In press, Journal of
Chemical Physics, 2009

An Exact Accelerated Stochastic Simulation Algorithm

Eric Mjolsness[1,2], David Orendorff[2], Philippe Chatelain[3], and Petros Koumoutsakos[3]

[1] Institute for Genomics and Bioinformatics, and

[2] Department of Computer Science, University of California, Irvine, USA

[3] Computational Science, ETH Zürich, CH-8092, Switzerland

UCI ICS TR # 08-09

November 2008

Abstract

An exact method for stochastic simulation of chemical reaction networks, which accelerates the Stochastic Simulation Algorithm (SSA), is proposed. The present "ER-leap" algorithm is derived from analytic upper and lower bounds on the multi-reaction probabilities sampled by SSA, together with rejection sampling and an adaptive multiplicity for reactions. The algorithm is tested on a number of well-quantified reaction networks and is found experimentally to be very accurate on test problems including a chaotic reaction network. At the same time ER-leap offers a substantial speed-up over SSA with a simulation time proportional to the 2/3 power of the number of reaction events in a Galton-Watson process.

1 Introduction

The Stochastic Simulation Algorithm (SSA) [1] is a widely used method for simulating the stochastic dynamics of chemical reaction networks. SSA executes every reaction event, and provides an accurate view of the system dynamics, albeit at a significant computational cost over the corresponding mass-action differential equations that approximate the mean numbers of each molecular species. A number of algorithms have been proposed for the acceleration of the SSA at the expense of its accuracy. The τ-*leaping* algorithm [2] and its recent variants [3-5] simulate leaps over several reaction events during a preselected time increment. Further developments include multiscale stochastic simulation algorithms such as "Nested Stochastic Simulation" [6], the multiscale methods [7] and [8], and the "slow-scale stochastic simulation" algorithm [9]. Another acceleration method [10] uses rejection sampling to achieve constant time scaling with the number of reaction channels; this differs from present work which uses rejection sampling to improve scaling with respect to the number of reaction events.

A related work is the *R-leaping* algorithm [11] which proposes the simulation of preselected numbers of reaction firings that occur over time intervals sampled from an Erlang distribution. An essential aspect of these approximate methods is the requirement that the changes to the reaction rate or "propensity" functions are small during each step.

We present a stochastic simulation algorithm which, similar to R-leap, accelerates SSA by executing multiple reactions per algorithmic step, but which samples the reactant trajectories from the same probability distribution as the SSA. This "Exact R-leap" or "*ER-leap*" algorithm is a modification of the R-leap algorithm which is both exact and capable of substantial speedup over SSA. The simplest versions of both τ-leap and R-leap have difficulties with the potential of producing negative numbers of reactants, which can be fixed by modifications such as Binomial tau-leap [3] and modified tau-leap [4]. Since ER-leap is exact, it intrinsically avoids this potential pitfall; stochastic moves to negative reactant states have zero probability and will be rejected. We

demonstrate by computational experiments that ER-leap can execute in time sublinear in the number of reaction events to be simulated, while remaining exact. The algorithm is based on the rejection sampling concept, using efficiently computable bounds on the SSA probability distribution.

The paper is organized as follows : In Section 2 we derive upper and lower bounds on the SSA reaction probabilities after multiple reactions, expressed using matrix notation for Markov processes, and use rejection sampling to derive the ER-leap algorithm. The algorithm itself is stated, analyzed for cost, and illustrated in Section 2.5 . In Section 3 we report on a series of numerical experiments designed to evaluate the accuracy and speedup of the ER-leap algorithm. In Section 4 we discuss the results, and conclude with an assessment of the method in the context of related works and an outline of directions for future work.

2 Theory

This section is organized as follows: Section 2.1, Section 2.2, and Section 2.3 introduce the required notations, reaction probabilities, and bounds on these probabilities, respectively. The ER-leap algorithm's key update equations are derived from these probability bounds in the calculations of Section 2.4 . The resulting algorithm is assembled from the key update equations, analyzed for cost, and illustrated in the case of a simple reaction network in Section 2.5 .

We consider a set of reactions, indexed by r, among chemical species C_a, indexed by a:

$$\{m_a^r\, C_a\} \longrightarrow \{m'^r_a\, C_a\} \quad \textbf{with} \quad \text{reaction rate } \rho_r \tag{1}$$

Here $\boldsymbol{m}^r = [m_a^r]$ and $\boldsymbol{m'}^r = [m'^r_a]$ are the input and output stoichiometries of the reaction r. In the following we derive an expression for the probability of states after a number of such reaction events.

2.1 Notations

We introduce the following notations. The definition of a version of the indicator function **1** from Boolean values to integers is:

$$\mathbf{1}\,(P) \equiv \begin{cases} 1 & \text{if predicate } P \text{ is true} \\ 0 & \text{otherwise} \end{cases} .$$

The Kronecker delta function $\delta(a, b)$ or $\delta(a - b)$ is:

$$\delta(a-b) = \delta_{a\,b} = \mathbf{1}\,(a = b) = \begin{cases} 1 & \text{if } a = b \\ 0 & \text{otherwise} \end{cases}$$

The function $V = \text{diag}(\boldsymbol{v})$ turns a d-dimensional vector $\boldsymbol{v}$ into a $d \times d$ square matrix V with components $V_{i\,j} = \delta_{i\,j}\, v_i$, i.e. zero everywhere except the diagonal which contains the components of $\boldsymbol{v}$. Given an ordered list of noncommuting matrices $V^{(k)}$ indexed by integers k, we defined the ordered product notation

$$\prod_{k=K_{\max}\searrow K_{\min}} V^{(k)} = V^{(K_{\max})} \cdot V^{(K_{\max}-1)} \cdot \ldots \cdot V^{(K_{\min}+1)} \cdot V^{(K_{\min})}$$

In addition to the standard set-builder notation $\{x \mid P(x)\}$ for defining the members of a set from a predicate P, we will build ordered sets or lists in a similar way using square brackets: $[x(i) \mid P(x(i), i) \,||\, i \in \mathcal{I}]$ imposes the image of a preexisting ordering on the index set $\mathcal{I}$ (such as the ordering of natural numbers if $\mathcal{I} \subseteq \mathbb{N}$)

onto any elements $x(i)$ selected for inclusion by the optional predicate P, and thus denotes a set together with a total ordering. For example, the B-tuple $[n_b \,||\, b \in \{1,\ ...\, B\}]$ denotes the components of a vector **n**.

2.2 Markov chain and multi-reaction probabilities

We denote states of the chemical reaction network by $I,\ J,\ K,$ time by t, and algorithm step number by k. Let n_a be the number of reactant molecules of type a present in a given state I at time t, so that I corresponds to the vector or ordered list of nonnegative integers $\boldsymbol{n} = [n_b \,||\, b \in \{1,\ ...\, B\}]$. Likewise if we are discussing are several such states that are present at different times t' and t'', we may denote them by $\boldsymbol{n}'$ and $\boldsymbol{n}''$ or correspondingly by J and K. The time interval between successive reactions is denoted by τ.

We wish to track the time evolution of the probabilities $\Pr(I, t)$, for all possible system states I by employing the governing Master (or Chapman-Kolmogorov) equation [12]. which we shall use here. We define $\Pr(I, t \,|\, J, k)$ as the "just-reacted state probability": the probability of being in state I at time t immediately after the k-th reaction event, given that the state is J at time zero. The Chapman-Kolmogorov equation [12] for such just-reacted state probabilities follows from taking k to be a discrete time coordinate, and can be written:

$$\Pr(I, t \,|\, J, k) \approx \sum_K \int_0^t d\tau \Pr(I, \tau \,|\, K, 1) \Pr(K, t - \tau \,|\, J, k - 1) \tag{2}$$

A key quantity in this equation is the "kernal" $\Pr(I, \tau \,|\, K, 1)$: the probability that if $k = 1$ reaction event has just occurred, and if the previous state was K, then a time τ has elapsed since the last reaction event and the new state is I. This kernal also provides the linear weights that advance the quantity $\Pr(K, t - \tau \,|\, J, k - 1)$, whis is the probability distribution over states K just after $k - 1$ reactions, to produce the probability distribution over states I after k reactions, $\Pr(I, t \,|\, J, k)$. So we can rename this kernal the conditional distribution

$$\mathcal{W}(I, t' \,|\, J, t) = \Pr(I, t' - t \,|\, J, 1)$$

using notation similar to that of [13]. This $\mathcal{W}$ is analogous to a matrix with two indices, each of which is a pair consisting of a discrete-valued systems state (such as I or J) and a continuous-valued time (such as t' or t).

Under the SSA algorithm $\mathcal{W}$ must factor into an update from time t to t' and then from state J to state I:

$$\mathcal{W}(I, t' \,|\, J, t) \approx \hat{W}_{I,J} \exp(-(t' - t)\, D_{J\,J})\, \mathbf{1}\,(t' \geqslant t)\,. \tag{3}$$

with

$$D = \mathrm{diag}\left(\boldsymbol{h} \cdot \hat{W}\right) \tag{4}$$

where $\boldsymbol{h}$ is the vector whose components are all 1, and "diag" turns a vector into the corresponding diagonal matrix. This result is derived in more detail in [14] and [13]. The state space transition matrix $\hat{W}$ contains the summed probability rates or "propensities" for all reactions that could move the system from state J to state I. The expenential term governs the distribution of waiting times between reaction events. The I'th component of the vector $\boldsymbol{h} \cdot \hat{W}$, which is defined as $D_{I\,I}$, is the the total probability per unit time for the system to leave state I. (In many papers the summed reaction rate $D_{I\,I}$ is denoted as $a_0\,(\boldsymbol{n})$ instead.)

Continuing with the matrix analogy for $\mathcal{W}$, and assuming that $t < 0 \wedge k \geqslant 0 \Rightarrow \Pr(I, t \,|\, J, k) = 0$,

$$\Pr(I, t \,|\, J, k) \approx \sum_K \int_{-\infty}^{\infty} d\tau\, \mathcal{W}(I, t \,|\, K, t - \tau) \Pr(K, t - \tau \,|\, J, k - 1) \tag{5}$$

Using vector notation $\Pr(. \mid J, k)$ for the (I, t) parameters, we may write

$$\Pr(. \mid J, k) \equiv \mathcal{W} \circ \Pr(. \mid J, k-1) \tag{6}$$

where the matrix-vector inner product $\circ$ is both a sum over states and an integral over all times t, as in Equation 5, and where

$$\mathcal{W} = \hat{W} \exp(-\Delta t\, D)\, \mathbf{1}(\Delta t \geqslant 0)\,. \tag{7}$$

Equation 7 expresses the Markov chain for the change of both chemical state and total time, after one reaction event. The matrix $\hat{W}$ contains probability rates or "propensities", the much larger matrix $\mathcal{W}$ contains only normalized probability densities for the combination of a discrete state change and a continuous time change Δt.

From Equation 6 and Equation 7, after k reaction events,

$$\Pr(. \mid J, k) = \mathcal{W}^k \circ \Pr(. \mid J, 0) = \left[\hat{W} \exp(-\Delta t\, D)\, \mathbf{1}(\Delta t \geqslant 0)\right]^k \circ \Pr(. \mid J, 0) \tag{8}$$

This expression is in accord with, for example, Theorem 10.1 of [14]. Furthermore it expresses concisely the SSA algorithm for sampling from $\Pr(I, t \mid J, k)$ by alternately updating the time t of most recent reaction, and the molecular state I.

The aim of the SSA algorithm is to sample from this distribution $\Pr(I, t \mid J, k)$. Equation 6 may be taken as a concise statement of the SSA algorithm update: it is a product of two conditional distributions, one $(\hat{W} D^{-1})$ for molecular state I given state J, and another one which samples time t' given time t and state J according to conditional distribution $D \exp(-\Delta t\, D)\, \mathbf{1}(\Delta t \geqslant 0)$, evaluated at state J. These two sampling steps are alternated and iterated k times as in Equation 6.

