SSALeaping: Efficient Leap Condition Based Direct Method Variant for the Stochastic Simulation of Chemical Reacting System

Davide Cangelosi
Department of Computer Science
University of Pisa
Pisa, Italy
cangelo@di.unipi.it

ABSTRACT

The τ-leaping methods are very known solutions for accelerating the Gillespie's Stochastic Simulation Algorithm in the simulation of well-stirred chemically reacting systems. In this paper, we propose a new variant of the stochastic simulation algorithm, that we call *SSAL*, which lays in the middle between the Gillespie's Direct Method and a τ-leaping. Essentially, SSAL works as a standard Direct Method. However, it uses the typical Leap Condition to incrementally build leaps, avoiding at the same time the risk of getting into negative populations. We compare SSAL with one of the most known and efficient τ-leaping methods, named Modified τ-leaping. We provide for both of them a detailed theoretical asymptotic analysis and some experimental tests upon three realistic biological models.

Categories and Subject Descriptors

I.6.8 [**SIMULATION AND MODELING**]: Types of Simulation—*Monte Carlo*

General Terms

ALGORITHMS, THEORY, EXPERIMENTATION, PERFORMANCE

Keywords

τ-leaping, Stochastic Simulation Algorithm, SSALeaping, Leap Condition, Direct Method

1. INTRODUCTION

The stochastic simulation of well-stirred chemically reacting systems has emerged as one of the most attractive challenge topics in the multidisciplinary area known as Systems Biology. In systems of living cells, few molecules can play fundamental roles in the processes governing many cellular activities. The population of these molecular species often exhibits random fluctuations. In other words, starting from the same initial conditions the system can evolve in many different ways. The inherent randomness in the possible evolutions can only be captured taking into account the fact that reactions are discrete and stochastic events. This is the core of the stochastic approach to the chemical kinetics.

The traditional way of the stochastic approach is to set up and solve the *Chemical Master Equation* (CME) of the system. Unfortunately, in many cases the CME is difficult to handle because an equation is necessary for each possible state (which is finite or infinite). Therefore, in 1976 Gillespie formulated a Monte Carlo procedure, called *Stochastic Simulation Algorithm* (SSA) [16]. This algorithm computes a single possible random evolution (realization, simulation) of the system state, despite of the infinite possibilities. The generated trajectory is an *exact* (probabilistically correct and theoretically founded) random description of the state evolution of the chemical system over time. SSA simulates the system evolution as a sequence of reactions. The critical point is to determine *when will the next reaction occur and what reaction it will be*. Gillespie proposed two equivalent formulations of SSA to answer to those questions: the *Direct Method* (DM) [17] and the *First Reaction Method* (FRM) [16]. DM is the first focus of this paper, we will survey it in the next section.

In general, when the populations of certain species and/or the number of reactions in the system are large, SSA requires a huge amount of steps and the performance slow down. To improve efficiency of SSA a number of methods have been proposed. Some maintain exactness, and they mainly speed up the selection of the next reaction [10, 21, 11]. Others sacrifice exactness using more sophisticated techniques [13, 24, 1, 23, 2, 4]. Among the seconds, the so called τ-leaping methods have been proved themselves very fast and accurate solutions, also when they are applied on realistic models [6]. The τ-leaping methods will be the second focus of this paper.

The original τ-leaping method [13] substituted the notion of reaction time with the notion of leap. A leap is a time interval within which hopefully many reactions fire. In general given a time interval τ', there is no way to predict the reactions firing in τ' without run the simulation. However, the τ-leaping overcomes the problem ground on a condition of the reactions activity, called *Leap Condition*. The leap condition enables to estimate the number of occurrences of

SIMUTools 2010 March 15–19, Torremolinos, Malaga, Spain.

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8665
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8665

a given reaction into the considered leap as a value taken from a Poisson distribution, this step is known as *Leap Approximation*. The state evolution advances from one state to the next, applying cumulatively all the reaction occurrences computed in the leap approximation step. So substantial gain in simulation speed can be achieved if each leap fires many reactions [13].
Unfortunately, sometimes an uncontrolled application of the leap approximation can lead some population to become negative. To solve this problem some authors provided interesting solutions [24, 8, 1]. One of the most known and efficient τ-leaping method is the Modified τ-leaping (MTL) [1, 3]. This method uses a new user defined parameter to split the set of reactions of the system. The division is useful to separate and manage differently, reactions that potentially risk to overdraw some of its reactants, from the others. Moreover, MTL shifts from the Gillespie's DM to the τ-leaping and viceversa according to a condition that depends on a further user defined parameter that we will survey in the next section.
In this paper we propose a new efficient stochastic simulation algorithm, called *SSAL*, which combines the advantages of Gillespie's Direct Method and of the τ-leaping. SSAL basically works as a standard DM but it verifies efficiently if the leap has to be interrupted or not. If it is the case, SSAL starts a new leap, otherwise, it computes a new pair (τ, j) reusing the same propensities and the sum of the preceding step. Moreover, the careful verification of the leap condition makes it impossible that some population becomes negative as we will prove later on.
This paper is organized as follows. In Sec. 2, we introduce some basic notation, we overview DM and MTL. We introduce our SSAL and its asymptotic time complexity analysis in Sec. 3. In Sec. 4, we test the accuracy and the performance of SSAL and we compare it with the MTL on three realistic biological models. Some concluding remark are given in Sec. 5.

2. BACKGROUND

Consider a system with N species $\{S_1, \cdots, S_N\}$ interacting through M different chemical reactions $\{R_1, \cdots, R_M\}$. Assume that the system is well-stirred, that it is confined in a constant volume V and in thermal (but not chemical) equilibrium [14]. The system state is the multivariate random variable $X(t) = \mathbf{x} = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$. We write $\mathbf{x}_i(t)$, or more precisely $\mathbf{x}_i$, to indicate the number of molecules of species S_i in the system at time t. Given an initial value $X(t_0)$ for the state, the problem is determining the value of $X(t)$ at any later time t. The evolution in time of the state of a chemical system is consequence of chemical reactions. In the stochastic formulation of chemical kinetics, chemical reactions are distinct, essentially instantaneous physical events involving two basic types. Unimolecular, written $S_1 \xrightarrow{c_j} Product(s)$, occurs as a result of processes internal to a single molecule with kinetic parameter c_j, and bimolecular, written $S_1 + S_2 \xrightarrow{c_j} Product(s)$ of two (distinct or not) molecular species occurs as a result of a collision. Any reaction R_j is characterized by two quantities. The first is the *state change vector* $\nu_j \equiv (\nu_{1j}, \cdots, \nu_{Nj})$, where ν_{ij} is the change in the population of S_i, caused by R_j. In other words, if the system is in state $\mathbf{x}$ and one R_j reaction occurs, the system immediately jumps to state $\mathbf{x} + \nu_j$. The second quantity is the *propensity function* a_j, also written $a_j(\mathbf{x})$, which is defined so that $a_j(\mathbf{x})dt$ gives the probability that the reaction R_j will occur in the next infinitesimal time interval $[t, t + dt)$.

