RSSA: a Rejection-based Stochastic Simulation Algorithm

Vo Hong Thanh
University of Trento, Italy
vo@disi.unitn.it

Roberto Zunino
University of Trento, Italy
roberto.zunino@unitn.it

April 24, 2013

Abstract

In this study we propose an improvement for the stochastic simulation algorithm (SSA), a standard method to properly realize the stochastic nature of biochemical reactions. Our algorithm is named *RSSA* after "rejection-based SSA", and is tailored for the efficient simulation of large models, which is typically done to understand the complex behavior of regulatory systems. The large models are highly coupled, and full of interconnection and feedback loops. In addition, in these models a complex propensity function is often used to express the stochastic rate of biochemical reactions. Evaluating and updating the reaction propensities is therefore rather computationally expensive. RSSA reduces this computational burden by postponing the full update and only computing the propensity as needed. We experiment with our algorithm on concrete biological models to demonstrate its efficiency.

Keywords: Stochastic simulation, Rejection-based SSA, RSSA, Systems biology.

1 RSSA

Stochastic simulation has widely been accepted as a standard for simulating biochemical reaction systems. The underpinning of stochastic kinetics is based on the probability the next reaction firing in the next infinitesimal time could be

expressed by a propensity function, which only depends on the current system state [6]. Hence, we could construct a realization of the given biochemical system by sampling a reaction with its corresponding probability through a simulation procedure called the stochastic simulation algorithm (SSA) [4, 5]. Basically, SSA uses a Monte Carlo simulation technique to sample the system state. A reaction is selected to fire with corresponding propensity, and the system is then evolved by updating propensity of affected reactions.

Even though a dependency graph, showing the dependency between reactions, can reduce the update to model-dependent so that only locally affected reactions have to recompute their propensities. Still, there many examples where costly update required. For example, let consider a highly coupled reaction network. In this model, the dependency graph becomes very dense. The number of affected reactions is asymptotically increasing by the order of reactions in the network. The simulation time for such systems is largely contributed by the propensity updates of affected reactions. The computational burden is further increasing while a complex propensity is applied. SSA only considers the intrinsic noise which is mainly due to the randomness and discrete nature of molecular species involved in biochemical reactions. However, other sources of noise, yet nevertheless has impact on the dynamic behavior of corresponding biological processes, should be considered in modelling of these processes. In this case, a complex propensity can be exploited in modelling. The power law [3], for example, has been successfully applied to model reactions in non-ideal kinetics to re-produce complex effects e.g., the allosteric effect. The evaluating of complex propensities is indeed very time-consuming. RSSA takes advantage over SSA in this sense. It reduces the computational cost by evaluating the corresponding reaction propensity only as needed and postponing the full propensity updates of affected reactions as possible.

The principle of RSSA is using the over approximation of reaction propensities to select a reaction. The upper-bound of a reaction propensity is controlled by supposing that the current system state is confined to a fluctuation interval. Since the reaction is chosen by a different probability, an acceptance-rejection based procedure is the applied to verify whether to accept or reject this candidate reaction. The exact propensity is used in this verification step in order to ensure the selection of the next reaction firing is correct. In case the candidate reaction is accepted, it is fired and the system state is updated after that. Otherwise, the selection step is repeated. The update of reaction propensities is only necessary as the system state jumps out of the assigned interval.

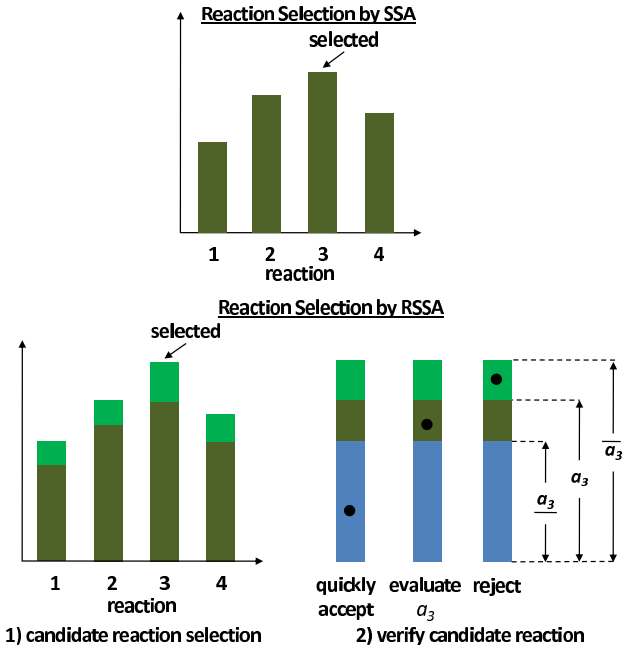
1.1 Selection of reaction firing

Let $X(t)$ be the current system state which is confined in the interval $[\underline{X}, \overline{X}]$. We will compute the upper-bound propensity of each reaction. Let $\overline{a_j}$ be the upper-bound propensity of reaction R_j given $X(t) \in [\underline{X}, \overline{X}]$. The computation of upper-bound propensity for the propensity function which satisfies the monotonic increasing condition is easy and efficient e.g., the mass-action kinetics propensity. In this case we have the relation $\overline{a_j} = a_j(\overline{X})$. For the propensity function which is a complex dependency on the system state e.g., a non-ideal kinetics applied, a numerical calculation procedure should be required to compute the upper-bound propensity given its fluctuation interval $[\underline{X}, \overline{X}]$.