To derive D and $\mathcal{W}$, and therefore (by Equation 7) the detailed SSA simulation process, we need only define the matrix $\hat{W}$ of probability rates for a chemical reaction network. For the reaction network of Equation 1, defining the net stoichiometry

$$\Delta m_a^r = {m'}_a^r - m_a^r\,,$$

the usual mass-action assumption for stochastic reactions corresponds to

$$\hat{W}(\boldsymbol{n}' \mid \boldsymbol{n}) = \sum_r \hat{W}^{(r)}(\boldsymbol{n}' \mid \boldsymbol{n})$$

where the probability rate matrix $\hat{W}^{(r)}$ for reaction r has elements given by a product of factors for all the input reactants (all a for which $m_a^r \neq 0$), times a product of Kronecker delta functions that enforce the net stoichiometries on the system state:

$$\hat{W}^{(r)}_{\boldsymbol{n}', \boldsymbol{n}} = \rho_r \left(\prod_{\{a \mid m_a^r \neq 0\}} \frac{n_a\,!}{(n_a - m_a^r)\,!} \right) \left[\prod_{\{a \mid \Delta m_a^r \neq 0\}} \delta(n'_a - n_a - \Delta m_a^r) \right];$$

the corresponding diagonal matrix $D^{(r)}$ is

$$D^{(r)}\left(\boldsymbol{n}' \mid \boldsymbol{n}\right) = \rho_r \left(\prod_{\{a \mid m_a^r \neq 0\}} \frac{n_a\,!}{(n_a - m_a^r)\,!} \right) \left[\prod_{\{a \mid \Delta m_a^r \neq 0\}} \delta(n'_a - n_a) \right],$$

with $D = \sum_r D^{(r)}$. (The elements of $\hat{W}^{(r)}$ of are essentially reaction "propensity functions", with a constant coefficient $\prod_a (1/(m_a^r)!)$ that can be absorbed into the definition of ρ_r to maintain notational consistency with the law of mass action, as discussed in Section 3.4 of [15] which also uses notation similar to that used here.) If we

define $W = \hat{W} - D$, SSA dynamics simulates trajectories drawn from the solution to the Master Equation, $d\,p/d\,t = W \cdot p$ [12].

2.3 Upper and Lower Bounds

In order to derive a new simulation algorithm, equivalent to SSA, using rejection sampling [16], we now seek simplified upper and lower bounds on the probability rate $\hat{W}^{(r)}_{I,J} \exp(-\Delta\, t\, D_{J\,J})$ (from Equation 7) for a single reaction event. However, we will assume that the reaction event to be bounded occurs within a run of L events in the SSA algorithm, in order to execute L reactions at once in the manner of the R-leap algorithm[11]. As we will see, this essentially comes down to bounding each combinatorial factor $n_a\,!/(n_a - m^r_a)!$ with a constant bound, even though it may change throughout the run of L events.

For step number l within the run we must find a simplifying upper bound for the key expression

$$F^{(r)}_{\boldsymbol{n}} \equiv \prod_{\{a|m^r_a \neq 0\}} \left(\begin{cases} \frac{n_a\,!}{(n_a - m^r_a)!} & \text{if } n_a \geqslant m^r_a \\ 0 & \text{otherwise} \end{cases} \right)$$

that occurs in $\hat{W}$ and D, and also to find a simplifying lower bound for its contribution to D, in order to lower-bound both factors in $\mathcal{W}$ under Equation 3. The products $\rho_r\, F^{(r)}_{\boldsymbol{n}}$ are usually called "propensity functions" denoted $a_r\,(\boldsymbol{n})$ for all R reaction channels:

$$\begin{aligned} a_r\,(\boldsymbol{n}) &\equiv \rho_r\, F^{(r)}_{\boldsymbol{n}}\,, \\ a_0\,(\boldsymbol{n}) &\equiv \sum_{r=1}^{R} a_r\,(\boldsymbol{n}) \end{aligned} \tag{9}$$

possibly with a different normalization convention as a function of m^r_a if $m^r_a \neq 1$ as mentioned in the previous section. In this work it is more convenient to keep separate the structural terms $F^{(r)}_{\boldsymbol{n}}$ and the reaction rates ρ_r, rather than combining them as in Equation 9. Fortunately every $F^{(r)}_{\boldsymbol{n}}$ is monotonic in each n_a, so we may find upper and lower bounds on $F^{(r)}_{\boldsymbol{n}}$ by finding upper and lower bounds on each n_a.

A very simple, though not very tight, set of bounds is:

$$n_a + l \min_r \{\Delta\, m^r_a\} \leqslant n'_a \leqslant n_a + l \max_r \{\Delta\, m^r_a\} \tag{10}$$

The corresponding upper and lower bounds $\tilde{F}$ and $\underset{\sim}{F}$ on F for the $l+1$-st reaction event (after l reaction events have already occurred) within a run of L events is:

$$\underset{\sim}{F}^{(r)}_{\boldsymbol{n},\,l} \leqslant F^{(r)}_{\boldsymbol{n}'} \leqslant \tilde{F}^{(r)}_{\boldsymbol{n},\,l}$$

where

$$\begin{aligned} \underset{\sim}{F}^{(r)}_{\boldsymbol{n},\,l} &\equiv F^{(r)}_{[n_a + l \min_r \{\Delta\, m^r_a\}\,||1 \leqslant a \leqslant A]} \\ \tilde{F}^{(r)}_{\boldsymbol{n},\,l} &\equiv F^{(r)}_{[n_a + l \max_r \{\Delta\, m^r_a\}\,||1 \leqslant a \leqslant A]} \end{aligned} \tag{11}$$

The sparsity structure of $\hat{W}^{(r)}$ is given by $S^{(r)} \in \{0, 1\}$:

$$\hat{S}^{(r)}_{\boldsymbol{n}',\boldsymbol{n}} = \mathbf{1}\left(\hat{W}^{(r)}_{\boldsymbol{n}',\boldsymbol{n}} > 0\right) \in \{0, 1\}$$

$$= \left(\prod_{\{a|m^r_a \neq 0\}} \mathbf{1}\,(n_a \geqslant m^r_a)\right)\left[\prod_{\{a|\Delta m^r_a \neq 0\}} \delta(n'_a - n_a - \Delta m^r_a)\right]$$

$$\hat{S}_{I,J} = \mathbf{1}\left(\sum_r S^{(r)}_{I,J}\right) = \mathbf{1}\left(\hat{W}_{I,J} > 0\right)$$

We will *assume* that reactions have unique outcomes (or, redefine the states I so this becomes true):

$$\sum_I \hat{S}^{(r)}_{I,J} = 1\,. \tag{12}$$

Taking l consecutive steps of this chain results in another sparsity structure of "reachability":

$$R_{I|J\,l} \equiv (\hat{S}^{l})_{I,J} = \mathbf{1}\left((\hat{W}^{l})_{I,J} > 0\right) \equiv \begin{cases} 1 & \text{if } (\hat{W}^{l})_{I,J} > 0, \\ 0 & \text{otherwise} \end{cases}$$

We now start the reactions from state $K = \boldsymbol{n} = [n_a \,||\, a \in \{1,\ \ldots A\}]$. Since

$$\hat{W}^{(r)}_{\boldsymbol{n}',\boldsymbol{n}} = \rho_r\, F^{(r)}_{\boldsymbol{n}}\, \hat{S}^{(r)}_{\boldsymbol{n}',\boldsymbol{n}}\,,$$

we have the bounds

$$R_{J|K\,l} = 1 \quad \Rightarrow \quad \underset{\sim}{W}^{(r)}_{I,J|K\,l} \leqslant \hat{W}^{(r)}_{I,J} \leqslant \tilde{W}^{(r)}_{I,J|K\,l}$$

where

$$\underset{\sim}{W}^{(r)}_{I,J|K,\,l} \equiv \rho_r\, \underset{\sim}{F}^{(r)}_{K,l}\, \hat{S}^{(r)}_{I,J}$$
$$\tilde{W}^{(r)}_{I,J|K,\,l} \equiv \rho_r\, \tilde{F}^{(r)}_{K,l}\, \hat{S}^{(r)}_{I,J}\,.$$

These quantities bound $\hat{W}^{(r)}_{I,J}$, in the circumstance that l reaction events have occurred since the system was in state K.

We also need to bound $-D$ in Equation 3. To this end, note from Equation 4 that

$$D_{I\,J} = \delta_{I\,J} \sum_{I'} \sum_r \hat{W}^{(r)}_{I',J} = \delta_{I\,J}\, D_{I\,I}\,.$$

Then

$$R_{J|K\,l} = 1 \ \Rightarrow$$
$$-\tilde{D}_{K\,l} = -\sum_r \sum_{I'} \tilde{W}^{(r)}_{I',J|K\,l} \leqslant -D_{J\,J} \leqslant -\sum_r \sum_{I'} \underset{\sim}{W}^{(r)}_{I',J|K\,l} = -\underset{\sim}{D}_{K\,l}$$

where

$$\underset{\sim}{D}_{K\,l} \equiv \sum_r \rho_r\, \underset{\sim}{F}^{(r)}_{K,l}$$
$$\tilde{D}_{K\,l} \equiv \sum_r \rho_r\, \tilde{F}^{(r)}_{K,l} \tag{13}$$

Thus, assuming $R_{J|K\,l} = 1$ and $\Delta t \geqslant 0$, upper and lower bounds on the elements of the Markov process $\mathcal{W}$ given by Equation 3 are determined as follows:

$$\rho_r \, F^{(r)}_{K,l} \, \hat{S}^{(r)}_{I,J} \exp\left(-\Delta t \, \tilde{D}_{K\,l}\right) \leqslant \hat{W}^{(r)}_{I,J} \exp(-\Delta t \, D_{J\,J}) \leqslant \rho_r \, \tilde{F}^{(r)}_{K,\,l} \, \hat{S}^{(r)}_{I,J} \exp\left(-\Delta t \, \underline{D}_{K\,l}\right) . \tag{14}$$

These desired bounds on reaction probability rates $\hat{W}^{(r)}_{I,J} \exp(-\Delta t \, D_{J\,J})$ follow from the simple bounds of Equation 10 on n'_a as a function of n_a and l.

2.4 Exploitation of probability bounds

We now use the bounds of Equation 14 to derive the key update equations of the ER-Leap algorithm. The resulting ER-leap algorithm will be assembled from these equations and discussed in Section 2.5, followed by computational experiments in Section 3. In this Section we perform the required calculations to derive the key update equations..

2.4.1 Rejection sampling

Rejection sampling [16] allows one to exploit probability bounds in exact sampling, as follows: given a target distribution $P(x)$ and an algorithm for sampling from a related distribution $P'(x)$ and from the uniform distribution $U(u)$ on [0,1], and if

$$P(x) < M \, P'(x)$$

for some constant $M > 1$, then $P(x)$ satisfies

$$P(x) = P'(x) \, \frac{P(x)}{M \, P'(x)} + (1 - 1/M) \, P(x)$$

and therefore also

$$P(x) = \int P'(x') \, dx' \int U(u) \, du \left[\mathbf{1}\left(u < \frac{P(x')}{M \, P'(x')}\right) \cdot \delta(x - x') + \mathbf{1}\left(u \geqslant \frac{P(x')}{M \, P'(x')}\right) \cdot P(x) \right] \tag{15}$$

which constitutes a mixture distribution, that can be applied recursively as needed to sample from $P(x)$. Pseudocode for sampling $P(x)$ according to Equation 15 is as follows (where "//" introduces a comment):

while not accepted {

sample $P'(x)$ and $U(u)$; // $P'(x)$ only approximates $P(x)$

compute Accept(x) = $P(x)/(M \, P'(x))$; // acceptance probability

if $u <$ Accept(x) **then** accept x;

} // now $P(x)$ is sampled exactly

What is essential in applying this algorithm is to find a provable strict upper bound $\tilde{P}(x) = M \, P'(x)$ for $P(x)$ (where $M > 1$), which is not a probability distribution but which when normalized yields a probability distribution $P'(x)$ that is easier to sample than $P(x)$. We also want acceptance to be likely, for computational efficiency; for that reason M should be as close to 1 as possible, so that the bound on $P(x)$ is as tight as possible for a given computational cost.