2.1 Stochastic Simulation Algorithm: DM

To simulate a system exist two historical approaches: the *deterministic approach* and the *stochastic approach*. The deterministic approach regards the time evolution of $X(t)$ as a continuous and wholly predictable process, the value of the ith $X_i(t)$ can be computed by set up and solve a set of coupled, first order, ordinary differential equations, called *Reaction Rate Equations*. In macroscopic systems, with a large number of interacting molecules, the randomness of this behavior averages out so that the overall macroscopic state of the system becomes highly predictable. It is this property of large scale random systems that enables a deterministic approach to be adopted; however, the validity of this assumption becomes strained in conditions as we examine small-scale cellular reaction environments with limited reactant populations. The stochastic approach of chemical kinetics is simply a consequence of taking seriously the fact that reactions in a system of molecules in thermal equilibrium occur in an essentially *random* manner. It regards the time evolution of a system as a kind of random-walk process which is governed by the Chemical Master Equation. As we already mentioned, the difficulties to handle the CME enabled the more simple strategy of advance the state evolution through single realizations. Now we survey the core of this method.
SSA moves the system forward in time by determining when will the next reaction occur and what reaction it will be. In the DM formulation the time τ is an exponential random variable with mean (and standard deviation) $1/a0$, where $a_0(\mathbf{x}) = \sum_{j=1}^{M} a_j(\mathbf{x})$, whereas, the index j is a statistically independent integer random variable with point probabilities $a_j/a0$. Formally, DM defines the probability at time t that the next reaction in the considered volume V will occur in the infinitesimal time interval $(t + \tau, t + \tau + d\tau)$, and this reaction will be R_j. It denotes this probability with the following $P(\tau, j \mid \mathbf{x}(t))d\tau$.

$$P(\tau, j \mid \mathbf{x}(t))d\tau = a_j(\mathbf{x})e^{-a_0(\mathbf{x})\tau}d\tau \tag{1}$$

As pointed out in [16], one way to generates a pair from Eq. 1 is to draw two independent uniformly distributed random samples $r_1$and r_2 in the unit interval $U(0, 1)$, taking

$$\tau = \frac{1}{a_0(\mathbf{x})} \ln\left(\frac{1}{r_1}\right) \tag{2}$$

and

$$j = \text{the smallest integer such that} \sum_{j'=1}^{j} a_{j'}(\mathbf{x}) > r_2 a_0(\mathbf{x}). \tag{3}$$

Giving M reactions and kinetic constants, N species, one initial state $X(t_0)$ and a stop time TIME, DM generates one possible random evolution of the state performing the elementary steps summarized in Algorithm 1. The algorithmic time complexity of DM is computed and analyzed as following. The costs are:

1. C_{a_j}, to compute M propensity functions a_j,
2. C_{a_0}, to sum M propensity a_j and obtain a_0,

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8665
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8665

Algorithm 1 Direct Method

```
while t < TIME do
  for j=1 to M do
    Compute a_j(x)
  end for
  Compute a_0
  Generates r_1 and r_2 in U(0,1) and generate values for
  τ and j according to Eq. 2 and Eq. 3
  t ← t + τ; x ← x + ν_j;
  print (t, x)
end while
```

3. C_{2r}, to generate the two uniformly distributed random numbers r_1 and r_2,
4. C_τ, to find the next occurring time τ,
5. C_j, to find the next reaction to fire,
6. C_{update}, to update the system state and the simulation time.

To simplify we assume that multiplication, division, sum, comparison, assignment and random number generation have the same unitary cost. So the costly operation within a single step are [5]:

1. C_{a_j} of order of magnitude $\Theta(M)$,
2. C_{a_0} of order of magnitude $\Theta(M)$,
3. C_j of order of magnitude $O(M)$.

According to the preceding costs and orders of magnitude, Eq. 4 summarizes the asymptotic time complexity for DM.

$$\begin{aligned} T_{DM}(M,N,n) &= (C_{a_j} + C_{a_0} + C_{2r} + C_\tau + C_j + C_{update})n \\ &= \Theta((M + M + 1 + 1 + M + 1)n) \\ &= \Theta(Mn), \end{aligned} \tag{4}$$

where n is the number of steps of the simulation.

2.2 Modified τ-leaping and Efficient τ Formula

The idea of the τ-leaping is that the simulation can be divided into contiguous subintervals, or *leap*. Substantial speed up can be achieved if many reactions can fire into a leap and if the leap computation can be done expeditiously [13]. The key constructs of the τ-leaping are the *Leap Condition* and the *Leap Approximation*. Suppose that the system is in state $\mathbf{x}$ at time t, the leap condition states the existence of a time value τ', such that, during the time interval $[t, t+\tau']$, every propensity function remains approximately constant to the value $a_j(\mathbf{x})$ at time t. The first leap condition formulation [13] required that, for every reaction R_j, the absolute fractional change $\Delta a_j(\mathbf{x})/a_j(\mathbf{x})$ during the leap never exceeded a *user-defined tolerance parameter* $\epsilon \ll 1$. Mathematically it is written as follows.

$$| a_j(\mathbf{x}(t+\tau')) - a_j(\mathbf{x}(t)) | \leq \max\{\epsilon a_j(\mathbf{x}(t)), 1\} \quad j = 1, \cdots, M. \tag{5}$$

The tolerance parameter ϵ determines the efficiency and the result accuracy of a τ-leaping method based on the preceding leap condition. Indeed, if we reduce the value of ϵ, we admit fewer changes in the propensity functions. This means shorter leaps, higher simulation time, but also greater accuracy. The *Leap Approximation*, instead, approximates the number of times a given reaction R_j fires during the leap as the Poisson distributed random variable $\mathcal{P}_j(a_j\tau')$. According to $\mathcal{P}_j(a_j\tau')$, a τ-leaping method generates a random sample k_j for each reaction R_j. So the state of the system at time $t+\tau'$ can be obtained from $\mathbf{x}$ at time t by applying the formula $X(t+\tau') = \mathbf{x} + \sum_{j=1}^{M} \nu_j k_j$, where ν_j is the state change vector. As the larger τ', the larger values for $\mathcal{P}_j(a_j\tau')$ and k_j, it is therefore important estimating the largest τ' consistent with the leap condition. In literature the procedure to select the largest τ' is named the *τ-selection procedure*[13, 12]. One of the most accurate, easier to implement and fast to execute has been recently proposed by Cao et. al in [3]. The underlying strategy of this procedure is to bound the relative change in molecular populations in such a way that the relative changes in the propensity functions are all bounded by the value ϵa_j. Cao et al. reformulated the leap condition definition as follows.

$$| \mathbf{x}_i(t+\tau') - \mathbf{x}_i(t) | \leq \max\{\epsilon_i \mathbf{x}_i(t), 1\} \qquad i \in I_{rs}. \tag{6}$$

In Eq. 6 I_{rs} denotes the set of indices of the species participating as reactant to at least one reaction, whereas, the values ϵ_i are selected to approximatively bound by ϵ the relative changes in all the propensity functions [3]. This means that the leap condition formulation in Eq. 6 implies that reviewed in Eq. 5.

Unfortunately, using the above procedure, sometimes the selected τ' can induce too large changes and the population of some reactant with few molecules can become negative. The analysis of the phenomenon revealed that negative population arise for two main reasons. Being the Poisson distribution unbounded, a sample k_j generated in the leap approximation can exceed the maximum number of times that R_j can fire before consuming one of its reactants. Than since each propensity function change gets estimated separately, two reactions sharing a common reactant, acting together may overdraw that reactant [24, 8, 1, 22]. To deal with negative populations, some methods substitutes the Poisson distribution with a bounded one, such as a Binomial or a Multinomial distribution [22, 8, 22]. These methods select the maximum number of firings of R_j permitted during the leap, and they force the next state changes to remain bounded according those L_j's. Some issues pointed out in [1] for the binomial τ-leaping methods have been resolved in [1] proposing the Modified τ-leaping (MTL). MTL became one of the most known and efficient method, for this reason it is one of the focus of this paper. Below we survey it in detail.

