The sorting direct method for stochastic simulation of biochemical systems with varying reaction execution behavior

ARTICLE *in* COMPUTATIONAL BIOLOGY AND CHEMISTRY · MARCH 2006

Impact Factor: 1.12 · DOI: 10.1016/j.compbiolchem.2005.10.007 · Source: PubMed

5 AUTHORS, INCLUDING:

Gregory Peterson
University of Tennessee
118 PUBLICATIONS **881** CITATIONS
SEE PROFILE

Chris D Cox
University of Tennessee
49 PUBLICATIONS **1,387** CITATIONS
SEE PROFILE

Mike Simpson
Oak Ridge National Laboratory
243 PUBLICATIONS **5,307** CITATIONS
SEE PROFILE

Nagiza F Samatova
North Carolina State University
153 PUBLICATIONS **1,716** CITATIONS
SEE PROFILE

Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computational Biology and Chemistry
Computational Biology and Chemistry 30 (2006) 39–49
www.elsevier.com/locate/compbiolchem

The sorting direct method for stochastic simulation of biochemical systems with varying reaction execution behavior

James M. McCollum [a,b,c], Gregory D. Peterson [b,c], Chris D. Cox [c,d], Michael L. Simpson [c,e,f], Nagiza F. Samatova [a,*]

[a] *Computational Biology Institute, Oak Ridge National Laboratory, P.O. Box 2008 MS6164, Oak Ridge, TN 37831, USA*
[b] *Department of Electrical and Computer Engineering, University of Tennessee, 414 Ferris Hall, Knoxville, TN 37996-2100, USA*
[c] *Center for Environmental Biotechnology, University of Tennessee, 676 Dabney Hall, Knoxville, TN 37996-1605, USA*
[d] *Department of Civil and Environmental Engineering, University of Tennessee, 233 Perkins Hall, Knoxville, TN 37996-2010, USA*
[e] *Molecular Scale Engineering and Nanoscale Technologies Research Group, Oak Ridge National Laboratory, P.O. Box 2008 MS6006, Oak Ridge, TN 37831, USA*
[f] *Department of Material Science and Engineering, University of Tennessee, 434 Dougherty Hall, Knoxville, TN 37996-2200, USA*

Received 27 June 2005; received in revised form 10 October 2005; accepted 10 October 2005

Abstract

A key to advancing the understanding of molecular biology in the post-genomic age is the development of accurate predictive models for genetic regulation, protein interaction, metabolism, and other biochemical processes. To facilitate model development, simulation algorithms must provide an accurate representation of the system, while performing the simulation in a reasonable amount of time. Gillespie's stochastic simulation algorithm (SSA) accurately depicts spatially homogeneous models with small populations of chemical species and properly represents noise, but it is often abandoned when modeling larger systems because of its computational complexity. In this work, we examine the performance of different versions of the SSA when applied to several biochemical models. Through our analysis, we discover that transient changes in reaction execution frequencies, which are typical of biochemical models with gene induction and repression, can dramatically affect simulator performance. To account for these shifts, we propose a new algorithm called the sorting direct method that maintains a loosely sorted order of the reactions as the simulation executes. Our measurements show that the sorting direct method performs favorably when compared to other well-known exact stochastic simulation algorithms.

Keywords: Stochastic simulation; Modeling biochemical systems; Gillespie algorithm; Gene networks; Systems biology

1. Introduction

During the last 20 years, high-throughput DNA sequencing technology has made possible the complete sequencing of hundreds of genomes. However, genomic sequence alone is insufficient to predict the phenotype of an organism. Rather it is the processing of environmental cues through various regulatory mechanisms at the transcriptional, translational, metabolic, and systematic levels that gives rise to the overall behavior (Strohman, 2002). The emerging discipline of systems biology attempts to examine these various levels of regulation in an integrated fashion (Ideker et al., 2001), often through use of mathematical models in conjunction with experiments. Chemical kinetic models, in which the behavior of the biological system is represented as a system of individual chemical reactions, are among the most commonly used.

The utility of chemical kinetic models within the biomedical research community has been repeatedly demonstrated. Recent work includes models of mammalian circadian clocks (Forger and Peskin, 2003, 2004, 2005), CD95-induced apoptosis (Bentele et al., 2004), cell cycle regulation in yeast (Stelling and Gilles, 2004), and quorum sensing in *Vibrio fischeri* (Cox et al., 2003). These examples represent only a subset of the analysis that has been enabled by biochemical predictive models and the role of such models in the biological research process is clearly increasing. Providing accurate and efficient general-purpose simulators and mathematical formulations that enable

* Corresponding author. Tel.: +1 865 241 4351; fax: +1 865 576 5491.
E-mail address: samatovan@ornl.gov (N.F. Samatova).

1476-9271/$ – see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compbiolchem.2005.10.007

the exploitation of these techniques will lead to their further adoption and use.

The most common mathematical formulation used for biochemical system modeling is deterministic. This approach treats chemical species as continuous valued concentrations and represents chemical interactions as ordinary differential equations. Stiff differential equation solvers like ODEPack (Hindmarsh, 2001), Matlab (Shampine and Reichelt, 1997), and Jarnac (Sauro et al., 2003) can be used to perform these simulations and their execution time is typically short.

While deterministic solutions are sufficiently accurate to capture the dynamics of systems with relatively large populations of the various chemical species, biological cells with length scales on the order of microns often contain populations of certain species on the order of 10–100. In these cases, the stochastic fluctuations of the molecular populations may result in phenotypic variation among a population of genetically identical cells exposed to uniform environmental conditions. Cells have evolved to either tolerate, or in some cases exploit, the noisy intracellular environment resulting from these stochastic fluctuations. These possibilities have stimulated a body of research on the dynamics and role of noise in regulatory networks in recent years (see Rao et al., 2002 and Kærn et al., 2005 for two recent reviews). To properly model noise-affected systems, a stochastic formulation that represents species as discrete-valued populations and chemical interactions (reactions) as random processes has been employed.