But what if $P(x)$ is expensive to compute? Then Accept(x) will also be expensive to compute and rejection sampling may be prohibitively expensive, even for a good approximating $P'(x)$. A solution to this problem is possible if a cheap lower bound for $P(x)$ is available. Suppose there is a function $\underline{A}(x)$ such that

$$0 \leqslant \underset{\sim}{A}(x) \leqslant \text{Accept}(x) \equiv P(x)/(M\,P'(x)) < 1\,. \tag{16}$$

Then

$$\text{Accept}(x) = \underset{\sim}{A}(x)\cdot 1 + (1-\underset{\sim}{A}(x))\cdot Q(x), \text{ where}$$
$$Q(x) \equiv \left(\frac{\text{Accept}(x) - \underset{\sim}{A}(x)}{1-\underset{\sim}{A}(x)}\right),$$

and Accept(x) becomes a mixture of probabilities defined over the pair of actions (accept, reject). Then we have the following "accelerated rejection sampling algorithm", in pseudocode:

while not accepted {
    sample $P'(x)$ and $U(u)$; // cheap but approximate
    compute $\underset{\sim}{A}(x)$; // cheap
    if $u < \underset{\sim}{A}(x)$ **then** accept x;
    else {
        compute Accept(x) = $P(x)/M\,P'(x)$; // expensive
        compute $Q(x)$ = (Accept(x) − $\underset{\sim}{A}(x)$)/(1 − $\underset{\sim}{A}(x)$); // $\underset{\sim}{A}(x) < 1 \Rightarrow 1-\underset{\sim}{A}(x) \neq 0$
        sample $U(u)$;
        if $u < Q(x)$ **then** accept x;
        else reject x;
    }
}

Again, the bound $\underset{\sim}{A}(x) \leqslant \text{Accept}(x)$ should be as tight as possible for a given level of computational cost, to maximize the probability of early and therefore low-cost acceptance. A natural measure of the tightness of this bound is $\int \underset{\sim}{A}(x)\,dx \leqslant 1$, which should be as close to 1 as possible given cost considerations. However, even if $\underset{\sim}{A}(x) = 0$ for some values of x, the algorithm still samples the distribution $P(x)$ exactly.

We now seek M, $P'(x)$, and $\underset{\sim}{A}(x)$ for a run of L successive reaction events in the SSA algorithm.

2.4.2 Equivalent Markov process

In this section we will use algebraic manipulations to transform the formula for SSA (Equation 8) into an equivalent form (Equation 18) that represents an accelerated rejection sampling algorithm, as outlined in the previous section.

The first step in the algebraic derivation is to identify a probability distribution equivalent to L steps of the original SSA Markov process, which can itself be iterated to create a new, equivalent Markov process. The target distribution P is (from Equation 8)

$$\left[\hat{W}\exp(-\Delta t\, D)\right]^L \circ \Pr(.\mid K, 0)$$

From Equation 14,

$$\hat{W}_{I,J}\exp(-t_k\,D_{J\,J}) = \left(\sum_r \rho_r\,\hat{S}^{(r)}_{I,J}\left(\frac{F^{(r)}_I}{\tilde{F}^{(r)}_{K,\,l-1}}\right)\tilde{F}^{(r)}_{K,\,l-1}\right)\exp\left(-t_k\left(D_{J\,J} - \underset{\sim}{D}_{K\,l}\right)\right)\exp\left(-t_k\,\underset{\sim}{D}_{K\,l}\right)$$

Expand out the ordered matrix product for states J reachable from K after L steps:

$$R_{J|K\,L} = 1 \;\Rightarrow$$

$$\left[\prod_{k=L-1\searrow 0} \hat{W}\;\exp(-\tau_k\, D)\right]_{I_L,\,I_0} = \sum_{\{I_k|k=1\,..\,L-1\}} \left[\prod_{k=L-1\searrow 0} \hat{W}_{I_{k+1},I_k}\;\exp(-\tau_k\, D_{I_k,I_k})\right]$$

$$= \sum_{\{I_k|k=1\,..\,L-1\}} \sum_{\{r_k\}} \prod_{k=L-1\searrow 0} \left[\left(\rho_{r_k}\,\hat{S}^{(r_k)}_{I_{k+1},I_k}\left(\frac{F^{(r_k)}_{I_k}}{\tilde{F}^{(r_k)}_{I_0,\,L-1}}\right)\tilde{F}^{(r_k)}_{I_0,\,L-1}\right)\exp\left(-\tau_k\left(D_{I_k,I_k} - D_{I_0\,L-1}\right)\right)\exp\left(-\tau_k\, D_{I_0\,L-1}\right)\right]$$

$$= \sum_{\{r_k|k=1\,..\,L-1\}} \sum_{\{I_k\}} \left[\prod_{k=L-1\searrow 0} \hat{S}^{(r_k)}_{I_{k+1},I_k}\right]\left[\prod_{k=L-1\searrow 0} \rho_{r_k}\,\tilde{F}^{(r_k)}_{I_0,\,L-1}\right]$$

$$\times\left[\prod_{k=L-1\searrow 0}\left(\frac{F^{(r_k)}_{I_k}}{\tilde{F}^{(r_k)}_{I_0,\,L-1}}\right)\exp\left(-\tau_k\left(D_{I_k,I_k} - D_{I_0\,L-1}\right)\right)\right]\exp\left(-\left(\sum_k \tau_k\right) D_{I_0\,L-1}\right)$$

Now $\sum_I \hat{S}^{(r)}_{I,J} = 1$ allows a change of representation to eliminate the inner state sums:

$$I_k = I_k(r_{k-1},\, I_{k-1}) = I_k\,(\boldsymbol{r} = [r_0,\; ...\, r_l],\, I_0)$$

$$\left[\prod_{k=l-1\searrow 0} \hat{W}\;\exp(-\tau_k\, D)\right]_{I_l,\,I_0} = \sum_{\{r_k|k=1\,..\,L-1\}} \left[\prod_{k=l-1\searrow 0} \rho_{r_k}\,\tilde{F}^{(r_k)}_{I_0,\,L-1}\right]\exp\left(-\left(\sum_k \tau_k\right) D_{I_0\,L-1}\right)$$

$$\times\left[\prod_{k=L-1\searrow 0}\left(\left(\frac{F^{(r_k)}_{I_k\,(\boldsymbol{r},I_0)}}{\tilde{F}^{(r_k)}_{I_0,L-1}}\right)\right)\exp\left(-\tau_k\left(D_{I_k\,(\boldsymbol{r},I_0),I_k\,(\boldsymbol{r},I_0)} - D_{I_0\,L-1}\right)\right)\right]$$

Define new rule probabilities

$$p_{r|K,\,l} = \rho_r\,\tilde{F}^{(r)}_{K,\,l} \,/\, \tilde{D}_{K\,l} \equiv \frac{\rho_r\,\tilde{F}^{(r)}_{K,\,l}}{\sum_r \rho_r\,\tilde{F}^{(r)}_{K,\,l}}\,. \tag{17}$$

Then,

$$\left[\prod_{k=l-1\searrow 0} \hat{W}\;\exp\left(-\tau_k\, D\right)\right]_{I_l,I_0} = \sum_{\{r_k|k=1\,..\,L-1\}} \left[\prod_{k=L-1\searrow 0} p_{r_k|I_0,\,L-1}\right]\left(\tilde{D}_{I_0\,L-1}\right)^l \exp\left(-\left(\sum_k \tau_k\right) D_{I_0\,L-1}\right)$$

$$\times\left[\prod_{k=L-1\searrow 0}\left(\left(\frac{F^{(r_k)}_{I_k\,(\boldsymbol{r},I_0)}}{\tilde{F}^{(r_k)}_{I_0,L-1}}\right)\exp\left(-\tau_k\left(D_{I_k\,(\boldsymbol{r},I_0),I_k\,(\boldsymbol{r},I_0)} - D_{I_0\,L-1}\right)\right)\right)\right]$$

$$= \sum_{\{r_k|k=1\,..\,L-1\}} e_1\,(\boldsymbol{r})\,e_2\,(\boldsymbol{r})$$

where

$$e_1(\boldsymbol{r}) \equiv \left[\prod_{k=L-1\searrow 0} p_{r_k|I_0,L-1}\right] \left(\tilde{D}_{I_0\,L-1}\right)^{l} \exp\left(-\left(\sum_k \tau_k\right) D_{I_0\,L-1}\right)$$

$$e_2(\boldsymbol{r}) \equiv \prod_{k=L-1\searrow 0} \left(\left(\frac{F^{(r_k)}_{I_k(\boldsymbol{r},I_0)}}{\tilde{F}^{(r_k)}_{I_0,L-1}}\right) \exp\left(-\tau_k \left(D_{I_k(\boldsymbol{r},I_0),I_k(\boldsymbol{r},I_0)} - D_{I_0\,L-1}\right)\right)\right)$$

We define an arbitrary ordering "$\leqslant$" on the reaction types or channels indexed by r, so the reactions events are "sorted" by type iff $r_0 \leqslant r_1 \leqslant \ldots \leqslant r_{L-1}$. Let σ denote a permutation on L elements which we may apply to this ordering to get an unordered sequence of rules $\boldsymbol{r} = \{r_k \mid k = 0\ ..\ L-1\}$. For a given unordered $\boldsymbol{r}$ we further restrict the permutations σ to be those which do not interchange equal r's; this will avoid double-counting.

Then in the foregoing expression $\sum_{\{r_k|k=1\,..\,L-1\}} e_1(\boldsymbol{r})\, e_2(\boldsymbol{r})$ we may replace the multiple sum over reactions with a sum over permutations σ that order the reactions, and an outer sum over the possible *ordered* reaction sets:

$$\sum_{\{r_k|k=1\,..\,L-1\}} e(\boldsymbol{r}) = \sum_{\{r_0\leqslant\ldots\leqslant r_{L-1}\}} \sum_{\{\sigma\,|\,\sigma \text{ permutes unequal } r\text{'s}\}} e_1(\sigma(\boldsymbol{r}))\, e_2(\sigma(\boldsymbol{r}))$$

The number of r's taking each possible value $1\,..\,R$ is denoted $[s_1, \ \ldots\ s_R] = \boldsymbol{s}(r)$; these are the number of times each type of reaction occurs in the sequence $\boldsymbol{r}$. The components of $\boldsymbol{s}$ and $\boldsymbol{r}$ are therefore related as follows:

$$s_r = \sum_{k=0}^{L-1} \delta(r_k - r), \text{ which satisfies}$$

$$s_r \in \mathbb{N} \quad \text{and} \quad \sum\nolimits_r s_r = L$$

Also the ordered list of r's is determined by the vector $\boldsymbol{s}$:

$$r_k = \min\left\{r \,\middle|\, k \leqslant \sum_{i=0}^{r} s_i\right\}.$$

Hence we may replace the sum over ordered $\boldsymbol{r}$ with a sum over constrained $\boldsymbol{s}$:

$$\sum_{\{r_k|k=1\,..\,L-1\}} e(\boldsymbol{r}) = \sum_{\left\{\boldsymbol{s}\,|\,s_r\in\mathbb{N},\,\sum_r s_r=L\right\}} \sum_{\{\sigma\,|\,\sigma \text{ permutes unequal } r\text{'s}\,|\,\boldsymbol{s}\}} e_1(\sigma(\boldsymbol{r}))\, e_2(\sigma(\boldsymbol{r}))$$

$e_1(\boldsymbol{r})$ however depends on $\boldsymbol{r}$ only through $\boldsymbol{s}$, which is permutation invariant:

$$e_1(\boldsymbol{r}) \equiv \tilde{e}_1(\boldsymbol{s}(\boldsymbol{r})) = \tilde{e}_1(\boldsymbol{s}(\sigma(\boldsymbol{r}))) = e_1(\sigma(\boldsymbol{r}))$$