MTL introduced the integer value $L_j(\mathbf{x})$, that denotes the maximum number of times a reaction R_j can fire before exhausting one of its reactants. $L_j(\mathbf{x})$ is function of the state $\mathbf{x}$ and the state change vector ν_j, as we recall in Eq. 7.

$$L_j(\mathbf{x}) = \min_{i \in I_{rs}} \left[\frac{x_i}{| \nu_{ij} |} \right]. \tag{7}$$

Given a new user-defined parameter, named n_c, MTL splits the reaction set into two sets: the critical set J_c and the non critical J_{nc}. It puts a reaction R_j in J_c if $L_j(\mathbf{x}) < n_c$ or in J_{nc} if $L_j(\mathbf{x}) > n_c$. The division allows to MTL to handle differently the critical and the non critical reactions. The non critical reaction set J_{nc} serves to generate the largest τ' consistent with leap condition in Eq. 6 according to the

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8665
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8665

Formula 8.

$$\tau' = \min_{i \in I_{rs}} \left\{ \frac{\max\{\epsilon x_i / g_i, 1\}}{\mid \sum_{j \in J_{nc}} \nu_{ij} a_j(\mathbf{x}) \mid}, \frac{\max\{\epsilon x_i / g_i, 1\}^2}{\sum_{j \in J_{nc}} \nu_{ij}^2 a_j(\mathbf{x})} \right\} \quad (8)$$

MTL shifts to DM for a number q of steps if τ' is smaller than $p\frac{1}{a_0(\mathbf{x})}$, where q and p are two others user-defined parameters and $1/a_0(\mathbf{x})$ is the mean waiting time for the next firing reaction. The critical set J_c is managed differently. MTL imposes that at most one critical reaction can occur during the leap. It first determines the occurrence of the next critical reaction τ'' and then it compares the times $t + \tau''$ and $t + \tau'$. If $t + \tau'' < t + \tau'$ then MTL reduces the leap time assigning $\tau' = \tau''$, then it selects the next critical reaction j_c setting $k_{j_c} = 1$ and $k_j = 0$ for all $j \in J_c/j_c$. For all $j \in J_{nc}$ it generates the samples k_j according to $\mathcal{P}_j(a_j\tau')$. If $t + \tau'' > t + \tau'$ MTL sets $k_j = 0$ for all $j \in J_c$ and it generates $k_j = \mathcal{P}_j(a_j\tau')$ for all $j \in J_{nc}$.
Schematically, giving M reactions and kinetic constants, N species, one initial state $X(t_0)$ and a stop time TIME, four user-defined parameter n_c, ϵ, q and p, MTL performs the elementary steps in Algorithm 2. Below, we analyze the algorithmic time complexity of MTL. The costs are:

1. C_{a_j} and C_{a_0}, the same costs seen for DM,
2. C_L, involves both the costs for the computation of the M quantity L_j and to split the reaction set,
3. $C_{\tau'}$, the cost for τ-selection formula Eq. 8,
4. C_{DM}, the cost to execute q DM steps,
5. $C_{\tau''}$, the cost to compute the firing time of the next critical reaction,
6. $C_{\tau' < \tau''}$, the cost to compute the leap approximation in case no critical reactions fires during the leap,
7. $C_{\tau' \geq \tau''}$, the cost to compute the leap approximation in case the next critical reaction fires during the leap,
8. C_{neg} and C_{update}, respectively, the cost of the N checks to find eventual negative populations and the cost to apply the formula $\mathbf{x}_i \leftarrow \mathbf{x}_i + \sum_{j=1}^{M} \nu_{ij} k_j$ and $t \leftarrow t + \tau$.

The above costs have the following order of magnitude.

1. C_{a_j} and C_{a_0} are $\Theta(M)$.
2. C_L is $\Theta(M)$ because L_j must be computed M times, and for each reaction it performs at least one division and one comparison to decide if it is critical or not.
3. $C_{\tau'}$ is $\Theta(N + M)$. The τ-selection formula finds the minimum among N tentative leap times, that is one for each species in I_{rs}, this is $O(N)$. Then as the maximum number of multiplications of each propensity a_j in $\sum_{j \in J_{nc}} \nu_{ij} a_j(\mathbf{x})$ or $\sum_{j \in J_{nc}} \nu_{ij}^2 a_j(\mathbf{x})$ of Eq. 8 is at most equal to the number of the reactants of R_j, Eq. 8 needs at most $\Theta(M)$ propensity multiplications. This is $\Theta(M)$.
4. C_{DM} is $\Theta(Mq)$.
5. $C_{\tau''}$ is $O(M)$ because for $\mid J_c \mid = M$ the firing time of the next critical reaction requires to sum at most M propensities for a_{0c}.

Algorithm 2 Modified τ-leaping

```
while t < TIME do
  for j=1 to M do
    Compute a_j(x)
  end for
  Compute a_0
  for j=1 to M do
    Compute L_j according to Eq. 7
    if L_j < n_c then
      J_c ← J_c ∪ j
    else
      J_nc ← J_nc ∪ j
    end if
  end for
  if J_nc ≠ {} then
    Compute τ' according to Eq. 8
  else
    τ' ← ∞
  end if
  repeat
    if τ' < (p · 1/a_0) then
      temp ← t
      Execute q DM steps
    else
      if J_c ≠ {} then
        Compute τ'' = 1/(a0_c(x)) ln(1/r_1)
      else
        τ'' = ∞
      end if
      if τ' < τ'' then
        for all j ∈ J_c do
          k_j ← 0
        end for
        for all j ∈ J_nc do
          k_j ← P_j(a_j τ');
        end for
      else
        τ' ← τ''
        j_c ← j' such that Σ_{j'∈J_c} a_{j'}(x) > r_2 Σ_{j∈J_c} a_j
        k_{j_c} ← 1
        for all j ∈ J_c/j_c do
          k_j ← 0
        end for
        for all j ∈ J_nc do
          k_j ← P_j(a_j τ');
        end for
      end if
      temp ← t
      t ← t + τ
      x_i ← x_i + Σ_{j=1}^{M} ν_ij k_j
      for j from 1 to N do
        if x_i < 0 then
          x_i ← x_i − Σ_{j=1}^{M} ν_ij k_j
          t ← t − τ;
          τ' = τ'/2
        end if
      end for
    end if
  until t = temp
  print (t, x)
end while
```

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8665
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8665

6. $C_{\tau' < \tau''}$ and $C_{\tau' \geq \tau''}$ are ΘM. They are mutually exclusive, and both compute M values k_j's. To simplify we assume that the Poisson random number generation is $O(1)$.

7. C_{neg} and C_{update} are $O(N)$ and $\Theta(M)$ because MTL checks at most N values of the state to find negative populations and it computes at most M unitary operations, one for each k_j, respectively.

