UNIVERSITY OF CALIFORNIA,
IRVINE

Exact and Hierarchical Reaction Leaping: Asymptotic Improvements to the Stochastic Simulation Algorithm

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Science

by

David Orendorff

Dissertation Committee:
Professor Eric Mjolsness, Chair
Professor Padhraic Smyth
Professor Qing Nie

2012

DEDICATION

To my family...

Contents

List of Figures

List of Tables

ACKNOWLEDGMENTS

Foremost, I would like to thank my wife for her unabating support, patience and editorial commentary. Without her this thesis may not have come to fruition. I would like to thank my advisor Eric Mjolsness for his expertise and mentorship during these past six years. He has done his job tirelessly, and I am extremely grateful to have worked with him. My thanks extend to my officemate, Todd Johnson, for useful discussions and for showing me that finishing is possible. Furthermore, I am grateful for my peer lunch group including, but not limited to, Lars Otten, Jimmy Folds, Drew Frank, Darren Davis, Chris DuBois, Matt Kayala and Levi Boyles for the interesting discussions. I am thankful for the Bayes Allstars. I am thank for my family and their heartfelt support. I am thankful for exemplary teachers throughout my formative years, including Sherry Leake and Sherry Bigham, for giving me the confidence to pursue difficult paths. I am thankful for our collaborators from ETH Zürich: Petros Koumoutsakos and Basil Bayati.

I am grateful for the Thesis committee. Padhraic Smyth has proven to be an outstanding educator and ally. My gratitude extends to Qing Nie for his commitment and support through the P50 grant. I am thankful for Xiaohui Xie and Max Welling for challenging me though the advancement process to become a more knowledgeable candidate. I gratefully acknowledge the American Institute of Physics for allowing me to incorporate some of their published material into this thesis. Finally, I am grateful for the support provided by NSF's Frontiers in Biological Research (FIBR) program award number #EF-0330786, and NIH P50-GM76516.

CURRICULUM VITAE

David Orendorff

EDUCATION

Doctor of Philosophy in Computer Science	**2012**
University of California, Irvine	*Irvine, California*
Master of Info and Computer Sci	**2008**
University of California, Irvine	*Irvine, California*
B.S. in Computer Science, with Departmental Distinction	**2006**
University of Washington	*Seattle, Washington*
B.A. in Mathematics	**2006**
University of Washington	*Seattle, Washington*

RESEARCH EXPERIENCE

Graduate Research Assistant	**2006–2012**
University of California, Irvine	*Irvine, California*
Intern	**June - September 2011**
Google	*Irvine, California*
Intern	**June - September 2010**
Google	*Irvine, California*
Research Assistant	**June-December 2005**
UrbanSim, University of Washington	*Seattle, Washington*
Intern	**Summer 2004**
Institute for Systems Biology	*Seattle, Washington*

REFEREED JOURNAL PUBLICATIONS

E. Mjolsness, D. Orendorff, P. Chatelain, and P. Koumoutsakos. An exact accelerated stochastic simulation algorithm. *The Journal of chemical physics*, 130(14):144110, 2009.

ABSTRACT OF THE DISSERTATION

Exact and Hierarchical Reaction Leaping:
Asymptotic Improvements to the Stochastic Simulation Algorithm

By

David Orendorff

Doctor of Philosophy in Computer Science

University of California, Irvine, 2012

Professor Eric Mjolsness, Chair

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. A second algorithm, "HiER-leap", is derived using some of the same principles used in the ER-leap derivation. HiER-leap utilizes a hierarchical organization of reaction channels into tightly coupled "blocks". Large portions of inter-block sampling may be done in parallel. An accept/reject step is used to synchronize across blocks. This method scales well when many reaction channels are present and has desirable asymptotic properties. The algorithm is exact, parallelizable and offers a significant speedup over SSA and ER-leap on certain problems. These two proposed algorithms offer a significant step towards efficient *in silico* modeling of entire organisms.

Chapter 1

Introduction

The world today is being assailed by a slew of problems: climate change, energy shortages, diseases, food shortage and overpopulation to name a few. These challenges touch all of humanity in one way or another. Without intervention, any one of these problems could have cataclysmic results.

Scientific inquiry represents a principled methodology to understand and eventually mitigate many of these global scale technical issues. Biology, in particular, has great promise for solving some of the more difficult problems. For example, the work of [3] rewrites some of the genetic circuitry in yeast to produce ethanol suitable for use as a petroleum substitute. The work of [40] aims to engineer *S. cerevisiae* to inexpensively produces an anti-miliaria drug. The World Health Organization estimates that 33.9% of American adults are obese [1]. Obesity has been linked to noncommunicable diseases such as cardiovascular disease, some cancers, diabetes and musculoskeletal disorders [1]. Genetic links have been found to be correlated with obesity [27], but relatively little is known about the consequences of these genetic links. Personalized medicine suggests a drastically increased efficacy in treatment and prevention of most diseases. Finally, the future leveraging of biology extends to even more dramatic possibilities. Space colonization and terraforming will likely rely heavily on genetically modified organisms: oxygen producing bacteria could one day blanket Mars, and space travelers may have in their bodies proteins and enhanced repair mechanisms built to resist the harmful effects of solar radiation. In these ways, the fate of humanity is closely linked to our understanding of biology.

All of the aforementioned examples have in common the need to understand and manipulate complex biological systems. Over the previous decade high throughput assays have become commonplace [33]. These technologies will often produce hundreds or thousands of indicators, the underlying system mechanics of which are rarely directly inferable. Instead, computational learning and inference techniques must hypothesise the underlying mechanisms.

The biological models of tomorrow will have a complexity that will likely yield present inference algorithms practically intractable. Maximum likelihood inference algorithms such as [53] and Bayesian inference algorithms such as [41], have in common the need for a great many simulations in their inner loops. Thus simulation is a major bottleneck for the computational inference of biological mechanisms. Overcoming this obstacle is the focus of this thesis.

A popular method for modeling biological systems describes a system in terms of 'chemically reacting species'. These models are independent of the exact inference algorithms used. In fact, the model may represent different distributions depending on the assumptions one makes about the underlying system as will be shown in section 2.1.

When molecule counts are low for species, it can be useful to describe reactions as occurring stochastically. This assumption may be especially useful when modeling regulatory networks, where particular protein or mRNA quantities are often very low.

A great deal of research has focused on understanding the stochastic behavior of these systems. Around 1976, Daniel Gillespie developed the Stochastic Simulation Algorithm (SSA) to exactly sample these systems [20]. Since then, this line of research has branched into various directions in order to speed up this sampling.

Accelerating the SSA algorithm has proven to be difficult. It has been shown that general chemically reacting networks are Turing complete [48] with error that is arbitrarily close to 0. By the halting problem, this implies that general chemically reacting models cannot be solved analytically.

One class of acceleration methods utilizes approximations. For example, [19, 10, 32, 24, 28, 8] assume reaction propensities do not change over short time periods. [5, 58] assume propensities do not change while time delayed reactions occur.

Another more recent class of SSA acceleration algorithms uses parallel architectures to accelerate SSA. There has been work on the parallelization of SSA via GPUs [31][28, 30] and multicore CPUs [18]. Significant speedup has been achieved by concurrently sampling SSA trajectories. This speedup is important if many samples from the posterior distribution need to be sampled.

Additionally, there has been work in speeding up the sampling of single trajectories with GPUs using approximate methods [56].

However, multicore GPUs and CPUs have not been effectively used to speed up the sampling of one Chemical Master Equation trajectory exactly. Arguably, this becomes the dominant problem when extremely large systems are being studied. For example, an *E. coli*'s genome contains between 4,000 to 5,500 genes [7]. This fact, in addition to the large number of non-genetic species present, suggests that tens of thousands of species will needed to be present if an *E. coli* specimen is to ever be comprehensively modeled *in silico*. If eukaryotic cells are ever to be fully modeled, the complexity will further increase by orders of magnitude. Using current technology and a standard implementation of SSA, the time required to simulate one trajectory for any meaningful duration becomes prohibitive.

This thesis demonstrates novel ways to algorithmically accelerate SSA without approximation, as we have reported for the first time in [37], and additionally utilizes parallel architecture and a hierarchical divide-and-conquer algorithm strategy to accelerate serial sampling of SSA.

These algorithms are found by manipulating the Chemical Master Equation to more efficiently sample from SSA equivalent distributions. Results indicate an efficiency not achievable by existing techniques that scales well across variables pertinent to organism scale simulation.

Chapter 2

Related Work

2.1 Modeling Chemically Reacting Species

A great deal of effort has been put into understanding the dynamics of chemically reacting species. Understanding the properties of reacting systems elucidates the study of chemistry, biochemistry, many components of intercellular processes, and microbiology.

For chemically reacting systems, we define a set of species types $\{C_a\}$, the amount of the corresponding species in solution as $\{C_a\}$, and set of reactions $\{R_j\}$. Each reaction takes a set of input species and instantaneously transforms these species into a set of output species. For example,

$$R_1 : C_3 \longrightarrow C_1$$

defines a process that converts one molecule of type C_3 into one of type C_1. A real-world example which creates water is $H^+ + OH^- \longleftrightarrow H_2O$. The general form, which also requires a kinetic rate, of the k^{th} reaction occurring with rate ρ_k is

$$R_k = \{m_i^k C_i\} \longrightarrow \{m'^k_j C_j\}, with\ \rho_k \tag{2.1}$$

where m_i^k and m'^k_j are the stoichiometries for the input and output of the i^{th} species respectively in the k^{th} reaction channel. The stoichiometric constants will be reexamined in section 3.2.2. The rate at which the reaction occurs is proportional to the rate ρ_k times the combinatorial number of ways the reactants may combine.

Assumptions are made which will determine the mathematical content of the resulting model.

2.1.1 Differential Equations

In the limit of a large number of reacting molecules, the law of mass action applies in differential form [25]. Instead of modeling individual molecules we look at concentrations $\{[c_a]\}$. The instantaneous rate for each reaction is the product of the input concentrations multiplied by the rate constant. The reactant and product concentrations change accordingly. This gives rise to a system of differential equations that can be solved analytically and/or numerically. For example, consider the chemically reacting system defined by

$$\begin{aligned} &R_1 : C_1 \longrightarrow C_1 + C_2,\ with\ \rho_1 \\ &R_2 : C_1 + C_2 \longrightarrow C_2,\ with\ \rho_2\ . \end{aligned} \tag{2.2}$$

The expected concentrations at time t can be found by solving for the concentrations given the differential equations,

$$\begin{aligned}\frac{d[c_1]}{dt} &= -\rho_2[c_1][c_2]\\ \frac{d[c_2]}{dt} &= \rho_1[c_1]\ ,\end{aligned}$$

and initial conditions.

A more thorough treatment of the system calls for the consideration of stochastic effects.

2.1.2 Stochastic Differential Equations

Stochastic differential equations (SDEs) [22] begin to address the inherent stochasticity of the system by adding random variables to the differential equation terms. For example, the system in 2.1.1 could be modeled with the stochastic differential equations in the form of the Langevin equation,

$$\begin{aligned}\frac{d[n_1]}{dt} &= -[n_1][n_2]\rho_2 + \eta_{\sigma_1}\sqrt{[n_1][n_2]\rho_2}\\ \frac{d[n_2]}{dt} &= [n_1]\rho_1 + \eta_{\sigma_2}\sqrt{[n_1]\rho_1}\ ,\end{aligned}$$

that add zero mean independent Guassian random variables, η_{σ_k}, with standard deviation σ.

SDEs may be solved numerically [36] although methods to do so are still an active area of research.

2.1.3 Master Equation

A more thorough treatment of the stochasticity in chemically reacting systems utilizes the master equation [50]. The master equation (ME) is a general formulation describing the continuous-time, discrete-state evolution of a random process.

We describe a state space, $\mathcal{J}$, where individual states are indexed by J. Furthermore, we define the column vector $P(t)$, indexed by $P_i(t)$, as the instantaneous probability at time t of being in state I_i.

$$P_i(t) = Prob(J|t)$$

Therefore, J and $P(t)$ are equinumerous.

Finally, we define a $|\mathcal{J}| \times |\mathcal{J}|$ matrix $\mathbf{A}(t)$ as the instantaneous 'flow' of probability between states. $\mathbf{A}_{jk}(t)$ is the instantaneous rate of going from J to K. This yields the time evolution equation:

$$\frac{dP(t)}{dt} = \mathbf{A}(t) \cdot P(t). \tag{2.3}$$

If $\mathbf{A}(t) = \mathbf{A}$ is constant for all t, this is a continuous-time, discrete-state Markov process and sometimes called a *kinetic scheme*. The waiting time between state transitions will be exponentially distributed. A sampled sequence of state and transition time pairs $\{(J, \tau_k)\}$ is called a *trajectory*. For the duration of this thesis, we will only consider constant $\mathbf{A}$.

Chemical Master Equation (CME)

When applying the ME to chemically reacting systems, we assume the molecules are homogenously distributed and internally at thermal equilibrium [21]. The molecules collide often. However, a reaction event occurs only if the molecules collide at the correct orientation and speed. Therefore, reactions occur much less often than collisions.

The rate at which a reaction event occurs (the *propensity*) is proportional to the number of ways the molecules of the required species may interact. This value will be described precisely in section 3.2.2.

Simulating atom locations and electron orbitals [35, 17], as many protein folding prediction algorithms do, does very little to help us if we already know which reactions occur and the kinetic rates at which they occur. In fact, for efficiency we do not model the location of reactant molecules. Instead, we keep track of the counts of each molecule species. This treatment is only valid for well "well-stirred" solutions.

It is assumed that effects which may invalidate the homogeneity distribution assumption, such as diffusion and just-reacted-locality, occur much faster than the rate at which species reactions occur. However, intermediate reaction channels may be added if these assumptions are not reasonable. For example, if the rate of diffusion is important, we can model spatial systems [15]. Here a common approach is to parse the reactant medium into a D-dimensional grid of separate and identical reacting systems connected by diffusion 'reactions'.

To use equation 2.3 for a spatially homogenous chemically reacting system, we construct $\mathbf{A}$ using the following formulation [55]. Each state J is one of a combinatorial number of possible sorted z-tuples of species $C_1 \ldots C_N$. For example, $J = (n_1 =$

$4, n_2 = 2)$. Note that $\mathcal{J}$ may have infinite cardinality, but the following method still works. The transition rate of J to K is non-zero if and only if there is exactly one reaction which can transform the species of J into the species of K. If we call such a reaction R_v, the rate of transition from J to K will be the possible number of ways that the input species of R_v may interact, times the rate constant ρ_v.

Going back to equation 2.2, we can derive $P(t)$ and $\mathbf{A}$ as follows:

$$P(t) = \begin{bmatrix} P(J(t) = (0,0)) \\ P(J(t) = (1,0)) \\ P(J(t) = (0,1)) \\ P(J(t) = (2,0)) \\ P(J(t) = (1,1)) \\ P(J(t) = (0,2)) \\ P(J(t) = (3,0)) \\ P(J(t) = (2,2)) \\ \cdots \end{bmatrix},$$

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & -\rho_1 & 0 & 0 & \rho_1 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & -2\rho_1 & 0 & 0 & 0 & 0 & 2\rho_1 & \cdots \\ 0 & 0 & \rho_2 & 0 & -(\rho_1 + \rho_2) & 0 & 0 & \rho_1 & 0 & \\ & & & \vdots & & & & & & \ddots \end{bmatrix}.$$

Notice the sparsity in $\mathbf{A}$. This property is common to chemically reacting systems and may be exploited for speedup purposes. Additionally, there will be at most two positive entries in each row. This is because there are two reaction channels and thus two viable state transitions. Finally, the negative diagonal terms make it so the rows add up to 0. This way probability will not be created or destroyed. The diagonal elements can be thought of as the waiting time rate for staying in state J.

Only for a few special models may $P(t)$ be solved analytically. By far the most used method to determine $P(t)$ is by sampling trajectories many times. Statistics are then accrued for $P(J|t)$ until a sufficient amount of confidence in the measurement is established. Existing methods to do so exactly (section 2.2) or with some error (section 2.3.1, section 2.3.2) are described in the following sections.

2.2 SSA

The Stochastic Simulation Algorithm (SSA) [20] is a widely used method for simulating trajectories of chemical reaction networks using a CME formulation. SSA executes every reaction event and provides an accurate view of the system dynamics, although the computational cost of doing so is high compared to the corresponding mass action differential equations.

The SSA samples a trajectory of the Chemical Master Equation (CME) 2.3 by repeatedly sampling the next reaction channel to fire and the amount of time τ that passes between reaction events. This process is seen in algorithm 2.1.

2.3 SSA Acceleration Techniques

A number of algorithms have been proposed for the acceleration of the SSA. Some of these do so at the expense of accuracy [19, 10, 32, 24, 28, 8, 5, 58, 37]. The τ-*leaping* algorithm [23] and its recent variants [14, 11, 13] simulate leaps over several

Algorithm 2.1 Stochastic Simulation Algorithm

```
Require: R reaction channels that involve only state C.
Require: $n_i \geq 0 | i = 1 \ldots |\mathbf{s}|$. $T \geq 0$.
Ensure: Returns sample of n after ‘evolving’ for duration $T$.
  function SSA(R, n, $T$)
      $t \leftarrow 0$
      while $t \leq T$ do
          $\mathbf{a} \leftarrow$ CalculatePropensities($\mathbf{R}, \mathbf{n}$)
          $\tau \leftarrow$ Exponential(Sum($\mathbf{a}$))
          $r \leftarrow$ Discrete ($\mathbf{a}$)                ▷ Same as Multinomial($\mathbf{a}, k = 1$)
          $t \leftarrow t + \tau$                          ▷ Update time and state.
          $\mathbf{n} \leftarrow$ ExecuteReaction($r$, $\mathbf{R}$, $\mathbf{n}$)
      end while
      return n
  end function
```

The stochastic simulation algorithm was the first popularized method to exactly sample from the CME. The **While** loop executes once per reaction event.

reaction events during a preselected time increment. Further developments include multiscale stochastic simulation algorithms such as “Nested Stochastic Simulation” [54], the multiscale methods [44] and [43], and the “slow-scale stochastic simulation” algorithm [12]. A different method finds cycles during simulation and skips over them to accelerate SSA [39]. Another acceleration method [47] uses rejection sampling to achieve constant time scaling with the number of reaction channels; this differs from the 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 [4] which proposes the simulation of preselected numbers of reaction firings that occur over time intervals whose duration is then 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.

Multicore CPU and GPU architectures have additionally been used to speed up SSA [31, 28, 30, 28, 18].

Those acceleration methods most relevant to our work will be discussed in the following sections.

2.3.1 $\tau-$leaping

The first popularized attempt to speed up SSA via approximation was called $\tau - leap$. This algorithm accelerates SSA by assuming stationarity during a predetermined interval τ. It is assumed that $P(J|t + d\tau, K(t))$ is constant for $d\tau = 0 \ldots \tau$. Multiple reactions events can then be sampled from a Poisson process without recalculating propensities after every channel is sampled. After time is progressed to $t' = t + \tau$, all the sampled reaction channels are simultaneously fired, the state is updated and propensities recalculated. When τ is large a greater speedup is achieved at the cost of accuracy. As $\tau \to 0$ this becomes an SSA-equivalent algorithm in accordance with equation 2.3.

The stationarity assumption breaks down more severely when species counts are low. This is because small changes in species molecule counts can drastically change event propensities. This becomes a serious problem when molecule counts become negative, as they may if the propensity is overestimated as positive when it should be zero. Numerous techniques have since been proposed to allow τ-leap to work under these conditions [11]. However, acceleration methods derived from τ-leap remain inexact.