The selection of reaction firing is composed of two steps as following. First, a candidate reaction R_j will be chosen with the probability $\overline{a_j}/\overline{a_0}$ in which $\overline{a_0}$ is the total sum of the upper-bound propensities i.e., $\overline{a_0} = \sum_{j=1}^{m} \overline{a_j}$. In this study, the alias method [14, 13], a fast lookup table method, is used in searching the next reaction firing. However, other search procedures could be applied e.g., a linear search, binary search (see e.g., [2, 11, 1, 12] for details). The alias method is a well-known algorithm for fast generating a random number given the corresponding probability distribution. The working of the alias method is based on the theorem that a distribution of m probability values can be equivalently described by an equi-probable mixture of m two-point distribution. More precisely, we use the alias method to select a random candidate reaction R_j with probability $\overline{a_j}/\overline{a_0}$. For a biochemical network of m reactions, we have to set-up two tables, each size m. A table, called *cut-off probability* table F, stores the value in the first two-point distribution, and a second table, called *alias table* L, contains the alias of the second of the two-point distribution. In running, it requires a random number from a uniform distribution $U(0, 1)$ to randomly lookup an equi-probable mixture. The random number is then rescaled to decide which part of the two-point. Hence, the alias method needs only one comparison and at most two table accesses to select a random candidate reaction. The compensation for the fast lookup search is two supported tables having to be built beforehand.

An acceptance-rejection step is then applied to verify whether to accept the selected candidate reaction R_j. The decision is made by its exact propensity a_j. More precisely, this reaction will be accepted to fire with probability $a_j/\overline{a_j}$. That is a second random number from the uniform distribution is generated. If it is less than $a_j/\overline{a_j}$, the candidate reaction R_j will be accepted to fire. Otherwise, the selection of the reaction firing is repeated. This selection procedure is looping until having a reaction was accepted to fire.

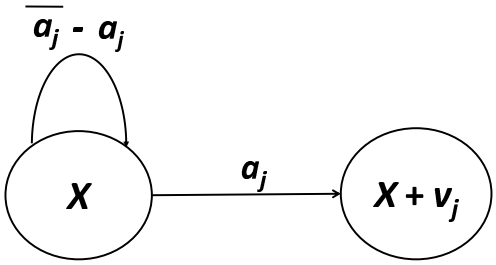
Figure 1: Reaction firing selection by SSA and RSSA



While the evaluation of reaction propensity is time-consuming we could save computational time by applying the squeeze principle. The squeeze is chosen to be the lower-bound propensity. Let $\underline{a_j}$ be lower-bound propensity of candidate reaction R_j. Before calculating the exact propensity of reactionR_j, we first check the condition the random number is less than $\underline{a_j}/\overline{a_j}$. If it is satisfied, we immediately accept the candidate reaction R_j firing without evaluating its propensity a_j. Thus, it only needs to evaluate the actual propensity in case this condition is invalid.

The clarification in reaction firing selection between SSA and RSSA is depicted in Fig. 1. The reaction selection of SSA is done in only one step by exact propensity value. In the figure, reaction R_3 selected by SSA will be fired immediately after selected. In contrast, candidate reaction R_3 has to be verified before it can be fired. This candidate reaction can even be rejected if the random value (the black dot in the figure) is larger than its exact propensity.

Figure 2: Transition rule for a candidate reaction



1.2 Reaction firing time

By introducing $\overline{a_j}$, rather than a_j, to select a reaction firing we add a probability the system state will remain in its current state. We can imagine we have built a new transition rule for candidate reaction R_j given the current state X at time t (see Fig. 2). It has two options: 1) moving to new state $X(t+\tau) = X(t) + v_j$ with rate a_j (the candidate reaction was accepted), or 2) still remaining in its current state $X(t+\tau) = X(t)$ with rate $(\overline{a_j} - a_j)$ (candidate reaction was rejected) where τ is the waiting time. The total rate for m candidate reactions is $\sum_{j=1}^{m} [a_j + (\overline{a_j} - a_j)] = \overline{a_0}$. Therefore, the waiting time for all these transitions is exponential distributed with mean $1/\overline{a_0}$. The firing time of an accepted reaction is the accumulated times of the consecutive rejected candidate reactions. In probability theory, the firing time of the accepted reaction is an $Erlang(k, \lambda)$ distribution with rate parameter $\lambda = \overline{a_0}$ and shape parameter k is the number of consecutive trials before a candidate reaction is accepted.