For readability, in Eq. 9 we group together some of the above costs and we named the group C_{TLEAP}.

$$C_{TLEAP} = C_{\tau''} + \max\{C_{\tau' < \tau''}, C_{\tau' \geq \tau''}\} + C_{neg} + C_{update} \tag{9}$$

Now, considering the orders of magnitude introduced above and assuming that M and N are of the same magnitude, we obtain the time complexity in Eq. 10. The values n', n'' and n''' denote the number of shifts to DM during the simulation, the number of leap and the number of reactions fired in all the shifts to DM, respectively. Eq. 10 summarizes the time complexity for MTL.

$$\begin{aligned} T_{MTL}(M, N, n', n'') &= (C_{a_j} + C_{a_0} + C_L + C_{\tau'})(n' + n'') + \\ (C_{DM} n' + C_{TLEAP} n'') &= \Theta(M((q+1)n' + 2n'')) \\ &= \Theta(M(n''' + n'')) \\ &= \Theta(Mn''' + Mn''). \end{aligned} \tag{10}$$

In Eq. 10 Mn''' regards the part of complexity of the shifts to DM, whereas Mn'' regards the part of the τ-leaping. As the efficiency of MTL depends very much by the sum $n''' + n'$, we compare MTL and DM according to distinct values of $n''' + n'$. If the sum $n''' + n'$ is of order of magnitude of n, the time complexity for MTL is $\Theta(Mn)$, and MTL and DM coincide. Instead, if $n''' + n'$ is of order of magnitude $\frac{n}{M}$, it results that $T_{MTL}(M, N, n', n'') = \Theta(n) < T_{DM}(M, N, n)$ and this means that asymptotically MTL performs better than DM.

3. OUR PROPOSAL: SSALEAPING (SSAL)

As already mentioned in Sec. 1 and confirmed by the analysis in the previous section, substantial gain in simulation speed can be achieved by MTL if each leap fires many reactions and few shifts occur (i.e. the n'' and n''' are small). Apart the optimistic case, it can happen that n'' and n''' are not so small. In other words, MTL can frequently shifts to DM, continuing to perform some of the extra operations identified with the costs C_L, $C_{\tau'}$, $C_{\tau' \geq \tau''}$, C_{neg} and C_{update}. When frequent shifts to DM occur, the burden introduced by these extra operations can slow down the performance. This can lead MTL to be slower than DM as well. In particular, when M and N are large. The minimum number of reactions to fire in a leap that guarantee a good speed up of MTL with respect to DM is fixed by the parameter p. However, this parameter is an heuristic and an arbitrary constant. To deal efficiently with the bad cases described above we propose a new method, that we call *SSALeaping* or SSAL for short. The idea of SSAL is very simple. SSAL generates values for τ and j according to Eq. 2 and Eq. 3, then it updates the system state $X(t)$ according to the state change vector ν_j and it checks if the changes in some population breaks down the leap condition in Eq. 6. If it is the case, SSAL recomputes all propensities and a_0, otherwise, for the next generation of the values τ and j it reuses the same a_j and a_0. The extra cost payed for the verification of the leap condition is small compared to the extra costs seen for MTL. The verification also allows to builds leap adaptively, and it makes impossible that some species population becomes negative. To do that the property used by SSAL is described in the next subsection.

3.1 No Negative Populations

We consider a reaction R_j and we solve the inequality $\mid a_j(\mathbf{x} + \nu_{j'}) - a_j(\mathbf{x}) \mid > \epsilon a_j(\mathbf{x})$, that is, if the leap condition for R_j is violated. To consider the most general case we identify the maximum state change that a propensity function $a_j(\mathbf{x})$ can undergo when a reaction $R_{j'}$ fires. Below we list all possible representative cases for R_j with their relative maximum changes.
The first case is when R_j is the unimolecular reaction $R_j : S_1 \xrightarrow{c_j} Product(s)$. For the unimolecular reaction the maximum change of the propensity $a_j(\mathbf{x})$ happens for $\nu_{1j'} = -2$.

$$\begin{aligned} \mid a_j(\mathbf{x} + \nu_{j'}) - a_j(\mathbf{x}) \mid &> \epsilon a_j(\mathbf{x}) \\ \mid (\mathbf{x}_1 - 2)c_j - \mathbf{x}_1 c_j \mid &> \epsilon \mathbf{x}_1 c_j \\ 2 &> \epsilon \mathbf{x}_1 \\ \frac{2}{\epsilon} &> \mathbf{x}_1 \end{aligned} \tag{11}$$

Here, the leap condition of R_j can be violated when $\mathbf{x}_1 < 2/\epsilon$.
The second case is for the bimolecular reaction $R_j : 2S_1 \xrightarrow{c_j} Products$. In this case, the maximum change of the propensity for R_j happens for $\nu_{1j'} = -2$ and we have the following inequality.

$$\begin{aligned} \mid a_j(\mathbf{x} + \nu_{j'}) - a_j(\mathbf{x}) \mid &> \epsilon a_j(\mathbf{x}) \\ \mid \frac{(\mathbf{x}_1 - 2)(\mathbf{x}_1 - 3)}{2} c_j - \frac{\mathbf{x}_1(\mathbf{x}_1 - 1)}{2} c_j \mid &> \epsilon \frac{\mathbf{x}_1(\mathbf{x}_1 - 1)}{2} c_j \\ \mid (\mathbf{x}_1 - 2)(\mathbf{x}_1 - 3) - \mathbf{x}_1(\mathbf{x}_1 - 1) \mid &> \epsilon \mathbf{x}_1(\mathbf{x}_1 - 1) \\ \mid -4\mathbf{x}_1 + 6 \mid &> \epsilon \mathbf{x}_1(\mathbf{x}_1 - 1) \end{aligned}$$

The absolute value $\mid -4\mathbf{x}_1 + 6 \mid$ has two distinct cases. If $-4\mathbf{x}_1 + 6 \geq 0$ we resolve in the following way.

$$\begin{aligned} \mid -4\mathbf{x}_1 + 6 \mid &> \epsilon \mathbf{x}_1(\mathbf{x}_1 - 1) \\ -4\mathbf{x}_1 + 6 &> \epsilon \mathbf{x}_1(\mathbf{x}_1 - 1) \\ \epsilon \mathbf{x}_1^2 - (\epsilon - 4)\mathbf{x}_1 - 6 &< 0 \\ \mathbf{x}_1 &< 2. \end{aligned} \tag{12}$$

The leap condition of R_j can be violated when $\mathbf{x}_1 < 2$. Instead, if $-4\mathbf{x}_1 + 6 < 0$ we treat it as follows.

$$\begin{aligned} \mid -4\mathbf{x}_1 + 6 \mid &> \epsilon \mathbf{x}_1(\mathbf{x}_1 - 1) \\ 4\mathbf{x}_1 - 6 &> \epsilon \mathbf{x}_1(\mathbf{x}_1 - 1) \\ \epsilon \mathbf{x}_1^2 - (\epsilon + 4)\mathbf{x}_1 + 6 &< 0 \\ 2 \leq \mathbf{x}_1 &< \frac{(\epsilon + 4) + \sqrt{}((\epsilon + 4)^2 - 24\epsilon)}{(2\epsilon)} \end{aligned} \tag{13}$$

Finally, consider $R_j : S_1 + S_2 \xrightarrow{c_j} Products$, we have two interesting state change cases: $\nu_{1j'} = -2, \nu_{2j'} = 0$ and $\nu_{1j'} = -1, \nu_{2j'} = -1$. In the first case the result is the

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8665
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8665

following.

$$\begin{aligned} \mid a_j(\mathbf{x}+\nu_{j'}) - a_j(\mathbf{x}) \mid &> \epsilon a_j(\mathbf{x}) \\ \mid (\mathbf{x}_1-2)\mathbf{x}_2 c_j - \mathbf{x}_1\mathbf{x}_2 c_j \mid &> \epsilon \mathbf{x}_1\mathbf{x}_2 c_j \\ \mid -2\mathbf{x}_2 \mid &> \epsilon \mathbf{x}_1\mathbf{x}_2 \\ \frac{2}{\epsilon} &> \mathbf{x}_1. \end{aligned} \tag{14}$$

Instead, the second case results as follows.