The dynamics of the stochastic formulation are described by the chemical master equation (CME) (Gillespie, 1992). For most biochemical system models, the CME is an infinite set of ordinary differential equations and is difficult to solve analytically. In such cases, it is common to predict the behavior of the CME using a Monte-Carlo simulation technique first developed by Gillespie called the stochastic simulation algorithm (SSA) (Gillespie, 1976). The SSA stochastically predicts the execution of each individual reaction event for the given system. Larger systems, such as whole-cell or multi-cell models that require the execution of billions of reactions, can take an unreasonable amount of time to simulate using the SSA (Endy and Brent, 2001). Compounding this performance problem is the fact that the algorithm only predicts a single potential time evolution (random-walk) of the chemically reacting system. Typically, multiple simulations with different initial random seeds are required to collect the data and statistics necessary to elucidate the system behavior. Improving the performance of the SSA is therefore critical to expanding the applicability of this technique.

Gillespie's original versions of the SSA are the first reaction method (FRM) and the direct method (DM) (Gillespie, 1977). Several mathematically equivalent algorithms have been proposed to help relieve some of the computational demands of the SSA. These algorithms include the next reaction method (NRM) by Gibson and Bruck (2000) and the optimized direct method (ODM) by Cao et al. (2004). Section 2 of this paper reviews each of these algorithms in detail. Section 3 analyzes their performance when applied to the simulation of several biochemical system models, specifically quorum sensing in *Vibrio fischeri* (Cox et al., 2003), the lysogeny decision in λ-phage (Arkin et al., 1998), and heat stress response in *Escherichia coli* (Kurata et al., 2001).

Through this analysis we discover that the performance of the ODM, currently believed to be the fastest known algorithm for stochastic simulation for most biological problems, can be negatively impacted by transient shifts in the frequency at which reactions occur. This behavior is common to biochemical reaction networks because of the inductive and repressive nature of genetic regulation. These shifts defeat the pre-simulation strategy employed within the ODM to reduce the time complexity of the SSA's reaction selection stage and thus degrade performance.

To reduce this degradation, we present an approach for handling reaction occurrence shifts by implementing an efficient sorting heuristic as the simulation executes. This approach, called the sorting direct method (SDM), eliminates the pre-simulations required by the ODM and allows the simulator to adapt to sharp changes in reaction execution frequencies. Section 4 provides a detailed description of the SDM and compares the performance of the SDM to current simulation algorithms. The results demonstrate that the SDM performs better or no worse than the ODM when it is applied to typical biochemical models.

2. Methods and algorithms

Given a vector of M molecular species $S = \{S_1, \ldots, S_M\}$ with initial species populations $X(t_0) = \{X_1(t_0), \ldots, X_M(t_0)\}$ and a vector of N chemical reactions $R = \{R_1, \ldots, R_N\}$, the function of the SSA is to predict a time evolution of the species populations $X(t) = \{X_1(t), \ldots, X_M(t)\}$ for a given time period t_0 to t_{end}. The SSA specifies that the system must be spatially homogeneous (well-stirred), allowing the algorithm to ignore the individual position of each molecule and only monitor the populations of the chemical species $X(t)$ over time. Each reaction R_i is defined by a stochastic rate constant k_i and its stoichiometry, represented as vectors of M reactant coefficients $r_i = \{r_{i1}, \ldots, r_{iM}\}$ and M product coefficients $p_i = \{p_{i1}, \ldots, p_{iM}\}$. The probability that a given reaction R_i occurs within a short time period $\mathrm{d}t$ given the current state of the system $X(t)$, can be calculated as

$$P_i(\mathrm{d}t) = \alpha_i(X(t))\,\mathrm{d}t + o(\mathrm{d}t). \tag{1}$$

The function α_i is called the propensity function. This function is the product of the reaction rate constant (adjusted for effects such as temperature and volume) and the number of possible ways the reaction can occur based on the reactant populations. Suppose the system includes the reaction $S_1 + S_2 \rightarrow S_3$ with the rate constant k_i. The propensity function for this reaction would be

$$\alpha_i(X(t)) = k_i X_1(t) X_2(t). \tag{2}$$

For a dimerization reaction of the form $S_4 + S_4 \rightarrow S_5$, with the rate constant k_i, the propensity function would be

$$\alpha_i(X(t)) = k_i \frac{X_4(t)(X_4(t) - 1)}{2}. \tag{3}$$

Table 1
Comparison of SSA versions

	First reaction	Direct	Next reaction	Optimized direct
Initialization	Read the model	Read the model	Read the model and generate DG[a] and IPQ[b]	Read the model, generate DG[a], pre-simulate, and reorder the reactions
Propensity calculation	Calculate propensities for all reactions	Calculate propensities for all reactions	Calculate only affected propensities using DG[a]	Calculate only affected propensities using DG[a]
Reaction time generation	Generate putative times for each reaction	Sum the propensities and generate a reaction time for the system	Generate a new putative time for the last reaction executed and update IPQ[b]	Update the total propensity using DG[a] and generate a reaction time for the system
Reaction selection	Find the minimum putative time using a linear search	Generate a scaled uniform random number and search the propensities using a linear search[c]	Get the minimum putative time, stored at the head of the IPQ[b]	Generate a scaled uniform random number and search the propensities using a linear search[c]

[a] Dependency graph.
[b] Indexed priority queue.
[c] Because the optimized direct method reorders the reactions following the pre-simulation at startup, the search depth required by the reaction selection stage will typically be less than the search depth of the direct method.

The time evolution of the probabilities of each state is defined by the chemical master equation:

$$\frac{\mathrm{d}P(X(t),t)}{\mathrm{d}t} = \sum_{i=1}^{N}[\alpha_i(X(t)+r_i-p_i)P(X(t)+r_i-p_i,t) - \alpha_i(X(t))P(X(t),t)]. \quad (4)$$

Because the chemical master equation is typically an infinite set of ordinary differential equations and is impractical to solve, the SSA uses a Monte-Carlo simulation to predict time evolutions of the system. The SSA consists of several stages that are listed below.

