A partial-propensity formulation of the stochastic simulation algorithm for chemical reaction networks with delays

Rajesh Ramaswamy[a] and Ivo F. Sbalzarini[b]
Institute of Theoretical Computer Science and Swiss Institute of Bioinformatics, ETH Zurich, CH-8092 Zürich, Switzerland

(Received 24 September 2010; accepted 9 November 2010; published online 7 January 2011)

Several real-world systems, such as gene expression networks in biological cells, contain coupled chemical reactions with a time delay between reaction initiation and completion. The non-Markovian kinetics of such reaction networks can be exactly simulated using the delay stochastic simulation algorithm (dSSA). The computational cost of dSSA scales with the total number of reactions in the network. We reduce this cost to scale at most with the smaller number of species by using the concept of partial reaction propensities. The resulting delay partial-propensity direct method (dPDM) is an exact dSSA formulation for well-stirred systems of coupled chemical reactions with delays. We detail dPDM and present a theoretical analysis of its computational cost. Furthermore, we demonstrate the implications of the theoretical cost analysis in two prototypical benchmark applications. The dPDM formulation is shown to be particularly efficient for strongly coupled reaction networks, where the number of reactions is much larger than the number of species. © *2011 American Institute of Physics*. [doi:10.1063/1.3521496]

I. INTRODUCTION

Stochastic kinetics of a well-stirred system of coupled chemical reactions is described by the chemical master equation (CME).[1] Time trajectories of the population (molecule copy numbers) of the chemical species can be sampled from the CME using one of the many exact formulations of the stochastic simulation algorithm (SSA).[1–8] All of these exact SSA formulations assume instantaneous execution of reactions; the population of species is instantaneously updated at the time of reaction firing. In many real-world systems, such as gene expression networks in biological cells, however, reactions do not finish instantaneously, but there is a time delay between reaction initiation and completion. The average transcription and translation speeds in eukaryotic cells, for example, are 20 nucleotides per second and 2 codons per second, respectively.[9,10] Transcription and translation processes hence incur some delay after initiation. Delay in stochastic chemical reactions has also been proposed as a mechanism to tune or induce circadian-rhythmic oscillation in drosophila.[11,12] In order to simulate chemical reaction networks with delays, the delay stochastic simulation algorithm (dSSA) is available,[13] extending SSAs to properly account for the effects of nonzero reaction durations. The main complication in doing so stems from the fact that the reaction propensities (i.e., the probability rate of a reaction firing) may change in the time *between* two reaction initiations (firings) as a result of pending reactions finishing meanwhile.

Chemical reaction networks can be represented by their dependency graph. In this graph, each node (vertex) represents a reaction, and an arrow (directed edge) is drawn from node p to node q if reaction p affects the propensity of reaction q.[3] The out-degree of node p is defined as the number of arrows leaving that node. Using this representation, we distinguish two coupling classes of chemical reaction networks: *weakly coupled* and *strongly coupled*.[7] In weakly coupled reaction networks, the maximum out-degree in the dependency graph (i.e., the degree of coupling of the network) is constant or bounded by a constant with increasing network size. Strongly coupled reaction networks have a degree of coupling that increases unboundedly with network size. The scaling of the computational cost (here formalized using the Bachmann–Landau "big-O" notation) of SSAs is determined by the coupling class of the network. For weakly coupled reaction networks, the computational cost (CPU time) of the delay direct method (dDM),[13] an exact dSSA formulation, is $O(p + M + \log\Delta)$, where p is the search depth to sample the time to the next reaction, M is the number of chemical reactions in the network, and Δ is the number of pending reactions at a given time. For strongly coupled reaction networks, the computational cost of dDM is $O(pM + \log\Delta)$.

We have recently introduced a class of novel exact SSA formulations that are based on the concept of partial propensities.[7,8] For reaction networks without delays, these formulations have a computational cost of $O(N)$ (N being the number of chemical species in the network) for strongly coupled networks and $O(1)$ for weakly coupled ones.

Here, we present a partial-propensity exact SSA for chemical reaction networks with delays: the delay partial-propensity direct method (dPDM). It is based on the concept of factored-out, partial reaction propensities[7,8,14] and is an exact formulation of dSSA. We show that the computational cost of dPDM is $O(pN + \log\Delta)$ for strongly coupled networks and $O(p + N + \log\Delta)$ for weakly coupled ones. As a result of using partial propensities, the number of reactions M in the computational cost of dDM is replaced by the smaller

[a]Electronic mail: rajeshr@ethz.ch.
[b]Electronic mail: ivos@ethz.ch.

