper, we assess the use of the rejection-based technique in the SSA-CR and RSSA algorithms. The right interpretation of the rejection-based

technique used in these algorithms is crucially important for the understanding of purposes and the clarification of these algorithms. The misinterpretations of the key rejection-based step used by these algorithms may lead to wrong conclusion. In reference [37], the authors seem to misinterpret this key step in SSA-CR and RSSA when they assume these algorithms serving the same purpose. Although both of these algorithms employ the rejection-based technique during the selection of the next reaction firing, the detailed implementations of these algorithms are different in which they focus on different aspects of the reaction networks. Second, we provide a thorough analysis of these algorithms. Through our analysis, we highlight facts that are often omitted in previous analysis of these algorithms. For instance, the analysis of the original work of SSA-CR [27] assumes the number of propensity updates after each reaction firing is bounded by a small constant. Under this assumption, the total simulation time in SSA-CR is dominated only by the search for next reaction firings. Ignoring this important underlying assumption when applying to practical models may lead to the 'surprising' low performance of SSA-CR [37]. Taking this fact into account, we can successfully give an explanation for the low performance of SSA-CR. In addition to theoretical analysis, we numerically compare SSA-CR and RSSA on concrete biological models to verify our analysis. Our ultimate aim for this analysis is to show the advantages and drawbacks of these algorithms in order to help researchers make an informed choice for the suitable algorithm for simulating a given reaction network based on its basic properties such as the number of reactions, reaction connectivity.

The paper is organized as follows. Section 2 gives the background on the stochastic simulation. Section 3 provides the detailed implementations as well as a comprehensive comparison of SSA-CR and RSSA in employing the rejection-based technique for improving the simulation large reaction networks. We also provide in this section the theoretical analysis of these algorithms. Section 4 shows the runtime analysis of these algorithms on concrete biological models to verify our analysis. The conclusion remarks are provided in Section 5.

2 Stochastic simulation

We consider a well-mixed volume consisting of N chemical species denoted as $S_1 \dots S_N$. The state of the system is represented by a population vector $X(t) = (X_1(t), \dots, X_N(t))$ where $X_i(t)$ is a discrete number (called population) of species S_i at a time t . Species can interact with others through M reactions $R_1 \dots R_M$. The probability that a reaction R_j fires in the next infinitesimal time $t + dt$ is $a_j(X(t))dt$ where $a_j(X(t))$ is called the reaction propensity [1]. The propensity function $a_j(X(t))$, in general, depends on the state $X_i(t)$ and its kinetics information. An exact formula for the propensity a_j ¹ is described by the reaction kinetics applied to model the system. For example, the propensity

¹ we often use a_j instead of $a_j(X(t))$ for a shorthand when the context is clear.

a_j of a reaction R_j with the common *mass-action* kinetics has the form:

$$a_j(X(t)) = c_j h_j(X(t)) \quad (1)$$

where c_j is stochastic rate constant and $h_j(X(t))$ counts the number of combinations of reactants involved in R_j .

If a reaction R_μ is selected to fire, the state changes according to the state change vector v_μ , which expresses the changes in population of species involved in R_μ . The state transition of the biochemical reaction can be modeled as a (continuous-time) jump Markov process and the probability distribution of the system is completely described by the chemical master equation (CME) [7]. The stochastic simulation algorithm (SSA) is an exact method to sample temporal behavior of the biochemical network encoded in CME.

SSA [1, 2] realizes the next state by simulating the joint probability density function (pdf) $p(\tau, \mu)$ with $p(\tau, \mu)d\tau$ being the probability that a reaction R_μ fires in the next infinitesimal time $t + \tau + d\tau$, given the state $X(t)$ at time t . Eq. 2 gives a closed form of $p(\tau, \mu)$:

$$p(\tau, \mu) = a_\mu \exp(-a_0 \tau) \quad (2)$$

where $a_0 = \sum_{j=1}^M a_j$. Note that the reaction R_μ fires with corresponding discrete probability a_μ/a_0 and the firing time τ is exponentially distributed with rate a_0 .

The direct method (DM) [1] is a common used implementation of SSA for sampling the pdf $p(\tau, \mu)$ to construct a simulation trajectory. It computes M propensities a_j for $j = 1 \dots M$ at beginning. Then, for each simulation step the next reaction firing R_μ and its firing time τ are realized from Eq. 2 by:

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

and

$$\mu = \text{smallest reaction index such that: } \sum_{j=1}^{\mu} a_j > r_2 a_0 \quad (4)$$

where r_1 and r_2 are two random numbers generated from a uniform distribution $U(0, 1)$. The state is updated according to the selected reaction R_μ and moves to a new state $X(t + \tau) = X(t) + v_\mu$. The propensities are updated to reflect the changes in the system state. For DM, the time complexity of the search for the next reaction is $O(M)$ because it linearly sums propensities until a reaction is selected. The update of propensities in the original work [1] is performed in $O(M)$ because M propensities are recomputed; however, this cost can be reduced by applying a reaction dependency graph [14] to decide which propensities need to be updated after each reaction firing. The time complexity of each simulation step of DM is thus $O(M)$.

3 Rejection-based technique for SSA

This section recalls the basics of the rejection-based technique. We then discuss its applications in the context of biological simulation for improving computational efficiency of SSA when dealing with different aspects of biochemical reactions. We focus here on two exact rejection-based simulation approaches published in the literature that are: the composition-rejection method (SSA-CR) by Slepoy *et al.* [27] and the rejection-based SSA (RSSA) by Thanh *et al.* [12]. For recent work on these algorithms, we refer to references [31, 34] for further improvements of these algorithms, to reference [32] for the combination of these algorithms and to references [22, 23] for approximate algorithms. After describing SSA-CR and RSSA, we compare the use of the rejection-based technique in these algorithms as well as provide the analytical analysis for these algorithms in order to highlight their advantages and drawbacks. The significance of our analysis is that we take into account important facts that are often omitted in the previous analysis.

3.1 Theoretical background

The rejection-based (or acceptance-rejection) technique provides a general framework for generating random numbers. Let X be a random variable with a given pdf f that we want to generate a realization. Instead of directly sampling the pdf f , the rejection-based method uses a proposal (or hat) function h satisfying $f(x) \leq ch(x)$, $\forall x \in \Omega_X$ where Ω_X denotes the state space of X and c is a constant. The pdf h is often selected so that it can easily generate a random number from h . The steps for the generation of random numbers with pdf f by using the rejection-based technique are as follows. Let y be a random number with pdf h and, respectively, $r \sim \mathcal{U}(0, 1)$ be a uniformly distributed random number. If the condition $r \leq f(y)/ch(y)$ holds, then the rejection-based method accepts $X = y$ as a random number with desired pdf f . Otherwise, the random number y is rejected and a new generation step is repeated. The average number of trials until a sample is accepted by the rejection-based method is c because the acceptance rate is $1/c$. The advantage of the rejection-based method is that the hat function h can be chosen arbitrarily so that the imposed condition holds. This feature is especially useful when the original pdf f is complex and difficult to sample directly. The reader is referred to, for example, Devroye [25] or Hormann *et al.* [26] for a correctness proof as well as improvements of the general rejection-based technique.