1. *Initialization*. Read the model file and initialize data structures.
2. *Propensity calculation*. Calculate the propensity function for each reaction.
3. *Reaction time generation*. Estimate the occurrence time of the next reaction.
4. *Reaction selection*. Select which reaction will occur next.
5. *Reaction execution*. Update the current simulation time and species variables to reflect the execution of the selected reaction.
6. *Termination.* If the simulation time has not yet reached the desired end time, go to stage 2.

The first reaction method (FRM), the direct method (DM), the next reaction method (NRM), and the optimized direct method (ODM) each handle stages 1–4 in different ways. A summary of these differences is given in Table 1.

2.1. The first reaction method

Fig. 1 outlines the FRM. During each iteration of the main loop (steps 2–6), each reaction propensity is recalculated based on the current values of the species populations (step 2). A randomly generated potential occurrence time is generated for each reaction (step 3) using the following equation:

$$\tau_i = -\frac{\ln(\mathrm{RAND})}{\alpha_i(X(t))} \quad (5)$$

RAND in Eq. (5) is a uniformly distributed random number between zero and one, but excludes zero because of the natural log function. The next reaction to execute is selected (step 4) by performing a linear search of the potential reaction occurrence times to find the reaction with the earliest occurrence time. The species populations are updated by decrementing the species populations of the reactants and incrementing the species populations of the products based on the stoichiometry of the selected reaction. The selected reaction occurrence time is added to the current time to complete step 5. This process is shown in Eqs. (6) and (7), where sel is the index of the reaction selected.

$$X(t+\tau_{\mathrm{sel}}) = X(t) - r_{\mathrm{sel}} + p_{\mathrm{sel}} \quad (6)$$

$$t = t + \tau_{\mathrm{sel}} \quad (7)$$

The loop continues until the desired end time t_{end} is reached.

2.2. The direct method

Because each iteration of the FRM main loop (steps 2–6) requires the generation of N exponentially distributed random numbers, the computation time per step of the algorithm is high and the scalability of the algorithm to models with many reaction channels is limited. To overcome this problem, Gillespie (1977) designed the direct method (DM) that requires the generation of only two random numbers per iteration regardless of the model size. This is accomplished by modifying step 3 of the algorithm to sum the propensities and generate a single reaction occurrence

```
CurrentTime = 0.0                                          1. Initialization
S[1..M] = Initial Species Populations
R[1..N] = Reactions
For I = 1..N                                               2. Propensity Calculation
  Prop[I] = CalcPropensity(S,R[I])
End For
For I = 1..N                                               3. Reaction Time
  T[I] = -ln(rand())/Prop[I]                                  Generation
End For
SelRxn = 1                                                 4. Reaction Selection
MinTime = T[1]
For I = 2..N
  If (MinTime > T[I])
    SelRxn = I
    MinTime = T[I]
  End If
End For
S = S - R[SelRxn].reactants + R[SelRxn].products           5. Reaction Execution
CurrentTime = CurrentTime + MinTime
If (CurrentTime < EndTime)                                 6. Termination
  Goto Propensity Calculation
End If
```

Fig. 1. Pseudo-code for the first reaction method (FRM).

time for the entire system as shown in the following equation:

$$\tau_{\text{next}} = -\frac{\ln(\text{RAND})}{\sum_{i=1}^{N}\alpha_i(X(t))} \tag{8}$$

A uniformly distributed random number, scaled to the total propensity is used to select the reaction r_{sel} in step 4, where sel is the index of the selected reaction, using the equation given in the following equation:

$$\sum_{i=1}^{\text{sel}}\alpha_i(X(t)) \le \text{RAND}\sum_{i=1}^{N}\alpha_i(X(t)) < \sum_{i=1}^{\text{sel}+1}\alpha_i(X(t)) \tag{9}$$

This algorithm is mathematically equivalent to the FRM and reduces the time complexity of generating random numbers from $O(N)$ to $O(1)$. Pseudo-code for the DM is provided in Fig. 2.

2.3. The next reaction method

The performance of exact stochastic simulation was later improved by Gibson and Bruck (2000) with their next reaction method (NRM). Realizing that for most models the execution of a single reaction only affects the value of a few reaction propensities, the NRM builds a reaction "dependency graph" during initialization that describes which propensities must be recalculated when a particular reaction occurs. An example dependency graph is given in Fig. 3.

Employing this graph significantly reduces the number of propensity calculations required by step 2 of the simulation for most biochemical models. A heap structure was also added to maintain a list of potential reaction occurrence times in sorted order to reduce the amount of time required to select which reaction will occur next (step 4). These enhancements significantly

```
CurrentTime = 0.0                                          1. Initialization
S[1..M] = Initial Species Populations
R[1..N] = Reactions
TotalPropensity = 0.0                                      2. Propensity Calculation
For I = 1..N
  Prop[I] = CalcPropensity(S,R[I])
  TotalPropensity = TotalPropensity + Prop[I]
End For
T = -ln(rand())/TotalPropensity                            3. Reaction Time Generation
Selector = TotalPropensity * rand()                        4. Reaction Selection
For I = 1..N
  Selector = Selector - Prop[I]
  If (Selector <= 0)
    SelRxn = I
    Break
  End If
End For
S = S - R[SelRxn].reactants + R[SelRxn].products           5. Reaction Execution
CurrentTime = CurrentTime + T
If (CurrentTime < EndTime)                                 6. Termination
  Goto Propensity Calculation
End If
```

Fig. 2. Pseudo-code for the direct method (DM).

Name	Reaction	Depends on	Affects	Update
R1	A -> B	A	A,B	R1,R2
R2	B -> C	B	B,C	R2,R3
R3	C + D -> E	C,D	C,D,E	R3,R4,R6
R4	E -> E + F	E	F	R5
R5	F -> A	F	A,F	R1,R5
R6	E -> B	E	B,E	R2,R4,R6

Fig. 3. Sample dependency graph.

reduce the time to simulate a model. An outline of the NRM is given in Fig. 4.