Hence, we will generate the reaction firing time by sampling its corresponding $Erlang$ distribution. We count the number of trials k until a new state transition occurring due to a reaction accepted. Then, k uniform random numbers, denoted by $u_i \sim U(0,1)$ for $i = 1 \ldots k$, are generated. The firing time is computed by:

$$\tau = (-1/\overline{a_0}) \ln(\prod_{i=1}^{k} u_i) \tag{1}$$

We can approximate the reaction firing time by the mean of the corresponding $Erlang$ distribution i.e., $k/\overline{a_0}$. If the number of trials k is large, a lot of random numbers used in generating the $Erlang$ distribution will be saved.

1.3 The RSSA algorithm

Following the discussions in previous section, we now present the overall of the RSSA. The outline of the RSSA procedure is given in the Algo. 1. The simulation repeats (by the **while** loop at line 1) until the current time t passes over a predetermined simulation time T_{max}. The code inside the simulation loop is logically divided into three parts: 1) preparing data structures for selecting the next reaction firing (line 2 - 5), 2) deciding which reaction fires next and its firing time (line 9-22), and 3) updating and maintaining the system state due to the reaction firing (line 23 - 26).

The preparation starts at line 2. First, the fluctuation interval $[\underline{X}, \overline{X}]$ of the current system state X is defined. Given the fluctuation interval, we will compute the upper-bound propensity $\overline{a_j}$ of a particular reaction R_j. The corresponding total upper-bound propensity $\overline{a_0}$ sums up all these values. We also compute the lower-bound propensity $\underline{a_j}$ of reaction R_j to quickly accept this reaction. The usage of lower-bound propensity will speed up the acceptance process when the evaluation of the propensity is time-consuming.

At line 5 we build supported tables for the alias method with the probability $\overline{a_j}/\overline{a_0}$ for $j = 1 \ldots m$. The generating of cut-off table F and alias table L for the alias method follows the description in [10, 13]. The implementation of alias method requires $O(m)$ storage space to store entries of supported tables. The time complexity in building up these tables is proportional to m i.e., $O(m)$.

The selection of the next reaction firing is done through a loop from line 9 - 21. The loop repeats until the flag $accepted$ is set to **true**. In each iteration three random numbers r_1, r_2, and $r_3 \sim U(0, 1)$ are generated, respectively. The first two numbers is used to decide which reaction occurring, while the last random number r_3 is accumulated up to calculate the reaction firing time (by line 22).

The next reaction firing selection is implemented as following. First, we randomly look up a candidate reaction R_j by the random number r_1. More precisely, we retrieve candidate reaction index stored at position $p = \lfloor m \cdot r_1 \rfloor$. The first value in two-point distribution in cut-off table F is loaded. It is then compared with $(m \cdot r_1 - p)$ to select the candidate reaction. We return the candidate reaction index $j = p$ if $((m \cdot r_1 - p) < F[p])$. Otherwise, the candidate reaction index is set $j = L[k]$. Second, we decide whether to accept this candidate reaction firing

Algorithm 1 RSSA procedure

while $t < T_{max}$ **do**
  define the fluctuation interval $[\underline{X}, \overline{X}]$ of current state X
  compute the upper-bound propensity $\overline{a_j}$ and lower-bound propensity $\underline{a_j}$ for each reaction R_j
  compute the total upper-bound sum $\overline{a_0}$
  build cut-off table F and alias table L for alias method with probability $\overline{a_j}/\overline{a_0}$ for $j = 1 \ldots m$
  repeat
    set $u = 1$
    set $accepted =$ **false**
    repeat
      generate three random numbers r_1, r_2, r_3 from uniform distribution $U(0, 1)$
      lookup a candidate reaction R_j by r_1 using alias method
      if $r_2 \leq (a_j/\overline{a_j})$ **then**
        $accepted =$ **true**
      else
        evaluate a_j with current state X
        if $r_2 \leq (a_j/\overline{a_j})$ **then**
          $accepted =$ **true**
        end if
      end if
      set $u = u \cdot r_3$
    until $accepted$
    set transition time $\tau = (-1/\overline{a_0}) \ln(u)$
    update time $t = t + \tau$
    update state $X = X + v_j$
    store/handle data
  until $X(t) \notin [\underline{X}, \overline{X}]$
end while

or to reject it. At line 12, we first compare $r_2 < a_j/\overline{a_j}$. If this inequality is satisfied, we immediately accept the candidate reaction R_j firing without evaluating its propensity. We only compute the actual propensity in case this condition is invalid. If this is the case, we evaluate the reaction propensity a_j (line 15). Then, if $r_2 < a_j/\overline{a_j}$ reaction R_j is accepted. We then move to calculate its firing time. We reject this candidate reaction otherwise. This selection step will repeat until having a reaction accepted.

Upon a reaction is selected to fire, its firing time τ is then computed by line 22 i.e., $\tau = (-1/\overline{a_0})\ln(u)$, in which variable u is defined at line 7. It is, in fact, the implementation of reaction firing time discussed in the previous section. The generating of this firing time uses u to store the accumulated multiplication of random number r_3 at line 20.

From line 23 - 25, we finish a simulation step. The system moves to new state $x+v_j$ caused by reaction R_j firing. We set simulation clock to new time $t+\tau$. The current simulation data could be store to external storage for further processing.