Hence

$$\sum_{\{r_k \,|\, k=1\,..\,L-1\}} e_1\,(\sigma(\boldsymbol{r}))\, e_2\,(\sigma(\boldsymbol{r})) = \sum_{\{\boldsymbol{s}\,|\,s_r\in\mathbb{N}\,,\,\sum_r s_r = L\}} \tilde{e}_1\,(\boldsymbol{s}(\boldsymbol{r})) \sum_{\{\sigma\,|\,\sigma \text{ permutes unequal } r\text{'s}\,|\,\boldsymbol{s}\}} e_2\,(\sigma(\boldsymbol{r}))$$

$$= \sum_{\{\boldsymbol{s}\,|\,s_r\in\mathbb{N}\,,\,\sum_r s_r = L\}} \tilde{e}_1\,(\boldsymbol{s}(\boldsymbol{r}))\left(\sum_{\{\sigma\,|\,\sigma \text{ permutes unequal } r\text{'s}\,|\,\boldsymbol{s}\}} e_2\,(\sigma(\boldsymbol{r}))\right)$$

$$= \sum_{\{\boldsymbol{s}\,|\,s_r\in\mathbb{N}\,,\,\sum_r s_r = L\}} \tilde{e}_1\,(\boldsymbol{s}(\boldsymbol{r}))\binom{L}{s_1\ \ldots\ s_R}\langle e_2\,(\sigma(\boldsymbol{r}))\rangle_{\{\sigma \text{ permutes unequal } r\text{'s}\,|\,\boldsymbol{s}\}}$$

where $\langle\,...\rangle_{\mathcal{S}}$ denotes averaging over the given set $\mathcal{S}$. On the other hand, $e_2\,(\boldsymbol{r})$ is invariant under any permutation σ which *only* exchanges equal r's, so

$$\langle e_2\,(\sigma(\boldsymbol{r}))\rangle_{\{\sigma \text{ permutes unequal } r\text{'s}\,|\,\boldsymbol{s}\}} = \langle e_2\,(\sigma(\boldsymbol{r}))\rangle_{\{\sigma \text{ permutes integers } 1..L\}}$$

and we find

$$\sum_{\{r_k \,|\, k=1\,..\,L-1\}} e_1\,(\sigma(\boldsymbol{r}))\, e_2\,(\sigma(\boldsymbol{r})) = \sum_{\{\boldsymbol{s}\,|\,s_r\in\mathbb{N}\,,\,\sum_r s_r = L\}} \binom{L}{s_1\ \ldots\ s_R}\tilde{e}_1\,(\boldsymbol{s}(\boldsymbol{r}))\,\langle e_2\,(\sigma(\boldsymbol{r}))\rangle_{\{\sigma \text{ permutes } r\text{'s}\,|\,\boldsymbol{s}\}}$$

Consequently,

$$\left[\prod_{k=l-1\searrow 0}\hat{W}\ \exp\left(-\tau_k\, D\right)\right]_{I_L, I_0} = \sum_{\{\boldsymbol{s}\,|\,s_r\in\mathbb{N}\,,\,\sum_r s_r = L\}} \binom{L}{s_1\ \ldots\ s_R}\left[\prod_{r=1}^{R}\left(p_{r|I_0\,,L-1}\right)^{s_r}\right]\left(\tilde{D}_{I_0\ L-1}\right)^{l}\exp\left(-\left(\sum_k \tau_k\right)\underset{\sim}{D}_{I_0\ L-1}\right)$$

$$\times\left\langle\left[\prod_{k=L-1\searrow 0}\left(\frac{F^{(r_k)}_{I_k\,(\sigma(\boldsymbol{r}),I_0)}}{\tilde{F}^{(r_k)}_{I_0,L-1}}\right)\exp\left(-\tau_k\left(D_{I_k\,(\sigma(\boldsymbol{r}),I_0),I_k\,(\sigma(\boldsymbol{r}),I_0)} - \underset{\sim}{D}_{I_0\ L-1}\right)\right)\right]\right\rangle_{\{\sigma|\boldsymbol{s}\}}.$$

This can be decomposed into more elementary probability distributions:

$$\left[\prod_{k=L-1\searrow 0}\hat{W}\ \exp\left(-\tau_k\, D\right)\right]_{I_L, I_0} = \frac{\left(\tilde{D}_{I_0\ L-1}\right)^{L}}{\left(\underset{\sim}{D}_{I_0\ L-1}\right)^{L}}\sum_{\{\boldsymbol{s}\,|\,s_r\in\mathbb{N}\,,\,\sum_r s_r = L\}}\text{Multinomial}(\boldsymbol{s}\,|\,\boldsymbol{p},\,L)$$

$$\times\,\text{Erlang}\left(\sum_k\tau_k\,\middle|\,L,\,\underset{\sim}{D}_{I_0\ L-1}\right)\text{UniformSimplex}(\boldsymbol{\tau};\,L)\,\text{Accept}(\boldsymbol{s},\,L,\,\boldsymbol{\tau}) \tag{18}$$

where

$$\text{Multinomial}(\boldsymbol{s}\,|\,\boldsymbol{p},\,L) = \binom{L}{s_1\ \ldots\ s_R}\left[\prod_{r=1}^{R}\left(p_{r|I_0\,,L-1}\right)^{s_r}\right],\ \text{with}$$

$$p_{r|I_0\,,L-1} = \frac{\rho_r\,\tilde{F}^{(r)}_{I_0\,,L-1}}{\sum_r \rho_r\,\tilde{F}^{(r)}_{I_0\,,L-1}}\ ;$$

$$\text{Erlang}(t;\,l,\,\lambda) \equiv \lambda^l\, e^{-\lambda\, t}\, t^{l-1}\,\big/\,(l-1)!$$
$$\text{where}\ \ \langle t\rangle_{\text{Erlang}} = l/\lambda\ ;$$

We note that the Erlang distribution is the Gamma distribution specialized to integer-valued shape parameter, l,

$$\text{UniformSimplex}(\boldsymbol{\tau}; L) = 1 \Big/ \left(\frac{t^{L-1}}{(L-1)!} \right);$$

and the acceptance probability

$$\text{Accept}(\boldsymbol{s}, l, \boldsymbol{\tau}) \equiv \langle P_\sigma \rangle_{\{\sigma|\boldsymbol{s}\}} ,$$

where

$$P_\sigma = \left[\prod_{k=L-1 \searrow 0} \left(\frac{F^{(r_k)}_{I_k(\sigma(\boldsymbol{r}),I_0)}}{\tilde{F}^{(r_k)}_{I_0,L-1}} \right) \right] \exp\left(-\sum_k \tau_k \left(D_{I_k(\sigma(\boldsymbol{r}),I_0),I_k(\sigma(\boldsymbol{r}),I_0)} - \underset{\sim}{D}_{I_0\,L-1} \right) \right). \tag{19}$$

From the definition of P_σ in Equation 19 and the fact that $\tilde{F}$ and $\underset{\sim}{D}$ are bounds, it follows that Accept $(\boldsymbol{s}, l, \boldsymbol{\tau}) \leqslant 1$. Also, if $R_{J|K\,L-1} = 0$ (so that state J is not reachable from state K after $L-1$ steps of SSA) then $P_\sigma = 0$, so that Equation 18 still agrees with Equation 8 despite the restriction to $R_{J|K\,L-1} = 1$ stated in the foregoing calculation.

Thus, Equation 18 provides an equivalent probability distribution and Markov process to Equation 8 .

2.4.3 Efficient rejection sampling algorithm

We now seek M and P' and $\underset{\sim}{A}(x)$ among the factors of Equation 18. We can upper-bound and lower-bound P_σ of Equation 19:

$$\underset{\sim}{P}\left(\boldsymbol{s}, \sum_k \tau_k, L \right) \leqslant P_\sigma \leqslant 1 \tag{20}$$

where

$$\underset{\sim}{P}\left(\boldsymbol{s}, \sum_k \tau_k, L \right) \equiv \left[\prod_{r=1}^{R} \left(\frac{\underset{\sim}{F}^{(r)}_{I_0,L-1}}{\tilde{F}^{(r)}_{I_0,L-1}} \right)^{s_r} \right] \exp\left(-\left(\sum_k \tau_k \right) \left(\tilde{D}_{I_0\,L-1} - \underset{\sim}{D}_{I_0\,L-1} \right) \right) \tag{21}$$

Note that $\underset{\sim}{P}$ does not depend on σ. This allows us to use rejection sampling [16] to transform samples of the bounding distribution

$$g(\boldsymbol{s}, \boldsymbol{\tau}) = \text{Multinomial}(\boldsymbol{s} \mid \boldsymbol{p}, L)\, \text{Erlang}\left(\sum_k \tau_k \,\middle|\, L, \underset{\sim}{D}_{I_0\,L-1} \right) \text{UniformSimplex}\left(\boldsymbol{\tau}; l, \sum_{k=0}^{L-1} \tau_k \right)$$

into samples of the target distribution

$$f(\boldsymbol{s}, \boldsymbol{\tau}) = g(\boldsymbol{s}, \boldsymbol{\tau}) \frac{(\tilde{D}_{I_0\,l})^L}{(\underset{\sim}{D}_{I_0\,l})^L} \text{Accept}(\boldsymbol{s}, L, \boldsymbol{\tau})$$

since the ratio $f(\boldsymbol{s}, \boldsymbol{\tau}) / g(\boldsymbol{s}, \boldsymbol{\tau})$ is bounded above by $M = \left(\tilde{D}_{I_0\,L-1} / \underset{\sim}{D}_{I_0\,L-1} \right)^L \geqslant 1$. $g(\boldsymbol{s}, \boldsymbol{\tau})$ plays the role of $P'(x)$ in the rejection sampling algorithm of Section 2.4.1, $f(\boldsymbol{s}, \boldsymbol{\tau})$ plays the role of $P(x)$, and M has just been defined. This bound is independent of all randomly chosen variables $\boldsymbol{s}$, t, $\boldsymbol{\tau}$, σ and just restores the probability otherwise

lost in rejection sampling due to the Accept($\boldsymbol{s}$, L, $\boldsymbol{\tau}$) factor being $\leqslant 1$. It remains to define $\underset{\sim}{A}(x)$ for the "efficient rejection sampling" algorithm.

In order to apply the "efficient rejection sampling" algorithm of Section 2.4.1, we need to find a lower bound $\underset{\sim}{A}(x)$ for Accept($\boldsymbol{s}$, l, $\boldsymbol{\tau}$) $= \langle P_\sigma \rangle_{\{\sigma|\boldsymbol{s}\}}$. Fortunately $P(\boldsymbol{s}, \sum_{\underset{\sim}{k}} \tau_k, L)$ is a lower bound for P_σ, so we can just average over σ compatible with $\boldsymbol{s}$. Then P_σ may be expressed as a mixture distribution:

$$P_\sigma = \underset{\sim}{P} \cdot 1 + (1 - \underset{\sim}{P}) \cdot Q_\sigma, \text{ where}$$

$$Q_\sigma = \left(\frac{P_\sigma - \underset{\sim}{P}}{1 - \underset{\sim}{P}} \right) \leqslant 1 \tag{22}$$

and thus

$$\langle P_\sigma \rangle_{\{\sigma|\boldsymbol{s}\}} = \underset{\sim}{P} \cdot 1 + (1 - \underset{\sim}{P}) \cdot \langle Q_\sigma \rangle_{\{\sigma|\boldsymbol{s}\}}$$

However, instead of numerically averaging over σ to compute $\langle Q_\sigma \rangle_{\{\sigma|\boldsymbol{s}\}}$ in each iteration, we will instead draw a single sample of σ and use that sample's value of Q_σ. This step is also exact since we can just define Accept($\boldsymbol{\sigma}$, L, $\boldsymbol{\tau}$) = Accept($\boldsymbol{s}$, L, $\boldsymbol{\tau}$) $\cdot$ Pr($\boldsymbol{\sigma} \mid \boldsymbol{s}$), where Pr($\boldsymbol{\sigma} \mid \boldsymbol{s}$) is uniform, and apply accelerated rejection sampling to $f(\boldsymbol{s}, \boldsymbol{\tau})$ Pr($\boldsymbol{\sigma} \mid \boldsymbol{s}$) using the corresponding bounds $f(\boldsymbol{s}, \boldsymbol{\tau})$ Pr($\boldsymbol{\sigma} \mid \boldsymbol{s}$) for $P'(x)$ and P_σ for $\underset{\sim}{A}(x)$.

