Technical Report CoSBi 19/2008

Notes on stochastic simulation of chemical kinetics with cycle-leaping

Stefano Teso
The DISI - University of Trento
teso@disi.unitn.it

Paola Lecca
The Microsoft Research - University of Trento
Centre for Computational and Systems Biology
lecca@cosbi.eu

Notes on stochastic simulation of chemical kinetics with cycle-leaping

Stefano Teso and Paola Lecca

Abstract

In this report we review the Riedel-Bruck stochastic simulation algorithm, which makes use of a cycle-leaping strategy to improve the simulation performance. We implemented the algorithmi and tested our implementation on some stochastic models, such as the Lotka-Volterra model of predation, the Brusselator model, and the Michaelis-Menten model of enzymatic catalysis. We discuss the advantages and the disadvantages of this algorithm from the viewpoint of its use in a systemic approach to modeling and simulation of biochemical and biological processes.

1 Introduction

The mathematical models of chemical kinetics can be grouped into two broad categories: (a) deterministic, continuous models, and (b) stochastic, discrete models.

The first category encompasses deterministic mathematical models in the form of systems of ordinary differential equations (ODEs). Each equation describes the change in *concentration* of one chemical species over time. The most general form to write such a system is:

$$\frac{dX_i}{dt} = F_i(X_1, X_2, \ldots, X_n; \mathbf{k}) \qquad \text{for } i = 1, 2, \ldots, n$$

where $\mathbf{X}(t) : \mathbb{R} \to \mathbb{R}_+^n$ is a continuous function representing the concentrations of n chemical species $X_1, X_2, \ldots, X_n$ for any given time t and parameter vector $\mathbf{k}$. The simulation of complex deterministic models involves the *numerical integration* of $\nabla\mathbf{X} = F(\mathbf{X}; \mathbf{k})$ to provide numerical estimates of the time behavior of the reactants concentration. The development of techniques and algorithms for the simulation of deterministic models has evolved to a stunning level of complexity [9].

The second category comprises stochastic models. In the stochastic framework, a reaction network is modeled as a random Markov process. The state vector of the system, $\mathbf{X}(t)$, represents the *populations* of the various chemical species, and is allowed to take only discrete values. The behavior in

time of the system is determined by the occurence of the involved chemical reactions. A typical reaction is defined by a set of stoichiometric coefficients s_i, r_i in the form

$$s_1 S_1 + s_2 S_2 + \cdots \longrightarrow r_1 S_1 + r_2 S_2 + \cdots \tag{1}$$

Each reactive collision changes the state $\{X_i\}$ of the mixture into $\{X_i + r_i - s_i\}$

The time behavior of the system is the realization of $\mathbf{X}(t)$, that is a continuous-time Markovian process taking values in the n-dimensional lattice of non-negative integer numbers. More precisely, the behavior of the system consists in jumps on the sub-lattice of $\mathbb{N}^n$ spanned by the reaction vectors $\{\mathbf{v}_j\}_{j=1}^m$ (where m is the number of reaction in the system) starting from the initial state $\mathbf{X}(0)$. This means that the simulation may exhibit significantly different properties when started from different initial states. The asymptotic behavior ultimately depends on whether the system is ergodic or not. The resolution of a stochastic model involves computing a realization of the random variable $\mathbf{X}(t)$. This is a form of integration as well: Monte Carlo simulation computes a numerical approximation of definite, high-dimensionality integrals. In this case the integral being evaluated is the time behavior of the probability of $\mathbf{X}$, $\Pr(\mathbf{X} \mid t)$, as given by the CME. It turns out that, under appropriate assumptions, this computation can be carried out *exactly*, meaning that the sampled probability distribution $\Pr(\mathbf{X}(t))$ approaches, *in the limit* of infinite realizations, the probability distribution given by the Chemical Master Equation (CME).