At line 26 we check the system state constraint to ensure that the system state is confined in its fluctuation interval i.e., $X \in [\underline{X}, \overline{X}]$. The reaction selection is kept working without updating the propensity of affected reactions while this constraint is satisfied. This takes advantage when each simulation step requires to update a huge number of affected reactions. However, in the other case we have to define a new fluctuation interval. The corresponding upper-bound and lower-bound propensity of reactions as well as the supported tables for the alias method also have to be recomputed.

1.4 Proof of correctness

We now show the correctness of the RSSA algorithm. The correctness, in this sense, means RSSA selects the next reaction firing R_j with the same probability as SSA i.e., a reaction R_j is selected with corresponding probability a_j/a_0. This result is stated in Proposition 1. In other words, RSSA produces the same stochastic behavior as SSA.

Proposition 1. *RSSA is exactly choosing a reaction R_j to fire with probability a_j/a_0. In addition, its firing time is exponential distribution with rate a_0.*

Proof. At a specific time t with current system state $X \in (\underline{X}, \overline{X})$, let $Pr(R_j)$ be the probability a candidate reaction R_j is selected and accepted to fire. $Pr(R_j)$, by the chain rule, is the multiplication of two probabilities: the probability of R_j is selected as a candidate, and the probability it is accepted. Hence, it given by:

$$\begin{aligned}\Pr(R_j) &= \Big(\frac{\overline{a_j}}{\overline{a_0}}\Big) \cdot \Big(\frac{a_j}{\overline{a_j}}\Big) \\ &= \frac{a_j}{\overline{a_0}}\end{aligned} \tag{2}$$

Now, let $Pr(R)$ be the probability an arbitrary reaction which is selected and accepted with current system state. We have $Pr(R)$

$$\begin{aligned}\Pr(R) &= \frac{\sum_{j=1}^{m} a_j}{\overline{a_0}} \\ &= \frac{a_0}{\overline{a_0}}\end{aligned} \tag{3}$$

Thus, using the conditional probability, we can derive the probability reaction R_j is selected and accepted given an arbitrary candidate reaction R is selected and accepted. That is:

$$\begin{aligned}\Pr(R_j|R) &= \Big(\frac{a_j}{\overline{a_0}}\Big) / \Big(\frac{a_0}{\overline{a_0}}\Big) \\ &= \frac{a_j}{a_0}\end{aligned} \tag{4}$$

For the second statement, let f_τ be the PDF of the firing time of the accepted reaction R_j. We will prove that it has the exponential distribution with rate a_0 i.e., $f_\tau(x) = a_0 \cdot e^{-a_0 x}$. In following let suppose the number of trials before reaction R_j accepted is denoted by k. We have:

$$
\begin{aligned}
f_\tau(x) &= \frac{\partial}{\partial x}\,\mathbb{P}(\tau \le x) \\
&= \frac{\partial}{\partial x}\,\textstyle\sum_{k_0=1}^{\infty}\mathbb{P}(\tau \le x \mid k = k_0)\cdot\mathbb{P}(k = k_0) \\
&= \frac{\partial}{\partial x}\,\textstyle\sum_{k_0=1}^{\infty} F_{Erlang(k_0,\overline{a_0})}(x)\cdot\frac{a_0}{\overline{a_0}}\cdot(1-\frac{a_0}{\overline{a_0}})^{k_0-1} \\
&= \textstyle\sum_{k_0=1}^{\infty}\frac{\partial}{\partial x}\,F_{Erlang(k_0,\overline{a_0})}(x)\cdot\frac{a_0}{\overline{a_0}}\cdot(1-\frac{a_0}{\overline{a_0}})^{k_0-1} \\
&= \textstyle\sum_{k_0=1}^{\infty} f_{Erlang(k_0,\overline{a_0})}(x)\cdot\frac{a_0}{\overline{a_0}}\cdot(1-\frac{a_0}{\overline{a_0}})^{k_0-1} \\
&= \textstyle\sum_{k_0=1}^{\infty}\frac{\overline{a_0}^{k_0}\cdot x^{k_0-1}\cdot e^{-\overline{a_0}x}}{(k_0-1)!}\cdot\frac{a_0}{\overline{a_0}}\cdot(\frac{\overline{a_0}-a_0}{\overline{a_0}})^{k_0-1} \\
&= a_0\cdot e^{-\overline{a_0}x}\cdot\textstyle\sum_{k_0=1}^{\infty}\frac{(\overline{a_0}-a_0)^{k_0-1}\cdot x^{k_0-1}}{(k_0-1)!} \\
&= a_0\cdot e^{-\overline{a_0}x}\cdot e^{x\cdot(\overline{a_0}-a_0)} \\
&= a_0\cdot e^{-a_0x} = f_{Exp(a_0)}(x)
\end{aligned}
$$

□

A direct conclusion from the above proof is the acceptance probability of the acceptance-rejection step in selecting a reaction firing of RSSA is bound. In fact, let $Pr(acceptance)$ be the acceptance probability. We have that

$$\underline{a_0}/\overline{a_0} \le Pr(acceptance) = a_0/\overline{a_0} \le 1 \tag{5}$$