2.3.2 R-leaping

R-leaping [4] is similar to τ-leap in that stationarity is assumed for a predefined interval. However, instead of time, a constant number of L reaction events occur between state update and propensity recalculation. See algorithm 2.2 for pseudocode.

Delta time is sampled as a sum of L exponentially distributed random variables with rate equal to the total propensity of the system. This distribution is equivalent to an Erlang distribution or equivalently a *Gamma* distribution with 'shape' parameter being an integer. Finally, the sampled reaction channels come from a multinomial distribution in which the probability of choosing reaction channel k is equal to its propensity normalized by the sum of all reaction channel propensities.

Like τ-leap, the exactness in R-leaping increases as we decrease its 'leaping' parameter. In this case when $L = 1$ R-leaping is equivalent to SSA.

A strong understanding of the *R-leaping* method [4] will facilitate the understanding of the algorithms proposed in chapters 3 and 4. These novel algorithms are based on R-leaping.

Algorithm 2.2 R-leaping Algorithm

```
Require: $\mathbf{R}$ reaction channels that involve only states $\mathbf{C}$.
Require: $n_i \geq 0 | i = 1 \ldots |\mathbf{n}|$. $T \geq 0$. $L \geq 1$.
Ensure: Returns (approximate) sample of $\mathbf{n}$ after 'evolving' for time duration $T$.
  function RLeaping($\mathbf{R}$, $\mathbf{n}$, $T$, $L$)
      $t \leftarrow 0$
      while $t \leq T$ do
          $\mathbf{a} \leftarrow$ CalculatePropensities($\mathbf{R}, \mathbf{n}$)
          ▷ Equivalent to sampling $L$ exponential random variables with $\lambda =$Sum($\mathbf{a}$)
          $\tau \leftarrow$ RandomErlang(Sum($\mathbf{a}$), $L$)
          $\mathbf{r} \leftarrow$ Multinomial ($\mathbf{a}, L$)
          $t \leftarrow t + \tau$                                   ▷ Update time and state.
          $\mathbf{n} \leftarrow$ ExecuteReactions($\mathbf{r}$, $\mathbf{R}$, $\mathbf{n}$)
      end while
      return n
  end function
```

The R-leaping algorithm is similar to SSA. The difference is that L reaction events are sampled per iteration. Since propensities do not need to be calculated during these L steps, R-leaping can efficiently sample the L reactions from a Multinomial.

2.3.3 SSA on Parallel Hardware

Another class of SSA acceleration methods use parallel hardware.

Current consumer level GPUs have hundreds of cores. This results in a new form of inexpensive desktop supercomputer many times faster than a CPU of comparable cost. Petzold et. al. [31] have had success speeding the simultaneous sampling of *many* SSA trajectories by an order of 200x. This work has helped spur a flowering of GPU SSA acceleration research. The work of Jenkins et. al. [28] offers a refined toolkit to produce ensembles of CME sampled trajectories. This software works with MPI or, for the direct method, simulates on GPUs using CUDA and here averages about a 30x speedup [28]. This work [28] does not provide any new algorithms, but instead optimizes existing ones as well as simulates ensembles on parallel architecture. Furthermore, the work of [56] has implemented $\tau - leaping$ on the GPU with some success.

The work of [46] interestingly aims to speedup single kinetic Monte Carlo trajectory sampling using multiple cores with MPI. However, their speedup comes at a minor loss of accuracy, although their methods find ever increasing acceleration when the number of processors goes up [45]. Their work simulates reaction on a lattice (spatial grid) and uses this as a basis for parallelization. In a similar way, our algorithm of chapter 4 distributes the sampling of reaction channels based on spatial location.

Chapter 3

ER-Leap

3.1 Introduction

We present a stochastic simulation algorithm which, similar to R-leap, accelerates SSA by executing multiple reactions per algorithmic step, but 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. Unlike R-leap or any previous algorithms, ER-leap 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 [14] and modified tau-leap [11]. Since ER-leap is exact, it intrinsically avoids this potential pitfall; stochastic moves to negative reactant states have exactly zero propensity 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 upper and lower bounds on the SSA propensities and therefore probability distributions.

The ER-leap algorithm was first presented in [37], which this chapter follows very closely. All ER-leap computer experiments were first down by the present author. Confirmatory computer experiments were done by P. Chatelain, coauther of [37] and [4].

This chapter is organized as follows: in section 3.2 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 3.2.5 . In section 3.3 we report on a series of numerical experiments designed to evaluate the accuracy and speedup of the ER-leap algorithm. In section 3.4 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.

3.2 Theory

This section is organized as follows: section 3.2.1, section 3.2.2, and section 3.2.3 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 3.2.4. 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 3.2.5.

3.2.1 Notations

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

$$\{m^r_a C_a\} \longrightarrow \{m'^r_a C_a\} \quad \textbf{with reaction rate } \rho_r \tag{3.1}$$

Here $\boldsymbol{m}^r = [m^r_a]$ 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.

We introduce the following notations. The definition of a version of the indicator function $\mathbf{1}$ from Boolean values to integers is:

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

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

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

The function $V = \text{diag}(\boldsymbol{v})$ turns a d-dimensional vector $\boldsymbol{v}$ into a $d \times d$ square matrix V with components $V_{ij} = \delta_{ij} v_i$, 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 define 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 $\{x|P(x)\}$ 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: $[x(i)|P(x(i),i)||i \in \mathcal{I}]$ 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 $\mathbf{n}$.

3.2.2 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 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 [50], 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 [50] for such just-reacted state probabilities follows from taking k to be a discrete time coordinate, and can be written [57]:

$$\Pr(I, t|J, k) \approx \sum_K \int_0^t d\tau \Pr(I, \tau|K, 1) \Pr(K, t-\tau|J, k-1) \tag{3.2}$$

A key quantity in this equation is the "kernel" $\Pr(I, \tau | K, 1)$: 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 kernel also provides the linear weights that advance the quantity $\Pr(K, t - \tau | J, k - 1)$, which 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 kernel the conditional distribution

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

using notation similar to that of Ref. 57. 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:

$$\mathcal{W}(I, t' | J, t) \approx \hat{W}_{I,J} \exp(-\,(t' - t)\, D_{JJ}) \mathbf{1}\,(t' \geqslant t)\,. \tag{3.3}$$

with

$$D = \text{diag}(\boldsymbol{h} \cdot \hat{W}) \tag{3.4}$$

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 Ref. 55 and Ref. 57. 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 exponential term governs the distribution of waiting times between reaction events, as in the SSA [14, 11, 13] and R-leap [4] algorithms. The I'th component of the vector $\boldsymbol{h} \cdot \hat{W}$, which is defined as D_{II}, is the the total probability per unit time for the system to leave state I. (In many papers the summed reaction rate D_{II} is denoted as $a_0(\boldsymbol{n})$ instead.)

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

$$\Pr(I,t|J,k) \approx \sum_K \int_{-\infty}^{\infty} d\tau \mathcal{W}(I,t|K,t-\tau) \Pr(K,t-\tau|J,k-1) \tag{3.5}$$

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

$$\Pr(.|J,k) \equiv \mathcal{W} \circ \Pr(.|J,k-1) \tag{3.6}$$

where the matrix-vector inner product $\circ$ is both a sum over states and an integral over all times t, as in equation 3.5, and where

$$\mathcal{W} = \hat{W} \exp(-\Delta t D) \mathbf{1} \, (\Delta t \geqslant 0) \, . \tag{3.7}$$

Equation 3.7 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 3.6 and equation 3.7, after k reaction events,

$$\Pr(.|J,k) = \mathcal{W}^k \circ \Pr(.|J,0) = \left[\hat{W} \exp(-\Delta t D) \mathbf{1} \, (\Delta t \geqslant 0) \right]^k \circ \Pr(.|J,0) \tag{3.8}$$

This expression is in accord with, for example, Theorem 10.1 of Ref. 55.

The aim of the SSA algorithm is to sample from the distribution $\Pr(I, t|J, k)$. Equation 3.6 may be taken as a concise statement of a single 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 $D\exp(-\Delta tD)\mathbf{1}(\Delta t \geqslant 0)$, evaluated at state J. These two sampling steps are alternated and iterated k times as in equation 3.8.

To derive D and $\mathcal{W}$, and therefore (by equation 3.7) 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 3.1, defining the net stoichiometry

$$\Delta m_a^r = m'^r_a - m_a^r,$$

the usual mass-action assumption for stochastic reactions corresponds to

$$\hat{W}(\boldsymbol{n}'|\boldsymbol{n}) = \sum_r \hat{W}^{(r)}(\boldsymbol{n}'|\boldsymbol{n}) \quad \text{and} \quad \hat{D}(\boldsymbol{n}'|\boldsymbol{n}) = \sum_r \hat{D}^{(r)}(\boldsymbol{n}'|\boldsymbol{n})$$

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:

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

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

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

(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 [38] which also uses notation similar to that used here.) If we define $W = \hat{W} - D$, SSA dynamics simulates trajectories [50] drawn from the solution to the Master Equation, $dp/dt = W \cdot p$.

3.2.3 Upper and Lower Bounds

In order to derive a new simulation algorithm, equivalent to SSA, using rejection sampling [51], we now seek simplified upper and lower bounds on the probability rate $\hat{W}_{I,J}^{(r)} \exp(-\Delta t D_{JJ})$ (from equation 3.7) 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[4]. As we will see, this essentially comes down to bounding each combinatorial factor $n_a!/(n_a - m_a^r)!$ 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

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

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 3.3. The products $\rho_r F_{\boldsymbol{n}}^{(r)}$ are usually called "propensity functions" denoted $a_r(\boldsymbol{n})$ for all R reaction channels:

$$\begin{aligned} a_r(\boldsymbol{n}) &\equiv \rho_r F^{(r)}_{\boldsymbol{n}}, \\ a_0(\boldsymbol{n}) &\equiv \sum_{r=1}^{R} a_r(\boldsymbol{n}) \end{aligned} \tag{3.9}$$

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 3.9. 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:

$$n_a + l\text{min}_r \{\Delta m^r_a\} \leqslant n'_a \leqslant n_a + l\text{max}_r \{\Delta m^r_a\} \tag{3.10}$$

The corresponding upper and lower bounds $\tilde{F}_{IJ}$ and $\underset{\sim}{F}_{IJ}$ 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

$$\begin{aligned} \underset{\sim}{F}^{(r)}_{\boldsymbol{n},l} &\equiv F^{(r)}_{[n_a + l\text{min}_r\{\Delta m^r_a\}||1\leqslant a\leqslant A]} \\ \tilde{F}^{(r)}_{\boldsymbol{n},l} &\equiv F^{(r)}_{[n_a + l\text{max}_r\{\Delta m^r_a\}||1\leqslant a\leqslant A]} \end{aligned} \tag{3.11}$$

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

$$\begin{aligned}
\hat{S}^{(r)}_{\boldsymbol{n}',\boldsymbol{n}} &= \mathbf{1}\left(\hat{W}^{(r)}_{\boldsymbol{n}',\boldsymbol{n}} > 0\right) \qquad \in \{0,1\} \\
&= \left(\prod_{\{a|m^r_a \neq 0\}} \mathbf{1}\left(n_a \geqslant m^r_a\right)\right)\left[\prod_{\{a|\Delta m^r_a \neq 0\}} \delta(n'_a - n_a - \Delta m^r_a)\right]
\end{aligned}$$

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

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

$$\sum_I \hat{S}^{(r)}_{I,J} = 1. \tag{3.12}$$

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

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

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

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

where

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

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 3.3. To this end, note from equation 3.4 that

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

Then

$$\begin{aligned}
&R_{J|Kl} = 1 \quad \Rightarrow \\
&-\tilde{D}_{Kl} = -\sum_{r} \sum_{I'} \tilde{W}^{(r)}_{I',J|Kl} \leqslant -D_{JJ} \leqslant -\sum_{r} \sum_{I'} \underset{\sim}{W}^{(r)}_{I',J|Kl} = -\underset{\sim}{D}_{Kl}
\end{aligned}$$

where

$$\begin{aligned}
\underset{\sim}{D}_{Kl} &\equiv \sum_{r} \rho_r \underset{\sim}{F}^{(r)}_{K,l} \\
\tilde{D}_{Kl} &\equiv \sum_{r} \rho_r \tilde{F}^{(r)}_{K,l}
\end{aligned} \tag{3.13}$$

Thus, assuming $R_{J|Kl} = 1$ and $\Delta t \geqslant 0$, upper and lower bounds on the elements of the Markov process $\mathcal{W}$ given by equation 3.3 are determined as follows:

$$\rho_r \underset{\sim}{F}^{(r)}_{K,l} \hat{S}^{(r)}_{I,J} \exp(-\Delta t \tilde{D}_{Kl}) \leqslant \hat{W}^{(r)}_{I,J} \exp(-\Delta t D_{JJ}) \leqslant \rho_r \tilde{F}^{(r)}_{K,l} \hat{S}^{(r)}_{I,J} \exp(-\Delta t \underset{\sim}{D}_{Kl}) \ . \tag{3.14}$$

These desired bounds on reaction probability rates $\hat{W}^{(r)}_{I,J} \exp(-\Delta t D_{JJ})$ follow from the simple bounds of equation 3.10 on n'_a as a function of n_a and l.

3.2.4 Exploitation of probability bounds

We now use the bounds of equation 3.14 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 3.2.5, followed by computational experiments in section 3.3. In this section we perform the required calculations to derive the key update equations.

Rejection sampling

Rejection sampling [51] 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

$$P(x) < MP'(x)$$

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

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

and therefore also

$$P(x) = \int P'(x')dx' \int U(u)du \\ \left[\mathbf{1}\left(u < \frac{P(x')}{MP'(x')}\right) \cdot \delta(x - x') + \mathbf{1}\left(u \geqslant \frac{P(x')}{MP'(x')}\right) \cdot P(x)\right] \quad (3.15)$$

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 3.15 is as follows (where "//" introduces a comment):

```
while not accepted {
    sample $P'(x)$ and $U(u)$;    // $P'(x)$ only approximates $P(x)$
    compute $\text{Accept}(x) = P(x)/(MP'(x))$;    // acceptance probability
    if $u < \text{Accept}(x)$ then accept $x$;
}  // now $P(x)$ is sampled exactly
```

What is essential in applying this algorithm is to find a provable strict upper bound $\tilde{P}(x) = MP'(x)$ 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 $\underset{\sim}{A}(x)$ such that

$$0 \leqslant \underset{\sim}{A}(x) \leqslant \text{Accept}(x) \equiv P(x)/\left(MP'(x)\right) < 1. \tag{3.16}$$

Then

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

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$(x) = P(x)/MP'(x)$;    // expensive
        compute $Q(x) = (\text{Accept}(x) - \underset{\sim}{A}(x))/(1 - \underset{\sim}{A}(x))$;    // $\underset{\sim}{A}(x) < 1 \Rightarrow 1 - \underset{\sim}{A}(x) \neq 0$
        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.

Equivalent Markov process

In this section we will use algebraic manipulations to transform the formula for SSA (equation 3.8) into an equivalent form (equation 3.18) 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 3.8)

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

From equation 3.14,

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

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

$$
\begin{aligned}
&R_{J|KL} = 1 \quad \Rightarrow \\
&\left[\prod_{k=L-1\searrow 0} \hat{W} \quad \exp(-\tau_k D)\right]_{I_L,I_0} \\
&= \sum_{\{I_k|k=1..L-1\}} \left[\prod_{k=L-1\searrow 0} \hat{W}_{I_{k+1},I_k} \quad \exp(-\tau_k D_{I_k,I_k})\right] \\
&= \sum_{\{I_k|k=1..L-1\}} \sum_{\{r_k\}} \prod_{k=L-1\searrow 0} \left[\left(\rho_{r_k} \hat{S}^{(r_k)}_{I_{k+1},I_k} \left(\frac{F^{(r_k)}_{I_k}}{\tilde{F}^{(r_k)}_{I_0,L-1}}\right) \tilde{F}^{(r_k)}_{I_0,L-1}\right)\right] \\
&\times \exp(-\tau_k(D_{I_k,I_k} - \underset{\sim}{D}_{I_0 L-1})) \exp(-\tau_k \underset{\sim}{D}_{I_0 L-1}) \\
&= \sum_{\{r_k|k=1..L-1\}} \sum_{\{I_k\}} \left[\prod_{k=L-1\searrow 0} \hat{S}^{(r_k)}_{I_{k+1},I_k}\right] \left[\prod_{k=L-1\searrow 0} \rho_{r_k} \tilde{F}^{(r_k)}_{I_0,L-1}\right] \\
&\times \left[\prod_{k=L-1\searrow 0} \left(\frac{F^{(r_k)}_{I_k}}{\tilde{F}^{(r_k)}_{I_0,L-1}}\right)\right. \\
&\qquad\qquad \left.\times \exp(-\tau_k(D_{I_k,I_k} - \underset{\sim}{D}_{I_0 L-1}))\right] \exp\left(-\left(\sum_k \tau_k\right) \underset{\sim}{D}_{I_0 L-1}\right)
\end{aligned}
$$

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

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

$$
\begin{aligned}
&\left[\prod_{k=l-1\searrow 0} \hat{W} \quad \exp(-\tau_k D)\right]_{I_l,I_0} = \sum_{\{r_k|k=1..L-1\}} \left[\prod_{k=l-1\searrow 0} \rho_{r_k} \tilde{F}^{(r_k)}_{I_0,L-1}\right] \\
&\times \exp\left(-\left(\sum_k \tau_k\right) \underset{\sim}{D}_{I_0 L-1}\right) \\
&\qquad \times \left[\prod_{k=L-1\searrow 0} \left(\left(\frac{F^{(r_k)}_{I_k(\boldsymbol{r},I_0)}}{\tilde{F}^{(r_k)}_{I_0,L-1}}\right)\right) \exp(-\tau_k(D_{I_k(\boldsymbol{r},I_0),I_k(\boldsymbol{r},I_0)} - \underset{\sim}{D}_{I_0 L-1}))\right]
\end{aligned}
$$

Define new rule probabilities

$$p_{r|K,l} = \rho_r \tilde{F}^{(r)}_{K,l} / \tilde{D}_{Kl} \equiv \frac{\rho_r \tilde{F}^{(r)}_{K,l}}{\sum\limits_r \rho_r \tilde{F}^{(r)}_{K,l}}. \tag{3.17}$$

Then,

$$\begin{aligned}
&\left[\prod_{k=l-1 \searrow 0} \hat{W} \quad \exp\left(-\tau_k D\right) \right]_{I_l, I_0} = \\
&\quad \sum_{\{r_k | k=1..L-1\}} \left[\prod_{k=L-1 \searrow 0} p_{r_k | I_0, L-1} \right] \left(\tilde{D}_{I_0 L-1} \right)^l \exp\left(- \left(\sum_k \tau_k \right) \underset{\sim}{D}_{I_0 L-1} \right) \\
&\quad \times \left[\prod_{k=L-1 \searrow 0} \left(\left(\frac{F^{(r_k)}_{I_k(\boldsymbol{r}, I_0)}}{\tilde{F}^{(r_k)}_{I_0, L-1}} \right) \exp(-\tau_k (D_{I_k(\boldsymbol{r},I_0), I_k(\boldsymbol{r},I_0)} - \underset{\sim}{D}_{I_0 L-1})) \right) \right] \\
&\quad = \sum_{\{r_k | k=1..L-1\}} e_1(\boldsymbol{r}) e_2(\boldsymbol{r})
\end{aligned}$$

where

$$\begin{aligned}
e_1(\boldsymbol{r}) &\equiv \left[\prod_{k=L-1 \searrow 0} p_{r_k | I_0, L-1} \right] \left(\tilde{D}_{I_0 L-1} \right)^l \exp\left(- \left(\sum_k \tau_k \right) \underset{\sim}{D}_{I_0 L-1} \right) \\
e_2(\boldsymbol{r}) &\equiv \prod_{k=L-1 \searrow 0} \left(\left(\frac{F^{(r_k)}_{I_k(\boldsymbol{r}, I_0)}}{\tilde{F}^{(r_k)}_{I_0, L-1}} \right) \exp\left(-\tau_k \left(D_{I_k(\boldsymbol{r},I_0), I_k(\boldsymbol{r},I_0)} - \underset{\sim}{D}_{I_0 L-1} \right) \right) \right)
\end{aligned}$$