Algorithmically this expression can be sampled from as follows. First compute $\underset{\sim}{P}$. Then with probability $\underset{\sim}{P}$, accept the "current" candidate move determined by all the other distributions. In the relatively unlikely event (probability 1-$\underset{\sim}{P}$) that the move is not immediately accepted this way, we then draw a random $\sigma \mid \boldsymbol{s}$ and compute its Q_σ. Then, accept the current move with probabilty Q_σ, and with probability $1 - Q_\sigma$ reject the current move, draw a new one, and iterate. For computational efficiency the initial acceptance rate $\underset{\sim}{P}$ should be high. Pseudocode for the resulting algorithm will be presented in the next section.

2.5 Exact R-leap algorithm

We now assemble the ER-leap algorithm from the key update equations derived in previous sections: Equation 11, Equation 13, Equation 17, Equation 18, Equation 21, Equation 19, and Equation 22 .

2.5.1 Algorithm summary

We adapt the efficient rejection sampling algorithm of Section 2.4.1, with the random variables $\boldsymbol{s}$, $\boldsymbol{\sigma}$, and $\boldsymbol{\tau}$, and the expressions for P, P', M and $\underset{\sim}{A}$ of Section 2.4.3, into pseudocode for the core of the resulting Exact R-leap algorithm:

set counters n, $\sum_k \underset{\sim}{P}_k$, $\sum_k \underset{\sim}{P}_k{}^2$ to zero
starting at state I_0, initial time t_0, and user-specified initial leap L
while $t \leq T$ {
    if L equals 1 **then** perform one SSA step, set $\underset{\sim}{P}$=1 (for dynamic L update counter) ;
    else repeat {
        compute or update the bounds on F's, D's for I_0,
            by Equation 11 and Equation 13;
        compute $\boldsymbol{p}$: $p_{r|I_0,l-1} = \rho_r \tilde{F}^{(r)}_{I_0,l-1} / \tilde{D}_{I_0\, l}$;

sample $\boldsymbol{s}$ from Multinomial($\boldsymbol{s} \mid \boldsymbol{p}, l$)
(using sorted sequential Binomials, for efficiency, as in R-leap) ;
sample $\sum_k \tau_k$ from Erlang$\left(\sum_k \tau_k \mid L, \underset{\sim}{D}_{I_0\, l}\right)$;
compute $\underset{\sim}{P}(\boldsymbol{s}, \sum_k \tau_k, L)$ by Equation 21 ; // cheap
with probability $\underset{\sim}{P}$ {
accept step;
if first-rejection-iteration **then** increment early-acceptance counter;
} **otherwise** {
// expensive
sample σ from permutations consistent with **s ;**
compute $P_\sigma \in [\underset{\sim}{P}, 1]$ by Equation 19 ;
compute $\underset{\sim}{Q}_\sigma = \left(\frac{P_\sigma - \underset{\sim}{P}}{1 - \underset{\sim}{P}}\right)$;
with probability $\underset{\sim}{Q}_\sigma$ accept step otherwise reject step;
}
} **until** step accepted ;
update I_0 ;
increment n, $\sum_k \underset{\sim}{P}_k = \sum_k \underset{\sim}{P}_k + \underset{\sim}{P}$, $\sum_k \underset{\sim}{P}_k{}^2 = \sum_k \underset{\sim}{P}_k{}^2 + \underset{\sim}{P}^2$;
if n ⩾ b **then** compute maximal L by Equation 25.
if L changed or Uniform $\in [0, 1]$ is below $1/L^2$ **then** set counters n, $\sum_k \underset{\sim}{P}_k$, $\sum_k \underset{\sim}{P}_k{}^2$ to zero ;
} **until** done

The implementation used in this paper is written in C++ and contains around 600 lines of code for the core components.

2.5.2 Acceptance ratio analysis

A preliminary analysis looks very permissive of large L:

$$\begin{aligned} \underset{\sim}{\Delta}_a &\equiv \min_r\ \Delta m_a^r \qquad \tilde{m}_a \equiv \max_r\ m_a^r \\ \tilde{\Delta}_a &\equiv \max_r\ \Delta m_a^r \qquad \tilde{m} \equiv \max_r\ \sum_a m_a^r \end{aligned} \tag{23}$$

Then for large n_a, such that

$$n_a \gg (L-1)\,|\,\underset{\sim}{\Delta}_a\,| + \tilde{m}_a\ ,$$

we further insist that

$$L(L-1) \leqslant \frac{\min_a n_a}{\tilde{m} \max_a \left(\tilde{\Delta}_a - \underset{\sim}{\Delta}_a + \tilde{m}_a\right)} \log(1/\alpha)$$

where $\alpha \in [0, 1]$ is the minimal early-acceptance rate (should be close to 1 for efficiency). If $\alpha = 1 - \epsilon$, this becomes roughly

$$L \leqslant \sqrt{\frac{\epsilon \min_a n_a}{\tilde{m} \max_a \left(\tilde{\Delta}_a - \underset{\sim}{\Delta}_a + \tilde{m}_a\right)}}\ .$$

2.5.3 Asymptotic cost of update

The asymptotic computational cost of simulating with ER-leap can be analyzed. The amount of computation required to calculate and sample $\underset{\sim}{P}$ is dominated by the time required to calculate the reaction probability rates or propensities. The asymptotic cost of this will be $O(R)$, where R is the number of reaction types or channels. In the event that an "early" sample is rejected, the more thorough sampling and calculation of P_σ, that becomes necessary, will be dominated by the recalculation of the reaction probability rates for each of the L reaction events. Therefore, computing P_σ will have asymptotic cost $O(L\,R)$. Thus, during simulation the expected computation per attempted leap will be the inevitable cost of calculating $\underset{\sim}{P}$ plus the cost of calculating P_σ, which is occurs with probability $(1 - \langle \underset{\sim}{P} \rangle)$. So the computational cost for one leap attempt can be estimated as

$$O(R + (1 - \langle \underset{\sim}{P} \rangle)\, L\, R) \tag{24}$$

To calculate the expected CPU cost per reaction event, we assume that all P_σ samples are rejected. This yields a lower bound on the expected number of accepted reaction events per leap, which will be $\langle \underset{\sim}{P} \rangle\, L$. Additionally, the cost for one SSA step will be $O(R)$ and the number of reactions events per step will be one. Thus the per-event costs for ER-leap and SSA will be

$$\text{ERleap cost} = \frac{\text{ERleap leap cost}}{\text{reaction events}} \leq \frac{R + (1 - \langle \underset{\sim}{P} \rangle)\, L\, R}{\langle \underset{\sim}{P} \rangle\, L},$$

$$\text{SSA cost} = \frac{\text{SSA step cost}}{\text{reaction events}} = \frac{R}{1}.$$

The cost ratio between SSA and ER-leap is therefore

$$\text{cost ratio} = \frac{\text{ERleap cost}}{\text{SSA cost}} \leq \frac{1 + (1 - \langle \underset{\sim}{P} \rangle)\, L}{\langle \underset{\sim}{P} \rangle\, L}.$$

When this cost ratio is less than one, ER-leap will be asymptotically faster than SSA. This is the case whenver $\langle \underset{\sim}{P} \rangle > (1 + L)/2\,L$ which in turn is $> 1/2$. Finally, taking the inverse of the cost ratio gives us the lower bound on the speedup of ER-leap over SSA, which is

$$\text{speedup} \propto \frac{\langle \underset{\sim}{P} \rangle\, L}{1 + (1 - \langle \underset{\sim}{P} \rangle)\, L}.$$

The required data structures and space requirements for ERLeap do not go significantly beyond what is conventional for SSA simulation: Each reaction needs a list of input/output species, so an array is used to remember the state of the system as well as a temporary state copy when calculating P_σ, and arrays are used to store σ, τ, and the maximal and minimal $\tilde{\Delta}_a$ values.

2.5.4 Dynamic choice of L

ER-leap efficiency depends on finding an L which optimally balances the benefits of having a large L versus the potential inefficiencies that would result from sample rejections. Our heuristic is described here.

Recall from Equation 24 the the cost of calculating early acceptance samples will be $O(R)$ and the expected cost of calculating the late acceptance samples is $O((1 - \langle \underset{\sim}{P} \rangle)\, L\, R)$ for each leap attempt. Balancing these costs yields $L = 1/(1 - \langle \underset{\sim}{P} \rangle)$, or $\langle \underset{\sim}{P} \rangle = (L - 1)/L$. So, during simulation the goal is to chose an L satisfying $\langle \underset{\sim}{P} \rangle \approx (L - 1)/L$. This is done by sampling $\underset{\sim}{P}$ to obtain an estimate of the 'true' value of $\langle \underset{\sim}{P} \rangle$ (for which we take at least five samples). Then L is increased or decreased by at most one, to minimize the error in the condition

$\langle \underset{\sim}{P} \rangle^{\beta} \approx (L-1)/L$, where the β parameter is introduced to tune differences in CPU running time between the $\underset{\sim}{P}$ and P_σ calculations. Experiments (not presented) show good performance when β=2/3 and this is used in all subsequent experiments.

Confidence intervals for our estimate of μ, the mean of $\underset{\sim}{P}$, come from the central limit theorem:

$$\mu = \overline{\mu} \pm z \sqrt{\frac{\overline{\sigma^2}}{n}}$$
$$= \overline{\mu} \pm E_{\text{rror}}$$

where statistics for calculating the sample mean and sample variance $\left(\overline{\mu}, \overline{\sigma^2}\right)$ are gathered from $\underset{\sim}{P}$ during simulatiion, z is a 'confidence factor' (we used z=1.7 in experiments), and n is the number of samples. Given the goal $\langle \underset{\sim}{P} \rangle$ for a given L, namely $h(L) = ((L-1)/L)^{1/\beta}$, the rule for updating L to a new L' is

$$L' = \begin{cases} L+1, & \text{if } h(L) < \overline{\mu} - E_{\text{rror}} \text{ and } h(L+1) < \overline{\mu} + E_{\text{rror}} \\ L-1, & \text{if } \overline{\mu} - E_{\text{rror}} < h(L-1) \text{ and } \overline{\mu} + E_{\text{rror}} < h(L) \\ L, & \text{otherwise} \end{cases} \tag{25}$$

which changes L whenever the interval $\{\overline{\mu} - E_{\text{rror}}, \overline{\mu} + E_{\text{rror}}\}$ doesn't contain $h(L)$, and changing L by one would either (a) put $h(L')$ within this interval, or (b) put $h(L')$ in between $h(L)$ and this interval, thereby bringing it closer to the desired interval.

2.5.5 An Illustrative Example

As a specific example of the use of the ER-leap algorithm, consider the two-reaction dimerization process $\left\{2\,S_1 \underset{\rho_2}{\overset{\rho_1}{\rightleftarrows}} S_2\right\}$ with forward and reverse reactions $r = 1$ and $r = 2$. Recall from Equation 9 that the instantaneous rates of firing, also called propensities, for each reaction are given by

$$a_1(\boldsymbol{n}) = \rho_1\, n_1(n_1 - 1), \quad a_2(\boldsymbol{n}) = \rho_2\, n_2\ .$$

(Some authors divide $a_1(\boldsymbol{n})$ by two to "avoid double counting", but our convention is to absorb this factor of two into ρ_1 and thereby remain notationally consistent with the law of mass action.) ER-leap requires upper and lower bounds on the propensities for each reaction at any of L reaction event "steps". The bounds are not required to be tight, but here it is easy to find the tightest bounds using Equation 11:

$$\tilde{a}_1(\boldsymbol{n}) = \rho_1\,(n_1 + 2(L-1))((n_1 + 2(L-1) - 1)), \quad \underset{\sim}{a}_1(\boldsymbol{n}) = \rho_1\,(n_1 - 2(L-1))((n_1 - 2(L-1) - 1))$$
$$\tilde{a}_2(\boldsymbol{n}) = \rho_2\,(n_2 + (L-1)), \qquad \underset{\sim}{a}_2(\boldsymbol{n}) = \rho_2\,(n_2 - (L-1)).$$

The upper bound $\tilde{a}_1$ comes from the extreme situation in which all L reactions are of type $r = 2$. Two S_1 are produced every time $r = 2$ fires. So we calculate the upper bounding propensities with an upper bound for S_1: $\tilde{n_1} = n_1 + 2(L-1)$. Recall that $(L-1)$ is used instead of L because about the bounds apply *just before* the L^{th} step occurs. The other bounds are calculated in the same way.