2 RSSA experiments

In this section we experiment the performance of RSSA with three biochemical reaction models: 1) Gene expression model, 2) Fully connected reaction model and 3) Multiscaled reaction model. The Table 1 summarizes the properties of simulated models. The first model considers different types of chemical reaction

Table 1: Summary of models

Model	Species	Reactions
Gene expression model	5	8
Fully connected reaction model	N	N(N-1)
Multiscaled reaction model	N + M	N(N-1) + M

kinetics (mass-action kinetics and Hill kinetics) and their effects to the simulation. The last two models consider the highly coupled of reaction network where the update propensity of affected reaction requires a very expensive computational resource. In these models, experiments have shown that the update of affected reactions, rather than the search of reaction firing, becomes the most contribution part to the total simulation runtime.

Three algorithms are tested including: DM, NRM and RSSA. All these simulation algorithms are implemented in Java and run on Intel i5-540M processor. The simulation was done after $2 \cdot 10^6$ steps. The simulation data are recorded after 10^5 steps. The experimental result excludes all initialization which is not a part of simulation loop.

The fluctuation interval used by RSSA is controlled by parameter δ. That is the system state X is confined to the interval $[X(1-\delta), X(1+\delta)]$. We run experiments by tuning the value of δ.

2.1 Gene expression model

The gene expression model is a collection of reactions which plays a key role in understanding of gene regulation mechanisms and functionalities. The result of gene expression is a collection of proteins encoded by the corresponding genes. It composes of two main consecutive processes: transcription and translation. The transcription initiates when an enzyme called RNA polymerase (RNAP) bind to gene promoter. In the transcription process, the gene is copied to intermediate form called mRNA. The mRNA will then bind to ribosomes to translate into the protein in the translation process.

The 8 reactions shown in Table 2 depict a typical gene expression model. In this table, protein P is encoded by its gene G. The intermediate product of transcription is denoted by RNA. The transcription was modelled by reaction R_1 where gene G transcribes to RNA. The RNA, after that, translates to protein P

Table 2: Gene expression model

R_1: $G \rightarrow G + RNA$	$k_1 = 0.09$
R_2: $RNA \rightarrow RNA + P$	$k_2 = 0.05$
R_3: $RNA \rightarrow \_$	$k_3 = 0.001$
R_4: $P \rightarrow \_$	$k_4 = 0.0009$
R_5: $P + P \rightarrow P_2$	$k_5 = 0.00001$
R_6: $P_2 \rightarrow P + P$	$k_6 = 0.0005$
R_7: $P_2 + G \rightarrow P_2G$	$k_7 = 0.005$
R_8: $P_2G \rightarrow P_2 + G$	$k_8 = 0.9$

in reaction R_2, will degrade by reaction R_3.

The proteins ussually interact to form a dimer P_2 rather than existing in the isolation form. Reaction R_5 and R_6, respectively, model the association and dissociation of dimer P_2. The dimer could bind to gene G to enhance the activation of the gene. Thus this is modelled by reaction R_7, R_8.

The Fig. 3 compares the runtime of simulation algorithms on the gene expression model. In this experiment the mass-action stochastic kinetics is applied. That is the propensity is a simple function which is the multiplication of rate constant with possible combinations of reactants involving in the reaction.

As shown in the Fig. 3, the runtime of NRM and DM is nearly the same, although DM is slightly faster than NRM. NRM exploited quite complex data structures for fast selecting a reaction firing, but it requires an expensive cost for heap update. In this experiment the update of NRM contributes 77% to the total simulation time, while in DM this is also up to 62%. In contrast, by RSSA we have reduced the update to only 2% in case $\delta = 20\%$. The overall result is the RSSA performance is roughly 20% faster than DM, NRM.

Table 3 compares the simulation runtime, percentage of update time in the simulation runtime and acceptance probability of RSSA by tuning the value of δ. It is clear that a narrow fluctuation interval (small value of δ) would yield a high acceptance probability but also the high percentage of update. The more frequently update the propensity does the slower simulation performance is. Thus, if δ is assigned extremely small value ($\delta = 1\%$), the RSSA performance is so poor even comparing with DM, NRM; however, if we slowly increase the value of parameter δ the simulation becomes faster. In our case the best performance is around