We define an arbitrary ordering "$\leqslant$" on the reaction types or channels indexed by r, so the reactions events are "sorted" by type iff $r_0 \leqslant r_1 \leqslant ... \leqslant r_{L-1}$. Let σ denote a permutation on L elements which we may apply to this ordering to get an unordered sequence of rules $\boldsymbol{r} = \{r_k | k = 0..L-1\}$. 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 $\sum\limits_{\{r_k|k=1..L-1\}} e_1(\boldsymbol{r})e_2(\boldsymbol{r})$ 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:

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

The number of r's taking each possible value $1...R$ is denoted $[s_1, \quad ... \quad s_R] = \boldsymbol{s}(r)$; 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:

$$s_r = \sum_{k=0}^{L-1} \delta(r_k - r), \text{which satisfies}$$
$$s_r \in \mathbb{N} \qquad \text{and} \qquad \sum_r s_r = L$$

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

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

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

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

$e_1(\boldsymbol{r})$ 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

$$\begin{aligned}\sum_{\{r_k|k=1..L-1\}} e_1(\sigma(\boldsymbol{r}))e_2(\sigma(\boldsymbol{r})) &= \sum_{\left\{\boldsymbol{s}|s_r\in\mathbb{N},\sum_r s_r=L\right\}} \tilde{e}_1(\boldsymbol{s}(\boldsymbol{r})) \sum_{\left\{\sigma|\sigma \text{ permutes} r\text{'s}|\boldsymbol{s}\right\}} e_2(\sigma(\boldsymbol{r}))\\ &= \sum_{\left\{\boldsymbol{s}|s_r\in\mathbb{N},\sum_r s_r=L\right\}} \tilde{e}_1(\boldsymbol{s}(\boldsymbol{r})) \left(\sum_{\left\{\sigma|\sigma \text{ permutes unequal } r\text{'s}|\boldsymbol{s}\right\}} e_2(\sigma(\boldsymbol{r}))\right)\\ &= \sum_{\left\{\boldsymbol{s}|s_r\in\mathbb{N},\sum_r s_r=L\right\}} \tilde{e}_1(\boldsymbol{s}(\boldsymbol{r})) \binom{L}{s_1 \quad \dots \quad s_R} \langle e_2(\sigma(\boldsymbol{r}))\rangle_{\left\{\sigma \text{ permutes unequal } r\text{'s}|\boldsymbol{s}\right\}}\end{aligned}$$

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

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

and we find

$$\begin{aligned}&\sum_{\{r_k|k=1..L-1\}} e_1(\sigma(\boldsymbol{r}))e_2(\sigma(\boldsymbol{r}))\\ &\qquad = \sum_{\left\{\boldsymbol{s}|s_r\in\mathbb{N},\sum_r s_r=L\right\}} \binom{L}{s_1 \quad \dots \quad s_R} \tilde{e}_1(\boldsymbol{s}(\boldsymbol{r}))\langle e_2(\sigma(\boldsymbol{r}))\rangle_{\left\{\sigma \text{ permutes } r\text{'s}|\boldsymbol{s}\right\}}\end{aligned}$$

Consequently,

$$\begin{aligned}&\left[\prod_{k=l-1\searrow 0} \hat{W} \quad \exp\left(-\tau_k D\right)\right]_{I_L,I_0} = \sum_{\left\{\boldsymbol{s}|s_r\in\mathbb{N},\sum_r s_r=L\right\}} \binom{L}{s_1 \quad \dots \quad s_R}\\ &\times \left[\prod_{r=1}^{R} \left(p_{r|I_0,L-1}\right)^{s_r}\right]\\ &\times \left(\tilde{D}_{I_0 L-1}\right)^l \exp\left(-\left(\sum_k \tau_k\right) \underset{\sim}{D}_{I_0 L-1}\right)\\ &\times \left\langle \left[\prod_{k=L-1\searrow 0} \left(\frac{F^{(r_k)}_{I_k(\sigma(\boldsymbol{r}),I_0)}}{\tilde{F}^{(r_k)}_{I_0,L-1}}\right) \exp(-\tau_k(D_{I_k(\sigma(\boldsymbol{r}),I_0),I_k(\sigma(\boldsymbol{r}),I_0)} - \underset{\sim}{D}_{I_0 L-1}))\right]\right\rangle_{\{\sigma|\boldsymbol{s}\}}.\end{aligned}$$

This can be decomposed into more elementary probability distributions:

$$\left[\prod_{k=L-1\searrow 0}\hat{W}\quad \exp\left(-\tau_k D\right)\right]_{I_L,I_0} = \frac{\left(\tilde{D}_{I_0 L-1}\right)^L}{\left(\underset{\sim}{D}_{I_0 L-1}\right)^L}$$
$$\times \sum_{\left\{\boldsymbol{s}|s_r\in\mathbb{N},\sum_r s_r=L\right\}} \text{Multinomial}(\boldsymbol{s}|\boldsymbol{p},L)$$
$$\times\,\text{Erlang}\left(\sum_k \tau_k|L,\underset{\sim}{D}_{I_0 L-1}\right)\text{UniformSimplex}(\boldsymbol{\tau};L)\,\text{Accept}(\boldsymbol{s},L,\boldsymbol{\tau}) \quad (3.18)$$

where

$$\text{Multinomial}(\boldsymbol{s}|\boldsymbol{p},L) = \begin{pmatrix} & L & \\ s_1 & \ldots & s_R\end{pmatrix}\left[\prod_{r=1}^{R}\left(p_{r|I_0,L-1}\right)^{s_r}\right], \text{with}$$
$$p_{r|I_0,L-1} = \frac{\rho_r \tilde{F}^{(r)}_{I_0,L-1}}{\sum\limits_r \rho_r \tilde{F}^{(r)}_{I_0,L-1}};$$

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

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

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

and the acceptance probability

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

where

$$P_\sigma = \left[\prod_{k=L-1 \searrow 0} \left(\frac{F^{(r_k)}_{I_k(\sigma(\boldsymbol{r}),I_0)}}{\tilde{F}^{(r_k)}_{I_0,L-1}} \right) \right] \exp \left(- \sum_k \tau_k (D_{I_k(\sigma(\boldsymbol{r}),I_0),I_k(\sigma(\boldsymbol{r}),I_0)} - \underset{\sim}{D}_{I_0 L-1}) \right). \tag{3.19}$$

From the definition of P_σ in equation 3.19 and the fact that $\tilde{F}$ and $\underset{\sim}{D}$ are bounds, it follows that $\text{Accept}(\boldsymbol{s}, l, \boldsymbol{\tau}) \leqslant 1$. Also, if $R_{J|KL-1} = 0$ (so that state J is not reachable from state K after $L-1$ steps of SSA) then $P_\sigma = 0$, so that equation 3.18 still agrees with equation 3.8 despite the restriction to $R_{J|KL-1} = 1$ stated in the foregoing calculation.

Thus, equation 3.18 provides an equivalent probability distribution and Markov process to equation 3.8.

Efficient rejection sampling algorithm

We now seek M and P' and $\underset{\sim}{A}(x)$ among the factors of equation 3.18. We can upper-bound and lower-bound P_σ of equation 3.19:

$$\underset{\sim}{P} \left(\boldsymbol{s}, \sum_k \tau_k, L \right) \leqslant P_\sigma \leqslant 1 \tag{3.20}$$

where

$$\underset{\sim}{P} \left(\boldsymbol{s}, \sum_k \tau_k, L \right) \equiv \left[\prod_{r=1}^{R} \left(\frac{\underset{\sim}{F}^{(r)}_{I_0,L-1}}{\tilde{F}^{(r)}_{I_0,L-1}} \right)^{s_r} \right] \exp \left(- \left(\sum_k \tau_k \right) \left(\tilde{D}_{I_0 L-1} - \underset{\sim}{D}_{I_0 L-1} \right) \right) \tag{3.21}$$

Note that $\underset{\sim}{P}$ does not depend on σ. This allows us to use rejection sampling [51] to transform samples of the bounding distribution

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

into samples of the target distribution

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

since the ratio $f(\boldsymbol{s},\boldsymbol{\tau})/g(\boldsymbol{s},\boldsymbol{\tau})$ is bounded above by $M = (\tilde{D}_{I_0 L-1}/\underset{\sim}{D}_{I_0 L-1})^L \geqslant 1$. $g(\boldsymbol{s},\boldsymbol{\tau})$ plays the role of $P'(x)$ in the rejection sampling algorithm of section 3.2.4, $f(\boldsymbol{s},\boldsymbol{\tau})$ plays the role of $P(x)$, and M has just been defined. This bound is independent of all randomly chosen variables $\boldsymbol{s}, t, \boldsymbol{\tau}, \sigma$ and just restores the probability otherwise lost in rejection sampling due to the $\text{Accept}(\boldsymbol{s},L,\boldsymbol{\tau})$ 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 3.2.4, we need to find a lower bound $\underset{\sim}{A}(x)$ for $\text{Accept}(\boldsymbol{s},l,\boldsymbol{\tau}) = \langle P_\sigma\rangle_{\{\sigma|\boldsymbol{s}\}}$. Fortunately $\mathrm{P}(\boldsymbol{s},\sum_{\mathrm{k}}\tau_{\mathrm{k}},\mathrm{L})$ 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:

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

$$Q_\sigma = \left(\frac{P_\sigma - \underset{\sim}{P}}{1-\underset{\sim}{P}}\right) \leqslant 1 \tag{3.22}$$

and thus

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

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 $\text{Accept}(\boldsymbol{\sigma}, L, \boldsymbol{\tau}) = \text{Accept}(\boldsymbol{s}, L, \boldsymbol{\tau}) \cdot \Pr(\boldsymbol{\sigma}|\boldsymbol{s})$, where $\Pr(\boldsymbol{\sigma}|\boldsymbol{s})$ is uniform, and apply accelerated rejection sampling to $f(\boldsymbol{s}, \boldsymbol{\tau}) \Pr(\boldsymbol{\sigma}|\boldsymbol{s})$ using the corresponding bounds $f(\boldsymbol{s}, \boldsymbol{\tau}) \Pr(\boldsymbol{\sigma}|\boldsymbol{s})$ for $P'(x)$ and P_σ for $\utilde{A}(x)$.

Algorithmically this expression can be sampled from as follows. First compute $\utilde{P}$. Then with probability $\utilde{P}$, accept the "current" candidate move determined by all the other distributions. In the relatively unlikely event (probability $1 - \utilde{P}$) that the move is not immediately accepted this way, we then draw a random σ given $\boldsymbol{s}$, 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 $\utilde{P}$ should be high. Pseudocode for the resulting algorithm will be presented in the next section.

3.2.5 Exact R-leap algorithm

We now assemble the ER-leap algorithm from the key update equations derived in previous sections: equation 3.11, equation 3.13, equation 3.17, equation 3.18, equation 3.21, equation 3.19, and equation 3.22.

Algorithm summary

We adapt the efficient rejection sampling algorithm of section 3.2.4, with the random variables $\boldsymbol{s}$, $\boldsymbol{\sigma}$ and $\boldsymbol{\tau}$, and the expressions for P, P', M and $\underset{\sim}{A}$ of section 3.2.4, into pseudocode (see algorithm 3.2.5) for the core of the resulting Exact R-leap algorithm.

Algorithm 3.1 ER-leap Algorithm

```
Require: $\mathbf{R}$ reaction channels that involve only states $\mathbf{C}$.
Require: $n_i \geq 0 | i = 1 \dots |\mathbf{n}|$. $T \geq 0$. $L > 1$.
Ensure: Returns sample of $\mathbf{n}$ after ‘evolving’ for time duration $T$.
  function ER-LEAP($\mathbf{R}$, $\mathbf{n}$, $T$, $L$)
      $t \leftarrow 0$
      while $t \leq T$ do
                                                   ▷ By equation 3.11 and equation 3.13.
          $\{\tilde{F}, \underset{\sim}{F}\} \leftarrow$ COMPUTEBOUNDS($n$, $L$, $R$)
                                              ▷ Equivalent to sampling $L$ exponential random...
          $\tau \leftarrow$ ERLANG($\tau; \underset{\sim}{D}_{(I_0,L)}, L$)                ▷ ...variables with $\lambda =$SUM($\mathbf{a}$)
          $\mathbf{s} \leftarrow$MULTINOMIAL($s; \left\{ p_i = \frac{\rho_i \widehat{F}^{r_i}_{(I_0 L)}}{\sum_{r'} \widehat{F}^{r'}_{(I_0 L)}} \right\}, L$)
          $z \leftarrow$ UNIFORMRANDOM(0,1)
                                              ▷ Compute by equation 3.21, a fast computation.
          $p \leftarrow$EARLYACCEPTANCEPROBABILITY($\tau, \tilde{D}_{(I_0,L)}, \underset{\sim}{D}_{(I_0,L)}), \mathbf{s}$)
          if $p \leq z$ then
                                     ▷ Early acceptance too small, calculate entire probability.
              $\sigma \leftarrow$ RANDOMPERMUTATION($s$)
                                              ▷ By equation 3.19; more involved computation.
              $p \leftarrow$ ACCEPTANCEPROBABILITY ($\tau$, $\tilde{D}_{(I_0,L)}$, $\underset{\sim}{D}_{(I_0,L)}$, $\mathbf{s}$, $\sigma$)
          end if
          if $p \geq z$ then
              $t \leftarrow t + \tau$                          ▷ Accept sample: update time and state.
              $\mathbf{n} \leftarrow$ EXECUTEREACTIONS($\mathbf{r}$, $\mathbf{R}$, $\mathbf{n}$)
          end if
      end while
      return $\mathbf{n}$
  end function
```

If it is required to update L automatically, the appropriate variables and counters from section 3.2.5 need to be added.

The implementation used in this paper is written in C++ and contains around 600 lines of code for the core components. The code is available at the ER-leap web site, *http://computableplant.ics.uci.edu/erleap/.*

Acceptance ratio analysis

A preliminary analysis looks very permissive of large L:

$$\begin{aligned} \underset{\sim}{\Delta}_a &\equiv \min{}_r \Delta m_a^r \qquad & \tilde{m}_a &\equiv \max{}_r m_a^r \\ \tilde{\Delta}_a &\equiv \max{}_r \Delta m_a^r \qquad & \tilde{m} &\equiv \max{}_r \sum_a m_a^r \end{aligned} \tag{3.23}$$

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

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

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 $O(LR)$. 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 occurs with probability $(1 - \langle \underset{\sim}{P} \rangle)$. So the computational cost for one leap attempt can be estimated as

$$O(R + \left(1 - \langle \underset{\sim}{P} \rangle\right) LR) \tag{3.24}$$

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

$$\begin{aligned}
\text{ERleap cost} &= \frac{\text{ERleap leap cost}}{\text{reaction events}} \leq \frac{R + \left(1 - \left\langle \underset{\sim}{P} \right\rangle\right) LR}{\left\langle \underset{\sim}{P} \right\rangle L}, \\
\text{SSA cost} &= \frac{\text{SSA step cost}}{\text{reaction events}} = \frac{R}{1}.
\end{aligned}$$

The cost ratio between SSA and ER-leap is therefore

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

When this cost ratio is less than one, ER-leap will be asymptotically faster than SSA. This is the case whenever $\langle \underset{\sim}{P} \rangle > (1 + L)/2L$ 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

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

The required data structures and space requirements for ER-leap 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 $\tilde{\Delta}_a$ and $\underset{\approx}{\Delta}_a$ values.

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 3.24 that the cost of calculating early acceptance samples will be $O(R)$ and the expected cost of calculating the late acceptance samples is $O((1 - \langle \underset{\sim}{P} \rangle)LR)$ 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 $b = 5$ 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 $\beta = 2/3$ 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:

$$\begin{aligned}\mu &= \overline{\mu} \pm z\sqrt{\tfrac{\overline{\sigma^2}}{n}} \\ &= \overline{\mu} \pm E_{\text{rror}}\end{aligned}$$

where statistics for calculating the sample mean and sample variance $(\overline{\mu}, \overline{\sigma^2})$ are gathered from $\underset{\sim}{P}$ during simulation, 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

$$L' = \begin{cases} L+1, & \text{if} \quad h(L) < \overline{\mu} - E_{\text{rror}} \text{ and } \ h(L+1) < \overline{\mu} + E_{\text{rror}} \\ L-1, & \text{if} \quad \overline{\mu} - E_{\text{rror}} < h(L-1) \text{ and } \ \overline{\mu} + E_{\text{rror}} < h(L) \\ L, & \text{otherwise} \end{cases} \tag{3.25}$$

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.

Finally, to avoid getting 'stuck' on a particular L, the counters are occasionally reset with probability $1/L^2$.

An Illustrative Example

As a specific example of the use of the ER-leap algorithm, consider the two-reaction dimerization process $\{2S_1 \underset{\rho_2}{\overset{\rho_1}{\rightleftarrows}} S_2\}$ with forward and reverse reactions $r = 1$ and $r = 2$. Recall from equation 3.9 that the instantaneous rates of firing, also called propensities, for each reaction are given by

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

(Some authors divide $a_1(\boldsymbol{n})$ by two to "avoid double counting", but our convention is to absorb this factor of two into $\rho_1$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 3.11:

$$\tilde{a}_1(\boldsymbol{n}) = \rho_1(n_1 + 2\,(L-1))\,((n_1 + 2\,(L-1) - 1))\,,$$

$$\underset{\sim 1}{a}(\boldsymbol{n}) = \rho_1(n_1 - 2\,(L-1))\,((n_1 - 2\,(L-1) - 1))\,,$$

$$\tilde{a}_2(\boldsymbol{n}) = \rho_2(n_2 + (L-1)),$$

$$\underset{\sim 2}{a}(\boldsymbol{n}) = \rho_2(n_2 - (L-1)).$$

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 $((\frac{\tilde{a}_1(n)}{\tilde{a}_0(x)}, \frac{\tilde{a}_2(n)}{\tilde{a}_0(n)}), L)$, where $\tilde{a}_0(n) = \tilde{a}_1(n) + \tilde{a}_2(n)$. Next, the total time step τ is sampled from the Erlang (gamma with an integer second argument) distribution with parameters $(\underset{\sim 0}{a}(n), L)$ such that

$$\boldsymbol{\tau} \sim Gamma(\underset{\sim 0}{a}(n), L)\ .$$

To compute the probability of early acceptance, equation 3.21 is used. This simplifies to

$$\begin{aligned}\text{Prob}_{\text{early}}(\boldsymbol{s},\boldsymbol{\tau}) &= \left(\frac{\underset{\sim}{F}^{(1)}_{I_0,L-1}}{\tilde{F}^{(1)}_{I_0,L-1}}\right)^{s_1}\left(\frac{\underset{\sim}{F}^{(2)}_{I_0,L-1}}{\tilde{F}^{(2)}_{I_0,L-1}}\right)^{s_2}\exp\left(-(\tau_1+\tau_2)\left(\tilde{D}_{I_0L-1}-\underset{\sim}{D}_{I_0L-1}\right)\right)\\ &= \left(\frac{\underset{\sim 1}{a}(\boldsymbol{n})}{\tilde{a}_1(\boldsymbol{n})}\right)^{s_1}\left(\frac{\underset{\sim 2}{a}(\boldsymbol{n})}{\tilde{a}_2(\boldsymbol{n})}\right)^{s_2}\exp(-\boldsymbol{\tau}(\tilde{a}_0(\boldsymbol{n})-\underset{\sim 0}{a}(\boldsymbol{n})).\end{aligned}$$

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 $\boldsymbol{\sigma} = \{r = 1, r = 1, r = 2, ...r = 1\}$. Next, we need to sample the length of individual time steps for each reaction, $\{\tau_1, \tau_2, ..., \tau_L\}$. This can be done by independently sampling L unit exponential random variables and "normalizing" them so their sum is τ. It is now possible to calculate the true probability of acceptance from equation 3.19

$$\begin{aligned}&\text{Prob}_{\text{accept}}(\boldsymbol{\sigma},\{\tau_i\})\\ &\quad= \left(\frac{1}{\tilde{F}^{(1)}_{I_0,L-1}}\right)^{s_1}\left(\frac{1}{\tilde{F}^{(2)}_{I_0,L-1}}\right)^{s_2}\prod_{i=1}^{L}\underset{\sim}{F}^{(\sigma_i)}_{I_i,L-1}\exp\left(-\tau_i\left(D_{I_i\ \ L-1}-\underset{\sim}{D}_{I_0L-1}\right)\right)\\ &\quad= \left(\frac{1}{\tilde{a}_1(\boldsymbol{n})}\right)^{s_1}\left(\frac{1}{\tilde{a}_2(\boldsymbol{n})}\right)^{s_2}\prod_{i=1}^{L}a_{\sigma_i}(\boldsymbol{n}_i)\exp(-\tau_i(a_0(\boldsymbol{n}_i)-\underset{\sim 0}{a}(\boldsymbol{n}))).\end{aligned}$$

Here $\tilde{a}(\boldsymbol{n})$ and $\underset{\sim 0}{a}(\boldsymbol{n})$ 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 3.11 and equation 3.13 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 3.23) 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 3.10. 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.