The two approaches to modeling have complementary disadvantages. The stochastic approach is more realistic because, as thoroughly argumented elsewhere [3] [10], chemical reaction networks are themselves non-deterministic. This is the case for several reasons: (a) The particle system is not isolated, but in contact with external sources of randomness, e.g., a heat bath. (b) Due to the microscopic scale at which the phenomena take place, quantum (random) effects have a non-negligible impact. (c) Even if the particle dynamics was deterministic, i.e. all particles were controlled by the Newtonian laws of dynamics, only part of their complete state (position, velocity) is known at any given time. The resulting projection is therefore inherently stochastic. (d) The particle system is extremely sensible to initial conditions: a small perturbation of the initial state gives rise to dramatically different outcomes. This is the same kind of random behavior imbued into pseudo-random number generators (RNGs), so it is not unreasonable to dub the reaction network random as well.

Stochasticity plays a major role when the system being modeled contains only a small number of molecules. In this case random fluctuations can influence the behavior of the system drastically, as the presence of a single molecule may have macroscopic long-term effects. Furthermore, continuous

concentrations become bad approximations when the size of the population is small. For larger populations the random fluctuations are averaged out, and ODEs offer a valuable approximation to the stochastic model.

The curves produced by the ODEs are usually treated as if they were the mean concentrations of the system being modeled, but they are not. This is particularly obvious when the system has multi-modal behavior.

Stochastic simulation algorithms such as Gillespie's algorithm [9] provide a methods to calculate $\mathbf{X}(t)$ without the need to derive it from the solution of the CME. The Gillespie algorithm is *exact*, in the sense that it does not introduce any discretization of the time, but derives the instant of occurrence of a reaction analytically from the distribution of reaction waiting times. However, this exactness comes at a cost.

The by far worst issue of SSAs is their poor performance when compared to deterministic simulators. This happens because biological processes involve a huge number of chemical reactions, that often are concurrent and parallel. To simulate a second's worth of reactions, a simulator has to execute them *serially*, an effort that takes several seconds of computing time. This is a serious problem indeed [6].

The performance of SSAs can be improved substantially, for instance, by making clever use of random numbers and caching techniques [1], or by allowing for suitable approximations [10]. The approach by Riedel and Bruck [4] takes a different route. The authors argue that stochastic simulators spend most of their time trapped in cycles of recurring reactions. To prevent this from happening, they devised a simple mechanism to avoid cycles altogether.

In the remainder of this report we describe the Riedel-Bruck (abbreviated RB) cycle-leaping method in some detail, and proceed at a critical discussion of the advantages and disadvantages of the proposed algorithm. We underline our remarks through experiments with simple and well-known chemical models.

2 Overview of the Riedel-Bruck Algorithm

Cycles are what render the subject of coupled chemical reactions most interesting, as they are *the* key ingredient to non-linearity. For instance, if cycles were forbidden, no control loops would be possible, e.g., in the regulatory gene expression network. According to [4] the stochastic simulators currently in use spend most of their time within cycles. Since no "real work" is being done in cycles, it can be argued that it would be wise not to waste too much time by blindly iterating through them. To this end, the authors in [4] devised a simple escape mechanism to leap over recurring cycles.

The RB algorithm works by generating a trajectory in the state space for the particular random walk induced by the exection of the reactions of

the network. Let $\mathbf{x}_t = \mathbf{X}(t)$. Starting at the initial state $\mathbf{x}_0$, at each step the algorithm chooses uniformly at random a transition vector $\mathbf{v}_j$, for some $1 \leq j \leq m$, according to the conditional probabilities of each reaction given the state at step t, that is, $\Pr(\mathbf{v}_j \mid \mathbf{x}_t)$. In particular, these probabilities turn out to be:

$$\Pr(\mathbf{v}_j \mid \mathbf{x}) = \frac{r(\mathbf{v}_j \mid \mathbf{x})}{\sum_{k=1}^{m} r(\mathbf{v}_k \mid \mathbf{x})} \qquad \text{where } \; r(\mathbf{v}_j \mid \mathbf{x}) \stackrel{\text{df}}{=} c_j \prod_{\ell \,:\, q_{j\ell} > 0} \binom{x_\ell}{q_{j\ell}}$$

where c_j is the stochastic reaction propensity for reaction j, and q_{ji} is the stoichiometric coefficient for reactant i as appearing in reaction j. The next state is set to be $\mathbf{x}_{t+1} = \mathbf{x}_t + \mathbf{v}_j$.