Figure 3: Compare simulation algorithms with mass-action kinetics

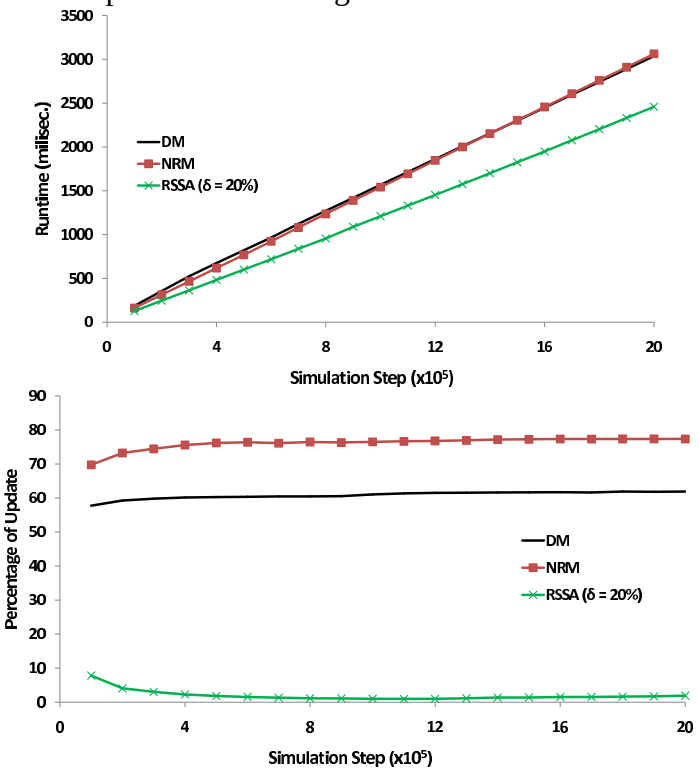


Table 3: Runtime of RSSA with different values of δ using mass-action kinetics

	Simulation time (milisec.)	Update (%)	Acceptance (%)
$\delta = 1\%$	4748	54.34	98.62
$\delta = 5\%$	2592	17.63	93.10
$\delta = 10\%$	2367	6.46	86.77
$\delta = 15\%$	2369	2.91	81.06
$\delta = 20\%$	2459	1.91	75.94
$\delta = 25\%$	2522	1.31	71.02
$\delta = 30\%$	2586	0.77	66.68
$\delta = 50\%$	2954	0.14	52.50
$\delta = 70\%$	3350	0.06	41.78

$\delta = 10\%$. But, we still increase the value of δ the performance is slowing down. This is beacause the candidate reaction is rejected most of time. The selection of reaction firing therefore repeats many times. And the computation of the search mostly contributes to the simulation time rather than update. However, since the marginal speed of alias search is very fast, the performance of RSSA with large δ ($\delta = 50\%, 70\%$) still produces a competitive performance as comparing to DM, NRM.

In our second experiment, we consider the effects of time-consuming propensity function to the update and the total simulation runtime. We modified the propensity function to use the Hill kinetics. It was first used to model the nonlinear effects of aggregation of the haemoglobin molecules with oxygen in the solution [8]. Hill equation recently has extensive applications in pharmacology to model the nonlinear relationship in drug-dose response on the target (see [7] for detailed discussions). In biology, Hill kinetics has been using to model the mechanism of enzymatic reactions. The Michaelis-Menten law, a well-known model of enzyme kinetics, is a special type of Hill kinetics. Hill kinetics is commonly used to describe the cooperativity of a ligand in binding to the enzyme. The binding of a ligand to a enzyme is often enhanced if there are already ligands binding to this enzyme.

In modelling of gene expression Hill kinetics is applied to describe the acti-

vation controlled in the gene regulation process. For example, in [9], it was used to model the switch-like behavior in the gene expression by protein activation. In our experiment, the propensity function is used with Hill equation which has a general form:

$$g(x) = \frac{x^n}{K^n + x^n} \tag{6}$$

where K is threshold constant and n is the steepness parameter (also called Hill coefficient), which is usually non-integer.

The the simulation runtime of algorithms is presented in Fig. 4 for the gene expression model with Hill kinetics. Due to expensive computational cost in evaluating of this complex propensity function the performance of DM, NRM is nearly $3x$ times slower than the mass-action kinetics propensity. In which the update contributes to the total simulation runtime up to 87% in DM and 91% in NRM, respectively. While the update of affected reactions is only kept in roughly 5% of the simulation runtime. Thus, the performance of RSSA is nearly 60% faster than DM.

We also compare different values of δ used by RSSA on the gene expression model with Hill kinetics. This is presented in Table 4. The same effect as seen in the previous experiment is repeated in this experiment. Except for the extremely small value of δ, all other cases yields better performance as comparing with DM, NRM. In this experiment, the best performance is achieved with δ is chosen around 15%.

2.2 Fully connected reaction model

The fully connected reaction network is an artificial model we used to benchmark the performance of RSSA. It consists of N chemical species S_i which reversibly converts into other species S_j at a reaction rate of k_i. The general form of reaction in this model is:

$$R_i : S_i \xrightarrow{k_i} S_j, i \neq j = 1 \ldots N$$

In our experiment the initial population of each species is set to 100. The propensity function is applied in the ideal kinetics (i.e., mass-action kinetics). The performance of algorithms is presented in Table 5. We consider the performance of algorithms by increasing the number of species N. In RSSA, three different values of $delta$ was tested is $10\%, 20\%$ and 30% respectively.