$$\begin{aligned} \mid a_j(\mathbf{x}+\nu_{j'}) - a_j(\mathbf{x}) \mid &> \epsilon a_j(\mathbf{x}) \\ \mid (\mathbf{x}_1-1)(\mathbf{x}_2-1) c_j - \mathbf{x}_1\mathbf{x}_2 c_j \mid &> \epsilon \mathbf{x}_1\mathbf{x}_2 c_j \\ \mid -\mathbf{x}_1 - \mathbf{x}_2 + 1 \mid &> \epsilon \mathbf{x}_1\mathbf{x}_2 \end{aligned}$$

The absolute value $\mid -\mathbf{x}_1 - \mathbf{x}_2 + 1 \mid$ has two distinct cases. If $-\mathbf{x}_1 - \mathbf{x}_2 + 1 < 0$ we resolve in the following way.

$$\begin{aligned} \mid -\mathbf{x}_1 - \mathbf{x}_2 + 1 \mid &> \epsilon \mathbf{x}_1\mathbf{x}_2 \\ \mathbf{x}_1 + \mathbf{x}_2 - 1 &> \epsilon \mathbf{x}_1\mathbf{x}_2 \\ \epsilon \mathbf{x}_1\mathbf{x}_2 - \mathbf{x}_1 - \mathbf{x}_2 + 1 &< 0 \\ \mathbf{x}_1(\epsilon \mathbf{x}_{2-1}) &< \mathbf{x}_2 - 1 \\ \mathbf{x}_1 &< \frac{\mathbf{x}_2 - 1}{(\epsilon \mathbf{x}_2 - 1)} \end{aligned} \tag{15}$$

Instead, if $-\mathbf{x}_1 - \mathbf{x}_2 + 1 \geq 0$ we treat it as follows.

$$\begin{aligned} \mid -\mathbf{x}_1 - \mathbf{x}_2 + 1 \mid &> \epsilon \mathbf{x}_1\mathbf{x}_2 \\ -\mathbf{x}_1 - \mathbf{x}_2 + 1 &> \epsilon \mathbf{x}_1\mathbf{x}_2 \\ \mathbf{x}_1(\epsilon \mathbf{x}_2 + 1) - \mathbf{x}_1) &< -\mathbf{x}_2 - 1 \\ \mathbf{x}_1 &= 0 \\ \mathbf{x}_2 &= 0 \end{aligned} \tag{16}$$

Next we give one numerical example to show how to use the bounds defined above.
Suppose to have $\epsilon = 0.03$, a unimolecular reaction $R_j : S_1 \xrightarrow{c_j} Product(s)$ and $\mathbf{x}_1 = 10$. Assume that a reaction $R_{j'}$ fires and $\nu_{1j'} = -1$. Than if we check the leap condition on R_j, we obtain the following result.

$$\begin{aligned} \mid a_j(\mathbf{x}+\nu_{j'}) - a_j(\mathbf{x}) \mid &\leq \epsilon a_j(\mathbf{x}) \\ \mid (\mathbf{x}_1-1)c_j - \mathbf{x}_1 c_j \mid &\leq \epsilon \mathbf{x}_1 c_j \\ \mid (9-10 \mid &\leq 0.03 * 10 \\ 1 &\leq 0.3 \end{aligned}$$

Now $1 \leq 0.3$ is false, so the leap condition for R_j is violated, but as we expect $\mathbf{x}_1 = 10 < 66 = 2/\epsilon$. Similar examples can be provided for any reaction.
Summarizing, Formulas 11-16 give the thresholds under which any state change, involving a population $\mathbf{x}_i$, violates the leap condition of some reaction of which the species S_i is reactant.
In addition an important observation about those thresholds is the following. The leap condition is violated frequently, when species with small populations are involved in fast reactions. In those cases, fast reactions occur frequently and their firing violates the leap condition many times because they change the population of species with few molecules. In literature exist some discussions about the worst case conditions for τ-leaping methods. Cao et al. [9, 4] stated that τ-leaping methods still have difficulty in effectively handling the situation when multiple time and population scale coexist, particularly when a species with a small population is involved in a fast reaction. Harris et al. [18] say that small reaction subnetworks (e.g. reversible reactions) that have small populations and large rate constants are the main bottlenecks for explicit leaping algorithms. In summary, small numbers and stiffness are considered the conditions causing worst cases.
This broadly accepted conclusions are undoubtedly true, but the Formulas 11-16 tell us more. They confirm numerically the above discussions providing also numerical thresholds under which a population lays into what is often named *small number* population.

3.2 Optimizations and Comparisons

SSAL uses some algorithmic optimization that we summarize below. The first is on the leap condition. We reformulate the leap condition by substituting $\epsilon_i x_i$ to value $\max\{\epsilon_i x_i, 1\}$ in Formula 6 yielding

$$\mid \mathbf{x}_i(t+\tau') - \mathbf{x}_i(t) \mid \leq \epsilon_i \mathbf{x}_i(t) \qquad i \in I_{rs}. \tag{17}$$

Here, I_{rs} is the set of species of the system that act as reactant at least in one reaction. Additionally, we introduce the constraint that if the leap condition of a given reaction is violated after the firing of a reaction $R_{j'}$, SSAL aborts the current leap maintaining $R_{j'}$ the last fired reaction. In this way, any leap fires at least one reaction. This makes Formulas 6 and 17 equivalent.
The second optimization increases efficiency in the selection of the reaction to fire. This optimization is the core of the *Logarithm Direct Method* (LDM) [20]. LDM accumulates the partial sums of the propensities and it stores them in an array A. The ordered sequence of partial sums enable a binary search that finds the position j such that subtotal satisfies $A[j] < a_0 r_2 < A[j+1]$.
Our algorithm depicted in Alg. 3 takes as inputs M reactions and kinetic constants, N species, one initial state $X(t_0)$, a stop time TIME and a tolerance parameter ϵ. Given a reaction R_j, we define the set $Reactants(R_j)$ as the indices $i \in \{1, \cdots, N\}$ such that $\nu_{ij} < 0$ and the set $Products(R_j)$ as the indices $i \in \{1, \cdots, N\}$ such that $\nu_{ij} > 0$. We also define the set $((Reactants(R_j) \cup Products(R_j)) \cap I_{rs})$, that is the set of reactants or products species of the reaction R_j that are also reactants of at least one reaction.
Now, as we did for DM and MTL we provide the algorithmic time complexity and the asymptotic analysis of SSAL. The costs of a single step are:

1. C_{a_j}, C_{a_0}, C_{2r}, C_τ,C_{update}, the same of DM
2. C_{copy}, the cost to make a copy of the state $X(t)$.
3. C_j, the cost for the binary search,
4. C_{Leap}, the cost to verify the leap condition in Formula 17.

The costs are of the following order of magnitude.

1. C_{a_j}, C_{a_0}, C_{2r}, C_τ,C_{update} are the same as for DM.
2. C_{copy} is $\Theta(N)$ because the state has N elements.
3. C_j is $\Theta(\log_2 M)$.
4. C_{Leap} is $O(1)$ because checking the leap condition involves few arithmetic operations for each reactant and product of the fired reaction. Each reaction involves at most four species.