3.3 Numerical Simulations

The foregoing 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 [4].

3.3.1 Accuracy

Here we verify ER-leap equivalence to SSA via numerical experiments. As an example of the tests performed in the CaliBayes test suite [16], 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 CaliBayes. 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:

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

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.

We performed this analysis on SSA and ER-leap. As figure 3.1

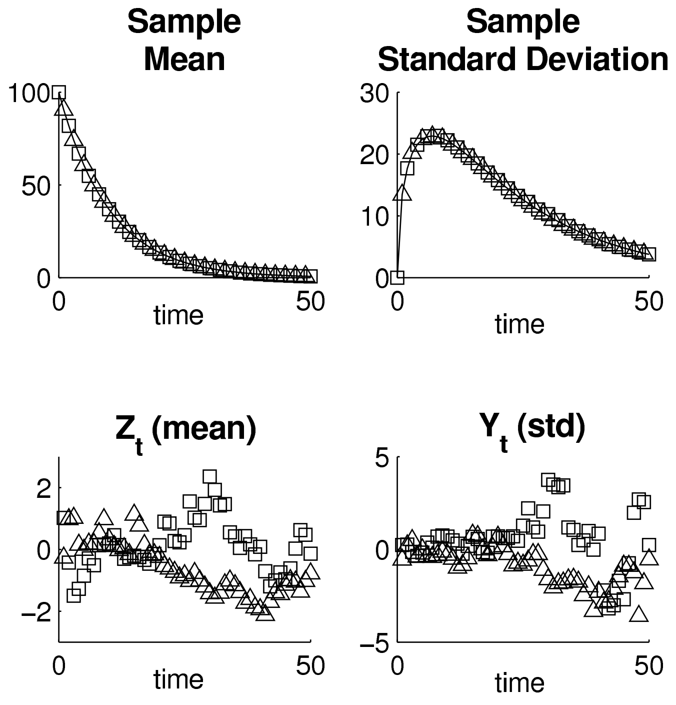

Figure 3.1: ER-leap (□) with $L = 4$ and SSA (△) compared with the analytical (—) mean and standard deviation. Y-axis in units of molecules. The Z_t and Y_t values will be normally distributed, assuming SSA equivalence. Therefore values in the range $(-3, 3)$ are considered reasonable. Galton-Watson 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.

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.

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 (see figure 3.2).

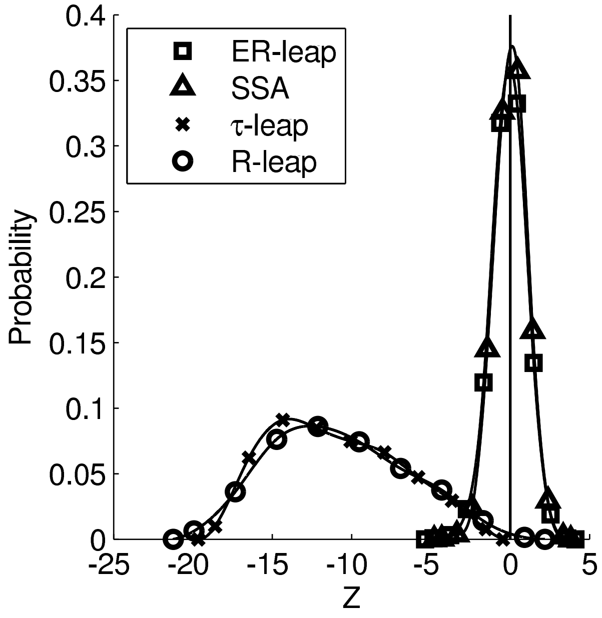

Figure 3.2: Distribution of Z_t for the four algorithms under consideration. ER-leap and SSA demonstrate a standard normal distribution whereas the approximate methods show Z_t values far outside the expected range. Reactions $\{X \longrightarrow 2X, X \longrightarrow \emptyset\}$ with rate parameters {0.11, 0.1} and $X(0) = 1.0 \times 10^5$. For ER-leap L=30. For R-leap $\theta = 0.1$ and $\varepsilon = 0.01$. For τ-leap $\varepsilon = 0.01$. 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.

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.

3.3.2 CaliBayes validation

Similar analysis as above is performed on several models in the CaliBayes test suite version DSMTS 21 [16]. 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 deviation 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 $L\upsilon/(\underset{\sim}{D} + \tilde{D})$ of t, with $\upsilon = 7$. In practice these small steps do not significantly affect running time.

Using the criteria above, 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.

3.3.3 Williamowski-Rössler Model

The Williamowski-Rössler model [52], which contains several bi-molecular reactions, is explored to demonstrate the usefulness 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:

$$X \underset{k_2}{\overset{k_1}{\longleftrightarrow}} 2X \quad Y \underset{k_6}{\overset{k_5}{\longleftrightarrow}} \emptyset \quad Z \underset{k_{10}}{\overset{k_9}{\longleftrightarrow}} 2Z$$
$$X + Y \underset{k_4}{\overset{k_3}{\longleftrightarrow}} 2Y \quad X + Z \underset{k_8}{\overset{k_7}{\longleftrightarrow}} \emptyset.$$

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

$$\dot{x} = k_1 x - k_3 xy - k_2 x^2 + k_4 y^2 - k_7 xz + k_8$$
$$\dot{y} = k_3 xy - k_5 y - k_4 y^2 + k_6$$
$$\dot{z} = -k_7 xz + k_9 z - k_{10} z^2 + k_8$$

and plot the solution of X vs. Y as in figure 3.3.

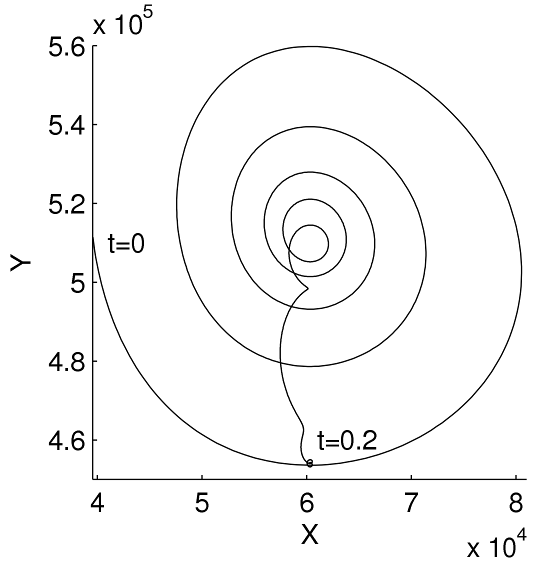

Figure 3.3: Mass-action deterministic solution of X vs. Y from time $t = 0$ to $t = 0.2$ for Williamowski-Rössler model. $k_1 = 900$, $k_2 = 8.3 \times 10^{-4}$, $k_3 = 0.00166$, $k_4 = 3.32 \times 10^{-7}$, $k_5 = 100$, $k_6 = 18.06$, $k_7 = 0.00166$, $k_8 = 18.06$, $k_9 = 198$, $k_{10} = 0.00166$. $X(0) = 39570$. $Y(0) = 511470$. $Z(0) = 0$.

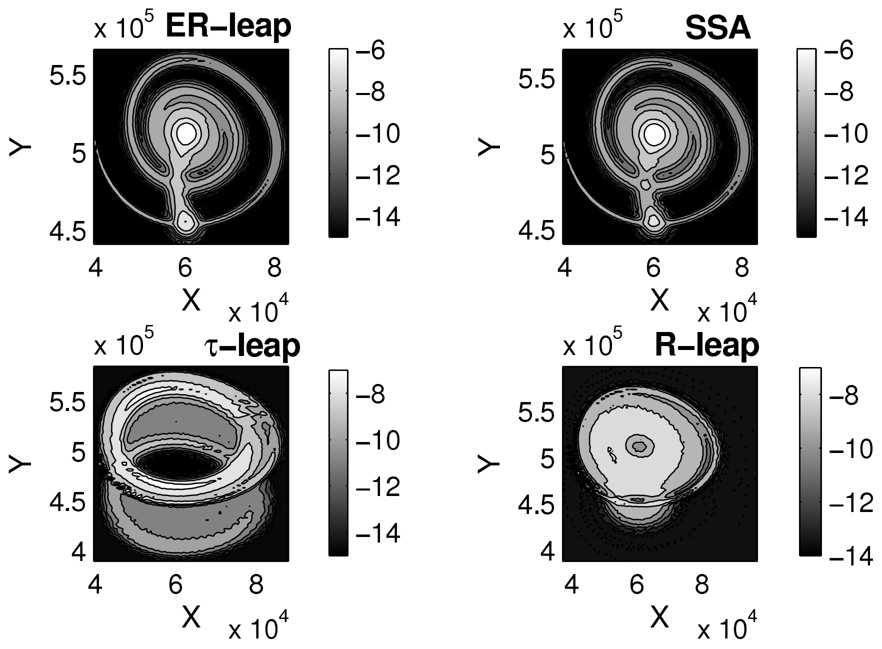

Figure 3.4: Comparing log probability densities for various simulation methods over time $t = 0$ to $t = 0.2$. Same parameters as figure 3.3. SSA and ER-leap appear identical. Total of $1,500$ samples for each simulator. For ER-leap L was chosen automatically and averaged $L = 23$. For τ-leap and R-leap $\varepsilon = 0.01$. For R-leap $\theta = 0.1$. Measurement taken every 10^{-4} sec.

As time progresses the mean trajectory spirals in towards an attraction point near $\{6.0 \times 10^4, 5.1 \times 10^5\}$. However, once the inner region is reached, the trajectory falls towards another attraction point around $\{6.0 \times 10^4, 4.5 \times 10^5\}$. The stochastic algorithms are run and we can observe the density plots over time for the exact and approximate algorithms in figure 3.4.

As figure 3.4 and figure 3.5 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.

We modify the foregoing Williamowski-Rössler model to have rate parameters in the chaotic regime as described in Ref. 52. 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 3.6. Notice the erratic behavior, which deterministic analysis may have difficulty capturing [52].

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.

In the corresponding mass-action ODE's in the chaotic regime, small simulation errors grow exponentially. Furthermore, deterministic mass action analysis has sometimes proven insufficient to model the system even for a large number of molecules [52]. 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.

3.3.4 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 $\alpha\simeq2/3$ (see figure 3.7). 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.

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 3.8.

This tradeoff can also be explored with a log-contour plot of CPU time and L (see figure 3.9). Notice that as simulation time increases, the optimal L changes. This fact is due to a change in the value of equation 3.21 as reactant numbers change. The presence of just a single local minimum in figure 3.9 suggests that dynamic optimization of L is not a hard problem.

3.3.5 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 addition, 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 $\{X_c \xrightarrow{\rho} X_{c'}\}$ 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, $d = 3$. As figure 3.10 demonstrates, ER-leap may be used to accelerate systems with many reaction channels. It also demonstrates the feasibility of applying ER-leap to spatially structured models.

Although the three-dimensional grid of compartments simulated here results in efficient simulation on a relatively highly connected network of reactions and reactants, with a graph diameter proportional to the cube root of the number of nodes, the ER-leap algorithm could be stressed to the point of inefficiency by other topologies. For example, a fully connected (diameter one) compartment graph, or a scale-free (logarithmic or sublogarithmic diameter) compartment graph, each have much higher connectivity than was tested here. Further work will be required to evaluate and possibly adapt the ER-leap algorithm for such alternative large-scale network structures.

3.4 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 sensitive 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 sampling each reaction event when there are many possible reactions to choose from [19]. An alternative acceleration of SSA has been proposed [39] based on exploiting cycle structure. The present ER-leap algorithm is based on the R-leap algorithm [4] that accelerates the SSA by efficiently executing a number of reaction firings together. ER-leap offers an acceleration that is more general than the efficient sampling of many reaction channels or types [19], or the exploitation of cycles [39]. Instead, like an approximate acceleration scheme, 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 3.3.5, along with previous work such as the use of tau-leap [42] and R-leap [6] in spatial models, show the feasibility 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.

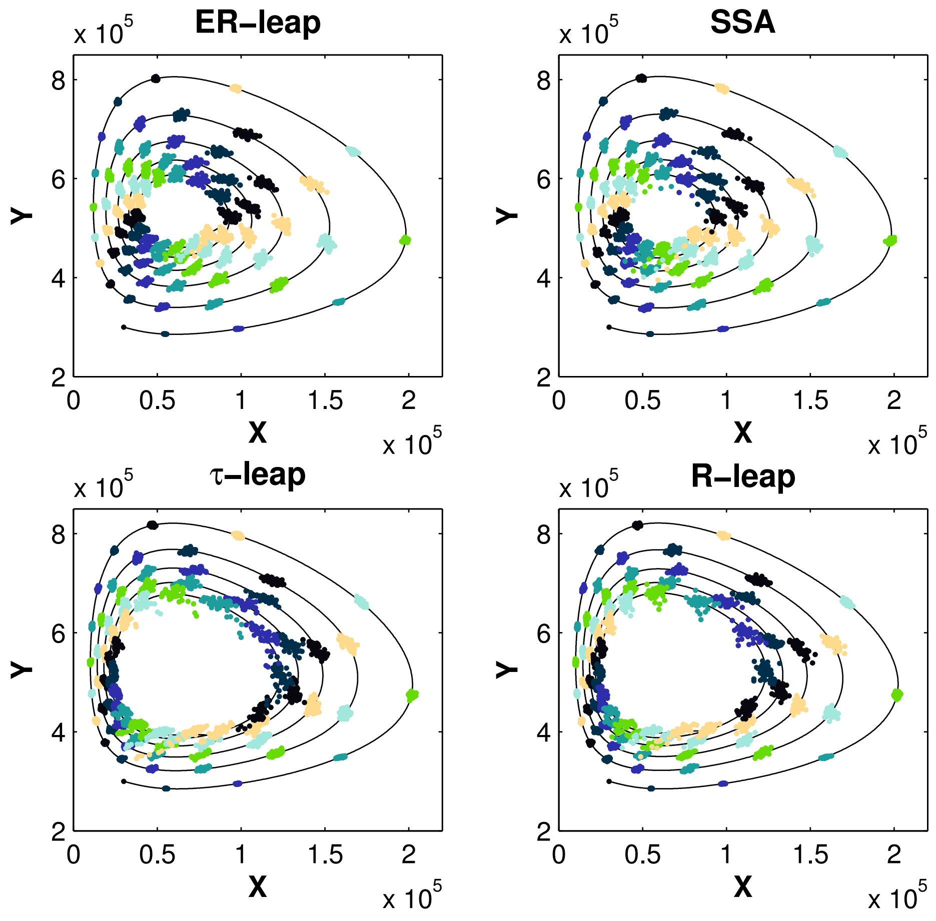

Figure 3.5: Another look at the differences in trajectories. Distribution of 50 runs for the four algorithms. Same network as figure 3.4. $X(0) = 30,000$. $Y(0) = 300,000$. (So we start further out in the spiral). Simulate from time $t = 0$ to $t = 0.13$, before the "escape" shown in figure 3.4. A constant 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.

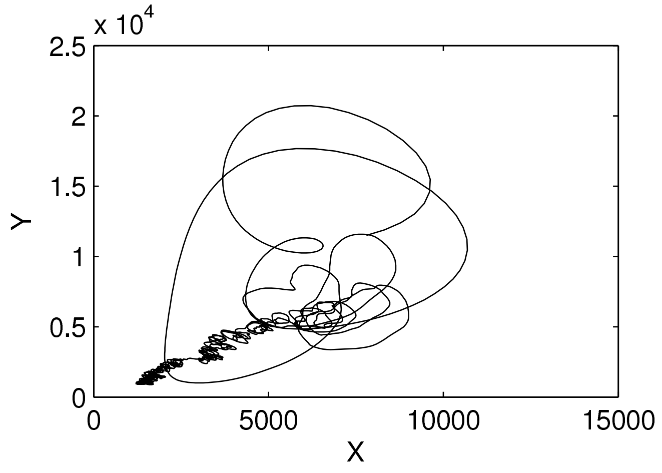

Figure 3.6: Mean number of molecules on chaotic system over $1,150$ SSA runs from time $t = 0$ to $t = 30$. $k_1 = 30$, $k_2 = 8.3\times10^{-4}$, $k_3 = 0.00166$, $k_4 = 3.32\times10^{-7}$, $k_5 = 10$, $k_6 = 0.602$, $k_7 = 0.00166$, $k_8 = 0.602$, $k_9 = 16.58$, $k_{10} = 0.00166$. $X(0) = 7800$. $Y(0) = 11500$. $Z(0) = 0$.