3.2 SSA-CR

We outline the detailed implementation of SSA-CR in Algorithm 1 that employs the rejection-based technique for improving the selection of next reaction firings. The key step of SSA-CR is the lines 7 - 12 where the composition-rejection search is used for selecting the next reaction firings.

Algorithm 1 SSA with Composition-Rejection Search (SSA-CR)

Require: a biochemical reaction network of M reactions in which each reaction R_j , $j = 1, \dots, M$, is accompanied with the state change vector v_j and the propensity a_j , the initial state $\mathbf{x}_0$ at time 0 and the simulation ending time T_{max}

Ensure: a trajectory $X(t)$, $0 \leq t \leq T_{max}$, of the biochemical reaction network

- 1: initialize time $t = 0$ and state $X = \mathbf{x}_0$
- 2: build the dependency graph G
- 3: partition M reactions into L groups $\{G_1, \dots, G_L\}$ such that a group G_l contains reaction R_j if $2^{u_l-1} \leq a_j < 2^{u_l}$
- 4: compute $a^l = \sum_{R_j \in G_l} a_j$, for $l = 1, \dots, L$ and total sum $a_0 = \sum_{l=1}^L a^l$
- 5: **while** ($t < T_{max}$) **do**
- 6: generate a random number $r_1 \sim \mathcal{U}(0, 1)$
- 7: select G_α with the smallest group index α s.t. $\sum_{l=1}^\alpha a^l \geq r_1 a_0$
- 8: **repeat**
- 9: generate a random number $r_2 \sim \mathcal{U}(0, 1)$
- 10: compute reaction index $\mu = \lceil r_2 |G_\alpha| \rceil$
- 11: set $r_3 = r_2 |G_\alpha| - \mu$
- 12: **until** ($r_3 \leq a_\mu / 2^{u_\alpha}$)
- 13: generate a random number $r_4 \sim \mathcal{U}(0, 1)$
- 14: compute $\tau = (1/a_0) \ln(1/r_4)$
- 15: update state $X = X + v_\mu$
- 16: set $t = t + \tau$
- 17: **for all** ($R_j \in \text{Dependents}(R_\mu)$) **do**
- 18: update a_j
- 19: **if** ($a_j \notin [2^{u_l-1}, 2^{u_l}]$) **then**
- 20: move R_j from G_l to an appropriate group G_m
- 21: update a^l and a^m
- 22: **else**
- 23: update a^l
- 24: **end if**
- 25: update total sum a_0
- 26: **end for**
- 27: **end while**

For the application of composition-rejection search, reactions are first partitioned into groups depending on their propensities a_j (line 3). Specifically, reactions are partitioned into L groups $G_1, \dots, G_L$ such that a reaction R_j is grouped into group G_l if its propensity a_j satisfies $2^{u_l-1} \leq a_j < 2^{u_l}$ where the exponent u_l is computed such that $u_l = \lceil \log_2(a_j) \rceil$ where $\lceil - \rceil$ denotes the ceiling operator. The number of groups L can be fixed if the global minimum propensity value and the global maximum propensity value for all reactions are known or can be estimated by using physical reasoning (e.g., bounding all possible values of the population of each species). Such extreme values, however, are often unknown in practice, thus the number of groups L must be dynamically increased during the simulation.

Consider the selection of the next reaction firing. Let $a^l = \sum_{R_j \in G_l} a_j$ be the sum of propensities of reactions in group G_l . It is obvious that the total propensity is $a_0 = \sum_{l=1}^L a^l$. The search for the next reaction firing R_μ in SSA-CR is composed of two steps. First, a group G_α is selected with probability a^l/a_0 . The selection of the group G_α can be performed by linearly summing values a^l until the smallest index α is found such that $\sum_{l=1}^\alpha a^l \geq r_1 a_0$ where

r_1 is a random number from $\mathcal{U}(0, 1)$. Second, a reaction in the selected group G_α is selected to fire by applying the rejection-based technique with the chosen proposal 2^{u_α} . Specifically, a random and uniform reaction index μ in group G_α is computed, i.e. $\mu = \lceil r_2 |G_\alpha| \rceil$ where $|G_\alpha|$ returns the size of the group G_α and r_2 is a random number from $\mathcal{U}(0, 1)$. The random selected reaction R_μ is tested to accept with probability $a_\mu/2^{u_\alpha}$. In other words, a random number $r_3 \sim \mathcal{U}(0, 1)$ is generated and compared against $a_\mu/2^{u_\alpha}$. The random number r_3 can be generated by recycling r_2 by noting that $r_3 = r_2 |G_\alpha| - \mu$ is uniformly distributed in $[0, 1]$. Thus, if the condition $a_\mu/2^{u_\alpha} \geq r_3$ holds, then R_μ is accepted to fire. Otherwise, the reaction is rejected. In case of rejection, a new random reaction index in G_α is generated and the check is performed again. The selection of reaction firing in the group G_α is repeated until there is a reaction R_μ accepted. The acceptance probability is bounded by $1/2$ because $a_\mu/2^{u_\alpha} \geq 1/2$ by the definition of the group G_α . Thus, in average, a reaction will be accepted after two attempts. The firing time τ of the reaction firing R_μ is generated by sampling the exponential distribution with rate a_0 .

The simulation then moves the system to the new state $X + v_\mu$ and advances the time to $t + \tau$. We note that after the firing of R_μ , reactions must update their propensities to reflect changes in the system state. The reaction dependency graph (built in line 2) is used to reduce the propensity updates for which reactions that actually change after each reaction firing. If the new propensity a_j of reaction R_j does not satisfy the condition of the current group G_l , i.e., $a_j \notin [2^{u_l-1}, 2^{u_l}]$, it must be moved to an appropriate group G_m , which is performed in lines 19 - 24.