2.4. The optimized direct method

The optimized direct method (ODM) developed by Cao et al. (2004) is the most recent algorithm designed to improve the performance of exact stochastic simulation. Building on the observation that when executing the NRM, a majority of computational time is spent maintaining the heap data structure, they developed a new approach for reducing the complexity of the reaction selection stage (step 4).

Performing the reaction selection stage (step 4) for the DM involves searching the propensity values to find the value of sel that satisfies Eq. (8). This is most easily accomplished using the pseudo-code found in Fig. 5. Notice that as soon as the selected reaction is found, we can break out of the search loop. We will refer to the number of steps the process takes as the search depth.

Cao et al. realized that by moving reactions that execute more frequently to the beginning of the reaction search order, they could reduce the average search depth for the run. Since most biological models have a few reactions that are executed frequently and a majority of the other reactions are executed infrequently, the modification can significantly reduce computation time. To determine which reactions will execute most frequently, the ODM executes pre-simulations that track reaction execution frequencies. This information is then used to presort the reactions in order of decreasing frequency of occurrence.

This algorithm also extends the use of the dependency graph of the NRM to increase the efficiency of the total propensity calculation. Pseudo-code for the ODM is given in Fig. 6. By reducing the overhead of the NRM heap structure, Cao et al. demonstrated that their algorithm outperforms the NRM and is the fastest known algorithm for simulating typical biochemical models.

2.5. Approximation methods

To address the computational performance challenges presented by exact stochastic simulation (“exact” refers to algo-

```
CurrentTime = 0.0                                                    1. Initialization
S[1..M] = Initial Species Populations
R[1..N] = Reactions
Dependencies = InitDependencyGraph(R)
For I = 1..N
  Prop[I] = CalcPropensity(S,R[I])
  T[I] = -ln(rand())/Prop[I]
  MinHeap.add(I,T[I])
End For
Goto Reaction Selection
DependentRxns = Dependencies[SelRxn]                                 2. Propensity Calculation
Foreach DepRxn in DependentRxns
  OldProp[DepRxn] = Prop[DepRxn]
  Prop[DepRxn] = CalcPropensity(S,R[DepRxn])
End Foreach
Foreach DepRxn in DependentRxns                                      3. Reaction Time
  NewT = OldProp[DepRxn] * (T[DepRxn] - CurrentTime)                    Generation
  NewT = NewT / Prop[DepRxn] + CurrentTime
  T[DepRxn] = NewT
  MinHeap.update(DepRxn,T[DepRxn])
End Foreach
T[SelRxn] = -ln(rand())/Prop[SelRxn]
MinHeap.update(SelRxn,T[SelRxn])
SelRxn = MinHeap.getMinIndex()                                       4. Reaction Selection
S = S - R[SelRxn].reactants + R[SelRxn].products                     5. Reaction Execution
CurrentTime = T[SelRxn]
If (CurrentTime < EndTime)                                           6. Termination
  Goto Propensity Calculation
End If
```

Fig. 4. Pseudo-code for the next reaction method (NRM).

```
Selector = TotalPropensity * rand()
For I = 1..N
  Selector = Selector - Prop[I]
  If (Selector <= 0)
    SelRxn = I
    Break
  End If
End For
```

Fig. 5. Direct method reaction selection.

rithms which are equivalent to the SSA and produce results which introduce no approximations to the CME), researchers have developed a variety of approximation methods. These methods attempt to collapse or condense computational work to generate large performance gains while sacrificing an acceptable amount of accuracy.

One of the more promising areas of approximation research that has received a significant amount of recent attention is called "tau-leaping." This technique, first proposed by Gillespie (2001), condenses the execution of multiple reactions events into a single simulation step by making "leaps" in simulation time and using Poisson random variables to determine how many reactions events occurred during the leap. The key to the success of this technique is to select a leap size big enough to allow many reactions to occur during the leap (reducing computation) and small enough that none of the propensity functions will change significantly in value (causing an error). This technique has continued to mature, particularly in the area of leap-size selection, through the work of a variety of researchers (Gillespie and Petzold, 2003; Rathinam et al., 2003; Tian and Burrage, 2004) and a further work in this area can be expected.

Other examples of approximation methods include multi-scale algorithms like the slow-scale stochastic simulation algorithm (Cao et al., 2005) and use of the quasi-steady state assumption (Rao and Arkin, 2003), the combination of deterministic and stochastic approaches (Adalsteinsson et al., 2004), using chemical Langevin equations (Gillespie, 2000; Simpson et al., 2003, 2004), using Poisson–Runge–Kutta methods (Burrage and Tian, 2003a), and a variety of others. This list is not intended to be complete and is provided merely to alert the reader that a wealth of methods exist. For a more comprehensive discussion of many of these techniques, readers are encouraged to consult

Simulate the system for a period of time < EndTime
  and monitor the number of times each reaction is
  executed
Sort the reactions, placing the most frequently
  executed reaction at R[1] and the least at R[N]
	Presimulation

CurrentTime = 0.0
S[1..M] = Initial Species Populations
R[1..N] = Reactions
Dependencies = InitDependencyGraph(R)
TotalPropensity = 0.0
For I = 1..N
  Prop[I] = CalcPropensity(S,R[I])
  TotalPropensity = TotalPropensity + Prop[I]
End For
Goto Reaction Time Generation
	1. Initialization

DependentRxns = Dependencies[SelRxn]
Foreach DepRxn in DependentRxns
  TotalPropensity = TotalPropensity - Prop[DepRxn]
  Prop[DepRxn] = CalcPropensity(S,R[DepRxn])
  TotalPropensity = TotalPropensity + Prop[DepRxn]
End Foreach
	2. Propensity Calculation

T = -ln(rand())/TotalPropensity	3. Reaction Time Generation

Selector = TotalPropensity * rand()
For I = 1..N
  Selector = Selector - Prop[I]
  If (Selector <= 0)
  SelRxn = I
  Break
  End If
End For
	4. Reaction Selection

S = S - R[SelRxn].reactants + R[SelRxn].products CurrentTime = CurrentTime + T	5. Reaction Execution

If (CurrentTime < EndTime)
  Goto Propensity Calculation
End If
	6. Termination

Fig. 6. Psuedo-code for the optimized direct method (ODM).

the detailed reviews by Turner et al. (2004) and Burrage et al. (2003b).