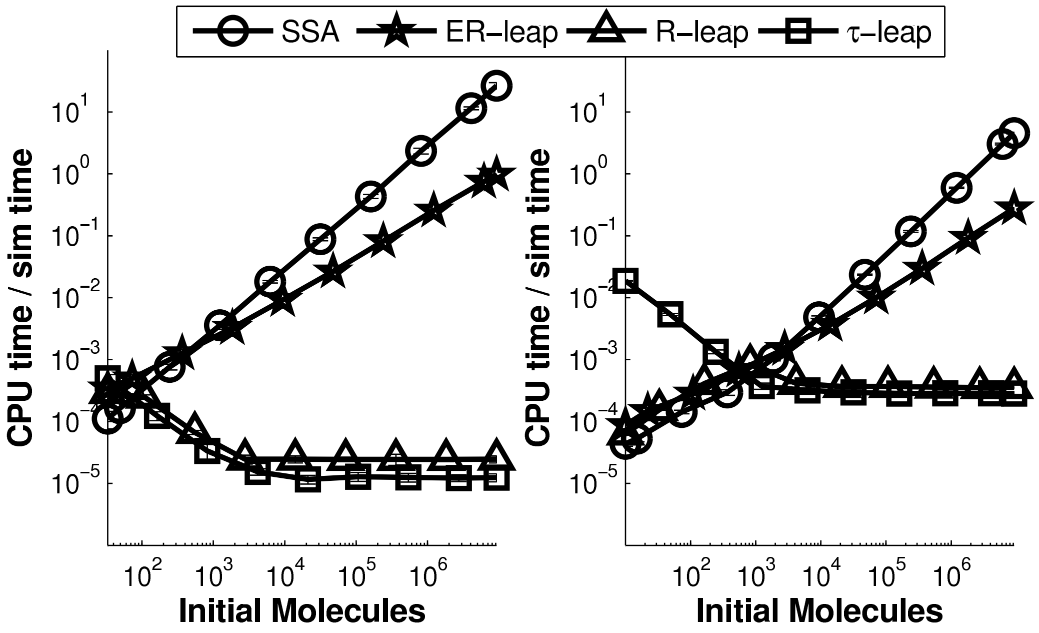

Figure 3.7: Log-log scaling of CPU running times for various stochastic simulation algorithms. The left panel plots results obtained for the Galton-Watson model with birth rate 0.101, death rate 0.10. Each test is simulated for 30 seconds. The slope of the ER-leap line is 0.65, and SSA is 0.99, about 1.0 as expected. Ratio is 0.66. L is chosen automatically for ER-leap. R-leap has accuracy parameters $\theta = 0.1$ and $\varepsilon = 0.01$. τ-leap has parameter $\varepsilon = 0.01$. The right panel plots results obtained for the dimerization process $\{2X \longrightarrow S, S \longrightarrow 2X\}$ with rate parameters $\{0.001/v, 0.01\}$ respectively, initial values $S(0) = n$, singleton molecule $X(0) = n/2$, and volume $v = n/100$. Slope of ER-leap line is 0.58 and slope of SSA line is 0.86 with a ratio of 0.68. Error bars are small compared to algorithm marks but represent one standard deviation.

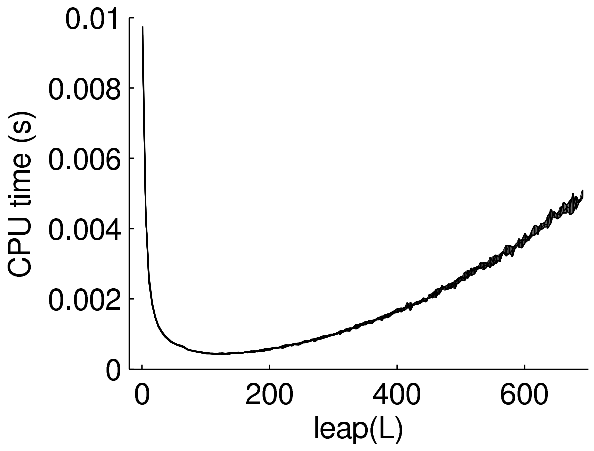

Figure 3.8: Varying L for birth/death process with rate of birth 0.1 and death 0.11. $X(0) = 1 \times 10^7$. $X(0) = 5 \times 10^6$. Simulation from $t = 0$ to $t = 5$. Initially as we increase L, CPU runtime drops dramatically until the optimum at about $L = 115$ which is about 22x faster than SSA. For larger L, the rejection of proposed samples starts to decrease performance and there is a monotonic increase in CPU computation time.

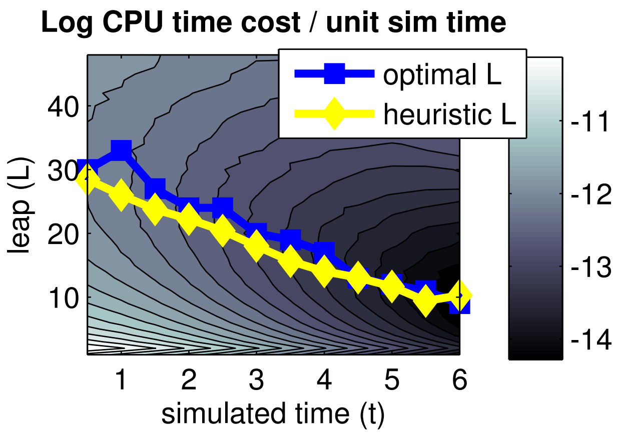

Figure 3.9: ER-leap contour plot of Log CPU time per unit simulation time vs. simulation time and leap size, L. Overlay of optimal and heuristic choice of L (from one run). Notice that the optimal leap L^* changes during simulation from $L^* = 34$ at $t = 0$ to about $L^* = 8$ at $t = 6$. Basic cascading network $\{S1 \longrightarrow S2, S2 \longrightarrow S3, S3 \longrightarrow S4\}$ and all rates 1.0. Initial values: $S1 = 4.2 \times 10^4$, $S2 = 4.0 \times 10^4$, $S3 = 3.5 \times 10^4$ and $S4 = 0$. Results averaged over 500 runs.

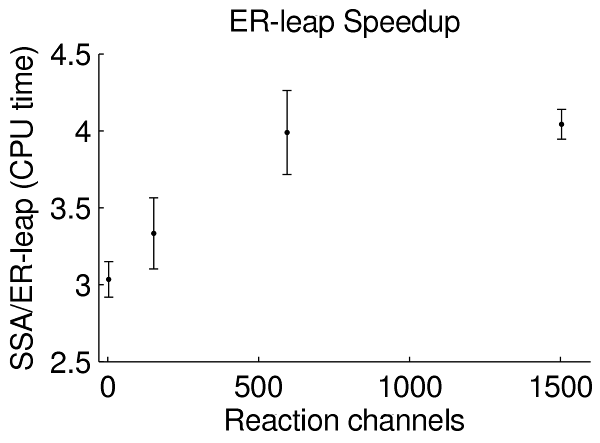

Figure 3.10: Speedup is calculated as SSA wall clock time divided by ER-leap wall clock time. It increases monotonically from a one-cell system with 10 reaction channels to a $4 \times 4 \times 4$ grid with 1504 reaction channels. In ER-leap L was chosen automatically, and averaged 23 over all experiments. Error bars are one standard deviation. Same rate parameters as figure 3.3. Rate of diffusion is 0.01 and the initial number of molecules in each cell is $X(0) = 5.0 \times 10^4$, $Y(0) = 4.5 \times 10^5$, $Z(0) = 3.0 \times 10^4$.

Chapter 4

HiER-Leap

4.1 Introduction

Computational biology is moving toward ever more complex, comprehensive and detailed biological models. It is becoming increasingly important to simulate and understand these models computationally.

There have been recent advances in consumer level multi-core CPU technology. There are indications that next generation CPU technology is moving from maximizing single core speed to increasing the number of cores by orders of magnitude.

The work of [30] suggests a great potential for parallel SSA algorithms. However, the parallelization was used to speedup sampling of *many* trajectories. In addition, the parallelization and acceleration of a single trajectory becomes a dominant problem when very large systems are modeled.

This chapter describes a new SSA-equivalent algorithm that can take advantage of parallel hardware, and *additionally* provides an algorithmic speedup for systems with many reaction channels. This “HiER-Leap” (Hierarchical Exact Reaction-Leaping) algorithm achieves these advancements without the loss of accuracy. This work therefore presents an important step towards organism-scale simulation.

4.2 Theory

4.2.1 Hierarchical Notation

The HiER-Leap algorithm uses a divide and conquer strategy to accelerate SSA. Evidence suggests that protein-protein interaction (PPI) networks tend to be modular [34]. These networks contain submodule clusters that interact heavily inside the cluster. Interactions with other clusters of proteins are less common. Although still an active area of research, the work of [26] suggests that similar modularity may exist in genetic regulatory networks as well. Additionally, when modeling spatial interactions, events spatially distant must interact through intermediate diffusion reaction channels. In this way, it is probably common that many reaction channels are weakly coupled to the majority of other channels. This observation suggests a potential avenue towards algorithm acceleration and parallelization for large biological networks.

Notation is introduced to describe a hierarchical organization of reaction channels. Next, following and generalizing the strategy of chapter 3, we will derive bounds on propensities and species. The bounds will be essential for deriving an algorithm for exact speedup of SSA for systems amenable to hierarchical organization.

Reaction channels must belong to exactly one *block*. A *block* is defined as a set of reaction channels. If reactions are “connected” by shared reactants, it is preferred that reactions should be more strongly connected within than between blocks. For this work, a two level hierarchy of reactions and blocks is used. However, it should be straightforward to expand our work to multiple levels.

Each reaction channel is indexed by its block ID r_1, and its within-block ID r_2, and will be designated as $R = (r_1 r_2)$ for $r_1 \in \{1 \ldots b\}$ and $r_2 \in \{1 \ldots b_{r_1}\}$. The "block propensity" for block r_1 and state I, denoted $D_I^{(r_1)}$ is the sum of propensities of constituent reaction channels. Specifically, using the notation of section 3.2.3 this means

$$D_I^{(r_1)} = \sum_{r_2 \in r_1} \rho_{r_1 r_2} F_I^{(r_1 r_2)}. \tag{4.1}$$

Furthermore, we denote the number of reaction events occuring within block r_1 as u_{r_1}. Finally, the number of events for the reaction channel indexed by $R = (r_1 r_2)$ is denoted by $v_{r_1 r_2}$.

4.2.2 Bounds on Propensities and Species Counts

Similar to equation 3.14, we now develop bounds on species counts and propensities. This enables us to derive a rejection sampling algorithm in many ways analogous to ER-leap. For reasons that will become evident in section 4.2.3, we first derive bounds on the block propensities given L and I_0. Afterwards, bounds will be developed on the species molecule counts and reaction channel propensities given u.

First, recall that in equation 3.10 we argued that species counts during the next L reaction events are bounded as

$$n_a + l \ \min_r \{\Delta m_a^r\} \leqslant n'_a \leqslant n_a + l \ \max_r \{\Delta m_a^r\} \tag{4.2}$$

which yields corresponding upper and lower bounds

$$\begin{aligned}\tilde{F}^{(r)}_{(I_0,L)} &\equiv F^{(r)}_{[n_a+l\ \max_r\{\Delta m_a^r\}||1\leqslant a\leqslant A]} \\ \underset{\sim}{F}^{(r)}_{(I_0,L)} &\equiv F^{(r)}_{[n_a+l\ \min_r\{\Delta m_a^r\}||1\leqslant a\leqslant A]}\end{aligned} \tag{4.3}$$

and can be calculated exactly as done in chapter 3. Additionally, note that the bound on the propensity of block r_1 is defined as

$$\tilde{D}^{r_1}_{(I_0,L)} = \sum_{r_2\in r_1} \tilde{F}^{(r_1r_2)}_{(I_0,L)}$$

which is similarly used for calculating the total network propensity bound $\tilde{D}_{(I_0,L)}$.

Optimized Block Level Bounds

If it is the case that we only need bounds on the block propensities, and not individual reaction channels, then we can take advantage of reaction event exclusion. Specifically, we no longer need to assume that all species counts are at the most extreme value possible after L reaction events.

We want to find a bound close to the optimal block propensity

$$\widehat{D}^{r_1}_{(I_0L)}{}^{*} = \max_{v_{r_1}||u_{r_1}=L} \sum_{r_2\in r_1} F^{r_1r_2}_{I(\mathbf{n}_0,\mathbf{v}_{r_1})}. \tag{4.4}$$

Unfortunately, naïvely solving this exactly for r_1 requires enumerating $(b_{r_1})^L$ possible choices for $\mathbf{v}_{r_1}$ upon every iteration. Fortunately the bound we seek, $\widehat{D}^{r_1}_{(I_0L)}$, is not required to be exactly optimal. Instead we only require that

$$\widehat{D}^{r_1}_{(I_0L)}{}^{*} \leq \widehat{D}^{r_1}_{(I_0L)} \leq \tilde{D}^{r_1}_{(I_0L)} \tag{4.5}$$

Symbol	Meaning
$\tilde{x}$	Upper bounding value for x after $L-1$ reaction events. Calculated by assuming each species type will be maximal.
$\hat{x}$	Upper bounding value for x after $L-1$ reaction events. May not depend on bounding all species values and therefore may be tighter than $\tilde{x}$.
$\bar{x}$	Upper bounding value given u. Will often involve inner block calculations.
$\utilde{x}, \underset{\vee}{x}, \underline{x}$	Lower bounding versions of the above definitions.
x^*	The optimal value of x with respect to some objective function.

Table 4.1: Notation, Accents and Meaning

such that ${\widehat{D}^{r_1}_{(I_0L)}}^* \leq \widehat{D}^{r_1}_{(I_0L)}$ is required for algorithmic correctness and $\widehat{D}^{r_1}_{(I_0L)} \leq \tilde{D}^{r_1}_{(I_0L)}$ is needed for improved efficiency.

A heuristic algorithm for $\widehat{D}^{r_1}_{(I_0L)}$ is developed. We demonstrate this falls between the requisite values and has 'nice' asymptotic properties that will be discussed later.

Derivation The idea is to find the maximum $\Delta\widehat{D}^{r_1}_{(I_0L)}$ possible resulting from one reaction channel firing sometime during the next L reaction events. If we determine this value, we can upper bound $D^{r_1}_{(I_0 \ldots I_{L-1})}$ with

$$D^{r_1}_{(I_0,L)} \leq D^{r_1}_{I_0} + (L-1)\Delta{\widehat{D}^{r_1}_{(I_0L)}}^* \tag{4.6}$$

with

$$\Delta{\widehat{D}^{r_1}_{(I_0L)}}^* = \max_{R_{r_1 r_2 || I_0 .. I_{L-1}}} \Delta\widehat{D}^{r_1}_{(I_0L)}.$$

Note how this is an upper bound on $D^{r_1}_{(I_0,L)}$. By construction, $\Delta{\widehat{D}^{r_1}_{(I_0L)}}^*$ is the largest amount that the block propensity may change for any of the upcoming possible $(L-1)$ reaction events in r_1. Since there are $(L-1)$ reaction events, and the most any of them may increase $D^{r_1}_{(I_0,L)}$ is $\Delta\widehat{D}^{r_1}_{(I_0L)}{}^*$, (equation 4.6) will always bound $D^{r_1}_{(I_0,L)}$.

This method improves upon our previous methods, which found the maximum $\tilde{n}_a$ for all species and then calculates the block propensity. Each within-block reaction channel propensity will be larger when $\tilde{n}_a$ vs n_a is used to calculate the propensity. Therefore, using the increased bound will result in a block's propensity being $O(b_{r_1} * L)$ larger than $D^{r_1}_{I_0}$. However, by calculating using the above $\Delta\widehat{D}^{r_1}_{(I_0L)}$ method will have the bound will be just $O(L)$ larger than $D^{r_1}_{I_0}$.

Again, naïvely solving for $\Delta\widehat{D}^{r_1}_{(I_0L)}{}^*$ requires an impractical amount of work. But as with our previous argument, we can upper bound $\Delta\widehat{D}^{r_1}_{(I_0L)}{}^*$ and still achieve an upper bound for $D^{r_1}_{(I_0,L)}$. To upper bound $\Delta\widehat{D}^{r_1}_{(I_0L)}{}^*$ we use the monotonic nature of $D^{r_1}_{I_k}$. If any species increases to $n'_a \geq n_a$ we know that $D^{r_1}_{n'_a} \geq D^{r_1}_{n_a}$. Therefore, if we find the reaction channel that increases the block propensity the most when $\tilde{n}_{I_0,L}$ is used for positive $\Delta m_a^{(r_1 r_2)}$, we are guaranteed that there does not exist a larger $\Delta\widehat{D}^{r_1}_{(I_0L)}$.

This yields

$$\Delta\widehat{D}^{r_1}_{(I_0L)} = \max_{r_2 \in r_1} \left[D^{r_1}(\mathbf{q}(\tilde{\mathbf{n}}, r_2)) - D^{r_1}(\tilde{\mathbf{n}})\right] \tag{4.7}$$

where

$$\mathbf{q}(\mathbf{n}, r_2)_a = \begin{cases} n_a + \Delta m_a^{(r_1 r_2)} & \text{if } \Delta m_a^{(r_1 r_2)} > 0, \\ n_a & \text{otherwise} \end{cases} \tag{4.8}$$

as our final equation for $\Delta\widehat{D}^{r_1}_{(I_0L)}$. A proof that this will be the maximum delta possible can be found in appendix A.

This tighter bound will result in a greater acceptance ratio. The basic reason for this improvement is that we need not overestimate *every* propensity in r_1 by $O(L)$, and add the overestimates up, since only L and not $b_1 L$ reactions will occur.

Naïvely finding the reaction channel $R = (r_1 r_2)$ that will increase $D^{r_1}(\tilde{n}_{I_0,L})$ by a maximal amount will cost $O(|R_{r_1}|)$ to compute. To accelerate this step further blockwise priority queues (PQ) are used to find this R efficiently. Nodes in the PQ are reaction channels and values are the $\Delta D^{r_1}_{(I(\tilde{n},b_{r_1}))}$ caused by each reaction channel firing. Upon acceptance of L reaction events we must update the priority queue for each block. Only nodes that interact with the species, which have changed, need to be adjusted. This, at worst, will be $O(\log b_{r_1})$ work for each node, although in practice the order rarely needs to change.

Propensity Bounds Given u

If we know u, the number of reaction events for r_1 and adjacent blocks, we can derive even tighter bounds on the reaction channels $R_{r_1 *}$. In fact, these tighter bounds help us to efficiently increase L when larger systems are considered, as will be demonstrated in section 4.2.3.

We determine $\bar{F}^{(r_1 *)}$ by finding bounds on species counts given u. In other words, we want to find

$$n_A(r, n_0, k) \leq \bar{n}_{Ar_1}(u, n_0), \text{ for } k = 0..[(\text{index of final } r_1 \text{ event}) - 1]$$

which is the maximum possible value of n_{Ar_1} prior to the last event in r_1 occurring. In this way $n_A(r, n_0, k)$ will never exceed the propensity calculated from $\bar{n}_{Ar_1}(u, n_0)$.

Finding the optimal value for $\bar{n}_{Ar_1}(u, n_0)$ is straightforward. We first need to consider blocks other than r_1 which may change n_{Ar_1}. Since the order of reactions is unknown, we must assume that all $u \backslash \{u_{r_1}\}$ reactions occur prior to those in u_{r_1}. It is desired that the number of neighbors relative to the total number of blocks will be small. This will decrease $n_A(r, n_0, k)$ and ultimately lead to a more efficient algorithm. Secondly, we need to consider reactions in r_1. In calculating the bound, it is assumed that all $(u_{r_1} - 1)$ reaction events chosen will behave adversarially. This is analogous to the method considered in section 4.2.2, with the modification that we will consider a subset of reaction channels. Thus

$$\bar{n}_{Ar_1}(u, n_0) \equiv n_A + \sum_{r_1{}'} (u_{r_1{}'} - \delta_{r_1 r_1{}'}) \max_{r_2{}'} \Delta m^{(r_1{}' r_2{}')}_{(a_1 a_2)}$$

will bound each n_{Ar_1} with respect to r_1 and u. Finally, the propensities of reaction channels inside of block r_1 are bound as,

$$\bar{F}^{(r_1 r_2)}(u, n_0) = F^{(r_1 r_2)}(u, \bar{n}_{Ar_1}).$$

Lower bounding the propensities and species is done with the same techniques as that used for upper bounding. These derived bounds are used in the following sections.

4.2.3 Equivalent Markov Process

Similar to section 3.2, we want to algebraically manipulate the distribution represented by the Chemical Master Equation (a special case of the Kolmogorov-Chapman equation [49]) into a form suitable for parallelization and acceleration. The hierarchical description from section 4.2.1 will aid us in this transformation.