3.3 RSSA

The detailed implementation of RSSA is outlined in Algorithm 2. Its principle is to abstract the exact propensity of each reaction into an interval, which is denoted by a propensity lower bound and a propensity upper bound, and then uses these propensity bounds to select the next reaction firing and generating its firing time. Specifically, a propensity bound $[a_j, \bar{a}_j]$ is defined for each reaction R_j , with $j = 1 \dots M$. These propensity bounds are derived by imposing an arbitrarily bound on the population of each species in the state X . For species S_i , a lower bound $\underline{X}_i$ and an upper bound $\bar{X}_i$ is defined around its current population $X_i(t)$. In particular, let $0 \leq \delta_i \leq 1$ be a parameter. The population interval (also called fluctuation interval) that bounds the current population of species S_i is computed as $[\underline{X}_i, \bar{X}_i] = [(1 - \delta_i)X_i, (1 + \delta_i)X_i]$. The state thus satisfies the inequality $\underline{X} \leq X(t) \leq \bar{X}$ for each species. Having the fluctuation interval $[\underline{X}, \bar{X}]$, the propensity bound $[a_j, \bar{a}_j]$ is derived such that it encloses all possible values of the propensity function a_j over such the fluctuation interval. For common reaction kinetics such as the mass action kinetics or the Michaelis-Menten kinetics where the propensity a_j is monotonic function of the state, the propensity bounds can be computed by putting $\underline{a}_j = a_j(\underline{X})$ and $\bar{a}_j = a_j(\bar{X})$. In cases that a_j is a complex function, a numerical opti-

mization technique can be applied to derive the bounds. We note that, for the simulation of RSSA, the exact minimum and maximum are not needed. The simulation works with any reasonable (tight) bounds for the propensity over the fluctuation interval and the selection of the next reaction firing by RSSA is always exact.

Algorithm 2 Rejection-Based SSA (RSSA)

Require: a biochemical reaction network of M reactions in which each reaction R_j , $j = 1, \dots, M$, is accompanied with the state change vector $\mathbf{v}_j$ and the propensity a_j , the fluctuation rate δ_i for each species S_i , $i = 1, \dots, N$, the initial state $\mathbf{x}_0$ at time 0 and the simulation ending time T_{max}

Ensure: a trajectory $X(t)$, $0 \leq t \leq T_{max}$, of the biochemical reaction network

- 1: initialize time $t = 0$ and state $X = \mathbf{x}_0$
- 2: build the species-reaction (SR) dependency graph $\mathcal{G}$
- 3: define a new $[X_i, \overline{X}_i]$ around X_i , for $i = 1, \dots, N$
- 4: compute propensity bounds $\overline{a}_j$ and $\underline{a}_j$ for $j = 1, \dots, M$
- 5: compute total sum $\overline{a}_0 = \sum_{j=1}^M \overline{a}_j$
- 6: **while** ($t < T_{max}$) **do**
- 7: **repeat**
- 8: set $u = 1$
- 9: set $accepted = \text{false}$
- 10: **repeat**
- 11: generate three random numbers $r_1, r_2, r_3 \sim \mathcal{U}(0, 1)$
- 12: select R_μ with minimum index μ satisfied $\sum_{j=1}^\mu \overline{a}_j \geq r_1 \overline{a}_0$
- 13: **if** ($r_2 \leq (\underline{a}_\mu / \overline{a}_\mu)$) **then**
- 14: set $accepted = \text{true}$
- 15: **else**
- 16: evaluate a_μ with state X
- 17: **if** ($r_2 \leq (a_\mu / \overline{a}_\mu)$) **then**
- 18: set $accepted = \text{true}$
- 19: **end if**
- 20: **end if**
- 21: set $u = u \cdot r_3$
- 22: **until** $accepted$
- 23: compute firing time $\tau = (-1/\overline{a}_0) \ln(u)$
- 24: update state $X = X + \mathbf{v}_\mu$
- 25: set time $t = t + \tau$
- 26: **until** (exists $(X_i \notin [X_i, \overline{X}_i])$)
- 27: **for all** $(X_i \notin [X_i, \overline{X}_i])$ **do**
- 28: define a new $[X_i, \overline{X}_i]$ around X_i
- 29: **for all** ($R_j \in \text{ReactionsAffectedBy}(S_i)$) **do**
- 30: compute new propensity bounds $\overline{a}_j$ and $\underline{a}_j$
- 31: update $\overline{a}_0$
- 32: **end for**
- 33: **end for**
- 34: **end while**

The selection of the next reaction firing by RSSA is in lines 10 - 22. First, a candidate reaction R_μ is selected with probability $\overline{a}_\mu / \overline{a}_0$ where $\overline{a}_0 = \sum_{j=1}^M \overline{a}_j$. RSSA realizes the candidate reaction by linearly summing propensity upper bounds until it finds the smallest reaction index μ satisfying the inequality

$\sum_{j=1}^{\mu} \bar{a}_j > r_1 \cdot \bar{a}_0$ where r_1 is a random number in $U(0, 1)$. The candidate reaction R_μ then enters a rejection test with success probability $a_\mu/\bar{a}_\mu$. This test requires to evaluate the exact propensity a_μ , but RSSA avoids computing it as much as possible. It uses the fact that if the candidate is accepted with probability $\underline{a}_\mu/\bar{a}_\mu$, then it is also accepted with $a_\mu/\bar{a}_\mu$ because of $\underline{a}_\mu \leq a_\mu$. Thus, RSSA generates a random number $r_2 \sim U(0, 1)$ and first checks whether the condition $r_2 \leq \underline{a}_\mu/\bar{a}_\mu$ holds. If it is in fact the case, then RSSA accepts R_μ without evaluating a_μ . If the test fails, a_μ is evaluated with the current state and then is used to check again the condition $r_2 \leq a_\mu/\bar{a}_\mu$. If R_μ is accepted after the rejection test, its firing time is then computed. Otherwise, a new selection is performed.

The firing time τ of the accepted reaction R_μ in RSSA is generated following an *Erlang*($k, \bar{a}_0$) distribution with the shape parameter k being the number of trials until the acceptance and the rate parameter $\bar{a}_0$ being the sum of propensity upper bounds. Such a distribution is in fact the sum of k exponentially distributed random numbers with the same rate $\bar{a}_0$. Line 23 uses the convolution method to implement the generation of the *Erlang*-distributed firing time as $\tau = (-1/\bar{a}_0) \ln(\prod_{i=1}^k u_i)$ where u_i is a random number from $U(0, 1)$. This implementation requires only one logarithm operation, instead of k logarithm operations by a naive sum of exponentially distributed random numbers.

Having the reaction R_μ and its firing time τ , the simulation updates the state X and simulation time t accordingly. However, RSSA does not recompute propensity bounds after each reaction firings if the state X is still confined in the current fluctuation interval $[\underline{X}, \bar{X}]$. Specifically, RSSA checks whether the condition $X(t) \in [\underline{X}, \bar{X}]$ holds after reaction firings. If the check returns true, the next simulation step can be performed without updating the propensity bounds. This is often the case because only a few species are changed by a reaction. In uncommon cases in which the state is outside the current fluctuation interval, i.e. $X(t) \notin [\underline{X}, \bar{X}]$, a new fluctuation interval is defined and new propensity bounds for reactions are derived. RSSA further reduces the number of reactions having to recompute their propensity bounds by applying a Species-Reaction (SR) dependency graph (built in line 2). The SR dependency graph is a bipartite directed graph showing the dependency of species and reactions. A directed edge between a species S_i and a reaction R_j is included in the SR graph if changes in the population of S_i affect the propensity a_j . Let $\text{ReactionsAffectedBy}(S_i)$ be the set of reactions affected by species S_i extracted from the SR graph, then only reaction $R_j \in \text{ReactionsAffectedBy}(S_i)$ should recompute their propensity bounds when species S_i exits its fluctuation interval.