While the authors acknowledge the promise of these approaches, these techniques have not matured to the point where general biochemical modelers can know when and when not to apply these methods. Although inferior computationally, exact stochastic simulation techniques remain effective, trusted, and accurate general-purpose tools for properly modeling stochastic effects in biochemical processes. Our work will therefore focus only on improving the performance of "exact" stochastic simulation techniques that are mathematically equivalent to Gillespie's SSA.

3. Performance analysis

To analyze the performance of the described exact stochastic simulation algorithms, three biological models are constructed. The first is an 8-cell model of quorum sensing in *Vibrio fischeri* based on the work of Cox et al. (2003).

3.1. Quorum sensing in Vibrio fischeri

Quorum sensing is a cell–cell communication mechanism used by bacteria to coordinate activities such as bioluminescence, virulence, or biofilm formation (Greenberg, 2000). *V. fischeri* colonize the light organs of the Hawaiian bobtail squid *Euprymna scolopes* with which it has a symbiotic relationship. Bioluminescence generated by *V. fischeri* helps camouflage the squid from predators; in exchange the squid provides *V. fischeri* a nutrient rich environment within its light organ.

V. fischeri uses quorum sensing to regulate bioluminescence and other colonization factors such that they occur only when the concentration of cells is relatively high (Lupp et al., 2003). The mechanism is primarily controlled by the action of two genes. The gene *luxI* codes for a protein that catalyzes the production of an acyl-homoserine lactone (AHL) autoinducer molecule. AHL can freely diffuse across the cell membrane and acts as a signaling molecule to surrounding cells. The gene *luxR* codes for a receptor protein, which upon complexation with AHL, positively regulates the activity of both genes. The bioluminescent genes are encoded downstream of *luxI* on the same operon. The positive feedback loop acts as a switch to turn on bioluminescence when the cell senses sufficiently large AHL concentrations to indicate that the cell density is high.

The quorum-sensing model used here is similar to the model of Cox et al. (2003), but differs in several respects. First, the present model includes abstract representations of cell growth and regulation of *luxR* by a secondary quorum sensing circuit. The present model also ignores the DNA-looping repression mechanism included in the earlier model. Most significantly, the present model captures the quorum-sensing dynamics of eight interacting cells. The model consists of 122 species and 201 reactions and is executed for 25,000 simulated seconds.

3.2. Lysogeny decision in phage-λ

The second model represents the lysogeny decision circuit in phage-λ infected *Escherichia coli* cells. When phage-λ virus infects *E. coli*, the phage DNA can direct the fate of the cell down one of two pathways. In the lytic case, the phage DNA hijacks the cell machinery to make several additional copies of the virus, the cell ruptures and the viruses are released to infect additional cells. Alternatively, the infected cell may be directed into a lysogenic state in which the phage DNA is incorporated into the bacterial DNA. In this case, as the cell divides the viral DNA is replicated along with the bacterial DNA and is thereby passed down to the cell's progeny. The core regulatory mechanisms controlling the decision are encoded into a four-promoter, five-gene network and are well studied (Ptashne, 1992). Arkin et al. (1998) formulated a stochastic model of this network and used it to demonstrate that the fraction of cells selecting each pathway was determined by the stochastic fluctuations in the gene expression levels. We used a simplified version of this model (see acknowledgements) for performance analysis of the SSA algorithms. It consists of 61 species, 117 reactions, and is executed for 2400 simulated seconds.

3.3. Heat stress response in Escherichia coli

The third model is provided by Cao et al. (2004) and is the same model they used to characterize the performance of their ODM. The model is a representation of heat stress response in *Escherichia coli.* Heat stress is the term used to describe the process of protein denaturing at elevated cell temperature. At normal temperatures, RNA polymerase is bound to the sigma factor σ^{70}. At elevated temperatures the activity of σ^{32} increases, which redirects RNA polymerase to a set of 20 or so heat stress genes. Some of these genes help refold denatured proteins, while others degrade denatured proteins. A regulatory mechanism controls the activity of σ^{32} in response to elevated temperature. The model of Cao et al. consists of 28 species and 61 reactions and is simulated for 500 s. For a detailed description of this model, consult (Cao et al., 2004; Kurata et al., 2001).

3.4. Results

Each of the four simulation algorithms were implemented in C++ and compiled with the maximum optimization flags using the GNU C++ compiler. To characterize the reaction execution frequency as required by the ODM, each model is pre-simulated for 5% of the total simulation time. The platform used to run the simulations is a 2.0 GHz Intel Pentium 4 processor with 512 kB L2 cache and 1 GB of RAM running Red Hat Linux v7.3. All computation time measurements reported throughout this paper are given in wall-clock time.

Results for executing the simulations using the four simulation algorithms described above are given in Table 2. The ODM results reflect the sum of the time spent executing the simulator and running the pre-simulation. Since each run is a different random-walk, the number of reactions executed by the simulator can vary. We therefore use reactions executed per second as the metric for comparison as shown in Table 3. The results support Cao et al.'s claim that the ODM is the fastest known algorithm for exact stochastic simulation for typical biochemical models.