Cycle detection is performed by keeping in memory a history of the k most recently traversed states, with k a user-chosen parameter. Whenever the simulated process jumps to a state $\mathbf{x}_t$, the history is scanned to check if $\mathbf{x}_t$ has been recently met. In the affirmative case, the cycle C is defined to be the sequence of states starting at $\mathbf{x}_t$ and ending at $\mathbf{x}_{t-1}$. The dynamics is illustrated in Figures 2 and 2.

Once a cycle has been identified, the algorithm computes the transition probabilities from the current state $\mathbf{x}_t$ to all the states reachable from the cycle in one step (in the "neighborhood" of the cycle). At this point the cycle is leaped over by picking one of the states in the neighborhood of C according to the calculated probability distribution. See Algorithm 1 for details. In this way, whenever a cycle shorter than k states is generated, it is automatically skipped by the algorithm: no time is "wasted" executing it, and the simulation can proceed further on to other regions of the state space. At the same time, we observe that the probabilities of each state in the neighborhood of C remain the same with and without cycle leaping. From this point of view the algorithm can be considered *exact*.

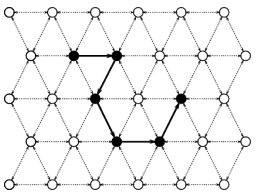

Figure 1: A random trajectory in the state graph. The path in the figure is an artificial example. However, it can be interpreted as the realization of a simple random walk with $n = 3$ species moving on the 3-dimensional simplex.

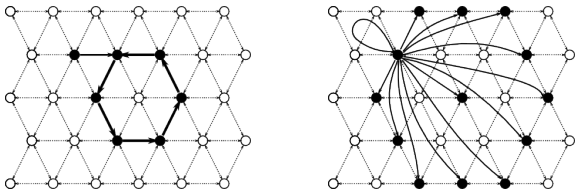

Figure 2: Formation and consequent short-circuiting of a cycle. Gray points represent states belonging to the "neighborhood" of the cycle.

Input: A history of states $H = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_h)$, and the current state $\mathbf{x}_t$
Output: The next state $\mathbf{x}_{t+1}$
if $\mathbf{x}_t = \mathbf{y}_{t'}$ *for some* $t' \in [1, h]$ **then**
  $q = 1$;
  foreach $k = t', t'+1, \ldots, h$ **do**
    foreach *reaction* j *in the network* **do**
      $p_{ij} = q \cdot \Pr(\mathbf{v}_j \mid \mathbf{y}_k)$;
    end
    if $k < h$ **then**
      Let $\mathbf{v}$ be $(\mathbf{y}_{k+1} - \mathbf{y}_k)$;
      $q = q \cdot \Pr(\mathbf{v} \mid \mathbf{y}_k)$;
    end
  end
  Let r be a random number uniformly drawn in the interval $[0, 1]$;
  $\pi = \sum_{i,j} p_{ij}$;
  $\sigma = 0$;
  foreach $(i, j) \in [t', h] \times [1, m]$ **do**
    $p_{ij} = \frac{1}{\pi} p_{ij}$;
    if $\sigma < r$ **then**
      Return $\mathbf{x}_{t+1} = \mathbf{y}_i + \mathbf{v}_j$;
    end
    $\sigma = \sigma + p_{ij}$;
  end
end

Algorithm 1: The cycle-leaping routine. Note that the algorithm requires the history to contain a *sequence of states without gaps*, that is, a trajectory of the form $\mathbf{y}_1 \to \mathbf{y}_2 \to \ldots \to \mathbf{y}_h$. If this is not the case, the computed transition probabilities will be wrong.