3.4 Comparison of SSA-CR and RSSA

We have shown through our descriptions of SSA-CR and RSSA in previous sections that although these algorithms employ the rejection-based technique

during the simulation, their use of this technique is different in detail. We follow in this section with a critical comparison of these algorithms as well as an analytical analysis for each of these algorithms. We compare the advantages and drawbacks of these algorithms in order to help making an informed choice of the suitable algorithm given properties of biochemical reactions.

SSA-CR and RSSA are both exact algorithms (we refer to the original work on these algorithms for their exactness proofs), but they focus on different computational bottlenecks of the stochastic simulation. SSA-CR focuses on improving the search for the next reaction firing by the composition-rejection method, while RSSA aims to reducing the average number of propensity updates during the simulation. More in detail, SSA-CR relies on the exact propensity a_j of reactions to select the next reaction firing and compute its firing time. The rejection-based technique is used to select the reaction in a group. The average number of rejection test is 2, because the acceptance probability of the reaction in a group is bound by $1/2$. We note that the acceptance probability can be tuned by changing the condition in partitioning of reactions into groups. The higher acceptance probability is, the smaller the base, hence the larger the number of groups L , should be used. SSA-CR will convert to SSA if the number of groups L is equal to the number of reactions M , i.e., each group contains one reaction and the acceptance probability is 1. The base 2 is often used in the implementation of SSA-CR because it can be easily implemented in a programming language. In contrast, RSSA relies on propensity bound $[\underline{a}_j, \bar{a}_j]$ of reactions. The propensity bounds are derived by constraining the population of species in state $X(t)$ into an arbitrary fluctuation interval $[\underline{X}, \bar{X}]$. Both the selection of the next reaction and the generation of its firing time used propensity bounds. The candidate reaction, which is selected by using propensity bounds, will be validated to fire by applying the rejection test. Although the rejection test in RSSA requires the exact propensity of the candidate reaction, the computation of the exact propensity a_j is only performed when necessary. The acceptance probability of the next reaction in RSSA is $a_0/\bar{a}_0$ and is bounded below by $\underline{a}_0/\bar{a}_0$ where $\underline{a}_0 = \sum_{j=1}^M \underline{a}_j$. Thus, tighter lower/upper bounds for propensities yield a better acceptance probability. This acceptance probability can be tuned by changing the size of the fluctuation interval $[\underline{X}, \bar{X}]$. In fact, we can increase the acceptance probability to 1 by setting $\underline{X} = \bar{X} = X(t)$, hence $a_j = \underline{a}_j = \bar{a}_j$ for all $j = 1, \dots, M$. RSSA in this case converges to the original SSA. In addition, using propensity bounds to generate reaction firing time makes RSSA easy for handling complex reaction mechanisms such as time-dependent reaction rates [9, 10]. This is the distinctive feature of RSSA that SSA-CR does not have.

A direct consequence of SSA-CR by relying on exact propensity to select the reaction firings is that it has to update propensities of reactions to reflect the changes in the state after each reaction firing because propensities a_j of reactions depend on the state $X(t)$. The reactions in a group as well may have to move to another group in order to maintain the group condition. For adding/deleting a reaction to/from a group, a complex data structure must be used. Specifically, the underlying data structure for implementing the group

in SSA-CR must support dynamic memory operations (e.g., a linked list) in order to dynamically increase/decrease the group size. A lookup table (e.g., a hash table) is also required to retrieve a reaction in a group during the update. We note that the operations for dynamically locating and freeing memory are often very computational intensive tasks, which negatively affect performance of SSA-CR. In contrast, RSSA does not update propensity bounds after each reaction firing. The same propensity bounds can be used for many simulation steps. The update of propensity bounds is performed only when state $X(t)$ moves out of its current fluctuation interval, i.e., $X(t) \notin [\underline{X}, \overline{X}]$. Often only a small number of species involved in a reaction, then by using SR dependency graph the number of propensity updates can be further reduced. We remark that the reaction dependency graph employed by SSA-CR and the SR dependency graph used in RSSA are different. The former one shows the relationship between reactions, while the latter focuses on the relationship between species and their corresponding reactions. We can obtain a reaction dependency graph from a SR dependency graph by removing the species nodes and adding edges for reactions.

Let us now analyze the time complexity of each algorithm. We consider two main tasks: 1) searching for the next reaction firing and 2) updating propensity of affected reactions because they largely dominate the total computational cost. Other tasks such as updating state and time, random number generations and writing outputs to files are small comparing to these costs or independent to the simulation. In fact, often only a few species are required to update their population after each reaction firing because each reaction often involves a small number of species, while the random number generations and writing outputs to files depend on the underlying operating systems. It is thus reasonable to assume these tasks taking constant time to perform in the networks having a large number of reactions. These costs, however, must be taken into account when considering models having a small number of reactions. We also remark that time complexity only shows the asymptotic running time of an algorithm. The actual performance of an algorithm requires to execute the algorithm on a specific platform.

Consider the computational time complexity of SSA-CR. Its search is composed of the cost for selecting the group and the cost for searching the next reaction firing in the group. The cost for selecting a group is proportional to the number of groups, i.e., $O(L)$. The time complexity of the acceptance-rejection for selecting the next reaction in a group is constant because it is independent of the number of reactions in the group. Furthermore, the average number of rejection tests is bound by 2. Thus, the time complexity of the search by SSA-CR is $O(L)$. For the update cost, let D be the average number of reactions depending on a reaction (i.e., the average out degree of the nodes of the reaction dependency graph). The cost for propensity updates after each reaction firing in SSA-CR is $O(D)$. The total time complexity of SSA-CR is thus $O(L + D)$. Because the number of groups L is independent of the number of reactions M and is tunable, we can constrain the number of groups L to a small constant during the simulation, then the search for next reaction firing

in SSA-CR is $O(1)$. The number of propensity updates in SSA-CR, however, cannot be changed because it is the property of the reaction network. In the original work of SSA-CR [27], the authors assumed that the number reactions D that needs to update their propensities after each reaction firing is a small constant. Based on this assumption the propensity update cost in each step of SSA-CR is $O(1)$, thus the total simulation time complexity is $O(1)$. This underlying assumption in the analysis of SSA-CR when applying for reaction models where the number of propensity updates D is proportional to the number of reactions, i.e., $D = O(M)$, is often broken. In this setting, the total time complexity of SSA-CR is $O(M)$, which is the same as the original SSA. The update cost of SSA-CR, however, is much more expensive than SSA because it needs to maintain also the complex underlying data structures. This explains for the reason why SSA-CR is slower than SSA in real-world biological systems where reactions are highly interconnected (see the FcRI model in Section 4.4).