TABLE I. Outline of the algorithm for the delay direct method (dDM) with global times. We only give the main steps and refer to the original publication for the detailed substeps (Ref. 13).

1. Initialization: set $t \leftarrow 0$, $\delta a_0 \leftarrow 0$, and the number of pending reactions $\Delta \leftarrow 0$; initialize the population vector $\mathbf{n}$, the propensities a_μ, and the total propensity $a_0 = \sum_\mu a_\mu$.
2. Sample the global time of firing of the next reaction, τ^g: First, perform linear search to find the search depth p such that $p \in [T_p^g, T_{p+1}^g)$ according to Eq. (3). Then compute τ^g according to Eq. (5). Update $\Delta \leftarrow \Delta - p$ and set $t \leftarrow \tau^g$.
3. Sample the index of the next reaction μ according to Eq. (6) using linear search.
4. If μ is a delay reaction, insert $t + d_\mu$ into the list that stores the global times of the pending reactions. Use bisection search to insert at the proper position such that the list is maintained in ascending order; increment $\Delta \leftarrow \Delta + 1$.
4. Update $\mathbf{n}$ depending on the delay type of reaction μ.
5. Update the affected a_μ's using a dependency graph and calculate the change in total propensity δa_0.
6. Update $a_0 \leftarrow a_0 + \delta a_0$.
7. Go to step 2.

number of species N. The dPDM formulation is thus especially efficient when p and $\log\Delta$, which are network-specific parameters that are independent of the simulation method, do not scale faster than $O(N)$. In addition, the linear dependence of the computational cost on N makes dPDM especially efficient for strongly coupled reaction networks, where M grows much faster than N with network size. As in all partial-propensity methods, these computational savings are realized by restricting ourselves to networks of elementary chemical reactions. Nonelementary reactions can be decomposed into sets of elementary ones[1,15] at the expense of linearly (in the order of the reaction) increased network size.

II. THE DELAY STOCHASTIC SIMULATION ALGORITHM (dSSA)

Consider a network of M chemical reactions among N species. Assume a subset of these M reactions incur a delay. If a reaction is a nondelay reaction (hereafter denoted as R_{D0}), it completes instantaneously and the populations of reactants and products are immediately updated. If a reaction μ is a delay reaction, its products are formed only after a delay d_μ from reaction initiation. We classify delay reactions depending on when the reactants are consumed into *nonconsuming* (denoted R_{D1}) and *consuming* (denoted R_{D2}) ones. In nonconsuming delay reactions, the population of reactants is only updated once the products are formed, thus after the delay d_μ. In consuming delay reactions, the population of the reactants is updated immediately upon reaction initiation, but the products only form after the delay d_μ. In the following, we measure time globally, i.e., relative to time $t = 0$. This is in contrast to the local (relative to the current time t) times used in the original dSSA publication.[13] We denote the global time of firing (initiation) of the next reaction as $\tau^g = t + \tau$ and the global time at which the products of a delay reaction μ are formed as $d_\mu^g = t + \tau + d_\mu$.

Assume that at some time t there are Δ pending (ongoing) delay reactions that will finish at later global times T_1^g, T_2^g, T_3^g, …, T_Δ^g. We assume that the list of pending reactions is ordered according to ascending global completion times, thus $T_i^g \le T_{i+1}^g$, $i = 1, \ldots, \Delta - 1$. Furthermore, we define $T_0^g = t$ and $T_{\Delta+1}^g = \infty$. As in classical SSA, the time to the next reaction τ (or the global time of firing of the next reaction, τ^g) and the index of the next reaction μ are sampled in order to propagate the system from reaction event to reaction event. In classical SSA, all reactions complete instantaneously, i.e., reaction initiation and completion happen at the same time. Therefore, the reaction propensities remain unchanged during the time interval $[t, t + \tau)$. This, however, is not the case in delay SSAs, where the reaction propensities change whenever a pending reaction completes. Accounting for these inter-firing changes of the propensities, the probability density functions for the global time of firing (initiation) of the next reaction $f_\tau(\tau^g)$ and of the index of the reaction $f_\mu(\mu)$ are given by:[13]