3 Implementation

Since the main goal the RB algorithm is to gain a substantial performance boost through cycle-leaping, the "system history" have to be implemented in an as efficient as possible way. Nevertheless, since our objective is to investigate the areas of

application of the algorithm, rather than to find the most efficient implementation, we did not consider further performance improvements. For the sake of simplicity, we implemented the history as a singly-linked list. It takes time proportional to $O(k)$ to search for a state, and constant time for insertion/removal of states from the extremes of the list. Indeed, simple experiments show how our implementation suffers quite a performance loss if the history is taken too large or is allowed to grow without bound. A better implementation would make use of $O(1)$-search hash maps.

The cycle-leaping technique comes at a cost, as it requires additional efforts to (a) maintain and search the history; (b) compute the transition probabilities for all states in the cycle. In our implementation (a) is clearly upper-bounded by $O(k)$, and (b) by $O(k|C|)$, where C is any cycle and k is the history size. Since the length of the target cycles is in the hundreds, history size may be chosen to be $k \approx 200$. This adds a considerable performance penalty to the algorithm, because it incurs into cost (a) at every step. As suggested in the original paper, if a region of the the state graph is to be visited frequently, the cost of (a) and (b) can be amortized by suitable caching of the transition probabilities for recurrent cycles.

In general, cycle-leaping is a time-saving technique as long as the cost of computing the transition probabilities, used for cycle-leaping, is less than the cost of actually executing the cycles. This is the case if there are few, highly probable cycles, which by their nature have a high probability of trapping the simulation for many steps. However, if the state graph is characterized by the presence of a large number of low-probability, "spurious" cycles, the RB algorithm will keep jumping from one cycle to the other, wasting time by leaping over cycles that would very unlikely be traversed many times. It remains to be checked which of the two scenarios chemical reaction networks give rise to.

We have an additional remark to make before examining the experimental results. There is one obvious fundamental difference between the RB algorithm and the mainstream stochastic simulators. By short-circuiting cycles, the RB algorithm renounces to track *time*. This choice has profound consequences. In this way, the results obtained by the simulations are more difficult to be read, interpreted. and compared to experimental outcomes. Due to the elimination of time, it is equally difficult to compare the results of the RB algorithm to those of mainstream stochastic simulators.

To mitigate this issue, [4] advocates a form of analysis of the results (that we will call for short "analysis of probabilities") based on logical predicates about the probability that populations have certain characteristics, such as for instance: "Population 1 has been in range $[A, B]$ with empirical probability at least p, and population 2 has never increased beyond a certain threshold T." A set of such propositions defines a number of possible system states, which should be able to properly discriminate different system behaviors. Applying this kind of analysis, it is finally possible to compare the behavior of experiments to that of simulations, despite the lack of time in the latter. It is interesting to note that this kind of analysis seems to be tightly connected to the work done in [5].

Since the object of the analysis here are the probabilities of certain subsets of states in the state graph, it seems more logical to directly analyze the properties of the underlying state graph. We note however that a complete analysis of the state graph, by which we mean the propagation of probabilities on the graph, amounts to solving the Chemical Master Equation, which has proved to be a challenging task

indeed. So studying the state graph induced by a chemical reaction network may prove to be equally difficult.

Discarding time from the simulations ultimately results in a (random) shrinking of the abscissae in the population plot. This effect is particularly evident for *isomerization* processes, that is, reactions of the form $R : X \longrightarrow Y$. The typical behavior is an exponential decay of the X population, and a corresponding increase of Y. Since time is absent in the simulations, the RB algorithm outputs a *linearly* decaying curve for X, with slope -1, and a *linearly* increasing curve for Y. A similar shrinking effect occurs in the simulation of the Michaelis-Menten enzyme catalysis model, see Figure 3. Here the substrate is supposedly following a decaying exponential as well, but in the plot it quickly drops to zero.

We guess that it should be possible to re-introduce time into the RB algorithm without drastically compromising the performance too much. When a cycle C is detected, it may be possible to (a) Randomly pick a cycle iteration to exit. (b) Randomly pick a state in the neighborhood of C to leap to. (c) Compute the *time* needed to perform these operations.