For RSSA, the computational cost for selecting the next reaction firing is composed of the cost to realize a candidate reaction and the number of rejection tests. RSSA selects the candidate reaction by linearly summing propensity upper bounds. Thus, the time complexity for selecting the candidate reaction is $O(M)$. Let α be the average number of times the search is performed until the candidate reaction is accepted. The number of rejection tests α is equal to the reciprocal average acceptance probability of a candidate reaction, i.e., $\alpha = \bar{a}_0/a_0 \leq \bar{a}_0/\underline{a}_0$, and is often bounded by a small constant. The search time complexity of RSSA is thus $O(M)$. Let D be the average number of reactions affected by a species extracting from the SR dependency graph. The propensity updates by RSSA is $O(D/\beta)$ where β is the average number of skipped updates during the simulation. β is the average frequency of $X(t) \in [\underline{X}, \bar{X}]$. In contrast to SSA-CR, the propensity update in RSSA can be controlled during the simulation by tuning the fluctuation interval $[\underline{X}, \bar{X}]$. In general, the narrower the interval $[\underline{X}, \bar{X}]$ we use, the higher the acceptance probability is, but the more frequently the propensity updates perform. In the extreme case where the fluctuation interval degenerates into the state $\underline{X} = \bar{X} = X(t)$, then $\alpha = \beta = 1$, the acceptance probability is 1 and reactions have to update their propensities after every reaction firing as in SSA. On the other hand, if the fluctuation interval is increased, the number of updates for propensity bounds is reduced, but also lower the acceptance probability. In particular, we can define a fluctuation interval so that no update occurs in the whole simulation ($\beta = \infty$), thus the update cost is zero and the computational cost of RSSA depends only the search cost. However, the acceptance probability of a candidate reaction in RSSA decreases significantly because $\underline{a}_j \ll a_j \ll \bar{a}_j$, in turn negatively affecting the search and the simulation performance. We, in principle, can obtain a fluctuation interval that optimizes both the search cost and the update cost of RSSA, but finding such a value might be computationally expensive. In practice, a fluctuation interval that gives a good performance is acceptable. We found that, in general, a fluctua-

Table 1 The search time and update time complexities of SSA, SSA-CR and RSSA. The time complexities are provided in terms of the number of reactions M , the number of groups L used to group reactions, the average number of propensity updates D after a reaction firing and the average number of skipped updates during the simulation β .

Algorithm	Search cost	Update cost
SSA (Direct Method)	$O(M)$	$O(D)$
SSA-CR	$O(L)$	$O(D)$
RSSA	$O(M)$	$O(D/\beta)$

tion defined around $\pm 10\%$ to $\pm 20\%$ of the current state often gives the good performance (see our numerical experiments in Section 4).

In Table 1, we summarize the time complexity of rejection-based algorithms discussed above in term of number reactions M , number of groups L and reaction connectivity D and use SSA as a reference. We note that the update cost of SSA shown in the table is $O(D)$ by applying the dependency graph [14] (we note that in the original work [1, 2], it is $O(M)$). The table informs that for a reaction network that has a large number of reactions $M \gg L$ and if the number of the average number of propensity updates D is a small constant, then SSA-CR will give the best performance. On the other case if D is large, RSSA should be used instead.

4 Case studies

This section numerically evaluates performance of SSA-CR and RSSA on four biological models that are: the linear chain model, the colloidal model, the Gemcitabine model and the FcεRI signaling model. We also include SSA in the comparison to serve as a reference. In order to compare performance of algorithms, we run 100 simulation runs for each algorithm and then compute the average simulation runtime (the variance were omitted because it is negligible). For each simulation run, the results are collected after 10^7 simulation iterations. For the simulation of RSSA, the population bound of each species for all simulated models is defined around $\pm 10\%$ of its population. The aim of the benchmark is to verify our analysis discussed the previous section. The first two models are theoretical models used to show the scalability of the algorithms under controlled conditions. The Gemcitabine model and the FcεRI model are real-world models used to demonstrate performance of considered algorithms in the practical settings. The Gemcitabine model is considered as a small model because it contains about dozens of reactions, while the FcεRI model is a large model with thousand of reactions. We implemented these simulation algorithms and run in an Intel i5-540M processor. All the implementations of algorithms discussed in the paper are freely available at <http://www.cosbi.eu/research/prototypes/rssa>.

Fig. 1 Performance of SSA, SSA-CR, RSSA on the linear chain model. The top figures show the search time and the propensity update time. The bottom figure show the total simulation time which is the sum of the the search time, propensity update time and all other tasks (e.g., recording state and writing result to file).

4.1 Linear chain model

The linear chain model is an artificial model that we use to verify the scalability of the considered algorithms when the search for next reaction frings contributes most to the simulation and the propensity update only contributes a small percentage.

The model consists of N species S_i with $i = 1 \dots N$ in which a species is transformed to the species S_j such that

$$S_i \xrightarrow{c_i} S_j, \text{ for } i = 1 \dots N \text{ and } j = (i + 1) \bmod N \quad (5)$$

where c_i is the rate constant of the transformation. The kinetics rate of all reactions is set to $c_i = 1$ for $i = 1 \dots N$. The initial population of each species S_i for $i = 1 \dots N$ is randomly taken in the interval between 0 and 10000. The number of reactions M in the linear chain is equal to the number of species, i.e., $M = N$, and the number of affected reactions which needs to update their propensities in the linear chain model is fixed by 2.

Figure 1 shows the performance of algorithms by increasing the values of N (100, 500, 1000, 5000, 10000 and 50000). For this model the propensity update

is constant because the number of reactions required updating propensities is constant. The main task that contributes to the total simulation is the search cost. The figure shows that the search cost of SSA-CR remains constant as increasing N . The performance of SSA-CR is thus the best because the search of both SSA and RSSA scales linear with N . In particular, for this model with $N = 50000$, SSA-CR is about 12 and 17.5 times faster than SSA and RSSA, respectively. We note that RSSA has the worst performance in simulating this model because its search requires more effort than SSA. This result is consistent with our theoretical analysis in previous sections.

4.2 Colloidal model

The colloidal aggregation is also an artificial model that we use to demonstrate the performance of considered algorithms when both the search and update increasing with the number of reactions. The colloidal aggregation is a process that forms big clusters from colloidal particles, e.g., proteins, nanobeads [44, 45].