Given bounds on a_1 and a_2, we can sample the reactions and time step. First, the number of times $r = 1$ and $r = 2$ are fired (s_1, s_2) is sampled from a multinomial distribution (here equivalent to a binomial) with parameters $\left(\left(\frac{\tilde{a}_1(n)}{\tilde{a}_0(x)}, \frac{\tilde{a}_2(n)}{\tilde{a}_0(n)}\right), L\right)$, where $\tilde{a}_0(n) = \tilde{a}_1(n) + \tilde{a}_2(n)$. Next, the total time step τ is sampled from the gamma distribution with parameters $\left(\underset{\sim}{a}_0(n), L\right)$.

To compute the probability of early acceptance, Equation 21 is used. This simplifies to

$$\text{Prob}_{\text{early}}(\boldsymbol{s}, \boldsymbol{\tau}) = \left(\frac{\underset{\sim}{F}^{(1)}_{I_0, L-1}}{\tilde{F}^{(1)}_{I_0, L-1}}\right)^{s_1} \left(\frac{\underset{\sim}{F}^{(2)}_{I_0, L-1}}{\tilde{F}^{(2)}_{I_0, L-1}}\right)^{s_2} \exp\left(-(\tau_1 + \tau_2)\left(\tilde{D}_{I_0\, L-1} - \underset{\sim}{D}_{I_0\, L-1}\right)\right)$$

$$= \left(\frac{\underset{\sim}{a}_1(\boldsymbol{n})}{\tilde{a}_1(\boldsymbol{n})}\right)^{s_1} \left(\frac{\underset{\sim}{a}_2(\boldsymbol{n})}{\tilde{a}_2(\boldsymbol{n})}\right)^{s_2} \exp\left(-\boldsymbol{\tau}\left(\tilde{a}_0(\boldsymbol{n}) - \underset{\sim}{a}_0(\boldsymbol{n})\right)\right).$$

We accept the sample $(\boldsymbol{s}, \boldsymbol{\tau})$ *early*, and with little computational cost, with $\text{Prob}_{\text{early}}$. If there is no early acceptance, the probability of late acceptance must be calculated. To calculate this first we must sample an *ordering* of reactions, $\boldsymbol{\sigma}$. This ordering is just a random shuffling of the L reactions. So our sample may look like $\boldsymbol{\sigma} = \{r = 1, r = 1, r = 2, \ldots r = 1\}$. Next, we need to sample the length of individual time steps for each reaction, $\{\tau_1, \tau_2, \ldots, \tau_L\}$. This can be done by independently sampling L unit exponential random variables and "normalizing" them so their sum is $\boldsymbol{\tau}$. It is now possible to calculate the true probability of acceptance from Equation 19:

$$\text{Prob}_{\text{accept}}(\boldsymbol{\sigma}, \{\tau_i\}) = \left(\frac{1}{\tilde{F}^{(1)}_{I_0, L-1}}\right)^{s_1} \left(\frac{1}{\tilde{F}^{(2)}_{I_0, L-1}}\right)^{s_2} \prod_{i=1}^{L} \underset{\sim}{F}^{(\sigma_i)}_{I_i, L-1} \exp\left(-\tau_i \left(D_{I_i\, L-1} - \underset{\sim}{D}_{I_0\, L-1}\right)\right)$$

$$= \left(\frac{1}{\tilde{a}_1(\boldsymbol{n})}\right)^{s_1} \left(\frac{1}{\tilde{a}_2(\boldsymbol{n})}\right)^{s_2} \prod_{i=1}^{L} a_{\sigma_i}(\boldsymbol{n_i}) \exp\left(-\tau_i \left(a_0(\boldsymbol{n_i}) - \underset{\sim}{a}_0(\boldsymbol{n})\right)\right).$$

Here $\tilde{a}(\boldsymbol{n})$ and $\underset{\sim}{a}_0(\boldsymbol{n})$ are held constant during the calculation, but the true propensities $a_{\sigma_i}(\boldsymbol{n_i})$ are recalculated after each reaction σ_i occurs. State I_0 corresponds to state vector $\boldsymbol{n}$, and I_i corresponds to $\boldsymbol{n_i}$, where $i \in \{1 .. L\}$ indexes the step number. With probability $(\text{Prob}_{\text{accept}} - \text{Prob}_{\text{early}})/(1 - \text{Prob}_{\text{early}})$ we accept the sample and update $\boldsymbol{n}$. Otherwise the sample is rejected.

In general calculating the propensity bounds with Equation 11 and Equation 13 can be made efficient by noting that the maximum and minimum amounts by which a species may change in one reaction event remains constant throughout the simulation. These $\tilde{\Delta}_a$ and $\underset{\sim}{\Delta}_a$ values (defined in Equation 23) are calculated prior to simulation, and the bounding $\tilde{n}_a$ is calculated as $\tilde{n}_a = n_a + (L-1)\tilde{\Delta}_a$, from Equation 10. Then the propensity upper and lower bounds are calculated as conventional propensities except that the bounding $\tilde{n}_a$ and $\underset{\sim}{n}_a$ are used for each reactant instead of n_a.

3 Numerical Simulations

The above stochastic algorithms are implemented in the C++ programming language and run on a MacBook running OS X v10.5 with an Intel dual-core 1.83Ghz processor and 2.0GB of RAM. Experiments are performed with emphasis on exploring accuracy and speedup. We compare the present algorithm with the software developed for the τ-leap and R-leap algorithms as reported in the R-leap paper [11].

3.1 Accuracy

Here we verify ER-leap equivalence to SSA via numerical experiments. As an example of the tests performed in the CaliBayes test suite [17], we consider the Galton-Watson stochastic process where analytic solutions for the mean and standard deviation are known. Mass-action stochastic kinetics are assumed. The solutions are compared to trajectories of many runs of SSA, ER-leap, τ-leap and R-leap.

Algorithm accuracy was validated using a statistical test as performed in CaliByaes. The i^{th} sample at time t will be denoted $X_t^{(i)}$ and is drawn from the random variable X^t. The analytic mean and standard deviation at time t are μ_t and σ_t. Additionally, $\overline{X_t}$ is the sample mean and $\overline{S_t}$ is the sample standard deviation assuming $E[X_t] = \mu_t$. Using the central limit theorem, we eventually arrive to:

$$Z_t = \sqrt{n}\ \frac{\overline{X_t} - \mu_t}{\sigma_t}, \quad Y_t = \sqrt{\frac{n}{2}}\left(\frac{\overline{S_t}^2}{{\sigma_t}^2} - 1\right).$$

Under the null hypothesis that the simulator is correct, the Z_t and Y_t values should have a standard normal distribution. So most Z_t values are expected to lie in the range $(-3, 3)$. We further relax this constraint for Y_t to lie in the range $(-5, 5)$ because the standard deviation is less likely to be normally distributed.

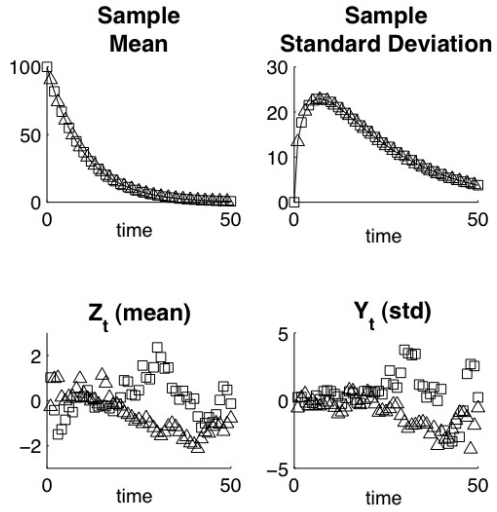

Figure 1. ER-leap (□) with $L = 4$ and SSA (△) compared with the analytical (——) mean and standard deviation. *Y*-axis in units of molecules. The Z_t and Y_t values will be normally distributed, assuming SSA equivalence. Therefore values in the range (-3,3) are considered reasonable. Galton-Watson stochastic process $\{X \longrightarrow 2X, X \longrightarrow \emptyset\}$ with rate parameters {1.0, 1.1} respectively and $X(0) = 100$. Simulation time is 50 seconds. Results from 20,000 runs.

We performed this analysis on SSA and ER-leap. As Figure 1 indicates, Z_t and Y_t are within the expected range for both simulation algorithms. This supports the notion that SSA and ER-leap draw from the same distribution.

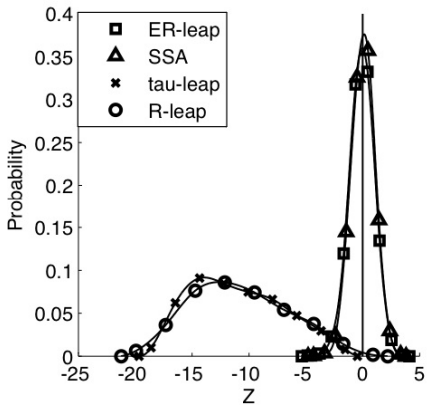

Figure 2. Distribution of Z_t for the four algorithms under consideration. ER-leap and SSA demonstrate a standard normal distribution whereas the approximate methods show Z_t values far outside the expected range. Reactions $\{X \longrightarrow 2X, X \longrightarrow \phi\}$ with rate parameters {0.11, 0.1} and $X(0) = 1.0 \times 10^5$. For ER-leap L=30. For R-leap θ=0.1 and ε=0.01. For τ-leap ε=0.01. Each Z_t calculated from 1000 time points for one second intervals up to time $t = 50$. The number of runs for each method varies in order to get smooth distributions and ranges from 1.0×10^5 to 2.0×10^5.

To demonstrate the sensitivity of this test we also compute Z_t and Y_t for the approximate algorithms. Interestingly, all algorithms do not show strong errors in Y_t. However, the absolute values of Z_t for R-leap and τ -leap are mostly greater than 3 [Figure 2]. This test indicates that SSA and ER-leap are equivalent with high certainty and it was sensitive enough to discover the error resulting from the assumptions made by R-leap and τ -leap.

3.2 CaliBayes validation

Similar analysis as above is performed on several models in the CaliBayes test suite version DSMTS 21 [17]. Three models with solvable mean and standard deviation are tested: the birth-death process, dimerization process and immigration-death process. Of these a total of 9 variations in initial conditions and parameters are simulated (the others not being tested due to limited ER-leap SBML support). The tested models are: 1-01, 1-03, 1-04, 1-05, 2-01, 2-02, 2-04, 3-01, 3-02.

Each test case has 50 time points where Z_t and Y_t values are calculated. A test is considered passing if $|Z_t| \leqslant 3.0$ for all 50 Z_t values with one exception per run. Likewise, since the standard devation normal assumption is not as strong, we require $|Y_t| \leqslant 5.0$ for all but one of the Y_t scores per test. This pass/fail criteria was also suggested in the CaliBayes documentation.

Furthermore, since the tests are made at discrete time points, a large leap may create a small but nonzero bias if we test at a state preceding the desired time *t*. To alleviate this problem we 'leap' to a time before *t* and then perform small SSA ($L = 1$) steps until *t* is reached. The SSA steps begin when the time is within $L\upsilon / \left(\underset{\sim}{D} + \tilde{D}\right)$ of *t*, with $\upsilon = 7$. In practice these small steps do not significantly affect running time.

Using the above criteria, we found all tested variations from the CaliBayes suite to pass, using ER-leap with $L = 3$ or automatically-selected L, and 20,000 simulations per model.