Using the matrix notation of 3.2.2 we begin with the SSA equivalent distribution for I and τ after L reaction events,

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

using the unique transformation of each reaction on state $I_0 \to I_1(R, I_0)$ we can write $\left[\hat{W}\exp(-\Delta t D)\right]^L$ in terms of ordered reaction sequences

$$\left[\prod_{k=l-1\searrow 0} \hat{W} \quad \exp(-\tau_k D)\right]_{I_l, I_0} =$$

$$= \sum_{\{R_k|k=1..L-1\}} \left[\prod_{k=L-1\searrow 0} \rho_{R_k} F^{(R_k)}_{I_k(\mathbf{R},I_0)} \exp(-\tau_k(D_{I_k(\mathbf{R},I_0),I_k(\mathbf{R},I_0)}))\right] \quad (4.9)$$

First, we impose an ordering on $\{R_k|k = 1..L-1\}$. Next, denote the number of R's being from block r_1 as $[u_{r_1}, \ldots u'_{r_1}] = \mathbf{u}(\mathbf{R})$ being the number of times reactions occur in block r_1 in the sequence of reactions $\mathbf{R}$. Finally, inside each block there will be u_{r_1} reactions events. These are further proportioned to reaction channels in block u_{r_1} as $[v_{r_2}, \ldots v'_{r_2}] = \mathbf{v}(\mathbf{R}; r_1)$. Finally, as in ER-leap 3.2.4 it was shown that (in ER-leap notation)

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

which can be converted to our hierarchical version with u's and v's strictly ordered

$$\sum_{\{R_k|k=1..L-1\}} e(\mathbf{R}) = \sum_{\left\{\mathbf{u}|u_R\in\mathbb{N},\sum_R u_R=L\right\}} \sum_{\left\{\mathbf{v_{r_1}}|v_R\in\mathbb{N},\sum_{r'_2} v_{r_1 r'_2}=u_{r_1}\right\}}$$
$$\sum_{\{\sigma_1|\sigma_1 \text{ permutes unequal } R'\text{s}|\mathbf{u}\}} \sum_{\left\{\sigma_2|\sigma_2 \text{ permutes unequal } R'\text{s}|\mathbf{v_{r_1}}\right\}} e(\sigma_1(\sigma_2(\mathbf{R})))$$

by taking an average of $e(\sigma(\mathbf{R}))$ and weighting by the number of ways the selection may occur we get

$$= \sum_{\left\{\mathbf{u}|u_R\in\mathbb{N},\sum_R u_R=L\right\}} \sum_{\left\{\mathbf{v_{r_1}}|v_R\in\mathbb{N},\sum_{r_2'} v_{r_1r_2'}=u_{r_1}\right\}} \binom{L}{R_1R_2\ldots R_n} \left\langle\langle e(\sigma_1(\sigma_2(\mathbf{R})))\rangle_{\sigma_2}\right\rangle_{\sigma_1}$$

and analogous to the way shuffling a deck of cards is the same as shuffling by suit and then, maintaining that order, shuffling by value independently for each suit

$$\begin{aligned}\binom{L}{R_1R_2\ldots R_n} &= \frac{L!}{R_1!R_2!\ldots R_n!}\\ &= \frac{L!}{u_{r_1}!u_{r_1'}!\ldots u_{r_1''}}\prod_{r_1}\frac{u_{r_1}!}{v_{r_1r_2}!v_{r_1r_2'}!\ldots v_{r_1r_2''}!}\\ &= \binom{L}{u_{r_1}\ldots u_{r_1'}}\prod_{r_1}\binom{u_{r_1}}{v_{r_1r_2}\ldots v_{r_1r_2'}}\end{aligned}$$

we arrive at an interesting form for our distribution, which is already suggestive of a parallel algorithm.

$$\begin{aligned}\sum_{\{r_k|k=1..L-1\}} e(\mathbf{r}) = \sum_{\left\{\mathbf{u}|u_R\in\mathbb{N},\sum_R u_R=L\right\}} \binom{L}{u_{r_1}\ldots u_{r_1'}} \sum_{\left\{\mathbf{v_{r_1}}|v_R\in\mathbb{N},\sum_{r_2'} v_{r_1r_2'}=u_{r_1}\right\}}\\ \prod_{r_1}\binom{u_{r_1}}{v_{r_1r_2}\ldots v_{r_1r_2'}} \left\langle\langle e(\sigma_1(\sigma_2(\mathbf{R})))\rangle_{\sigma_2}\right\rangle_{\sigma_1} \quad (4.10)\end{aligned}$$

To go further, we need to re-examine $e(\ldots)$.

Introduction of Probability Bounds

We now make use of our previously derived propensity bounds to derive a parallel algorithm. From 4.9 we have,

$$\left\langle \langle e(\sigma_1(\sigma_2(\mathbf{R})))\rangle_{\sigma_2}\right\rangle_{\sigma_1} = \left\langle\left\langle \prod_{k=L-1\searrow 0} \rho_{R_k} F^{(R_k)}_{I_k(\mathbf{R},I_0)} \exp(-\tau_k(D_{I_k(\mathbf{R},I_0),I_k(\mathbf{R},I_0)}))\right\rangle_{\sigma_2}\right\rangle_{\sigma_1}$$

with inclusion of derived bounds,

$$= \left\langle\left\langle \prod_{k=L-1\searrow 0} \frac{F^{(R_k)}_{I_k(\mathbf{R},I_0)}}{\bar{F}^{(r_1r_2)}(u,I_0)} \frac{\rho_{R_k}\bar{F}^{(r_1r_2)}(u,I_0)}{\bar{D}^{(r_1)}(u,I_0)} \frac{\bar{D}^{(r_1)}(u,I_0)}{\widehat{D}^{r_1}_{(I_0L)}} \widehat{D}^{r_1}_{(I_0L)} \times \right.\right.$$
$$\left.\left. \exp(-\tau_k(D_{I_k(\mathbf{R},I_0),I_k(\mathbf{R},I_0)} - \underset{\frown}{D}_{(I_0L)}) \exp(-\tau_k \underset{\frown}{D}_{(I_0L)})\right\rangle_{\sigma}\right\rangle_{\sigma_1}$$

we can immediately begin to separate out terms based on independence of σ_1, σ_2 and v, arriving at

$$= \exp(-\tau \underset{\frown}{D}_{(I_0L)}) \left(\prod_{r_1} \widehat{D}^{r_1}_{(I_0L)}{}^{u_{r_1}}\right) \left(\prod_{r_1}\prod_{r_2} \left(\frac{\rho_{R_k}\bar{F}^{(r_1r_2)}(u,I_0)}{\bar{D}^{(r_1)}(u,I_0)}\right)^{v_{r_2}}\right) \times$$
$$\left(\prod_{r_1} \left(\frac{\bar{D}^{(r_1)}(u,I_0)}{\widehat{D}^{r_1}_{(I_0L)}}\right)^{u_{r_1}}\right) \times$$
$$\left\langle \prod_{k=L-1\searrow 0} \frac{F^{(R_k)}_{I_k(\mathbf{R},I_0)}}{\bar{F}^{(r_1r_2)}(u,I_0)} \left\langle \exp(-\tau_k(D_{I_k(\mathbf{R},I_0),I_k(\mathbf{R},I_0)} - \widehat{D}_{(I_0L)})\right\rangle_{\sigma_2}\right\rangle_{\sigma_1}$$

We now plug in our expression for $\left\langle \langle e(\sigma_1(\sigma_2(\mathbf{R})))\rangle_{\sigma_2}\right\rangle_{\sigma_1}$ into 4.10, combining terms where appropriate:

$$\begin{aligned}
&\left[\prod_{k=l-1\searrow 0} \hat{W} \quad \exp(-\tau_k D)\right]_{I_l,I_0} = \\
&\left(\frac{\sum_{r_1'} \widehat{D}^{r_1'}_{(I_0 L)}}{\underset{\frown}{D}_{(I_0,L)}}\right)^{L} \sum_{\left\{\mathbf{u}|u_R\in\mathbb{N},\sum_R u_R=L\right\}} \left[\binom{L}{u_{r_1}\ldots u_{r_1'}} \prod_{r_1}\left(\frac{\widehat{D}^{r_1}_{(I_0 L)}}{\sum_{r_1'}\widehat{D}^{r_1'}_{(I_0 L)}}\right)^{u_{r_1}}\right]\times \\
&\sum_{\left\{\mathbf{v_{r_1}}|v_R\in\mathbb{N},\sum_{r_2'} v_{r_1 r_2'}=u_{r_1}\right\}} \left[\prod_{r_1}\binom{u_{r_1}}{v_{r_1 r_2}\ldots v_{r_1 r_2'}}\prod_{r_2}\left(\frac{\rho_{R_k}\bar{F}^{(r_1 r_2)}(u,I_0)}{\bar{D}^{(r_1)}(u,I_0)}\right)^{v_{r_2}}\right]\times \\
&\left(\underset{\frown}{D}_{(I_0,L)}\right)^{L} \exp(-\tau \underset{\frown}{D}_{(I_0 L)}) \times AcceptCoarse(u; I_0, L)\times \\
&\qquad \left(\prod_{r_1} AcceptBlock(v_{r_1},\sigma_2;u)\right) \times AcceptFine(\sigma_1;u,v,I_0,\sigma_2) \qquad (4.11)
\end{aligned}$$

The acceptance probabilities are as follows:

$$AcceptCoarse(u;I_0,L) = \prod_{r_1}\left(\frac{\bar{D}^{(r_1)}(u,I_0)}{\widehat{D}^{r_1}_{(I_0 L)}}\right)^{u_{r_1}} \qquad (4.12)$$

$$AcceptBlock(v_{r_1},\sigma_2;u,r_1) = \prod_{k\in r_1}\frac{F^{(R_k)}_{I_k(\mathbf{R},I_0)}}{\bar{F}^{(r_1 r_2)}(u,I_0)} \qquad (4.13)$$

$$AcceptFine(\sigma_1;u,v,I_0,\sigma_2) = \prod_{k=L-1\searrow 0} \exp(-\tau_k(D_{I_k(\mathbf{R},I_0),I_k(\mathbf{R},I_0)} - \widehat{D}_{(I_0 L)})) \qquad (4.14)$$

Furthermore, prior to turning these equations into an algorithm, we note that we can lower-bound these acceptance probabilities. This will enable us to do an early acceptance or rejection without always doing all of the work to calculate these values exactly.

Lower Bounding Acceptance Probabilities

We begin by lower bounding $AcceptFine(\ldots)$. This probability requires the most work to calculate and as we will see may be bound fairly tightly. The bound only requires that τ has been sampled.

The lower bound $\underset{\frown}{AcceptFine}(\ldots)$ is sought such that

$$\underset{\frown}{AcceptFine}(\ldots) \leq \prod_{k=L-1\searrow 0} \exp(-\tau_k(D_{I_k(\mathbf{R},I_0),I_k(\mathbf{R},I_0)} - \underset{\frown}{D}_{(I_0L)})).$$

In the above, note that $\underset{\frown}{D}_{(I_0L)}$ is constant with respect to k. Therefore, when we also upper bound

$$\widehat{D}_{I_0L} \geq D_{I_k(\mathbf{R},I_0),I_k(\mathbf{R},I_0)}$$

this creates an easily computable expression for our lower bound

$$AcceptFine(\sigma_1; u, v, I_0, \sigma_2, \tau) \geq \prod_{k=L-1\searrow 0} \exp(-\tau_k(\widehat{D}_{I_0L} - \underset{\frown}{D}_{(I_0L)}))$$

so that

$$\underset{\frown}{AcceptFine}(\tau; I_0, L) = \exp(-\tau(\widehat{D}_{I_0L} - \underset{\frown}{D}_{(I_0L)})) \tag{4.15}$$

Furthermore, recall that $E[\tau] = \frac{L}{\underset{\frown}{D}_{(I_0,L)}}$. If we assume that $\Delta\widehat{D}_{(I_0L)} \propto L$ when computing $\underset{\frown}{D}_{(I_0L)}$ and $\widehat{D}_{(I_0L)}$ (see Eq. 4.6) in the limit of many non-zero propensity reaction channels

$$E\left[\lim_{|\mathcal{R}|\to\infty}\exp(-\tau(\widehat{D}_{I_0L}-\underset{\frown}{D}_{(I_0L)}))\right]\to 1$$

which implies that both $\underset{\frown}{\mathit{AcceptFine}}(\ldots)$ and $\mathit{AcceptFine}(\ldots)$ tend to unity as the number of reaction channels increases.

Next, we set out to lower bound $\mathit{AcceptBlock}(\ldots)$. This acceptance probability depends on σ_2. Therefore, work will be saved if we can calculate the lower bound without sampling σ_2. This can be accomplished by noting that $\mathit{AcceptBlock}(\ldots)$ is a product of fractions. If we have a numerator and denominator that are independent of σ_2 we can re-write this equation in terms of r_2. Specifically, using $\underline{F}^{(r_1r_2)}_{u,I_0}$ allows us to bound the equation.

$$\mathit{AcceptBlock}(v_{r_1},\sigma_2;u,r_1)=\prod_{k\in r_1}\frac{F^{(R_k)}_{I_k(\mathbf{R},I_0)}}{\bar{F}^{(r_1r_2)}(u,I_0)}\geq\prod_{k\in r_1}\frac{\underline{F}^{(r_1r_2)}(u,I_0)}{\bar{F}^{(r_1r_2)}(u,I_0)}$$

yielding

$$\underset{\frown}{\mathit{AcceptBlock}}(v_{r_1};u)=\prod_{r_2\in r_1}\left(\frac{\underline{F}^{(r_1r_2)}(u,I_0)}{\bar{F}^{(r_1r_2)}(u,I_0)}\right)^{v_{r_2}}\tag{4.16}$$

4.2.4 Algorithm

The above equations, along with rejection sampling, allow us to create an efficient algorithm that will allow much of the work be done in parallel. From equation 4.11 observe there are two pmf expressions for a multinomial distribution. Specifically,

$$\mathit{Multinomial}(u;\left\{p_{r_1}=\frac{\widehat{D}^{r_1}_{(I_0L)}}{\sum_{r_1'}\widehat{D}^{r_1'}_{(I_0L)}}\right\},L)=\binom{L}{u_{r_1}\ldots u_{r_1'}}\prod_{r_1}\left(\frac{\widehat{D}^{r_1}_{(I_0L)}}{\sum_{r_1'}\widehat{D}^{r_1'}_{(I_0L)}}\right)^{u_{r_1}}$$

is the multinomial distribution for sampling u. And for each r_1 the vector v_{r_1} is sampled as

$$Multinomial(v_{r_1}; \left\{ p_{r_1 r_2} = \frac{\rho_{R_k} \bar{F}^{(r_1 r_2)}(u, I_0)}{\bar{D}^{(r_1)}(u, I_0)} \right\}, u_{r_1}) = \\ \binom{u_{r_1}}{v_{r_1 r_2} \ldots v_{r_1 r_2'}} \prod_{r_2} \left(\frac{\rho_{R_k} \bar{F}^{(r_1 r_2)}(u, I_0)}{\bar{D}^{(r_1)}(u, I_0)} \right)^{v_{r_2}}$$

which is interesting and implies $(v_{r_1} \perp\!\!\!\perp v_{r_1'})|u$ for all blocks.

We now present algorithm 4.2.4, which is a realization of the aforementioned equations. First, note that if we have an early global acceptance, then most of the computational effort will be put into line 10. The subroutine from this line is shown in algorithm 4.2.4. Notice that this function is independent for all blocks and needs to be done for all blocks with at least one reaction event. This is an ideal scheme for parallelization and is done so with good efficacy as shown in figure 4.1. Furthermore, for the tests in in section 4.3.2 the full calculation $AcceptFine(\ldots)$ was rare because in general $\underset{\frown}{AcceptFine}(\ldots) \geq 0.995$.

Algorithm 4.1 HiER-Leap L SSA Steps

```
Require: $\widehat{D}_{(I_0,L)}, \underset{\frown}{D}_{(I_0,L)}, \{\widehat{D}^{r_1}_{(I_0,L)}\}$ precomputed for $L \geq 1$ and $I_0$.
Ensure: Return $(I_L, \Delta t)$                    ▷ return updated state and duration of L steps
    function HiER-Leap($I_0, L$)
        $\tau \leftarrow$ Erlang($\tau; \underset{\frown}{D}_{(I_0,L)}, L$)
        $u \leftarrow$Multinomial($u; \left\{ p_{r_1} = \frac{\widehat{D}^{r_1}_{(I_0 L)}}{\sum_{r_1'} \widehat{D}^{r_1'}_{(I_0 L)}} \right\}, L$)
        Compute $\bar{D}$'s
5:                                                   ▷ In accordance with equation 4.12.
        if UniformRandom(0,1) $\geq$AcceptCoarse($u; I_0, L$) then
            return HiER-Leap($I_0, L$)                ▷ Early Rejection. Try again.
        end if

10:     for all $r_1 \in R$ do
                                                     ▷ See algorithm 4.2.4.
            $(v_{r_1}, \sigma_2) \leftarrow$ SampleBlock($r_1, u, I_0$)       ▷ May be done in parallel.
        end for
        $z \leftarrow$ UniformRandom(0,1)
15:     if $z \leq \widehat{AcceptFine}(\tau; I_0, L)$ then              ▷ See equation 4.15.
            return $(I_L(I_0, v, u), \tau)$                   ▷ Early Acceptance.
        end if
                                   ▷ Computation should be rare; see equation 4.14.
        if $z \leq$ AcceptFine($\sigma_1; u, v, I_0, \sigma_2, \tau$) then
20:         return $(I_L(I_0, v, u), \tau)$
        else
            return HiER-Leap($I_0, L$)                            ▷ Try again.
        end if
    end function
```

Algorithm 4.2 HiER-Leap Sample Block

```
function SampleBlock($r_1$, $u$, $I_0$)
    $v_{r_1} \leftarrow$Multinomial $\left(v_{r_1}; \left\{p_{r_1 r_2} = \frac{\rho_{R_k} \bar{F}^{(r_1 r_2)}(u, I_0)}{\bar{D}^{(r_1)}(u, I_0)}\right\}, u_{r_1}\right)$
    $pBlockEarly \leftarrow 1$
    for $r_2 \in r_1$ and $v_{(r_1 r_2)} \geq 1$ do
        $pBlockEarly \leftarrow pBlockEarly \times \left(\frac{\underline{F}^{(r_1 r_2)}(u, I_0)}{\bar{F}^{(r_1 r_2)}(u, I_0)}\right)^{v_{r_2}}$
    end for
    $z \leftarrow$ UniformRandom(0,1)
    if $z \leq pBlockEarly$ then
        return $(v_{r_1}, \sigma_2)$                                   ▷ Early Accept.
    end if
    ▷ If neighboring blocks do not early accept will need to do the following jointly.
    $pBlock \leftarrow 1$
    $\sigma_2 \leftarrow$ Permutation($u_{r_1}$)          ▷ Must compute exact acceptance probability.
    for all $k = 1 \ldots u_{r_1}$ do
        $r'_2 \leftarrow \sigma_2(v_{r_1}, k)$
        $pBlock \leftarrow pBlock \times \frac{F^{(r_1 r'_2)}_{I_k(\sigma_2(v_{r_1}), I_0)}}{\bar{F}^{(r_1 r'_2)}(u, I_0)}$          ▷ Calculating $I_k$ takes the most work.
    end for
    if $z \leq pBlock$ then
        return $(v_{r_1}, \sigma_2)$                                   ▷ Accept sample.
    else
        return SampleBlock($r_1$, $u$, $I_0$)                 ▷ Sample rejected, try again.
    end if
end function
```

4.3 Numerical Experiments

4.3.1 CaliBayes Validation

We test the HiER-leap algorithm correctness with the CaliBayes test suite similar to the work in section 3.3.1. If is possible to solve analytically for $P(X|t)$, this allows us to compare many simulated trajectories to the true distribution defined by the CME. Since HiER-leap reduces to ER-leap when the number of blocks goes to one, and it has already been show that ER-leap samples the correct distribution, we are required to test across a large variety of reaction channel quantities and organization structure.