The model contains aggregates with up to N colloidal particles. The reaction network that models the colloidal aggregation process is defined by:

where $c_{n,m}$ and $c_{p,q}$ are rate constant of forward and reverse processes, respectively. The initial population of all species in this experiment is set to $\#S_i = 1000$ for $i = 1 \dots N$. The kinetics rate of all reaction is set to $c_{n,m} = c_{p,q} = 1$. For this model, the number of reactions M in the colloidal aggregation network is increasing in quadratically in the number of particles, i.e., $M = O(N^2)$, and the average number of affected reactions which needs to update their propensities is linear with the number of particles, i.e., $D = O(N)$.

Figure 2 shows performance of algorithms in simulating the colloidal aggregation model by increasing N (10, 50 and 100). As shown in the figure, although the search cost of the SSA-CR is constant, its update cost is significantly high, negatively affecting its performance. The figure also confirms our analysis that the performance of SSA-CR is even slower than SSA when the update has important effects in the simulation performance, but often ignored in previous analysis. Reducing propensity update cost returns a huge advantage for RSSA. For example, in case $N = 100$, RSSA is about 5 and 10 times faster than SSA and SSA-CR, respectively. We remark that we do not include in the figure the runtimes of algorithms with higher values of N . In fact, we have tried SSA, SSA-CR and RSSA with $N = 500$, but only RSSA is able to simulate in reasonable time. Specifically, RSSA takes, in average, 3371.129s to simulate the model, while SSA and SSA-CR do not finish their simulation jobs.

Fig. 2 Performance of SSA, SSA-CR, RSSA on the colloidal model. The top figures show the search time and the propensity update time. The bottom figure show the total simulation time which is the sum of the the search time, propensity update time and all other tasks (e.g., recording state and writing result to file).

4.3 Gemcitabine model

We use the gemcitabine model [40] which is developed to understand the mechanisms of resistance to gemcitabine efficacy. Gemcitabine (2,2-difluorodeoxycytidine, dFdC) is an anti-cancer chemotherapy drug. It has been used to treat different types of cancer including non-small-cell lung cancer, pancreatic cancer, bladder cancer, and breast cancer [38,39].

The model consists of 22 species and 29 reactions listed in Table 2. It details mechanisms for the race between gemcitabine and natural nucleoside triphosphate dCTP for DNA incorporation which includes the mechanisms of resistance by considering the role of ribonucleotide reductase (RR), deoxycytidine kinase (dCK) and human equilibrative nucleoside transporter1 (hENT1). The initial condition is assigned to $\#dCK = 1000$, $\#RR = 1000$, $\#dCMPD = 1000$, $\#CDP = 2000$, $\#dFdC\text{-out} = 100000$ and all other species are zero. The stochastic rates of reactions are: $c_1 = 99.7234$, $c_2 = 2.61675 \times 10^{-3}$, $c_3 = 4.72336 \times 10^{-5}$, $c_4 = 5.08194 \times 10^{-1}$, $c_5 = 1.04994 \times 10^{-2}$, $c_6 = 8.75208 \times 10^{-1}$, $c_7 = 23.7162$, $c_8 = 2.12216$, $c_9 = 25.2037$, $c_{10} = 14.4908$, $c_{11} = 9.68 \times 10^{-3}$, $c_{12} = 5.60415 \times 10^{-4}$, $c_{13} = 0.7844$, $c_{14} = 4.20541 \times 10^{-2}$, $c_{15} = 1.64322$, $c_{16} = 9.05139 \times 10^{-4}$, $c_{17} = 4.76746 \times 10^{-3}$, $c_{18} = 4.559 \times 10^{-5}$,

$c_{19} = 0.544456$, $c_{20} = 7.37496 \times 10^{-3}$, $c_{21} = 100$, $c_{22} = 0.5$, $c_{23} = 252.037$,
 $c_{24} = 5$, $c_{25} = 1.0 \times 10^{-4}$, $c_{26} = 0.1$, $c_{27} = 1.0 \times 10^{-4}$, $c_{28} = 100$, $c_{29} = 0.1$.

Table 2 Gemcitabine model

R_1 : dFdC-out $\xrightarrow{c_1}$ dFdC	R_2 : dFdC $\xrightarrow{c_2}$ dFdC-out
R_3 : dFdC-out $\xrightarrow{c_3}$ dFdU	R_4 : dFdU $\xrightarrow{c_4}$ dFdU-out
R_5 : dFdC + dCK $\xrightarrow{c_5}$ dFdC-MP + dCK	R_6 : dFdC-MP $\xrightarrow{c_6}$ dFdC
R_7 : dFdC-MP $\xrightarrow{c_7}$ dFdC-DP	R_8 : dFdC-DP $\xrightarrow{c_8}$ dFdC-MP
R_9 : dFdC-DP $\xrightarrow{c_9}$ dFdC-TP	R_{10} : dFdC-TP $\xrightarrow{c_{10}}$ dFdC-DP
R_{11} : dFdU + dCK $\xrightarrow{c_{11}}$ dFdU-MP + dCK	R_{12} : dFdU-MP $\xrightarrow{c_{12}}$ dFdU
R_{13} : dFdU-MP $\xrightarrow{c_{13}}$ dFdU-DP	R_{14} : dFdU-DP $\xrightarrow{c_{14}}$ dFdU-MP
R_{15} : dFdU-DP $\xrightarrow{c_{15}}$ dFdU-TP	R_{16} : dFdU-TP $\xrightarrow{c_{16}}$ dFdU-DP
R_{17} : dFdC $\xrightarrow{c_{17}}$ dFdU	R_{18} : dFdC-MP + dCMPD $\xrightarrow{c_{18}}$ dFdU-MP + dCMPD
R_{19} : dFdC-TP $\xrightarrow{c_{19}}$ dFdC-TP-DNA	R_{20} : dFdU-TP $\xrightarrow{c_{20}}$ dFdU-TP-DNA
R_{21} : $\emptyset \xrightarrow{c_{21}}$ CDP	R_{22} : CDP + RR $\xrightarrow{c_{22}}$ dCDP + RR
R_{23} : dCDP $\xrightarrow{c_{23}}$ dCTP	R_{24} : dCTP $\xrightarrow{c_{24}}$ CTP-DNA
R_{25} : dCTP + dCK $\xrightarrow{c_{25}}$ dCTP-dCK	R_{26} : dCTP-dCK $\xrightarrow{c_{26}}$ dCTP + dCK
R_{27} : dFdC-DP + RR $\xrightarrow{c_{27}}$ dFdC-DP-RR	R_{28} : dFdC-TP + dCMPD $\xrightarrow{c_{28}}$ dFdC-TP-dCMPD
R_{29} : dFdC-TP-dCMPD $\xrightarrow{c_{29}}$ dFdC-TP + dCMPD	

Figure 3 shows the performance of SSA, SSA-CR and RSSA on simulating the Gemcitabine model while detailed runtime of simulation tasks is listed in Table 3. As shown in the figure and the table, both the search time and update time of SSA-CR are slower than SSA. It is because for this small model the complex selection procedure of SSA-CR negatively affects its performance. In contrast, although the search time of RSSA is slower than SSA, its update cost is faster than SSA. Specifically, we observe for this model, the average number of reactions needed updating their propensities after each reaction firing is 4.27 and the maximum is 6. RSSA only performs 2.44×10^6 updates (about 15% of the 10^7 simulation steps). The result is the performance of RSSA is about 2 times faster than SSA-CR and 24% faster than SSA.