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8665
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8665

Algorithm 3 SSALeaping

while $t < TIME$ **do**
  Compute $a_1(\mathbf{x})$
  for j=2 to M **do**
    Compute $a_j(\mathbf{x})$
    $A[j] \leftarrow A[j-1] + a_j(\mathbf{x})$
  end for
  $a_0 \leftarrow A[M-1]$
  Store a copy of $\mathbf{x}(t)$
  $OK \leftarrow true$
  $\tau' \leftarrow 0$
  while $t + \tau' < TIME$ and $OK = true$ **do**
    Generate r_1 and r_2 in $U(0,1)$ and generate values for τ according to Eq. 2. Through binary search find j according to Eq. 3.
    $\mathbf{x}(t + \tau' + \tau) \leftarrow \mathbf{x}(t + \tau') + \nu_j$;
    $\tau' \leftarrow \tau' + \tau$;
    for all $i \in ((Reactants(R_j) \cup Products(R_j)) \cap I_{rs})$ **do**
      if $\mid \mathbf{x}_i(t + \tau') - \mathbf{x}_i(t) \mid > \epsilon_i \mathbf{x}_i(t)$ **then**
        $OK \leftarrow false$
      end if
    end for
  end while
  $t \leftarrow t + \tau'$
  print $(t, \mathbf{x})$
end while

For SSAL, a leap is a time interval between two consecutive violations of the leap condition. A leap involves one or more steps. Because for each step, SSAL pays the costs C_{2r}, C_τ, C_j, C_{update} and C_{Leap}, whereas, for each leap, SSAL pays the costs C_{a_j}, C_{a_0} and C_{copy}, we denote the number of steps with n and the number of leap with k. Eq. 18 summarizes the asymptotic time complexity for SSAL.

$$\begin{aligned} T_{SSAL}(M,N,n) &= (C_{a_j} + C_{a_0} + C_{copy})k + (C_{2r} + C_\tau + C_j + \\ &+ C_{update} + C_{Leap})n \\ &= \Theta((M + M + N)k + (log(M) + 1)n) \\ &= \Theta(Mk + \log(M)n). \end{aligned} \tag{18}$$

Now we are able to compare the complexity of SSAL with the complexity given for DM and MTL.
In the worst case $k = \Theta(n)$, the complexity of SSAL and DM coincide. If we consider $n''' + n'' = \Theta(n)$, the complexity of SSAL and MTL coincide. Without loss of generality, suppose that k and n'' are of the same order of magnitude. In this case, theoretically SSAL performs better than MTL if it holds the inequality $n''' > \frac{\log(M)n}{M}$. Whereas MTL performs better than SSAL otherwise. In other words, SSAL performs better than MTL if the number of reactions fired by MTL when it shifts to DM exceeds the bound $\frac{\log(M)n}{M}$.

4. NUMERICAL EXPERIMENTS

The asymptotic analysis estimates theoretical information about efficiency, costly operations and relations among those operations and simulation parameters. However, to give a more pragmatic comparison of SSAL, MTL and DM we investigated how efficiency changes in practice. We provided experimental tests that consider different model parameters taken from realistic biological models. Our tests are made upon the Decaying-Dimerizing, Map Kinase Cascade and LacZ/LacY models, which span from four up to hundreds of reactions. In the LacZ/LacY model the cell volume is assumed to grow in time. The population of two species are randomly determined from two Normal random variables, and the mean values of these variables grow together with the volume of the cell. To the best of our knowledge, existing toolkits that implement MTL (i.e. Stochkit), even though extensible, they do not allow to specify those model features yet. For this reason, we realized our C language implementations of DM, MTL and SSAL and we used those implementations to simulate LacZ/LacY and the others models. All experiments run on a WINDOWS XP personal computer with a 3.0 GHz CPU and 1 Gbyte memory. For each experiment, we collected the final states of a selected species taken from 1000 independent simulations. We estimate the accuracy of the results by computing first the histogram and the Kolmogorov distances [7] between 1000 samples of DM and MTL or SSAL. Then we compare those distance values with the so called *self distance* [7], interpreting the comparison as follows. The closer to the self distance a distance value is, the more accurate the method who has generated that samples will be. We performed the mean and variance of the histogram and Kolmogorov self distances according to the formulas given in [7]. The mean and variance for the histogram self distance are 0.079 and 0.49 respectively. Whereas for the Kolmogorov self distance are 0.0389 and 0.00136 respectively.
Then in order to cover the most large possible range of cases and parameters, we repeated experiment considering different ϵ values. For each test we take the values for: (ϵ), CPU Time ($CPUTimeSSAL/MTL$), number of reactions fired per leap for SSAL and for MTL (m'/m''), histogram distance ($HDSSAL/HDMTL$) and Kolmogorov distance ($KDSSAL/KDMTL$). Whereas, only for MTL, we consider: (ϵ), the number of MTL shifts to DM (n'), the number of leap and reaction fired ($n''' + n''$), the number of executions of the branch ($\tau' < \tau''$), the number of executions of the branch ($\tau'' < \tau'$).

4.1 Decaying-Dimerizing Model

This first test model is taken from [12]. It consists of three species S_1, S_2 and S_3 (N=3) and four reactions (M=4). A monomer S_1 reversibly dimerises to an unstable form S_2, which can convert to a stable form S_3. We simulate the model using the following stochastic coefficients: $c_1 = 1.0$, $c_2 = 0.002$, $c_3 = 0.5$ and $c_4 = 0.04$ and the initial state $\mathbf{X}(t_0) = (\mathbf{x}_1 = 4150, \mathbf{x}_2 = 39565, \mathbf{x}_3 = 3445)$. We set upped also stop time $TIME = 10$, $n_c = 10$, $q = 100$ and $p = 10$. To run 1000 independent simulations, DM required 43.625 seconds. Table 1 and Table 2 summarize the results for SSAL and MTL. Table 1 shows that m' is greater than m'' for small values of ϵ. In particular for $\epsilon = 0.004$. As we can see, SSAL fired $m' = 18.53$ reactions, whereas MTL only $m'' = 3.65$. It can be also noted from Table 2 that, for small ϵ values, MTL computes a lot of shifts. Being $\frac{\log(M)n}{M} = \frac{279036}{2} = 139138$ and $n''' \approx n' * q$, we have that $n''' > \frac{\log(M)n}{M}$ for any value $\epsilon \leq 0.001$. Although theoretically for $n''' > \frac{\log(M)n}{M}$ SSAL ought to perform better than MTL, for this specific biological model, the CPU time of MTL results always smaller than the CPU time of SSAL

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8665
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8665

Table 1: Results comparison between SSAL and MTL for 1000 independent simulations for the Decaying-Dimerizing reactions. Accuracy have been taken for the values $X_0(10.0)$.

ϵ	$CPUTime\ SSAL/MTL$	m'/m''	$HDSSAL/HDMTL$	$KDSSAL/KDMTL$
0.0	58.75/51.26	1.0/1.0	0.518/0.544	0.035 / 0.04
0.001	59.9/51.29	1.5/1.0	0.518 / 0.542	0.029 / 0.035
0.004	44.3/43.87	18.53/3.65	0.56 / 0.502	0.046 / 0.028
0.01	42.37/9.32	91.87/78.29	0.494 / 0.54	0.037 / 0.025
0.03	43.39/1.2	698.36/699.57	0.482 / 0.558	0.05 / 0.047
0.06	43.48/0.37	2247.2/2258.4	0.56/ 0.594	0.077 / 0.114
0.1	42.6/0.2	4231.3/4107.6	0.646/0.78	0.103 / 0.191

($CPUTimeSSAL > CPUTimeMTL$), as we can see in Table 1. This happens because the model involves only four reactions, so it takes the same time to SSAL to compute and sum $M = 4$ propensity functions and to verify the leap condition. For the same reason, the CPU time of SSAL and DM almost coincide for $\epsilon > 0.004$.
Apart for $\epsilon = 0.1$, the histogram and Kolmogorov distances are very close to the histogram and Kolmogorov self distances provided.