Figure 4: Compare simulation algorithms with Hill kinetics

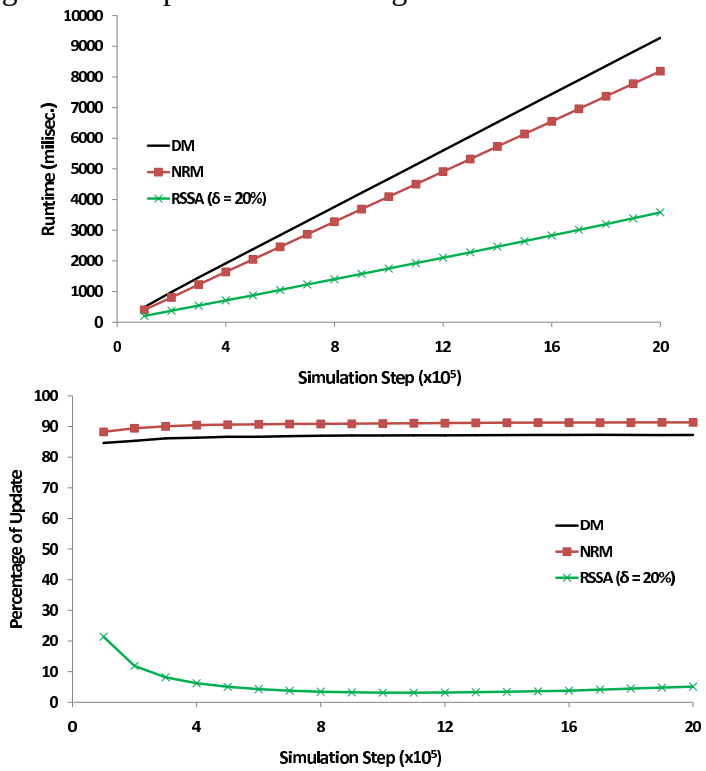


Table 4: Runtime of RSSA with different values of δ using Hill kinetics

	Simulation Time (milisec.)	Update Time (%)	Acceptance (%)
$\delta = 1\%$	12470	82.61	98.62
$\delta = 5\%$	4245	43.36	93.08
$\delta = 10\%$	3370	18.99	86.79
$\delta = 15\%$	3366	9.12	81.06
$\delta = 20\%$	3581	5.11	75.84
$\delta = 25\%$	3825	3.11	71.10
$\delta = 30\%$	4141	1.79	66.70
$\delta = 50\%$	5423	0.26	52.58
$\delta = 70\%$	7030	0.03	40.48

The numbers of affected reactions is linearly increasing with the number of species N. In fact, each time a reaction fires there are total $N-1$ affected reactions having to update their propensities. Since the propensity updates is asymptotically increasing with N the update time is thus increasing if we increase N. It will largely contributes to the simulation runtime. For example, the percentage of the update time of NRM and DM in case $N = 100$ is up to 99% and 93% of the total simulation runtime, respectively. In result, the performance of these algorithms is very slow. Sine RSSA effectively controls the update time, its simulation runtime is significantly reduced. For example, with $delta = 20\%$ RSSA is rougly 10 times faster than the DM, NRM.

From the experiments we also see the effect of the high coupling of reactions to the choosing of δ. For small N, choosing δ around 20% seems to achieve a better performance for RSSA. The search time in these models dominates the simulation performance while the update is less significant. Therefore, increasing δ would not yield a better performance. If we increase N, the update is increasing and largely affects the simulation performance. Thus, the larger fluctuation interval would yield better performance. For example, the performance of RSSA with $\delta = 30\%$ is 2 times faster than $\delta = 20\%$ in the model with $N = 100$.

Table 5: Performance of algorithms on fully connected model

N	Algorithm	Simulation Time (milisec.)	Update Percentage (%)	Acceptance Prob. (%)
5	DM	4112	77.35	
	NRM	4721	85.64	
	RSSA ($\delta = 10\%$)	1734	9.80	91.27
	RSSA ($\delta = 20\%$)	1685	2.14	83.59
	RSSA ($\delta = 30\%$)	1738	0.52	77.22
10	DM	8369	86.65	
	NRM	9481	92.55	
	RSSA ($\delta = 10\%$)	2189	25.67	91.27
	RSSA ($\delta = 20\%$)	1832	7.59	83.64
	RSSA ($\delta = 30\%$)	1841	2.77	77.16
50	DM	75352	90.70	
	NRM	76997	98.06	
	RSSA ($\delta = 10\%$)	19357	88.94	91.26
	RSSA ($\delta = 20\%$)	6487	65.65	83.63
	RSSA ($\delta = 30\%$)	4004	41.93	77.21
100	DM	359439	92.79	
	NRM	371058	99.10	
	RSSA ($\delta = 10\%$)	135644	96.94	91.26
	RSSA ($\delta = 20\%$)	36839	89.03	83.61
	RSSA ($\delta = 30\%$)	18103	77.38	77.20

2.3 Multiscaled reaction model

The multiscaled reaction model consists of N chemical species A_i and M species B_i. This model is separated into fast reactions, involving only fast species A_i, and slow reactions. Slow reaction contains not only slow species B_j but also involving fast species A_i. To form the slow reaction a fast species is randomly selected in the collection of N fast species.

$$\text{Fast reaction } R_i : A_i \xrightarrow{k_i} A_j$$
$$\text{Slow reaction } R_j : A_i + B_j \xrightarrow{k_j} B_k$$

In this model the reaction rate of fast reaction is chosen many times faster than the slow reactions ($k_i \gg k_j$). The initial population of fast species is set to 1000 and slow species is 100. The propensity function is in the simple form of the mass-action kinetics. In our experiment we fix the number of fast species is $N = 5$, and adjust the number slow species M. The number of slow reactions is adjusted from 25 to 1000. Table 6 compares the simulation runtime of algorithms applied to multiscaled reaction model.