4 Experiments

We have validated our implementation of the RB algorithm by experimenting with some simple stochastic models: (a) The Lotka-Volterra model of predation. (b) The Brusselator model. (c) The toy model illustrated in Figure 2 of the original paper [4]. (d) The Michaelis-Menten model of enzyme catalysis. The qualitative behavior of models (a), (b) and (c) is well known. For this reason they are often used as standard tests to assess the correctness of stochastic simulators [9] [8] [4].

We found that the population curves generated are qualitative agreement[1] to those obtained by other stochastic simulators. For a comparison with StochSim [7] simulations, see, `http://www.ebi.ac.uk/~lenov/stochsim.html`. Figure 3 illustrates our point.

This outcome was to be expected because the RB algorithm, once stripped of the cycle-leaping logic, is identical to Gillespie's First Reaction Method without time tracking. Of course changing the history size has no effect on the simulation outcomes.

5 Final remarks

Turning our attention back to the kind of analysis of probabilities, mentioned in Section 2, suggested by the original authors, we note that, when used with the RB algorithm, it comes with a slight imperfection. The issue is that while the algorithm computes the transition probabilities correctly, it doesn't do so with the probabilities of individual states. This occurs because when a cycle is leaped over, none of the states it contains is traversed, and therefore their empirical probabilities get spoiled. The result is that the probabilities are shifted from high-probability areas (cycles) to the neighboring regions. At the same time there is no immediate fix to the algorithm, since that would require to know how many times a state in a cycle is traversed. These informations are impossible to determine, unless of course

[1] About the quantitative agreement, it is not possible at this stage to determine the goodness of the agreement, since the time scale is not considered in RD algorithm

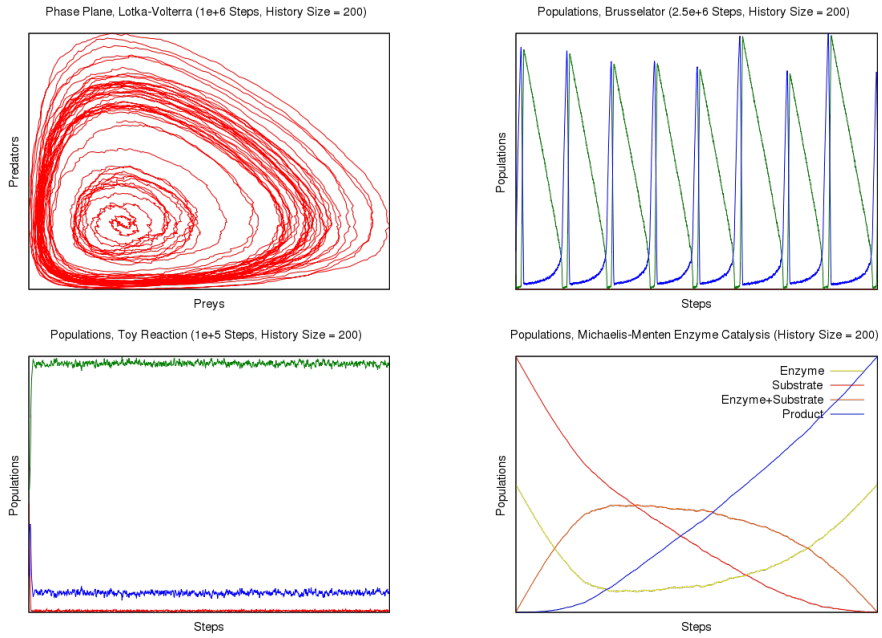

Figure 3: The results of some simulations performed by our RB implementation. (Upper Left) The phase-plane portrait of a single simulation of the Lotka-Volterra predation model. (Upper Right) The population plot of a single simulation of the Brusselator model. (Lower Left) The population plot of a single simulation of the toy reaction model described in Figure 2 of [4]. (Lower Right) The population plot of a single simulation of the Michaelis-Menten enzyme catalysis model. Here it can be noted that, as an effect of the absence of time, the substrate population drops much faster than in usual simulations, e.g., using StochSim.

the algorithm is modified in a way similar to that suggested above to reintroduce time in the simulation.