Table 2: MTL statistics for the Decaying-Dimerizing reactions for 1000 independent run.

ϵ	$n''' + n''$	n'	$tau1 < tau2$	$tau2 < tau1$
0.0	279036.8	2790.85	0	0
0.001	279000.0	2790.5	0	0
0.004	76264.4	613.69	14943.8	0
0.01	3564.1	0.21	2562.8	0
0.03	399.2	0.027	399.1	0
0.06	123.1	0.011	123.83	0
0.1	68.39	0.003	68.37	0

4.2 Map Kinase Cascade Model

Recently Chatterjee et. al [6] applied their Binomial τ-leaping method to the signaling pathway of epidermal growth factor (EGF) receptor (EGFR) activated mitogen activated protein (MAP) kinase cascade. EGFRs belong to the receptor tyrosine kinase (RTK) family of receptors and play an important role in many physiological processes among which cell proliferation. This biological model involves 106 species and 296 reactions. The reaction set, initial amount and kinetic coefficients have been taken from the web site `http://www.dion.che.udel.edu/multiscale/software.html`. Addditionally, we set upped stop time $TIME = 0.1$, $n_c = 10$, $q = 100$ and $p = 10$. To run 1000 independent simulations, DM required 209.53 seconds. Table 3 and 4 summarize the results for SSAL and MTL. Table 3 shows that m' is greater than m'' for $\epsilon \leq 0.01$. Then being $\frac{\log(M)n}{M} = \frac{8.2*60237}{296} = 1668.7$ and $n''' \approx n' * q$, we have $n''' > \frac{\log(M)n}{M}$ for $\epsilon \leq 0.01$. Although theoretically when $n''' < \frac{\log(M)n}{M}$ MTL ought to perform better than SSAL, Table 3 shows that SSAL performs better than MTL in that range as well. This happens because the extra costs of MTL requires substantial computational extra time for $M = 296$ and $N = 106$. In Table 4 this has a peak in correspondence of $\epsilon = 0.001$. SSAL is almost one order of magnitude faster. Here, MTL executes $tau1 < tau2 = 3515.7$ times the τ-leaping branch, but the number of reaction fired in each of these leap is very small. This means that MTL computes τ' for $M = 296$, but unfortunately often the leap fires only one reaction. This case is a realistic example of the impact of the extra operations of MTL in the simulation time. Again the accuracy is very close to the self distances both for SSAL and MTL.

Table 4: MTL statistics for the Map Kinase Cascade reactions for 1000 independent run.

ϵ	$n''' + n''$	n'	$tau1 < tau2$	$tau2 < tau1$
0.0	60237	608.9	0	0
0.001	22266.1	189.87	3515.7	0.82
0.004	10742.3	88.58	2018.6	1.94
0.01	3725.3	22.24	1435.6	136.86
0.03	853.3	0.65	625.3	223.5
0.06	588.4	0.63	362.3	221.8
0.1	465.2	0.65	269.12	224.49

4.3 LacZ/LacY Model

This model was first introduced by Kierzek et al. in [19] but we consider the model reviewed in [22]. It consists of 23 species (N=23) and 22 reactions (M=22), details and kinetic parameters can be found in those references. In this model the cell volume is assumed to grow in time according to the formula $V(t) = V_0(1 + t/T_{gen})$. The initial cell volume is $V_0 = 10^{-15}$ liter and $T_{gen} = 2100$ seconds. Than the population of the two species RNAP and Ribosome are randomly determined from the two Normal random variables $N(35 * (1 + t/2100), 3.5)$ and $N(350 * (1 + t/2100), 35)$. The mean values of these variables grow together with the volume of the cell so that the concentrations of these molecules remain constant [24]. We simulate this system in the time interval $[300 - 330]$ because test in [0-300] takes too much time. In fact, 1000 independent simulations of DM required 4840 seconds and $n = 6.409324e^6$. For this reason, we simulated DM in the interval [0,300], we have taken the state of the simulation at time $t = 300$ and we used it as initial state for the SSAL and MTL simulations. MTL used the parameters: $n_c = 10$, $q = 100$ and $p = 10$. For this model 1000 independent simulations of DM in the interval [300-330] required 2053.1 seconds. Table 5 and Table 6 summarize the results for SSAL and MTL. Table 5 shows that the number of reaction fired m' is greater than m'' for each value ϵ. In particular, for $\epsilon = 0.03$ the number of reactions fired in a leap for SSAL averages at $m' = 10.75$, instead, for MTL it is $m'' = 1.056$. For the LacZ/LacY model, we have $\frac{\log(M)n}{M} = \frac{4.45*2877222}{22} = 581983.5$ and $n''' \approx n' * q$. Whereas it results that $n''' > \frac{\log(M)n}{M}$ for $\epsilon \leq 0.03$. Again CPU times show that SSAL performs better than MTL for any value ϵ in the table. In particular, SSAL is four times faster for

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8665
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8665

Table 3: Results comparison between SSAL and MTL for 1000 independent simulations for the Map Kinase Cascade reactions. Accuracy have been taken for the values $X_2(0.1)$.

ϵ	$CPUTime\ SSAL/MTL$	m'/m''	$HDSSAL/HDMTL$	$KDSSAL/KDMTL$
0.0	219.7 / 274.9	1.0 /1.0	0.476 / 0.462	0.04 / 0.049
0.001	45.77 / 434.0	6.37 /2.7	0.472 / 0.454	0.029 / 0.033
0.004	30.64 / 242.3	12.3 /5.6	0.48 /0.466	0.022 / 0.03
0.01	20.7 / 175.32	27.89 /16.67	0.454 / 0.478	0.043 / 0.021
0.03	16.29 / 88.5	59.68 /70.59	0.45 / 0.452	0.037 / 0.046
0.06	15.5 / 64.98	85.18 /102.38	0.522 / 0.518	0.049 / 0.041
0.1	14.59 / 52.48	106.1 /129.5	0.486 / 0.528	0.038 / 0.058

Table 5: Results comparison between SSAL and MTL for 1000 independent simulations for the LacZ/LacY reactions. Accuracy have been taken for the values $X_{trrbslacy}(330)$.

ϵ	$CPUTime\ SSAL/MTL$	m'/m''	$HDSSAL/HDMTL$	$KDSSAL/KDMTL$
0.0	1962.5/2546.8	1.0/1.0	0.36/0.28	0.03/0.21
0.001	1500.0/2534.0	1.5/1.0	0.35/0.34	0.023/0.026
0.004	1545.7/2555.9	1.58/1.0	0.362/0.358	0.034/0.024
0.01	1550.2/2503.8	1.5/1.0	0.36/0.358	0.024/0.022
0.03	670.9/2768.1	10.75/1.056	0.362/0.358	0.039/0.057
0.06	573.4/1809	33.29/28.05	0.372/0.37	0.038/0.026
0.1	546.9/682.4	77.28/76.15	0.402/0.378	0.026/0.028

$\epsilon = 0.03$. Accuracy of the results in Table 5 are very similar for SSAL and MTL. Table 6 shows that for small ϵ MTL shifts to DM, while in the other cases the simulation turns into the branches $tau1 < tau2$ and $tau2 < tau1$. We omit the negative population branch in all the tables because no negative populations occurred for the parameters chosen for the models.

Table 6: MTL statistics for the LacZ/LacY reactions for 1000 independent run.