Since a slow reaction is formed by combining of a slow species and a randomly selected fast species, we have on average $M/N + 1$ affected reactions which must update their propensities each time a fast reaction firing. If we fix $N = 5$, the update is linearly increasing with the number of slow species M. Thus, the update time will increase and dominate the simulation time as increasing M. From the result Table 6 we have that it contributes roughly 90% for NRM even with $M = 25$ and this value is over 97% when $M \geq 100$. The update time of DM is also roughtly 95% of the simulation time. In result, the performane of these algorithm is very slow. RSSA achieved better performance because it has an effective control over the time-consuming update. The result is RSSA performance with $\delta = 20\%$ is 195 times faster than DM in case $M = 1000$.

This result table also consolidates our guideline of choosing of δ. While the search dominates, choosing small value of $delta$ will yield better performance. In contrast, if the update dominates, the larger value of δ is better. For example, the performance of RSSA with $\delta = 10\%$ is better when M is small, but in case M is large $\delta = 20\%$ has better performance.

Table 6: Performance of algorithms on multiscaled reaction model

M	Algorithm	Simulation Time (milisec.)	Update Percentage (%)	Acceptance Prob. (%)
25	DM	9805	89.25	
	NRM	11061	93.68	
	RSSA ($\delta = 10\%$)	1582	0.25	90.94
	RSSA ($\delta = 20\%$)	1694	0	83.10
	RSSA ($\delta = 30\%$)	1752	0	76.51
50	DM	15486	92.52	
	NRM	17252	96.02	
	RSSA ($\delta = 10\%$)	1631	0.61	90.93
	RSSA ($\delta = 20\%$)	1698	0	82.97
	RSSA ($\delta = 30\%$)	1793	0	76.59
100	DM	28166	94.87	
	NRM	30917	97.35	
	RSSA ($\delta = 10\%$)	1742	1.44	90.82
	RSSA ($\delta = 20\%$)	1835	0	82.39
	RSSA ($\delta = 30\%$)	1889	0	76.16
500	DM	171109	97.89	
	NRM	178607	99.29	
	RSSA ($\delta = 10\%$)	1752	4.51	90.81
	RSSA ($\delta = 20\%$)	1806	0	79.32
	RSSA ($\delta = 30\%$)	1895	0	73.13
1000	DM	367541	98.28	
	NRM	397735	99.62	
	RSSA ($\delta = 10\%$)	2024	9.19	90.64
	RSSA ($\delta = 20\%$)	1880	0	80.28
	RSSA ($\delta = 30\%$)	2018	0	68.75

References

[1] James Blue, Isabel Beichl, and Francis Sullivan. Faster monte carlo simulations. *Phys. Rev. E*, 51(2):867–868, 1995.

[2] 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.

[3] E. J. Crampina, S. Schnella, and P. E. McSharryd. Mathematical and computational techniques to deduce complex biochemical reaction mechanisms. *Progress in Biophysics and Molecular Biology*, 86(1):77112, 2004.

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

[5] Daniel T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.*, 81(25):2340–2361, 1977.

[6] Daniel T. Gillespie. A rigorous derivation of the chemical master equation. *Physica A*, 188(1-3):404–425, 2007.

[7] S. Goutelle, M. Maurin, F. Rougier, X. Barbaut, L. Bourguignon, M. Ducher, and P. Maire. The Hill equation: a review of its capabilities in pharmacological modelling. *Fundamental & Clinical Pharmacology*, 22, 2008.

[8] A. V. Hill. The possible effects of the aggregation of the molecules of haemoglobin on its dissociation curves. *Journal of Physiology*, 40:iv–vii, 1910.

[9] Haseong Kim and Erol Gelenbe. Stochastic gene expression modeling with hill function for switch-like gene responses. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 9(4):973–979, 2012.

[10] Richard A. Kronmal and Arthur V. Peterson. On the alias method for generating random variables from a discrete distribution. *The American Statistician*, 33(4):214–218, 1979.

[11] James McCollum and *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.

[12] Vo Hong Thanh and Roberto Zunino. Tree-based search for stochastic simulation algorithm. In *Proc. of ACM-SAC*, 2012.

[13] Michael D. Vose. A linear algorithm for generating random numbers with a given distribution. *IEEE Transactions on Software Engineering*, 17(9):972–974, 1991.

[14] Alastair J. Walker. An efficient method for generating discrete random variables with general distributions. *ACM Transactions on Mathematical Software (TOMS)*, 3(3):253–256, 1977.