Table 2
Performance measurements for simulators executing various biological models

	Heat stress response		Phage-λ		Quorum-sensing	
	Reactions	Time	Reactions	Time	Reactions	Time
First (FRM)	48392805	1620.71	695552	65.02	4561472	488.38
Direct (DM)	46604186	232.36	531470	7.83	4824786	84.86
Next (NRM)	46786907	244.49	661999	15.21	4785310	12.27
Optimized (ODM)	46312042	52.56	605892	1.71	4438137	6.94

4. The sorting direct method

4.1. Transient reaction execution behavior shifts

Although the ODM is efficient, we discovered that there are still ways to improve the computational performance of reaction selection (step 4). The ODM requires the execution of several pre-simulations to determine the proper ordering of reactions. For this to be effective, it is assumed that the reaction execution behavior exhibited at the beginning of the simulation will be characteristic of the long-term reaction execution behavior. The following example shows that this assumption may not be valid for many biochemical networks.

Suppose we are modeling the gene regulation network given in Fig. 7. Here we see that Gene1 and Gene2 transcribe at a particular basal rate. When the transcription factor Protein1 binds to Gene2, it acts as an inducer and transcribes at a greatly accelerated rate. A stochastic simulation of this system is given in Fig. 8. Notice that just before time 200.0, Protein1 binds to Gene2 and the Protein2 population increases dramatically. If the pre-simulations used to order the reactions were based solely on the first 100 s (the first 25%) of the simulation, the pre-simulation would overemphasize Gene1's transcription, translation, and decay reactions, when over the entire simulation Gene2's reactions would be the most frequently executed. This would increase the reaction search depth dramatically and decrease the performance of the simulator.

4.2. Algorithm definition

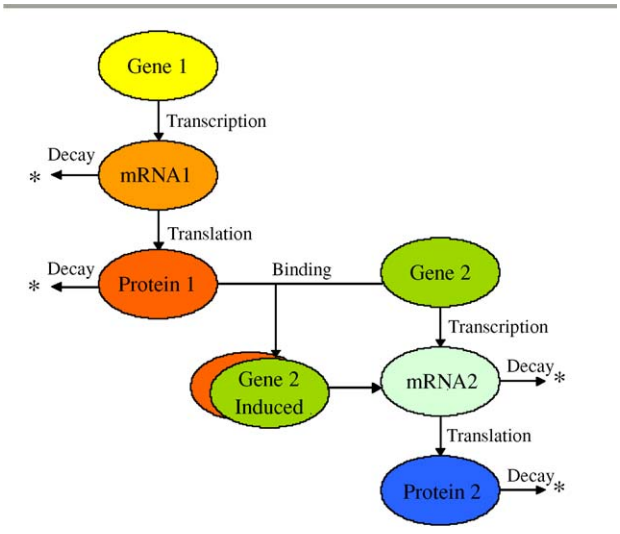
To account for this problem and to eliminate costly pre-simulations, we developed the sorting direct method (SDM), which maintains a reaction selection order that is approximately sorted throughout the simulation. Each time a reaction is executed, its position is moved up in the reaction selection order. This moves reactions that have occurred recently toward the top of the reaction search list, effectively reducing the search depth for this reaction the next time it is executed. Since the sort requires only a swap of two memory addresses, it adds negligible overhead to the simulation as we will show in our performance analysis. The sorting allows the algorithm to adapt to sharp changes in propensity, thus handling the on/off behavior of natural switches in genetic regulatory networks. Pseudo-code for the algorithm is given in Fig. 9.

Table 3
Performance results measured in reactions executed per second

Simulator	Heat stress response	Phage-λ	Quorum-sensing
First (FRM)	29859	10697	9339
Direct (DM)	200561	67840	56855
Next (NRM)	191363	43515	389953
Optimized (ODM)	881043	354675	639697

4.3. Performance comparison

To demonstrate our performance improvement, we expand the model given in Fig. 7 to produce a 20-gene model with




```
GENE1 -> GENE1 + mRNA1              10.0
mRNA1 -> mRNA1 + Protein1           10.0
mRNA1 -> *                           1.0
Protein1 -> *                        1.0

GENE2 -> GENE2 + mRNA2               5.0
mRNA2 -> mRNA2 + Protein2            5.0
mRNA2 -> *                           1.0
Protein2 -> *                        1.0

Gene2 + Protein1 -> Gene2_IND    0.0001
Gene2_IND -> Gene2_IND + mRNA2    100.0
```

Fig. 7. Gene regulation network model. This model includes the transcription and translation of Gene1 into Protein1 and Gene2 into Protein2. When Protein1 binds to Gene2, it induces the production of Protein2.

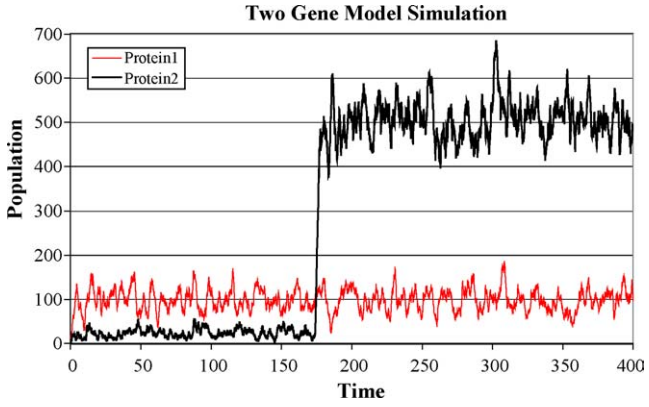


Fig. 8. Stochastic simulation of the gene regulation network model. Notice that the dynamics of the simulation change just before time 200 when the inducer protein binds to the gene and induces the production of Protein2.

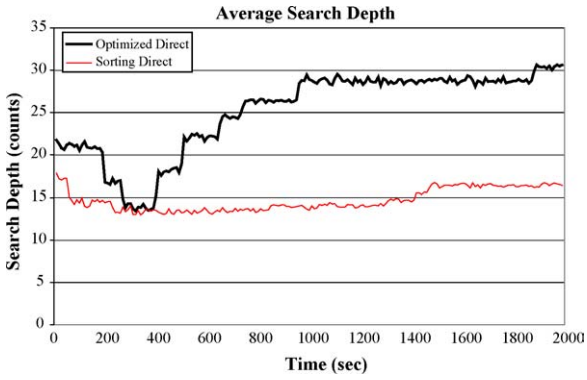


Fig. 10. Measured average search depth for the ODM and the SDM while running the 20-gene model. Sharp decreases or increases occur in the search depth for the ODM when genes are induced, while the SDM remains small and constant throughout the simulation.