The reaction networks for which we know the analytical solution involve at most two species types. However, simulating many replicates of these networks on a grid, not connected with diffusion, will allow us to treat each block as an independent sample. We can then treat the simulation of many network replicates as many sampled trajectories of a single network.

We perform tests over a number of network replicates $m = 2 \dots 1000$. The number of blocks range from $b = 1 \dots m$. The leap is in the range $L = 3 \dots 18$, where the leap used depends on the specific reaction network, m and b, but is held constant throughout the simulation.

CaliBayes models 1-01, 1-03, 1-04, 2-01, 2-02, 3-01 and 3-02 [16] are tested, on the spaced defined by the Cartesian product of the possible values for the m, b and L parameters as described above, for parameters which result in an acceptance probability greater than about 0.05. These tests pass on these cases using the criteria of section 3.3.1.

4.3.2 Acceleration

Being an exact algorithm, the key performance metric of relevance to HiER-leap is the amount of acceleration achievable. As discussed earlier, in principle, adding more reaction channels and processors should increase the relative speedup over SSA. We can see this trend experimentally in figure 4.1 and figure 4.2.

The following tests are all run on an Apple Macintosh Pro with a Quad-Core Intel Xeon processes running a total of 8 cores at 2.26 GHz and 13 GB of RAM using OS X 10.6.8. The algorithms are coded in C++ and *Boost.Thread* [29] and the Intel Threading Building Blocks [2] are used for multithreading. We compiled the code using the LLVM compiler 1.0.2. The HiER-leap code may be found at *http://computableplant.ics.uci.edu/hierleap/*.

4.3.3 HiER-Leap Properties

The algorithm parameters, such as leap and hierarchical organization, require tweaking before the fastest possible execution time is achieved. To find the ideal methods with which to optimize our algorithm, we explore various trade-offs here.

In looking at figure 4.3 we observe that the optimal b and L are interdependent for a given network. However, it is interesting to note that for this experiment, there is a relatively large plateau of nearly equivalent optimal running times. This means that the range of reasonably good parameters is large. Furthermore, the contour plot of figure 4.3 indicates that there is only one global optimum. This seemingly convex behavior indicates that finding the optimum should require only a simple hill climbing algorithm.

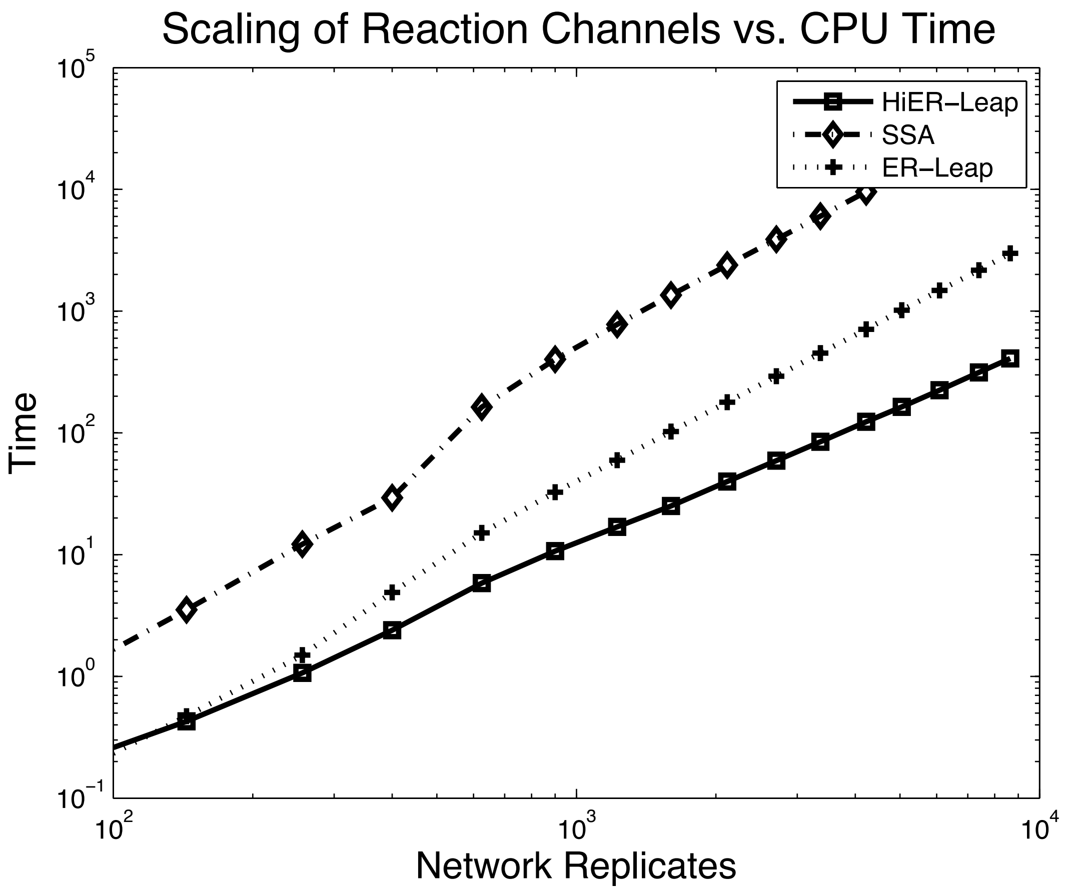

Figure 4.1: The Williamoski-Rossler model as seen in section 3.3.3 is used for this experiment. There are different number of network replicates on a 2D square grid with diffusion rate of 0.1. The number of replicates ranges from $4\ldots8649$ which equates to $64\ldots189612$ reaction channels.

The results from figure 4.3 indicate that finding the optimal L and hierarchical organization for a spatially distribution system is an easy optimization problem. These results, and those from ER-leap, suggest that L will always have a local optimum that is also a global optimum. However, the optimal configuration of the blocks and reaction channels for networks not specifically representing a spatial distribution remains an open problem.

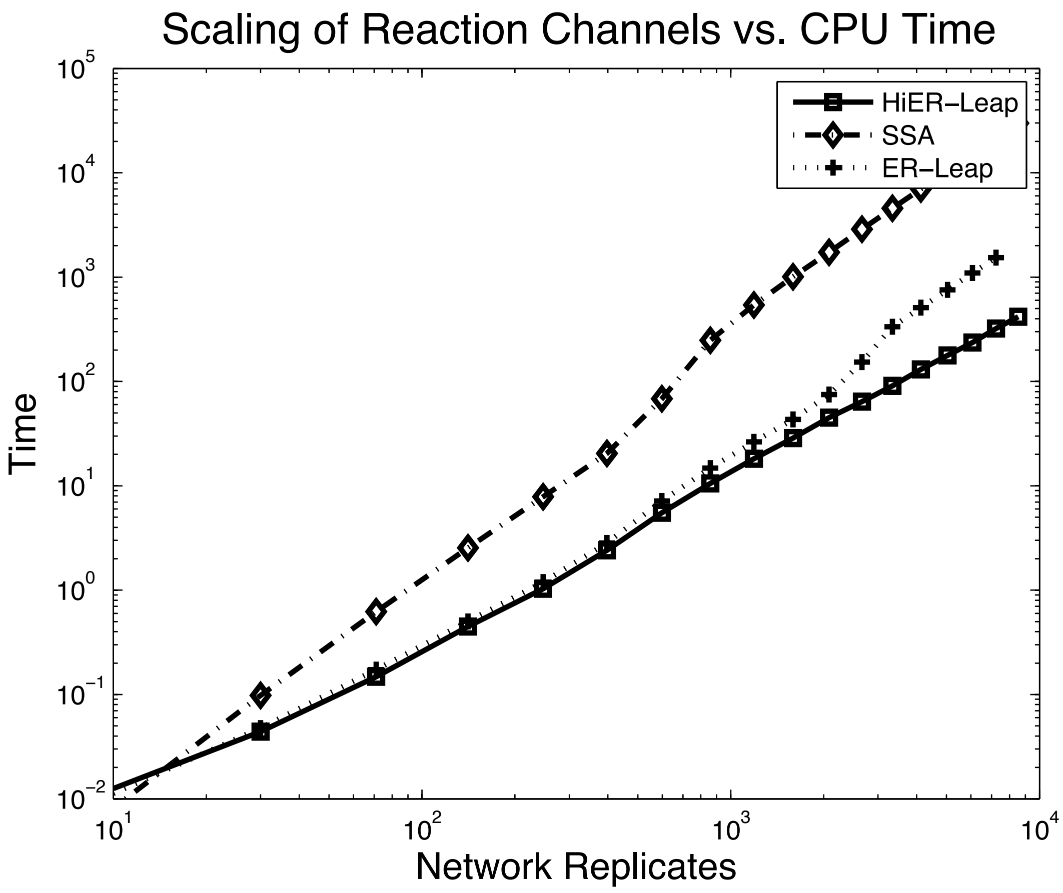

Figure 4.2: The same experimental setup as used for figure 4.1 except 1D diffusion is used.

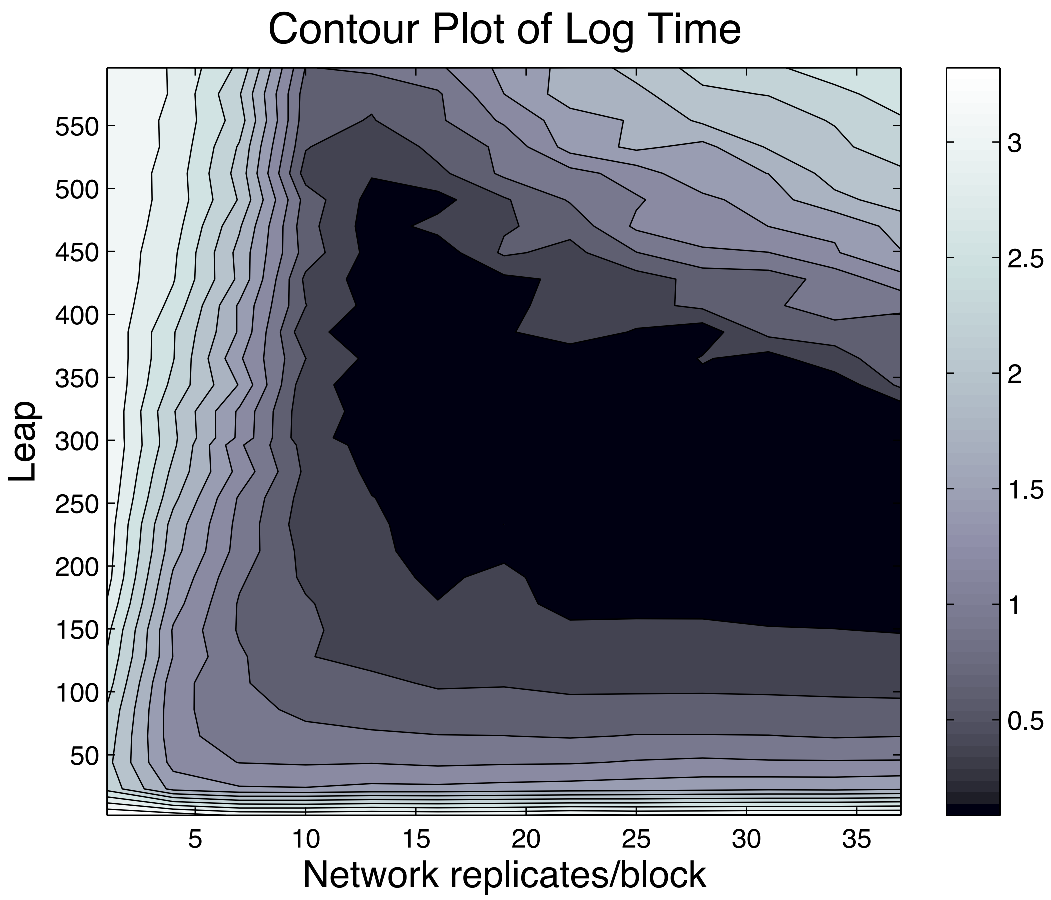

Figure 4.3: The Williamoski-Rossler model as seen in section 3.3.3 is used for this experiment. There are 400 network replicates on a 1D grid with diffusion rate of 0.1. The model execution time depends on leap and hierarchical organization. As leap increases the amount of work per iteration goes up but the acceptance ratio goes down. Furthermore, if there are many reaction channels per block the total acceptance probability of the system goes down. However, in this situation the inner-block acceptance probability goes up. When the number of reaction channels per block goes down, the opposite trends occur. In this way the chosen leap and block organization will determine the total execution time.

4.4 Summary

We have presented a novel algorithm which has demonstrated an ability to sample from the CME without a loss of accuracy. Due to the hierarchical design, this method scales very well with the number of reaction channels and simultaneously takes advantage of parallel hardware for single trajectory samples. Our literature search suggests this is the first algorithm to do both of those tasks and is therefore of potential significance to the computational biology community.

Open questions and future work abound. For example, it is not know how well this method works on 'real networks' of substantial complexity. We believe that modular structure in biological networks will make this method particularly useful. Additionally, it is unknown how substantial increases in the number of cores of next generation CPUs will increase performance.

Chapter 5

Conclusion

We have derived two new algorithms to accelerate SSA sampling, and validated them with computer experiments.

Our first algorithm, Exact R-Leap or ER-Leap, is the first known algorithm to exactly accelerate SSA in time sub-linear to the number of reaction events. This algorithm is derived from the Chemical Master Equation and is accelerated using rejection sampling with an early acceptance step. It has been commonly assumed that approximate methods are sufficient for systems with a large number of molecules. However, our experiments in section 3.3.3 demonstrate otherwise. This makes the ER-Leap algorithm relevant to simulation and inference on virtually any reaction network that is representable by chemically reacting species.

Our work next generalizes ER-Leap to a hierarchical version, HiER-Leap. HiER-Leap begins with the Chemical Master Equation and algebraically arrives at an algorithm which is naturally suitable for parallelization. For each iteration of sampling L reaction events HiER-Leap uses multiple independent acception/rejection steps, one per reactions block. In this it is unlike ER-Leap which has one acception/rejection step per iteration. By essentially ‘factoring out’ the accept/reject steps we are able to use a larger L with greater efficiency than that found in ER-Leap. In fact, the experiment shown in figure 4.1 demonstrates a scenario where HiER-Leap is around 70x faster than SSA and 5.1x faster than ER-Leap.

The reasons for the speedup are twofold. First, by taking advantage of loosely coupled modules in the reaction network, we are able to achieve much tighter bounds on individual reaction channel propensities. This is especially important when the number of molecules is small and the difference between L and the number of inner-block events is large. Additionally, when considering network-wide dynamics, we only need to bound block propensities. We have found an algorithm to solve for these bounds, which results in the width of the block level propensity bound on the order of $O(L)$. This is independent of the number of reaction channels and leads to greater acceleration when the number of reaction channels goes to infinity. Also implied is that for fixed L the global level acception/rejection step will accept with very high probability. This means most of the work will be done on the inner-block level, leading to the second reason for speedup. Secondly, HiER-Leap allows us to use parallel hardware on an algorithm which has traditionally been serially implemented. When the number of reaction channels goes to infinity, the majority of work is done independently for each block. Since these are independent operations, we can theoretically increase performance linearly when increasing the number of processors up to the total number of blocks.

The wide variety of possible network topologies presents several opportunities for future work. Currently little is known about how to organize reaction channels that are not on a spatial grid to achieve maximal speedup using HiER-Leap. It may be possible to infer optimal topologies using concepts from information theory. For example, it is likely desirable for reaction channels that are tightly coupled to be in the same block to achieve high efficiency. Mostly uncorrelated reaction channels, on the other hand, do not need to know much about distant channels in order to sample from the correct distribution. This idea might be the basis for hierarchy optimization. Additionally, some work has sped up SSA using the notion of multiscale simulation [9]. In this work one makes a distinction between fast and slow reactions. The fast reactions are sped up using approximation and the slow reactions are simulated accurately. Our HiER-Leap algorithm may be amenable to this optimization. We may then be able to introduce a two-tiered hierarchy: one tier separates reactions based on speed and the next separates reactions based on proximity to one another similar, to HiER-Leap. These ideas are speculation at this point, although the possibilities are interesting.

Additionally, there are potential "tweaks" to HiER-Leap which could lead to an even more accelerated algorithm. For example, if two adjacent blocks are sampled and neither one changes a species relevant to the propensities in the other one, we should be able to accept/reject them independently. This concept may be difficult to translate into the algorithm, and it is currently unknown how to prove this algebraically. However, further improvements in parallelization and a higher acceptance ratio could be found if this were true.

The algorithms we have presented speed up SSA in a way, using rejection sampling, that has not been seen before. The proofs and experimental validation with CaliBayes show correctness. In the limit of many reaction channels or many molecular species, we are guaranteed compounding acceleration over SSA. It is our hope that these algorithms will help make biological modeling of highly complex systems a reality.

Bibliography

[1] WHO global database on body mass index (BMI): An interactive surveillance tool for monitoring nutrition transition. *Public Health Nutrition*, 9(5805):658–660, 2006.

[2] Thread building blocks. http://threadingbuildingblocks.org/, last accessed May 2012.

[3] H. Alper, J. Moxley, E. Nevoigt, G. R. Fink, and G. Stephanopoulos. Engineering yeast transcription machinery for improved ethanol tolerance and production. *Science*, 314(5805):1565–1568, 2006.

[4] A. Auger, P. Chatelain, and P. Koumoutsakos. R-leaping: Accelerating the stochastic simulation algorithm by reaction leaps. *The Journal of Chemical Physics*, 125(8):084103–+, Aug. 2006.

[5] B. Bayati, P. Chatelain, and P. Koumoutsakos. D-leaping: Accelerating stochastic simulation algorithms for reactions with delays. *J. Comput. Phys.*, 228(16):5908–5916, Sept. 2009.

[6] K. P. Bayati B., Chatelain P. Multiresolution stochastic simulations of reaction-diffusion processes. *Physical Chemistry Chemical Physics*, 10:5963–5966, 2008.

[7] F. R. Blattner, G. Plunkett, C. A. Bloch, N. T. Perna, V. Burland, M. Riley, J. Collado-Vides, J. D. Glasner, C. K. Rode, G. Mayhew, F., J. Gregor, N. W. Davis, H. A. Kirkpatrick, M. A. Goeden, D. J. Rose, B. Mau, and Y. Shao. The complete genome sequence of *Escherichia coli* K-12. *Science*, 277(5331):1453–1462, Sept. 1997.

[8] D. Cangelosi. Ssaleaping: Efficient leap condition based direct method variant for the stochastic simulation of chemical reacting system. In *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques*, SIMUTools '10, pages 36:1–36:10, ICST, Brussels, Belgium, Belgium, 2010. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[9] Y. Cao, D. Gillespie, and L. Petzold. Multiscale stochastic simulation algorithm with stochastic partial equilibrium assumption for chemically reacting systems. *J. Comput. Phys.*, 206(2):395–411, 2005.

[10] Y. Cao, D. T. Gillespie, and L. R. Petzold. Accelerated stochastic simulation of the stiff enzyme-substrate reaction. *Journal of Chemical Physics*, 123(14):144917, 2005.

[11] Y. Cao, D. T. Gillespie, and L. R. Petzold. Avoiding negative populations in explicit Poisson tau-leaping. *The Journal of Chemical Physics*, 123(5):054104, 2005.