3.3 Williamowski-Rössler Model

The Williamowski-Rössler model [18], which contains several bi-moleclar reactions, is explored to demonstrate the usefullness of the ER-leap algorithm. Results indicate that the approximate methods do not model well the true stochastic behavior for particular instances of the system. Consider the following set of reactions:

$$X \underset{k_2}{\overset{k_1}{\longleftrightarrow}} 2\,X \qquad Y \underset{k_6}{\overset{k_5}{\longleftrightarrow}} \emptyset \qquad Z \underset{k_{10}}{\overset{k_9}{\longleftrightarrow}} 2\,Z$$

$$X + Y \underset{k_4}{\overset{k_3}{\longleftrightarrow}} 2\,Y \quad X + Z \underset{k_8}{\overset{k_7}{\longleftrightarrow}} \emptyset.$$

We can numerically solve for the corresponding set of deterministic mass action differential equations

$$\begin{aligned} \dot{x} &= \; k_1\,x - k_3\,\mathrm{xy} - k_2\,x^2 + k_4\,y^2 - k_7\,\mathrm{xz} + k_8 \\ \dot{y} &= \; k_3\,\mathrm{xy} - k_5\,y - k_4\,y^2 + k_6 \\ \dot{z} &= \; -k_7\,\mathrm{xz} + k_9\,z - k_{10}\,z^2 + k_8 \end{aligned}$$

and plot the solution of X vs. Y as in Figure 3.

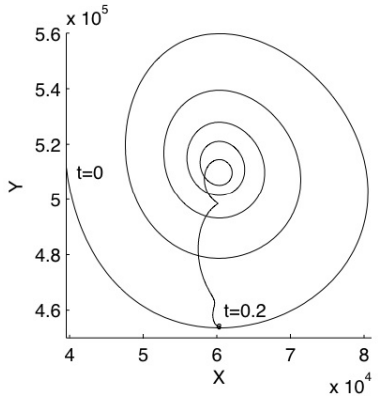

Figure 3: Mass-action deterministic solution of X vs. Y from time t=0 to t=0.2 for Williamowski-Rössler model. k_1=900, k_2=8.3×10^{-4}, k_3=0.00166, k_4=3.32×10^{-7}, k_5=100, k_6=18.06, k_7=0.00166, k_8=18.06, k_9=198, k_{10}= 0.00166. X(0)=39570. Y(0)=511470. Z(0)=0.

As time progresses the mean trajectory spirals in towards an attraction point near $\{6.0\times10^4, 5.1\times10^5\}$. However, once the inner region is reached, the trajectory falls towards another attraction point around $\{6.0\times10^4, 4.5\times10^5\}$. The stochastic algorithms are run and we can observe the density plots over time for the exact and approximate algorithms in Figure 4.

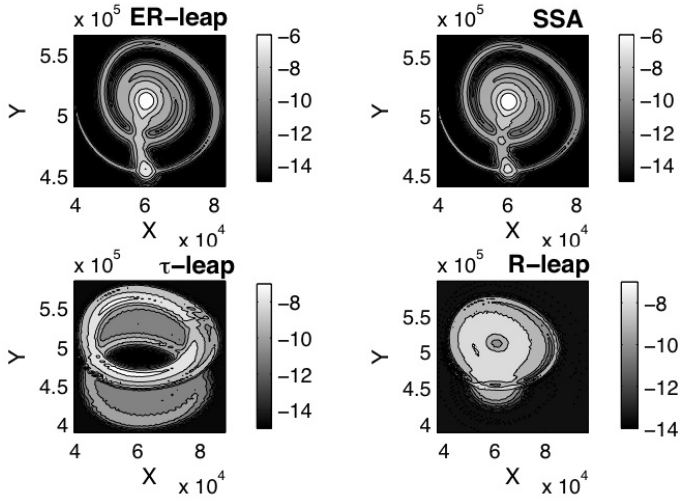

Figure 4: Comparing log probability densities for various simulation methods over time t=0 to t=0.2. Same parameters as Figure 3 . SSA and ER-leap appear identical. Total of 1,500 samples for each simulator. For ER-leap L was chosen automatically and averaged L=23. For τ-leap and R-leap ε=0.01. For R-leap θ=0.1. Measurement taken every 10^{-4} sec.

As Figure 4 and Figure 5 demonstrate, there is a substantial difference between the probability densities from the exact and approximate simulation methods. However, ER-leap is able to produce an answer similar to that of SSA and is about 4.5 times faster on this example.

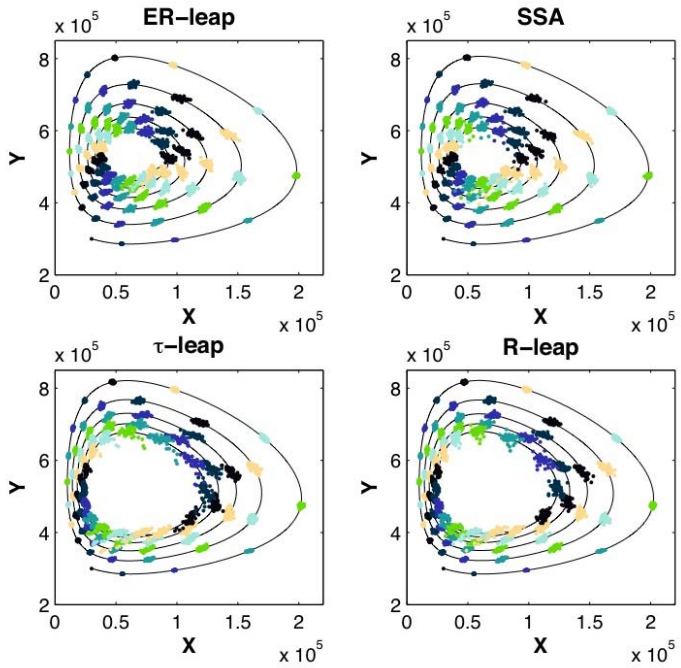

Figure 5. Another look at the differences in trajectories. Distribution of 50 runs for the four algorithms. Same network as Figure 4 . X(0)=30,000. Y(0)=300,000. (So we start further out in the spiral). Simulate from time t=0 to t=0.13, before the "escape" shown in Figure 4. A constant amount of time passes between time samples. Each cluster of points represents a group of trajectories that started at the same initial condition and has run for the same amount of time, varying only stochastically, ie. by the choice of the seed for a random number generator.

We modify the foregoing Williamowski-Rössler model to have rate parameters in the chaotic regime as described in [18]. The idea is that small simulation errors may grow into large errors as time progresses. The SSA mean of X vs. Y over 1,150 runs is shown in Figure 6. Notice the erratic behavior, which deterministic analysis may have difficulty capturing [18].

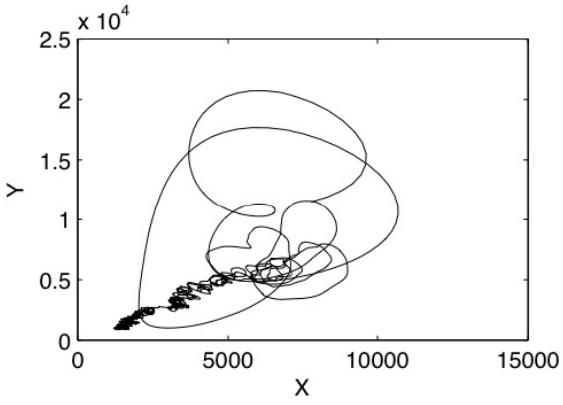

Figure 6. Mean number of molecules on chaotic system over 1,150 SSA runs from time t=0 to t=30. k_1=30, k_2=8.3×10^{-4}, k_3=0.00166, k_4=3.32×10^{-7}, k_5=10, k_6=0.602 k_7=0.00166, k_8=0.602, k_9=16.58, k_{10}= 0.00166. X(0)=7800. Y(0)=11500 Z(0)=0.

When we examine log-densities accumulated over time we observe that ER-leap and SSA have densities that appear very similar whereas the approximate methods display greater departures from SSA.

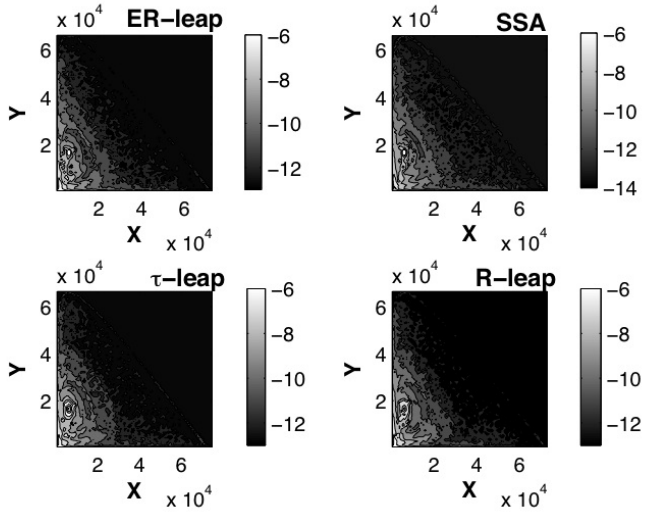

Figure 7. Comparing X vs. Y log probability density for various simulation methods over time t=0 to t=30. Same parameters as in Figure 6. Total of 1,150 samples runs for each simulator. ER-leap L was chosen automatically and averaged around 11.5. For τ -leap and R-leap ε=0.01. For R-leap θ=0.1. Measurement taken every 0.1 sec.

In the corresponding mass action ODE's in the chaotic regime, small simulation errors grow exponentially. Furthermore, mass action analysis has sometimes proven insufficient to model the system even for a large number of molecules [18]. To elucidate model dynamics stochastic simulation methods need to be applied. To our knowledge ER-leap is the fastest such algorithm to do this exactly.

3.4 Scaling of computational cost with reaction events

The acceleration of SSA by ER-leap depends on the number of molecules n (along with other factors not explored here). We run the Galton-Watson model with initial molecule number n ranging from 10 to 9×10^7. As expected the SSA CPU running time scales linearly with n. The ER-leap CPU time appears to scale as $O(n^{\alpha})$ where $\alpha\approx2/3$ [Figure 8]. R-leap and τ-leap scale much better to large number of molecules, but are not exact algorithms. Notice that the slope of the approximating methods is nearly 0. This is due to the fact that the leap sizes are determined from bounds on relative propensity changes. Because this system only involves first order reactions, this leap control results in sizes that are proportional to n. Substantial room remains for the improvement of exact algorithms.

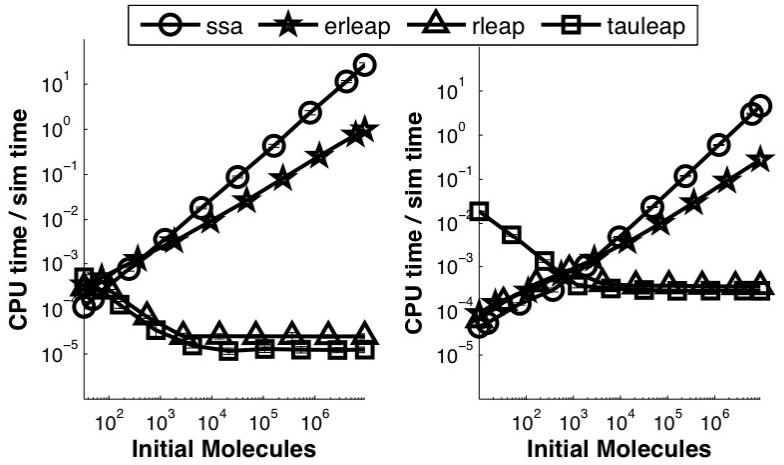

Figure 8. Log-log scaling of CPU running times for various stochastic simulation algorithms. The left panel plots results obtained for the Galton-Watson model with birth rate 0.101, death rate 0.10. Each test is simulated for 30 seconds. The slope of the ER-leap line is 0.65, and SSA is 0.99, about 1.0 as expected. Ratio is 0.66 . L is chosen automatically for ER-leap. R-leap has accuracy parameters θ=0.1 and ε=0.01. τ-leap has parameter ε=0.01. The right panel plots results obtained for the dimerization process {2X⟶S, S⟶2X} with rate parameters {0.001 / υ, 0.01} respectively, initial values $S(0) = n$, singleton molecule $X(0) = n/2$, and volume $\upsilon = n/100$. Slope of ER-leap line is 0.58 and slope of SSA line is 0.86 with a ratio of 0.68. Error bars represent one standard deviation.

Additionally, we can explore the trade-off between the potential gain of large L and loss of efficiency from rejecting samples from too-ambitious L values. There is an optimal L that is model- and time-specific. We explore this relationship by varying L for a particular simulation and observing the CPU cost, as plotted in Figure 9.