10 genes induced by proteins that the other 10 genes produce. Fig. 10 shows periodic measurements of the average search depth as the simulation executes for both simulators. The ODM shows sharp changes in the reaction search depth as the simulation executes. Our results show that these sharp changes occur as genes are induced. As the run continues, the average search depth increases because the pre-simulation has not effectively predicted the reaction execution behavior for the run. The SDM successfully adapts to changes in reaction execution behavior and consistently remains at or below the average search depth measured by the ODM. Because the SDM is able to maintain a relatively small average search depth, it is able to perform the simulation at a rate of 1.55 MRxns/s (millions of reactions executed per second) while the ODM is only able to perform the simulation at a rate of 1.29 MRxns/s.

```
CurrentTime = 0.0
S[1..M] = Initial Species Populations
R[1..N] = Reactions
RSO[1..N] = 1..N  (Reaction Search Order)
Dependencies = InitDependencyGraph(R)
TotalPropensity = 0.0                                        1. Initialization
For I = 1..N
  Prop[I] = CalcPropensity(S,R[I])
  TotalPropensity = TotalPropensity + Prop[I]
End For
Goto Reaction Time Generation
-------------------------------------------------------------------------------
DependentRxns = Dependencies[SelRxn]
Foreach DepRxn in DependentRxns
  TotalPropensity = TotalPropensity - Prop[DepRxn]
  Prop[DepRxn] = CalcPropensity(S,R[DepRxn])                 2. Propensity Calculation
  TotalPropensity = TotalPropensity + Prop[DepRxn]
End Foreach
-------------------------------------------------------------------------------
T = -ln(rand())/TotalPropensity                              3. Reaction Time Generation
-------------------------------------------------------------------------------
Selector = TotalPropensity * rand()
For I = 1..N
  Selector = Selector - Prop[RSO[I]]
  If (Selector <= 0)                                         4. Reaction Selection
    RSOIndex = I
    Break
  End If
End For
SelRxn = RSO[RSOIndex]
-------------------------------------------------------------------------------
S = S - R[SelRxn].reactants + R[SelRxn].products
CurrentTime = CurrentTime + T
If (RSOIndex != 1)                                           5. Reaction Execution
  Temp = RSO[RSOIndex]
  RSO[RSOIndex] = RSO[RSOIndex - 1]
  RSO[RSOIndex] = Temp
End If
-------------------------------------------------------------------------------
If (CurrentTime < EndTime)                                   6. Termination
  Goto Propensity Calculation
End If
```

Fig. 9. Pseudo-code for the sorting direct method (SDM).

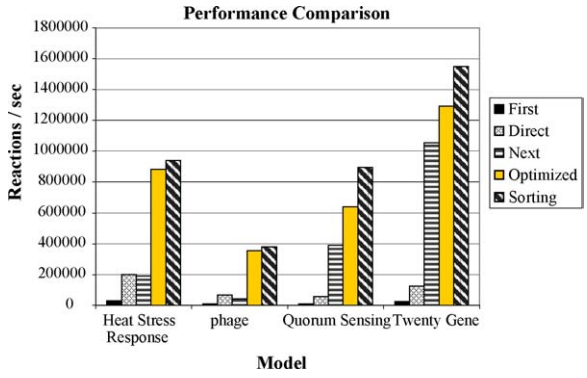


Fig. 11. Comparison of simulator performance for a variety of models.

Even in situations where the reaction execution frequency remains constant throughout the simulation and the ODM accurately predicts the optimal order of reactions, the SDM performs as well as the ODM and adds negligible overhead. To show this, we use the same 20-gene model and set the initial species populations to their long-term steady state values. With this configuration, the pre-simulation properly sorts the reactions in the model to provide the ideal reaction search order for the ODM. Performing 100,000 reactions for the pre-simulation consumes 0.14 s and executing 10,000,000 reactions takes an additional 6.27 s for a total run time of 6.40 s. The SDM is able to execute 10,000,000 reactions in 6.44 s. This demonstrates that the overhead added by the SDM to maintain the reaction sort order is small and may be less than the pre-simulation time required for the ODM.

Finally, we execute the biological models from Section 3. Fig. 11 shows a chart comparing the performance of the SDM to the four other simulation algorithms using the biological models from Section 3 and the 20-gene model. The results show that the SDM performs as well or better than the ODM for all models tested. The SDM shows a notable performance improvement when executing the 20-gene and quorum-sensing models, because these models include significant state transitions (certain reactions are turned on or off throughout the simulation). The *E. coli* and λ-phage models show fairly consistent reaction execution behavior throughout the simulation, so the pre-simulation performed by the ODM reasonably characterizes the long-term reaction behavior for the system and performs comparably well to the SDM.

5. Conclusion

The structure of regulatory networks in cells leads to large shifts in the transient behavior of protein populations. Failure to account for this behavior when stochastically simulating such models can negatively impact computational performance. By loosely sorting the reactions as the simulation executes, the sorting direct method employs an effective strategy for handling large shifts in reaction propensities and eliminates the need for pre-simulations.

Acknowledgements

This work was funded in part by the U.S. Department of Energy's Genomes to Life program (www.doegenomestolife.org) under the ORNL-PNNL project, "Exploratory Data Intensive Computing for Complex Biological Systems." This work was also developed in conjunction with the DARPA BioSPICE Bio-Computation program and the Keck Futures Initiative. We gratefully acknowledge funding support from the National Science Foundation and the Defense Advanced Research Projects Agency through National Science Foundation grants EIA-0130843 and CCS-0311500.

The authors would also like to thank Dr. Yang Cao, Dr. Hong Li, and Dr. Linda Petzold of the University of California Santa Barbara for providing the source code for their implementation of the optimized direct method and the *E. coli* heat stress response model. The authors would also like to thank Dr. Chris Myers of the University of Utah for kindly sharing his SBML model of the phage-λ system. We thank anonymous reviewers who provided many useful suggestions for improving the paper.