The magnitude of this "probability-spreading" effect depends on (a) the history size, (b) the number of times that a cycle is iterated. In our case the history size is set to $k \approx 200$, which probably renders (a) negligible, but (b) varies with the model at hand. The point is that the whole reason of resorting to cycle-leaping is to gain speed while avoiding approximations. But then cycle-leaping requires a new kind of analysis which, as a downside, remains exact only as long as cycle-leaping is not used. The initial goal of avoiding approximations is therefore not reached.

A possible way is to find precise bounds on the error introduced by cycle-leaping. The estimation of the error range requires to find out the distribution of cycles in the state graph. At the moment of writing, this study is a work in progress.

As a final note we stress that, under the effect of cycle-leaping, the simulation process doesn't jump on the full state graph anymore, but rather on its *contractions*. In an (edge) contraction of the state graph, a group of states, namely those belonging to a cycle, are aggregated into one single state. This is the main reason

why state probabilities are spread towards neighboring regions. See Figure 4 for an illustration of the process.

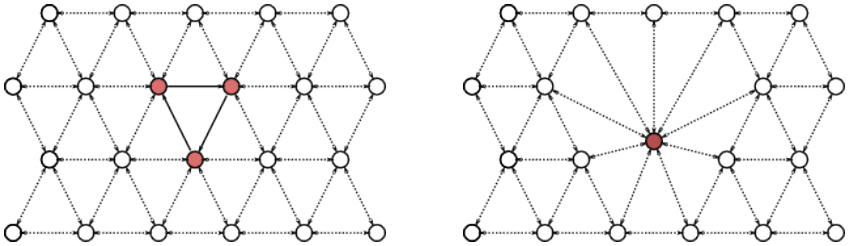

Figure 4: Diagram of a state graph contraction (Right) following the detection of a cycle (Left).

Generalizing, it is possible to imagine different ways to group related states together to speed up the simulation, for instance grouping strongly connected states — states with highly probable transitions between them. In this case probabilities would diffuse in more complex ways than on a simple sequential cycle of states anyway.

It can be proved that graph contractions are related to the finite state projection approach, that is a clever truncation method recently proposed for solving the CME. In this method the state graph is projected to a finite number of states, for which there is an easily computable analytical solution to the corresponding CME. If the error due to the projection is larger than acceptable, the projection is iteratively enlarged. See [2] for details.

References

[1] Gibson M. A. and Bruck J. Efficient exact stochastic of chemical systems with many specifes and many channels. *J. Phys. Chem. A*, 2000.

[2] Munsky B. and Khammash M. The finite state projection algorithm for the solution of the chemical master equation. *The Journal of Chemical Physics*, 124(4), 2006.

[3] Meng T. C., Somani S., and Dhar P. Modeling and simulation of biological systems with stochasticity. In Silico *Biology*, 4(0024), 2004.

[4] Riedel M. D. and Bruck J. Exact stochastic simulation of chemical reactions with cycle leaping, 2006.

[5] Soloveichik D., Cook M., Winfree E., and Bruck J. Computation with finite stochastic chemical reaction networks. *ETR085*, 2008.

[6] Lok L. The need for speed in stochastic simulation. *Nature Biotechnology*, 22:964–965, 2004.

[7] N. Le Novère and T.S. Shimizu. Stochsim: modelling of stochastic biomolecular processes. *Bioinformatics*, 17(6):575–576, June 2001.

[8] Barbuti R., Maggiolo-Schettini A., Milazzo P., and Troina A. An alternative to gillespie's algorithm for simulating chemical reactions. *IN procs. Computational Methods in Systems Biology (CMSB05).*

[9] Gillespie D. T. Exact stochastic simulation of coupled chemical reactions. *Journal of Chemical Physics*, 81(25):2340–2361, 1977.

[10] Gillespie D. T. Approximate accelerated stochastic simulation of chemically reacting systems. *Journal of Chemical Physics*, 115(4):1716–1733, 2001.