ϵ	$n''' + n''$	n'	$tau1 < tau2$	$tau2 < tau1$
0.0	2877222	28772.6	0	0
0.001	2871591	28716.3	0	0
0.004	2880443	28736.8	0	0
0.01	2723601	28738.6	0	0
0.03	2724048	27064	14621.7	125.4
0.06	102529.3	10.8	99298.2	2405.4
0.1	37771	0.097	35386.5	2409.7

Now, we conclude by providing in Table 7 and Table 8 the speed-up of SSAL against DM and MTL respectively. The speed-up is taken by dividing the CPU Time of a method for the CPU Time of SSAL. Table 7 shows that, apart for some ϵ in the Decaying-Dimerizing model, SSAL performs better than DM. While Table 8 confirms that SSAL performs better than MTL in the non trivial cases.

Table 7: Speed-up between SSAL and DM.

ϵ	$Decay - Dimer$	$MapKinase$	$LacZ/LacY$
0.0	0.74	0.953	1.046
0.001	0.73	4.577	1.368
0.004	0.98	6.838	1.328
0.01	1.029	10.122	1.324
0.03	1.005	12.86	3.06
0.06	1.003	13.518	3.58
0.1	1.024	14.361	3.75

Table 8: Speed-up between SSAL and MTL.

ϵ	$Decay - Dimer$	$MapKinase$	$LacZ/LacY$
0.0	0.872	1.251	1.297
0.001	0.856	9.482	1.68
0.004	0.99	7.907	1.65
0.01	0.219	8.469	1.615
0.03	0.027	5.432	4.125
0.06	0.008	4.192	3.154
0.1	0.004	3.596	1.247

5. CONCLUSIONS

We proposed SSAL, a new method which lays in the middle between the direct method (DM) and a τ-leaping. SSAL *adaptively* builds leap and stepwise updates the system state. Differently from MTL, SSAL neither shifts from τ-leaping to DM nor pre-selects the time leap τ'. Additionally whereas MTL prevents negative populations taking apart critical and non critical reactions, SSAL generates sequentially the reactions to fire verifying the leap condition after each generation. We proved that a reaction overdraws one of its reactants if and only if the leap condition is violated. Therefore, this makes it impossible for the population to become negatives, because SSAL stops the leap generation in advance. In order to compare SSAL with existing methods, we focused on the Modified τ-leaping (MTL) and the direct method (DM). For them we provided the time complexity and a detailed asymptotic analysis. These allowed to abstract from many implementation details and model specifications, and it highlights both the bottleneck operations and the specific features of the model that make these methods inefficient. We showed that in the worst case, the complexity of MTL and SSAL reduce themselves to that of DM. Instead, SSAL performs better than MTL if the number of reactions fired sequentially by MTL exceeds the bound $\frac{\log(M)n}{M}$.
We also integrated the analysis with some numerical experiments to test how the above methods work in practice.

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8665
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8665

We run our implementations of SSAL, DM and MTL upon the Decaying-Dimerizing, the Map Kinase Cascade and the LacZ/LacY models. Results substantially agrees with the theoretical analysis provided. They showed that for MAPK and LacZ/LacY and for $\epsilon > 0$ our SSAL implementation performs better than the two implementations of MTL and DM. Additionally, they highlighted that $n''' > \frac{\log(M)n}{M}$ is the main range emerged for two of the three realistic biological models considered. In this range SSAL performs better than MTL.

However, the biological models tested in our experiments are not representative for all possible classes of models. For instance, we have not provided experiments with biological models in the class of stiff systems (systems with very fast and stable reactions), for which very efficient methods exist. In conclusion, results confirmed that SSAL is very promising to simulate realistic biological models even though, for the future, further analysis, experiments and comparisons are necessary to investigate as many as possible biological cases and methods.

6. REFERENCES

[1] Y. Cao, D. T. Gillespie, and L. R. Petzold. Avoiding negative populations in explicit poisson tau-leaping. *J. Chem. Phys.*, 123:4104, August 2005.

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

[3] Y. Cao, D. T. Gillespie, and L. R. Petzold. Efficient step size selection for the tau-leaping simulation method. *J Chem Phys*, 124(4), January 2006.

[4] Yang Cao, Daniel T. Gillespie, and Linda R. Petzold. Adaptive explicit-implicit tau-leaping method with automatic tau selection. *The Journal of Chemical Physics*, 126(22), 2007.

[5] Y. Cao, H. Li, and L. Petzold. Efficient formulation of the stochastic simulation algorithm for chemically reacting systems. *J Chem Phys*, 121(9):4059–4067, September 2004.

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

[7] Y. Cao and L. Petzold. Accuracy limitations and the measurement of errors in the stochastic simulation of chemically reacting systems. *Journal of Computational Physics*, 212(1):6–24, 2006.

[8] A. Chatterjee, D. G. Vlachos, and M. A. Katsoulakis. Binomial distribution based tau-leap accelerated stochastic simulation. *J Chem Phys*, 122(2), January 2005.

[9] Y. Cao and L. Petzold. Slow-scale tau-leaping method. *Computer Methods in Applied Mechanics and Engineering*, 197(43-44):3472–3479, 2008.

[10] Cristian Dittamo and Davide Cangelosi. Optimized parallel implementation of Gillespie's First Reaction Method on graphics processing units. *Computer Modeling and Simulation, International Conference on*, 0:156–161, 2009.

[11] Michael A. Gibson and Jehoshua Bruck. Efficient exact stochastic simulation of chemical systems with many species and many channels. *J. Phys. Chem. A*, 104:1876–1889, 2000.

[12] D. T. Gillespie and L. R. Petzold. Improved leap-size selection for accelerated stochastic simulation. *J. Chem. Phys.*, 119:8229–8234, October 2003.

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

[14] Daniel T. Gillespie. Stochastic simulation of chemical kinetics. *Annual Review of Physical Chemistry*, 58(1):35–55, 2007. PMID: 17037977.

[15] D. Gillespie. *Formal Methods for Computational Systems Biology*, volume 5016/2008, chapter Simulation Methods in Systems Biology, pages 125–167. Springer Berlin / Heidelberg, May 2008.

[16] Daniel T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, 22(4):403–434, December 1976.

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

[18] L. A. Harris, A. M. Piccirilli, E. R. Majusiak, and P. Clancy. Quantifying stochastic effects in biochemical reaction networks using partitioned leaping. *Physical Review E*, 79:051906, 2009.

[19] A. M. Kierzek, J. Zaim, and P. Zielenkiewicz. The Effect of Transcription and Translation Initiation Frequencies on the Stochastic Fluctuations in Prokaryotic Gene Expression. *Journal of Biological Chemistry*, 276(11):8165–8172, 2001.

[20] Hong Li and Linda Petzold. Logarithmic direct method for discrete stochastic simulation of chemically reacting systems. July 2006.

[21] James M. Mccollum, Gregory D. Peterson, Chris D. Cox, Michael L. Simpson, and Nagiza F. Samatova. The sorting direct method for stochastic simulation of biochemical systems with varying reaction execution behavior. *Computational Biology and Chemistry*, 30(1):39–49, February 2006.

[22] M. F. Pettigrew and H. Resat. Multinomial tau-leaping method for stochastic kinetic simulations. *The Journal of Chemical Physics*, 126(8):084101, 2007.

[23] Asawari Samant, Babatunde Ogunnaike, and Dionisios Vlachos. A hybrid multiscale monte carlo algorithm (hymsmc) to cope with disparity in time scales and species populations in intracellular networks. *BMC Bioinformatics*, 8(1):175, 2007.

[24] Tianhai Tian and Kevin Burrage. Binomial leap methods for simulating stochastic chemical kinetics. *The Journal of Chemical Physics*, 121(21):10356–10364, 2004.

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8665
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8665