References

Adalsteinsson, D., McMillen, D., Elston, T.C., 2004. Biochemical network stochastic simulator (BioNetS): software for stochastic modeling of biochemical networks. BMC Bioinform. 5, 24.

Arkin, A., Ross, J., McAdams, H.H., 1998. Stochastic kinetic analysis of developmental pathway bifurcation in phage λ-infected *Escherichia coli* cells. Genetics 149, 1633–1648.

Bentele, M., et al., 2004. Mathematical modeling reveals threshold mechanism in CD95-induced apoptosis. J. Cell Bio. 166, 839–851.

Burrage, K., Tian, T., 2003a. Poisson–Runge–Kutta methods for chemical reaction systems. In: Proc. Hong Kong Conf. Sci. Comput.

Burrage, K., Tian, T., Burrage, P., 2003b. A multi-scaled approach for simulating chemical reaction systems. Prog. Biophys. Mol. Bio. 85, 217–234.

Cao, Y., Li, H., Petzold, L.R., 2004. Efficient formulation of the stochastic simulation algorithm for chemically reacting systems. J. Chem. Phys. 121, 4059–4067.

Cao, Y., Gillespie, D.T., Petzold, L.R., 2005. The slow-scale stochastic simulation algorithm. J. Chem. Phys. 122, 14116.

Cox, C.D., et al., 2003. Analysis of noise in quorum sensing. OMICS 9, 317–334.

Endy, D., Brent, R., 2001. Modeling cellular behavior. Nature 409, 391–395.

Forger, D.B., Peskin, C.S., 2003. A detailed predictive model of the mammalian circadian clock. Proc. Natl. Acad. Sci. 100, 14806–14811.

Forger, D.B., Peskin, C.S., 2004. Model based conjectures on mammalian clock controversies. J. Theo. Bio. 230, 533–539.

Forger, D.B., Peskin, C.S., 2005. Stochastic simulation of the mammalian circadian clock. Proc. Natl. Acad. Sci. 102, 321–324.

Gibson, M., Bruck, J., 2000. Efficient exact stochastic simulation of chemical systems with many species and many channels. J. Phys. Chem. A 104, 1876–1889.

Gillespie, D.T., 1976. General method for numerically simulating the stochastic time evolution of coupled chemical reactions. J. Comp. Phys. 22, 403–434.

Gillespie, D.T., 1977. Exact stochastic simulation of coupled chemical reactions. J. Phys. Chem. 81, 2340–2361.

Gillespie, D.T., 1992. A rigorous derivation of the chemical master equation. Physica A 188, 404–425.

Gillespie, D.T., 2000. The chemical Langevin equation. J. Chem. Phys. 113, 297–306.

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

Gillespie, D.T., Petzold, L.R., 2003. Improved leap-size selection for accelerated stochastic simulation. J. Chem. Phys. 119, 8229–8234.

Greenberg, E.P., 2000. Acyl-homoserine lactone quorum sensing in bacteria. J. Microbio. 38, 117–121.

Hindmarsh, A., 2001. Brief description of ODEPACK—a systematized collection of ODE solvers. http://www.netlib.org/odepack/opkd-sum.

Ideker, T., Galitski, T., Hood, L., 2001. A new approach to decoding life: systems biology. Annu. Rev. Genom. Human Genet. 2, 343–372.

Kærn, M., et al., 2005. Stochasticity in gene expression: from theories to phenotypes. Nat. Rev. Genet. 6, 451–464.

Kurata, H., et al., 2001. Feedback regulation of the heat shock response in *E. coli*. In: Proceedings of the IEEE Conference on Decision and Control, pp. 837–842.

Lupp, C., et al., 2003. The *Vibrio fischeri* quorum-sensing systems ain and lux sequentially induce luminescence gene expression and are important for persistence in the squid host. Mol. Microbio. 50, 319–331.

Ptashne, M., 1992. A Genetic Switch: Lambda Phage and Higher Organisms, 2nd ed. Blackwell Scientific Publications, Cambridge, MA.

Rao, C.V., Wolf, D.M., Arkin, A.P., 2002. Control, exploitation, and tolerance of intracellular noise. Nature 420, 231–237.

Rao, C.V., Arkin, A.P., 2003. Stochastic chemical kinetics and the quasi-steady-state assumption: application to the Gillespie algorithm. J. Chem. Phys. 118, 4999–5010.

Rathinam, M., et al., 2003. Stiffness in stochastic chemically reacting systems: the implicit tau-leaping method. J. Chem. Phys. 119, 12784–12794.

Sauro, H.M., et al., 2003. Next generation simulation tools: the Systems Biology Workbench and BioSPICE integration. OMICS 7, 355–372.

Shampine, L.F., Reichelt, M.W., 1997. The MATLAB ODE suite. SIAM J. Sci. Comput. 18, 1–22.

Simpson, M.L., Cox, C.D., Sayler, G.S., 2003. Frequency domain analysis of noise in autoregulated gene circuits. Proc. Natl. Acad. Sci. 100, 4551–4556.

Simpson, M.L., Cox, C.D., Sayler, G.S., 2004. Frequency domain chemical Langevin analysis of stochasticity in gene transcriptional regulation. J. Theo. Bio. 229, 383–394.

Stelling, J., Gilles, E.D., 2004. Mathematical modeling of complex regulatory networks. IEEE Trans. Nanobiosci. 3, 172–179.

Strohman, R., 2002. Manuvering in the complex path from genotype to phenotype. Science 296, 701–703.

Tian, T., Burrage, K., 2004. Binomial leap methods for simulating chemical kinetics. J. Chem. Phys. 121, 10356–10364.

Turner, T.E., Schnell, S., Burrage, K., 2004. Stochastic approaches for modeling in vivo reactions. Comp. Bio. Chem. 28, 165–178.