4.4 FceRI model

We use the high-affinity IgE receptor, referred to as FceRI, model developed by Liu *et al.* [43] to show the performance of algorithms on the large biological

Table 3 Detailed runtime of simulation tasks by algorithms in simulating Gemcitabine model.

Algorithm	Search cost (s)	Update cost (s)	Other tasks (s)
SSA	0,348	9,732	4,136
SSA-CR	3,222	15,054	3,496
RSSA	3,044	4,159	3,463

Fig. 3 Performance of SSA, SSA-CR, RSSA on the Gemcitabine model. The simulation runtime of a algorithms is the sum of search time, propensity update time and all other tasks (e.g., recording state and writing result to file).

Fig. 4 Performance of SSA, SSA-CR, RSSA on the FcεRI model. The simulation runtime of a algorithms is the sum of search time, propensity update time and all other tasks (e.g., recording state and writing result to file).

model. FcεRI is a high-affinity receptor for the antigen-specific immunoglobulin E (IgE). It is the target for many immunological pathways related to allergic disease. The FcεRI receptor is consisting of three subunits that are: an α chain (FcεRI α), a β chain (FcεI β), and two disulfide bridge connected γ chains (FcεRI γ). The activation of the FcεRI pathway initiates when the antigen IgE bind to the binding site of the α chain. The cross-linking of the IgE-antigen complex and the aggregation of the FcεRI leads to the degranulation and release of allergic mediators from the immune system [41,42].

The FcεRI model that we consider in this section contains 380 species involved in 3862 reactions. Figure 4 depicts the simulation runtime of algorithms in simulating the model. Table 4 shows the detailed runtimes of simulation tasks in each of the considered algorithm.

Table 4 Detailed runtime of simulation tasks by algorithms in simulating FccRI model.

Algorithm	Search cost (s)	Update cost (s)	Other tasks (s)
SSA	79,403	868,384	7,876
SSA-CR	3,522	980,418	5,102
RSSA	97,477	0,586	8,114

As shown in Figure 4 and Table 4, the search by SSA-CR achieves the best performance. Its update cost, however, is the worst. For simulating this model, the update cost of SSA-CR contributes up to 99% of its total simulation time. The update cost of SSA also contributes about 90% of its total cost. The reason for the high update cost in SSA and SSA-CR is due to the large number of reactions that need their propensities after each reaction firing. For this model, there are, in average, 115.80 propensity updates per reaction firing and at most 388 propensity updates (corresponding to 10% of reaction network). Improving the search cost in this situation does not improve the performance of SSA-CR. In fact, the performance of SSA-CR is even slower than SSA. The huge performance improvement achieved by RSSA comes from reducing the update cost. In fact, RSSA only performs 93322 propensity updates (about 1% of the total simulation steps as comparison with SSA and SSA-CR). The result is that RSSA is about 9 and 9.3 times faster than both SSA and SSA-CR, respectively.

5 Conclusions

We critically assessed the use of the key rejection-based technique in the implementations of SSA-CR and RSSA for improving efficiency of stochastic simulation. Although both of the algorithms employ the rejection-based technique during the simulation to select the next reaction firings, the detailed implementations of these algorithms are different. SSA-CR relies on exact propensities of reactions to select the next reaction firing and compute the firing time. It reduces the search time complexity by grouping reactions and applying rejection-based technique in selecting the reaction in a group. In contrast, RSSA uses propensity bounds, an interval that bound all possible values of the exact propensity, to select the next reaction and its firing time. The exact propensity of a reaction in RSSA is only computed when necessary. Its aim by doing so is to reduce the average number of propensity updates during the simulation. We provided a detailed analysis of these algorithms taking both the search and update of reactions into account. The omission of one of these facts in the assessment may lead to a misleading choice of suitable algorithm when applying for a given model. Our numerical analysis confirmed that for reaction networks where the search cost dominates the simulation time then SSA-CR should be used, and for reaction networks where the update cost is the dominant task, which often the case for practical models, then RSSA is the choice.

Acknowledgements

This work has been partially done when VHT was at the Microsoft Research - University of Trento Centre for Computational and Systems Biology (COSBI), Italy. The research supported by Academy of Finland grant 311639, "Algorithmic Designs for Biomolecular Nanostructures (ALBION)".