[12] Y. Cao, D. T. Gillespie, and L. R. Petzold. The slow-scale stochastic simulation algorithm. *J Chem Phys*, 122(1), January 2005.

[13] Y. Cao, D. T. Gillespie, and L. R. Petzold. Efficient step size selection for the tau-leaping simulation method. *The Journal of Chemical Physics*, 124(4), 2006.

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

[15] J. Elf and M. Ehrenberg. Spontaneous separation of bi-stable biochemical systems into spatial domains of opposite phases. *Systems Biology, IEE Proceedings*, 1(2):230 – 236, dec. 2004.

[16] T. W. Evans, C. S. Gillespie, and D. J. Wilkinson. The SBML discrete stochastic models test suite. *Bioinformatics*, 24(2):285–286, January 2008.

[17] A. R. Fersht. Optimization of rates of protein folding: The nucleation-condensation mechanism and its implications. *Proceedings of the National Academy of Sciences of the United States of America*, 92(24):10869–10873, 1995.

[18] A. Gabrieli, P. Demontis, F. G. Pazzona, and G. B. Suffritti. Speeding up simulation of diffusion in zeolites by a parallel synchronous kinetic Monte Carlo algorithm. *Physical Review E - Statistical, Nonlinear and Soft Matter Physics*, 83(5 Pt 2):056705, 2011.

[19] M. A. Gibson and J. Bruck. Efficient exact stochastic simulation of chemical systems with many species and many channels. *J. Phys. Chem. A*, 104(9):1876–1889, March 2000.

[20] D. T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 81(25):2340–2361, 1977.

[21] D. T. Gillespie. A rigorous derivation of the chemical master equation. *Physica A: Statistical Mechanics and its Applications*, 188(13):404 – 425, 1992.

[22] D. T. Gillespie. The chemical Langevin equation. *The Journal of Chemical Physics*, 113(1):297–306, 2000.

[23] D. T. Gillespie. Approximate accelerated stochastic simulation of chemically reacting systems. *J Chem Phys*, 115:1716–1733, 2001.

[24] D. T. Gillespie and L. R. Petzold. Improved leap-size selection for accelerated stochastic simulation. *The Journal of Chemical Physics*, 119(16):8229, 2003.

[25] C. Guldberg and P. Waage. Studies concerning affinity. *C.M. Forhandlinger: Videnskabs-Selskabet i Christiana*, 35, 1864.

[26] A. Hintze and C. Adami. Evolution of complex modular biological networks. *PLoS Comput Biol*, 4(2):e23, 02 2008.

[27] J. Hirsch and R. L. Leibel. The genetics of obesity. *Hosp. Pract. (Minneap.)*, 33(3):55–9, 62–5, 69–70 passim, 1998.

[28] D. D. Jenkins and G. D. Peterson. AESS: Accelerated exact stochastic simulation. *Computer Physics Communications*, 182(12):2580 – 2586, 2011.

[29] B. Kempf. The boost.threads library. *C/C++ Users Journal*, 20(5), May 2002.

[30] G. Klingbeil, R. Erban, M. Giles, and P. K. Maini. STOCHSIMGPU: Parallel stochastic simulation for the Systems Biology Toolbox 2 for Matlab. *Bioinformatics*, 27(8):1170–1171, 2011.

[31] H. Li and L. Petzold. Efficient parallelization of the stochastic simulation algorithm for chemically reacting systems on the graphics processing unit. *Int. J. High Perform. Comput. Appl.*, 24(2):107–116, May 2010.

[32] H. Lu and P. Li. Stochastic projective methods for simulating stiff chemical reacting systems. *Computer Physics Communications*, 183(7):1427 – 1442, 2012.

[33] U. Maskos and E. M. Southern. Oligonucleotide hybridizations on glass supports: A novel linker for oligonucleotide synthesis and hybridization properties of oligonucleotides synthesised in situ. *Nucleic Acids Res*, 20(7):1679–84.+, 1992.

[34] S. Maslov and K. Sneppen. Specificity and stability in topology of protein networks. *Science*, 296(5569):910–913, 2002.

[35] A. Matouschek, J. T. Kellis, L. Serrano, and A. R. Fersht. Mapping the transition state and pathway of protein folding by protein engineering. *Nature*, 340(6229):122–126, 1989.

[36] G. N. Milstein. Numerical Integration of Stochastic Differential Equations. In *Math. Appl. 313.* Kluwer Academic, Dordrecht, The Netherlands, 1995.

[37] E. Mjolsness, D. Orendorff, P. Chatelain, and P. Koumoutsakos. An exact accelerated stochastic simulation algorithm. *The Journal of chemical physics*, 130(14):144110, 2009.

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

[39] M. D. Riedel and J. Bruck. Exact stochastic simulation of chemical reactions with cycle leaping. Technical report, California Institute of Technology, 2006.

[40] D.-K. Ro, E. M. Paradise, M. Ouellet, K. J. Fisher, K. L. Newman, J. M. Ndungu, K. A. Ho, R. A. Eachus, T. S. Ham, J. Kirby, M. C. Y. Chang, S. T. Withers, Y. Shiba, R. Sarpong, and J. D. Keasling. Production of the antimalarial drug precursor artemisinic acid in engineered yeast. *Nature*, 440(7086):940–943, 2006.

[41] S. Rogers and M. Girolami. A Bayesian regression approach to the inference of regulatory networks from gene expression data. *Bioinformatics*, 21(14):3131–3137, July 2005.

[42] D. Rossinelli, B. Bayati, and P. Koumoutsakos. Accelerated stochastic and hybrid methods for spatial simulations of reaction-diffusion systems. *Chemical Physics Letters*, 451(1-3):136 – 140, 2008.

[43] H. Salis and Y. N. Kaznessis. An equation-free probabilistic steady-state approximation: Dynamic application to the stochastic simulation of biochemical reaction networks. *The Journal of Chemical Physics*, 123(21):214106, 2005.

[44] A. Samant and D. G. Vlachos. Overcoming stiffness in stochastic simulation stemming from partial equilibrium: A multiscale Monte Carlo algorithm. *The Journal of Chemical Physics*, 123(14):144114, 2005.

[45] Y. Shim and J. G. Amar. Synchronous sublattice algorithm for parallel kinetic Monte Carlo. *eprint arXiv:cond-mat/0406379*, June 2004.

[46] Y. Shim and J. G. Amar. Rigorous synchronous relaxation algorithm for parallel kinetic Monte Carlo simulations of thin film growth. *Phys. Rev. B*, 71:115436, Mar 2005.

[47] A. Slepoy, A. P. Thompson, and S. J. Plimpton. A constant-time kinetic monte carlo algorithm for simulation of large biochemical reaction networks. *The Journal of Chemical Physics*, 128(20):205101, 2008.

[48] D. Soloveichik, M. Cook, E. Winfree, and J. Bruck. Computation with finite stochastic chemical reaction networks. *Natural Computing*, 7:615–633, 2008. 10.1007/s11047-008-9067-y.

[49] M. Ullah and O. Wolkenhauer. A family tree of Markov models in systems biology. *IET SYST BIOL.*, 1:247, 2007.

[50] N. G. Van Kampen. *Stochastic Processes in Physics and Chemistry, Third Edition (North-Holland Personal Library)*. North Holland, April 2007.

[51] J. von Neumann. *Various Techniques Used in Connection with Random Graphs*, pages 36–38. Washington: US Government Printing Office, 1951.

[52] H. Wang and Q. Li. Master equation analysis of deterministic chemical chaos. *The Journal of Chemical Physics*, 108(18):7555–7559, 1998.

[53] Y. Wang, S. Christley, E. Mjolsness, and X. Xie. Parameter inference for discretely observed stochastic kinetic models using stochastic gradient descent. *BMC Systems Biology*, 4(1):99+, 2010.

[54] E. Weinan, D. Liu, and E. V. Eijnden. Nested stochastic simulation algorithm for chemical kinetic systems with disparate rates. *The Journal of Chemical Physics*, 123(19), 2005.

[55] D. J. Wilkinson. *Stochastic Modelling for Systems Biology (Mathematical and Computational Biology)*. Chapman & Hall/CRC, April 2006.

[56] L. Xu, M. Taufer, S. Collins, and D. G. Vlachos. Parallelization of tau-leap coarse-grained Monte Carlo simulations on GPUs. In *24th IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2010, Atlanta, Georgia, USA, 19-23 April 2010 - Conference Proceedings*, pages 1–9. IEEE, 2010.

[57] G. Yosiphon and E. Mjolsness. Towards the inference of stochastic biochemical network and parameterized grammar models. In N. D. Lawrence, M. Girolami, M. Rattray, and G. Sanguinetti, editors, *Learning and Inference in Computational Systems Biology*. MIT Press, 2010.

[58] W. Zhou, X. Peng, Z. Yan, and Y. Wang. Accelerated stochastic simulation algorithm for coupled chemical reactions with delays. *Computational Biology and Chemistry*, 32(4):240–242, 2008.

Appendices

A Appendix A

We will show that for $\Delta\widehat{D}^{r_1}_{(I_0L)}{}^*$ from equation 4.6 and $\Delta\widehat{D}^{r_1}_{(I_0L)}$ from equations 4.7 and 4.8, it is the case that $\Delta\widehat{D}^{r_1}_{(I_0L)}{}^* \leq \Delta\widehat{D}^{r_1}_{(I_0L)}$.

Proof by contradiction. Assume there is some $r_2 \in r_1$ and state $I' = \mathbf{n'}$ with $\forall_a n'_a \leq \tilde{n}_a$ and $\exists_a n'_a < \tilde{n}_a$, reachable from I_0 in at most $L-1$ reaction events used to find $\Delta\widehat{D}^{r_1}_{(I_0L)}{}^*$ such that $\Delta\widehat{D}^{r_1}_{(I_0L)}{}^* > \Delta\widehat{D}^{r_1}_{(I_0L)}$. Plugging in our definitions for $\Delta\widehat{D}^{r_1}_{(I_0L)}{}^*$ and $\Delta\widehat{D}^{r_1}_{(I_0L)}$, using equation 4.1, and introducing the notation that $I(r_2)$ will be the results of r_2 applied to I and $I({r_2}^+)$ is the result of r_2 applied to I only for species which have net gain ($\Delta m_a^{(r_1r_2)} > 0$), yeilds

$$\begin{aligned}
\Delta\widehat{D}^{r_1}_{(I_0L)}{}^* &= D^{r_1}_{I'(r_2)} - D^{r_1}_{I'} \\
&= \sum_{r''_2 \in r_1} \rho_{(r_1r''_2)} F^{(r_1r''_2)}_{I'(r_2)} - \sum_{r''_2 \in r_1} \rho_{(r_1r''_2)} F^{(r_1r''_2)}_{I'} \\
&= \sum_{r''_2 \in r_1} \rho_{(r_1r''_2)} \left(F^{(r_1r''_2)}_{I'(r_2)} - F^{(r_1r''_2)}_{I'} \right)
\end{aligned}$$

and

$$\Delta \widehat{D}^{r_1}_{(I_0 L)} = \sum_{r_2'' \in r_1} \rho_{(r_1 r_2'')} \left(F^{(r_1 r_2'')}_{\tilde{I}(r_2+)} - F^{(r_1 r_2'')}_{\tilde{I}} \right).$$

Therefore, we can equivalently say that we are trying to disprove

$$\sum_{r_2'' \in r_1} \rho_{(r_1 r_2'')} \left(F^{(r_1 r_2'')}(\mathbf{n}' + \mathbf{\Delta m}^{(r_1 r_2)}) - F^{(r_1 r_2'')}(\mathbf{n}') \right) > \sum_{r_2'' \in r_1} \rho_{(r_1 r_2'')} \left(F^{(r_1 r_2'')}(\tilde{\mathbf{n}} + \mathbf{\Delta m}^{(r_1 r_2)}) - F^{(r_1 r_2'')}(\tilde{\mathbf{n}}) \right). \quad \text{(A.1)}$$

Note that by grouping terms by r_2'', there is a one-to-one correspondence between the summation terms on each side of the inequality.

If true, equation A.1 implies that there is at least one reaction channel $r_2' \in r_1$ for $\Delta m_a^{(r_1 r_2)} > 0$ such that

$$F^{(r_1 r_2')}(\mathbf{n}' + \mathbf{\Delta m}^{(r_1 r_2)}) - F^{(r_1 r_2')}(\mathbf{n}') > F^{(r_1 r_2')}(\tilde{\mathbf{n}} + \mathbf{\Delta m}^{(r_1 r_2)}) - F^{(r_1 r_2')}(\tilde{\mathbf{n}}) \quad \text{(A.2)}$$

but we will show that this is impossible for any $\tilde{n}_a > n_a' \geq 0$. Note that we do not need to consider $\Delta m_a^{(r_1 r_2)} \leq 0$ because F is monotonic, the LHS will be decreased and the RHS will not change as per the definition of $\Delta \widehat{D}^{r_1}_{(I_0 L)}$ (negative $\Delta m_a^{(r_1 r_2)}$ are ignored).

Before proceeding we will introduce the forward difference operator, $\Delta_{F(i)}$, such that

$$\Delta_{F(i)} f(z) \equiv f(z+i) - f(z) \quad \text{(A.3)}$$

for any function $f(z)$.

Furthermore, $F^{(r_1 r_2)}(\mathbf{n})$ can be decomposed by species into terms including chemical species C_a and those which do not. Following from section 3.2.3, this allows us to rewrite $F^{(r_1 r_2)}(\mathbf{n})$ as

$$F^{(r_1 r_2')}(\mathbf{n}) = G^{(r_1 r_2')}(\mathbf{n}\backslash\{n_a\}) \times (n_a)_k$$

for some constant $G^{(r_1 r_2)}(\mathbf{n}\backslash\{n_a\}) \geq 0$ which does not depend on n_a, where $k = m_a^{r_1 r_2'}$ is the input stoichiometry for reaction r_2' and species C_a, and

$$(n)_k \equiv \frac{n!}{(n-k)!}.$$

For equation A.2 to be true there must exist a C_a such that

$$F^{(r_1 r_2')}(n_a' + \Delta m_a^{r_1 r_2}) - F^{(r_1 r_2')}(n_a') > F^{(r_1 r_2')}(\tilde{n}_a + \Delta m_a^{r_1 r_2}) - F^{(r_1 r_2')}(\tilde{n}_a) \tag{A.4}$$

is true. All of the above $F^{(r_1 r_2)}$ are calculated using $n_b = \mathbf{n}'\backslash\{n_a\}$ and $n_a \in \{n_a', \tilde{n}_a\}$. When we show that n_a' will not result in a greater delta than that offered by using $\tilde{n}_a$ instead, this implies that equation A.2 may never be true.

Equivalent to equation A.4, by dividing out $G^{(r_1 r_2)}(\mathbf{n}') \geq 0$, using equation A.3, and setting $m = \Delta m_a^{r_1 r_2}$ we arrive at

$$\Delta_{F(m)}(\tilde{n}_a)_k - \Delta_{F(m)}(n_a')_k < 0. \tag{A.5}$$

However, because $n_a' < \tilde{n}_a$, if it is shown that $\Delta_{F(m)}(n)_k$ is monotonic in n then this will imply equation A.5 is false.

Therefore, it just remains to be shown that $\Delta_{F(m)}(n)_k$ is monotonic in n. Consider the following equation which tests for monotonicity

$$\begin{aligned}
&\Delta_{F(m)}(n+1)_k - \Delta_{F(m)}(n)_k \\
&\qquad = \left[\Delta_{F(1)}(n+m)_k + \ldots + \Delta_{F(1)}(n+1)_k\right] - \\
&\qquad\qquad \left[\Delta_{F(1)}(n+m-1)_k + \ldots + \Delta_{F(1)}(n)_k\right] \\
&\qquad = \Delta_{F(1)}(n+m)_k - \Delta_{F(1)}(n)_k \\
&\qquad = k(n+m)_{k-1} - k(n)_{k-1} \\
&\qquad = k\left[\frac{(n+m)!}{(n+m-k+1)!} - \frac{n!}{(n-k+1)!}\right] \\
&\qquad = k\frac{n!}{(n-k+1)!}\left[\frac{n+m}{n+m-k+1} \times \ldots \times \frac{n+1}{n-k+2} - 1\right] \\
&\qquad \geq 0
\end{aligned}$$

because $k \geq 1$ implies every factor in the long product is ≥ 1. This implies monotonicity. Therefore equation A.4 is false for all C_a, implying equations A.2 is false.

□

Proof by contradiction. Assume there is some $0 \leq n'_a < \tilde{n}_a$ and reaction r_2 such that $\Delta D^{r_1}_{r_2}(n'_a) > \Delta D^{r_1}_{r_2}(\tilde{n}_a)$.

That means there is at least on reaction channel r'_2 for which

$$F^{(r_1 r'_2)}(n'_a + \Delta m_a^{(r_1 r_2)}) - F^{(r_1 r'_2)}(n'_a) > F^{(r_1 r'_2)}(\tilde{n}'_a + \Delta m_a^{(r_1 r_2)}) - F^{(r_1 r'_2)}(\tilde{n}_a)$$

where, assuming at most two reactants, there are three possible forms for $F^{(r_1 r'_2)}(\ldots)$.

If F is of the form

$$1 : \varnothing \to X$$

then the propensity is independent of n_a and therefore is constant with respect to n_a.

If F is of the form:

$$2 : C_a + C_b \to Z$$

then we have

$$\begin{aligned}
\Delta F^{(r_1 r_2')}(n_a') &> \Delta F^{(r_1 r_2')}(\tilde{n}_a)\\
(n_a' + \Delta m_a^{(r_1 r_2)})(n_b + \Delta m_b^{(r_1 r_2)}) - n_a' n_b &> (\tilde{n}_a + \Delta m_a^{(r_1 r_2)})(\tilde{n}_b + \Delta m_b^{(r_1 r_2)}) - \tilde{n}_a \tilde{n}_b\\
n_a' \Delta m_b^{(r_1 r_2)} + \Delta m_a^{(r_1 r_2)} n_b + \Delta m_a^{(r_1 r_2)} \Delta m_b^{(r_1 r_2)} &> \tilde{n}_a \Delta m_b^{(r_1 r_2)} + \Delta m_a^{(r_1 r_2)} \tilde{n}_b + \Delta m_a^{(r_1 r_2)} \Delta m_b^{(r_1 r_2)}\\
n_a' \Delta m_b^{(r_1 r_2)} + \Delta m_a^{(r_1 r_2)} n_b &> \tilde{n}_a \Delta m_b^{(r_1 r_2)} + \Delta m_a^{(r_1 r_2)} \tilde{n}_b
\end{aligned}$$

for m constant and supposing $n_a' < \tilde{n}_a$ and $n_b \leq \tilde{n}_b$. However, we know $\tilde{n}_a > n_a'$ by construction, therefore this statement cannot be true.

Finally, F could be of the form

$$3 : C_a + C_a \to X$$

which represents a dimerization process. Setting $q = \Delta m_a^{(r_1 r_2)}$ for notational brevity this implies that

$$\begin{aligned}
\Delta F^{(r_1 r_2')}(n_a') &> \Delta F^{(r_1 r_2')}(\tilde{n}_a)\\
(n_a' + q)(n_a' + q - 1) - n_a'(n_a' - 1) &> (\tilde{n}_a + q)(\tilde{n}_a + q - 1) - \tilde{n}_a(\tilde{n}_a - 1)\\
n_a' q + q n_a' + q^2 - q &> \tilde{n}_a q + q \tilde{n}_a + q^2 - q\\
2 n_a' q &> 2 \tilde{n}_a q
\end{aligned}$$

which is impossible when $\tilde{n}_a > n'_a$ and $q \geq 0$ is constant.

Since none of the three possible scenarios support the assumed hypothesis, the proposition that our supposition is false has been proven false. $\square$