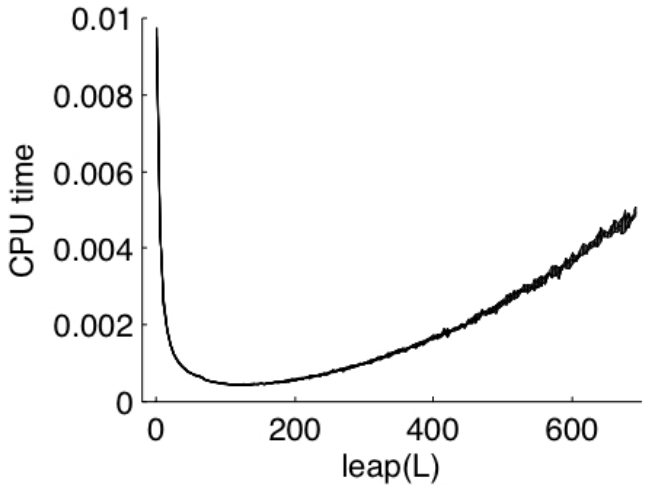

Figure 9: Varying L for birth/death process with rate of birth 0.1 and death 0.11. $X(0)=1\times10^7$. $X(0)=5\times10^6$. Simulation from t=0 to t=5. Initially as we increase L, CPU runtime drops dramatically until the optimum at about L=115 which is about 22x faster than SSA. For larger L, the rejection of proposed samples starts to decrease performance and there is a monotonic increase in CPU computation time.

This tradeoff can also be explored with a log-contour plot of CPU time and L [Figure 10]. Notice that as simulation time increases, the optimal L changes. This fact is due to a change in the value of Equation 21 as reactant numbers change. The lack of multiple local minima in Figure 10 suggests that dynamic optimization of L is not a hard problem.

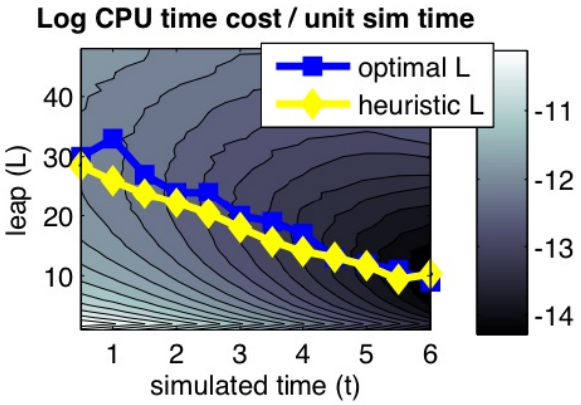

Figure 10. ER-leap contour plot of Log CPU time per unit simulation time vs. simulation time and leap size, L. Overlay of optimal and heuristic choice of L (from one run). Notice that the optimal leap L^* changes during simulation from $L^* = 34$ at $t = 0$ to about $L^* = 8$ at $t = 6$. Basic cascading network {S1⟶S2, S2 ⟶ S3, S3⟶S4} and all rates 1.0. Initial values: S1=4.2×10^4, S2=4.0×10^4, S3=3.5×10^4 and S4=0. Results averaged over 500 runs.

3.5 Scaling of computational cost with reaction channels

The acceleration of ER-leap over SSA is explored as a function of the number of reaction channels. The Williamowski-Rössler model is replicated over a d-dimensional grid. In each compartment of the grid there is a copy of the Williamowski-Rössler reaction network, including all of its chemical species and their intracompartmental reactions. In additiion, molecules diffuse (stochastically) between adjacent grid compartments. This is accomplished by replicating all WR reactions over the set of compartments, and adding new reactions of the form $\left\{X_c \xrightarrow{\rho} X_{c'}\right\}$ where c is the grid coordinate for molecules of type X and c' is any neighboring compartment. Diffusion is to adjacent compartments only, so the L_1 distance between c and c' is one. In the experiments shown below, $d = 3$.

Exact_R_LeapTR08_09V2.nb

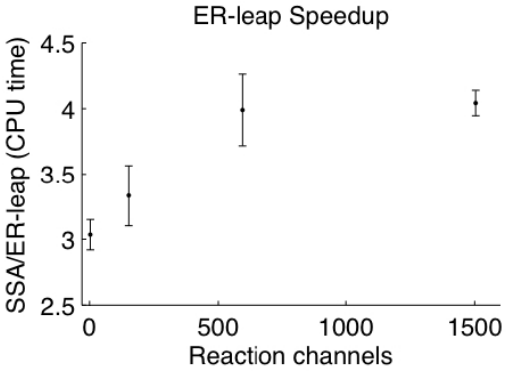

Figure 11: Speedup up is calculated as SSA wall clock time divided by ER-leap wall clock time. It increases monotonically from a one-cell system with 10 reaction channels to a 4×4×4 grid with 1504 reaction channels. In ER-leap L was chosen automatically, and averaged 23 over all experiments. Error bars are one standard deviation. Same rate parameters as Figure 3. Rate of diffusion is 0.01 and the initial number of molecules in each cell is $X_0=5.0\times10^4$, $Y_0=4.5\times10^5$, $Z_0=3.0\times10^4$.

As Figure 11 demonstrates, ER-leap may be used to accelerate systems with many reaction channels. It also demonstrates the feasability of applying ER-leap to spatially structured models.

4 Conclusions

We have derived an exact accelerated algorithm for stochastic simulation of chemical reactions, using rejection sampling together with upper and lower bounds on the probability of an outcome of a run of L reactions. We have demonstrated a speedup to sublinear time for simulating a large number of reactions. We have verified the accuracy of the method with senstive tests including examples from the Calibayes test suite and a chaotic reaction network.

We note that the SSA has also been accelerated, without approximation, by executing one reaction event at a time, lowering the cost of simulating each reaction event when there are many possible reaction to choose from [19]. An alternative acceleration of SSA has been proposed [20] based on exploiting cycle structure. The present ER-leap algorithm is based on the R-leap algorithm [11] that accelerates the SSA by specifying a number of reaction firings, and does not exploit a large number of reaction types as in [19-20]. Instead, it exploits the scaling possible for large numbers of reactant particles (molecules) and of reaction events. In these conditions, and for reaction networks (such as the Williamowski-Rossler oscillator) for which high-accuracy or exact simulation is necessary to find the correct long-time behavior, ER-leap may turn out to be the currently preferred algorithm. In any case, the existence of ER-Leap demonstrates that it is possible to create exact, accelerated stochastic simulation algorithms which scale better than SSA with respect to the number of reactant particles and reaction events. Among these exact methods, only ER-leap has been demonstrated to have an asymptotically sublinear (roughly 2/3 power of SSA) simulation time as a function of the number of reaction events for a regular family of simulation problems, namely two exactly solvable networks (Galton-Watson and dimerization) in a test suite for stochastic simulation algorithms.

Future work includes the hybridization of the present ER-algorithm with techniques from other exact simulation algorithms that more directly address scaling with the number of reaction channels, as well as improvements in the extension of the ER-algorithm to spatially dependent stochastic simulations. The numerical experiments of Section 3.5, along with previous work such as the use of tau-leap [21] and R-leap [22] in spatial models,

show the feasability of spatial stochastic simulations but do not, we think, exhaust the avenues for their acceleration.

Software for the ER-leap algorithm is provided at http://computableplant.ics.uci.edu/erleap.

4.1 Acknowledgements

EM wishes to thank ETH and Prof. Joachim Buhman for supporting a visit to ETH Zürich. Other support was provided by NSF's Frontiers in Biological Research (FIBR) program award number #EF-0330786, and NIH P50-GM76516.

References

[1] Gillespie, D. T. (1976). *Exact Stochastic Simulation of Coupled Chemical Reactions*. Journal of Computational Physics, **22**, 403–434.

[2] Gillespie, D. T. (2001). *Approximate Accelerated Stochastic Simulation of Chemically Reacting Systems*. Journal of Chemical Physics, **115**, 1716–1733.

[3] Chatterjee, A., Mayawala, K., Edwards, J. S., & Vlachos, D. G. (2005). *Time accelerated Monte Carlo simulations of biological networks using the binomial tau -leap method*. Bioinformatics, **21**(9), 2136–2137.

[4] Cao, Y., Gillespie, D. T., & Petzold, L. R. (2005). *Avoiding Negative Populations in Explicit Tau Leaping*. Journal of Chemical Physics, **123**, 054104–054112.

[5] Cao, Y., Gillespie, D. T., & Petzold, L. R. (2006). *Efficient Stepsize Selection for the Tau-Leaping Method*. Journal of Chemical Physics, **124**, 044109–144109-11.

[6] E, W., Liu, D., & vanden-Eijnden, E. (2005). *Nested stochastic simulation algorithm for chemical kinetic systems with disparate rates*. J. Chem. Phys., **123**, 194107.

[7] Samant, A., & Vlachos, D. G. (2005). *Overcoming stiffness in stochastic simulation stemming from partial equilibrium: A multiscale Monte Carlo algorithm*. Journal of Chemical Physics, **123**, 144114.

[8] Salis, H., & Kaznessisa, Y. N. (2005). *An equation-free probabilistic steady-state approximation: Dynamic application to the stochastic simulation of biochemical reaction networks*. Journal of Chemical Physics, **123**, 214106.

[9] Cao, Y., Gillespie, D. T., & Petzold, L. R. (2006). *The slow-scale stochastic simulation algorithm*. Journal of Chemical Physics, **122**, 014116.

[10] Slepoy, A., Thompson, A. P., & Plimpton, S. J. (2008). *A constant-time kinetic Monte Carlo algorithm for simulation of large biochemical reaction networks*. Journal of Chemical Physics, **128**, 205101.

[11] Auger, A., Chatelain, P., & Koumoutsakos, P. (2006). *R-leaping: Accelerating the stochastic simulation algorithm by reaction leaps*. Journal of Chemical Physics, **125**, 084103.

[12] van Kampen, N. G. (1981). *Stochastic Processes in Physics and Chemistry*. North-Holland.

[13] Yosiphon, G., & Mjolsness, E. (2008). *Towards the Inference of Stochastic Biochemical Network and Parameterized Grammar Models*. In N. Lawrence et al., eds., Learning and Inference in Computational Systems Biology (title with MIT Press), to appear 2009. **UCI ICS TR # 08-07**. http://computableplant.ics.uci.edu/papers .

[14] Wilkinson, D. J. (2006). *Stochastic Modelling for Systems Biology*. Boca Raton: Chapman and Hall/CRC.

[15] Mjolsness, E., & Yosiphon, G. (2006). *Stochastic process semantics for dynamical grammars*. Annals of Mathematics and Artificial Intelligence, **47**(Issue 3-4), 329–395.

[16] von Neumann, J. (1951). *Various Techniques Used in Connection with Random Graphs*. In A. S. Householder, G. E. Forsythe & H. H. Germond (Ed.), *Monte Carlo Method* (pp. 36–38). Washington: US Government Printing Office.

[17] Evans, T. W., Gillespie, C. S., & Wilkerson, D. J. (2008). *The SBML discrete stochastic models test suite*. Bioinformatics, **24**(2), 285–286.

[18] Wang, H., & Li, Q. (1998). *Master equation analysis of deterministic chemical chaos*. J. Chem. Phys., **108**(18), 7555–7559.

[19] Gibson, M. A., & Bruck, J. (2000). *Efficient exact stochastic simulation of chemical systems with many species and many chanels*. Journal of Physical Chemistry, **104**(9), 1876–1889.

[20] Riedel, M. D., & Bruck, J. (2006). *Exact Stochastic Simulation of Chemical Reactions with Cycle Leaping*. http://paradise.caltech.edu/bruck_etr.html

[21] Rossinelli, D., Bayati, B., & Koumoutsakos, P. (2008). *Accelerated stochastic and hybrid methods for spatial simulations of reaction–diffusion systems*. Chemical Physics Letters, **451**, 136–140.

[22] Bayati, B., Chatelain P., & Koumoutsakos, P. (2008). *Multiresolution stochastic simulations of reaction-diffusion processes*. Physical Chemistry Chemical Physics, **10**,5963-5966.