References

1. Daniel Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *J. Comp. Phys.*, 22(4):403–434, 1976.
2. Daniel Gillespie. Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.*, 81(25):2340–2361, 1977.
3. Adam Arkin, John Ross, and Harley H. McAdams. Stochastic kinetic analysis of developmental pathway bifurcation in phage lambda-infected escherichia coli cells. *Genetics*, 149(4):1633–1648, 1998.
4. Harley H. McAdams and Adam Arkin. It’s a noisy business! genetic regulation at the nanomolar scale. *Trends in Genetics*, 15(2), 1999.
5. Harley H. McAdams and Adam Arkin. Stochastic mechanisms in gene expression. *PNAS*, 1997.
6. Jonathan M. Raser and Erin K. O’Shea. Noise in gene expression: Origins, consequences, and control. *Science*, 309:2010–2013, 2005.
7. Daniel Gillespie. A rigorous derivation of the chemical master equation. *Physica A*, 188(1-3):404–425, 1992.
8. David F. Anderson. A modified next reaction method for simulating chemical systems with time dependent propensities and delays. *J. Chem. Phys.*, 127(21):214107, 2007.
9. Vo H. Thanh and Corrado Priami. Simulation of biochemical reactions with time-dependent rates by the rejection-based algorithm. *J. Chem. Phys.*, 143(5):054104, 2015.
10. Vo H. Thanh, Luca Marchetti, Federico Reali *et al.* Incorporating Extrinsic Noise into the Stochastic Simulation of Biochemical Reactions: A Comparison of Approaches. *J. Chem. Phys.*, 148(6):064111, 2018.
11. Xiaodong Cai. Exact stochastic simulation of coupled chemical reactions with delays. *J. Chem. Phys.*, 126(12):124108, 2007.
12. Vo H. Thanh, Corrado Priami and Roberto Zunino. Efficient rejection-based simulation of biochemical reactions with stochastic noise and delays. *J. Chem. Phys.*, 141(13), 2014.
13. Vo H. Thanh, Roberto Zunino and Corrado Priami. Efficient Stochastic Simulation of Biochemical Reactions with Noise and Delays. *J. Chem. Phys.*, 146(8):084107, 2017.
14. Michael Gibson and Jehoshua Bruck. Efficient exact stochastic simulation of chemical systems with many species and many channels. *J. Phys. Chem. A*, 104(9):1876–1889, 2000.
15. Yang Cao, Hong Li, and Linda Petzold. Efficient formulation of the stochastic simulation algorithm for chemically reacting systems. *J. Chem. Phys.*, 121(9):4059, 2004.
16. James McCollum Gregory D. Peterson, Chris D. Cox *et al.* The sorting direct method for stochastic simulation of biochemical systems with varying reaction execution behavior. *Comp. Bio. Chem.*, 30(1):39–49, 2006.
17. James Blue, Isabel Beichl, and Francis Sullivan. Faster monte carlo simulations. *Phys. Rev. E*, 51(2):867–868, 1995.
18. Vo H. Thanh and Roberto Zunino. Tree-based search for stochastic simulation algorithm. In *Proc. of ACM-SAC*, 2012.
19. Vo H. Thanh and Roberto Zunino. Adaptive tree-based search for stochastic simulation algorithm. *Int. J. Computational Biology and Drug Design*, 7(4):341–357, 2014.
20. Rajesh Ramaswamy, Nlido Gonzalez-Segredo, and Ivo F. Sbalzarini. A new class of highly efficient exact stochastic simulation algorithms for chemical reaction networks. *J. Chem. Phys.*, 130(24):244104, 2009.
21. Daniel Gillespie. Approximate accelerated stochastic simulation of chemically reacting. *J. Chem. Phys.*, 115:1716–1733, 2001.
22. Vo H. Thanh, Corrado Priami and Roberto Zunino. Accelerating rejection-based simulation of biochemical reactions with bounded acceptance probability. *J. Chem. Phys.*, 144(22):224108, 2016.
23. Luca Marchetti, Corrado Priami and Vo H. Thanh. HRSSA - efficient hybrid stochastic simulation for spatially homogeneous biochemical reaction networks. *J. Comp. Phys.*, 317:301–317, 2016.
24. Luca Marchetti, Corrado Priami and Vo H. Thanh. *Simulation Algorithms for Computational Systems Biology*. Springer, 2017.

25. Luc Devroye *Non-Uniform Random Variate Generation*. Springer-Verlag, 1986.
26. Wolfgang Hörmann, Josef Leydold and Gerhard Derflinger. *Automatic Nonuniform Random Variate Generation*. Springer-Verlag, 2004.
27. Alexander Slepoy, Aidan P. Thompson, and Steven J. Plimpton. A constant-time kinetic Monte Carlo algorithm for simulation of large biochemical reaction networks. *J. Chem. Phys.*, 128(20):205101, 2008.
28. Tim Schulze. Efficient kinetic Monte Carlo simulation. *J. Comp. Phys.*, 227(4):2455–2462, 2008.
29. Vo H. Thanh. *On Efficient Algorithms for Stochastic Simulation of Biochemical Reaction Systems*. PhD thesis, University of Trento, Italy. <http://eprints-phd.biblio.unitn.it/1070/>, 2013.
30. Sean Mauch and Mark Stalzer. Efficient formulations for exact stochastic simulation of chemical systems. *IEEE/ACM Trans. on Computational Biology and Bioinformatics*, 8(1):27–35, 2011.
31. Rajesh Ramaswamy and Ivo F. Sbalzarini. A partial-propensity variant of the composition-rejection stochastic simulation algorithm for chemical reaction networks. *J. Chem. Phys.*, 132(4):044102, 2010.
32. Vo H. Thanh, Roberto Zunino and Corrado Priami. Efficient constant-time complexity algorithm for stochastic simulation of large reaction networks. *IEEE/ACM Trans. Comput. Biol. Bioinf.*, 14(3):657–667, 2017.
33. Vo H. Thanh. Stochastic simulation of biochemical reactions with partial-propensity and rejection-based approaches. *Mathematical Biosciences*, 292:67–757, 2017.
34. Vo H. Thanh, Roberto Zunino, and Corrado Priami. On the rejection-based algorithm for simulation and analysis of large-scale reaction networks. *J. Chem. Phys.*, 142(24):244106, 2015.
35. Iain Hepburn, Weiliang Chen, Stefan Wils and Erik De Schutter. STEPS: efficient simulation of stochastic reaction-diffusion models in realistic morphologies *BMC Systems Biology*, 6:36, 2012.
36. Luca Marchetti and Rosario Lombardo and Corrado Priami. HSimulator: Hybrid Stochastic/Deterministic Simulation of Biochemical Reaction Networks *Complexity*, vol. 2017, Article ID 1232868, 12 pages, 2017.
37. Abhishek Gupta and Pedro Mendes. An Overview of Network-Based and-Free Approaches for Stochastic Simulation of Biochemical Systems. *Computation* 6(1):9, 2018.
38. Stephan A. Veltkamp and Jos H. Beijnen and Jan H.M. Schellens. Prolonged Versus Standard Gemcitabine Infusion: Translation of Molecular Pharmacology to New Treatment Strategy. *The Oncologist*, 13(3):261-276, 2008.
39. Niccola Funel *et al.*. Laser microdissection and primary cell cultures improve pharmacogenetic analysis in pancreatic adenocarcinoma. *Laboratory Investigation*, 88:773-784, 2008.
40. Ozan Kahramanogullari, Gianluca Fantaccini, Paola Lecca, Daniele Morpurgo and Corrado Priami. Algorithmic Modeling Quantifies the Complementary Contribution of Metabolic Inhibitions to Gemcitabine Efficacy. *PLOS ONE*, 8(5), 2013.
41. James R. Faeder, William S. Hlavacek, Ilona Reischl *et al.* Investigation of early events in FcεRI-mediated signaling using a detailed mathematical model. *J Immunol*, 170:3769–3781, 2003.
42. Lily A. Chylek, David A. Holowka, Barbara A. Baird *et al.* An interaction library for the FcεRI signaling network. *Front. Immunol.*, 5(172):1664–3224, 2014.
43. Yanli Liu, Dipak Barua, Peng Liu *et al.* Single-cell measurements of ige-mediated FcεRI signaling using an integrated microfluidic platform. *PLoS ONE*, 8(3):60159, 2013.
44. Paul Meakin. Models for colloidal aggregation. *Annual Review of Physical Chemistry*, 39:237–267, 1988.
45. M. Y. Lin, H. M. Lindsay, D. A. Weitz *et al.* Universality in colloid aggregation. *Nature*, 339:360–362, 1989.