$$f_\tau(\tau^g) = a_0\left(T_i^g\right)\exp\left(-\sum_{j=0}^{i-1} a_0\left(T_j^g\right)\left(T_{j+1}^g - T_j^g\right) - a_0\left(T_i^g\right)\left(\tau^g - T_i^g\right)\right),$$
$$\tau^g \in \left[T_i^g, T_{i+1}^g\right), \quad i = 0, \ldots, \Delta, \tag{1}$$

and

$$f_\mu(\mu) = \frac{a_\mu\left(T_i^g\right)}{a_0\left(T_i^g\right)}, \quad \mu = 1, \ldots, M, \quad \tau^g \in \left[T_i^g, T_{i+1}^g\right). \tag{2}$$

Here, $a_\mu(t)$ is the reaction propensity of reaction μ at global time t and $a_0(t)$ is the total propensity of all reactions at global time t.

A. The delay direct method

In dDM as presented by Cai et al.[13] and summarized in Table I, the global time of firing of the next reaction τ^g is obtained from Eq. (5) using linear search in order to sample the interval p such that

$$p = \max\left[i \ : \ r_1 \ge F\left(T_i^g\right)\right] \tag{3}$$

with $\tau^g \in [T_p^g, T_{p+1}^g)$ and r_1 a uniform random number in [0,1). Here, $F(\cdot)$ is the cumulative distribution function of the probability density function $f_\tau(\tau^g)$ [Eq. (1)]. It is given by:

$$F(\tau^g) = 1 - \exp\left(-\sum_{j=0}^{i-1} a_0\left(T_j^g\right)\left(T_{j+1}^g - T_j^g\right) - a_0\left(T_i^g\right)\left(\tau^g - T_i^g\right)\right),$$
$$\tau^g \in \left[T_i^g, T_{i+1}^g\right), \quad i = 0, \ldots, \Delta. \tag{4}$$

Note that in order to find p, we have to keep track of the time evolution of a_0. This is done by successively updating the propensities a_μ and the total propensity a_0 every time a pending reaction completes. Therefore, p is the search depth needed to sample τ^{g}.

Once the interval p is determined, τ^{g} is calculated as

$$\tau^{\mathrm{g}} = T_p^{\mathrm{g}} + \frac{-\log(1-r_1) - \sum_{j=0}^{p-1} a_0\left(T_j^{\mathrm{g}}\right)\left(T_{j+1}^{\mathrm{g}} - T_j^{\mathrm{g}}\right)}{a_0\left(T_p^{\mathrm{g}}\right)}, \tag{5}$$

such that

$$\tau^{\mathrm{g}} \in \left[T_p^{\mathrm{g}}, T_{p+1}^{\mathrm{g}}\right).$$

The index μ of the next reaction is also obtained by linear search. Unlike in Gillespie's original direct method (DM), however, the probability density function of μ depends on the interval p. The next reaction is hence always sampled *after* p has been found. Using a uniform random number $r_2 \in [0, 1)$, μ is found such that

$$\mu = \min\left[\mu' \ : \ r_2 a_0\left(T_p^{\mathrm{g}}\right) \ < \ \sum_{\mu'=1}^{\mu} a_{\mu'}\left(T_p^{\mathrm{g}}\right)\right]. \tag{6}$$

The computational cost of dDM is $O(pM + \log\Delta)$ for strongly coupled reaction networks and $O(p + M + \log\Delta)$ for weakly coupled ones, as shown in Appendix A.

III. THE DELAY PARTIAL-PROPENSITY DIRECT METHOD (dPDM)

The delay partial-propensity direct method (dPDM) is based on the concept of factorized reaction propensities, called *partial propensities*. These partial propensities are then grouped according to the common factored-out species, and the next reaction is sampled using linear search.[7,8]

A. Prerequisites for dPDM

The basic concepts underlying dPDM are partial propensities and partial-propensity SSAs.[14] For conciseness, we briefly review these concepts below. For a more detailed description, the reader is referred to the corresponding original publications.[7,8]

1. Partial propensities

The partial propensity of a reaction with respect to one of its reactants is defined as the propensity per unit number of molecules of that reactant.[7] For example, the partial propensity $\pi_\mu^{(i)}$ of reaction μ with respect to (perhaps the only) reactant S_i is a_μ/n_i, where a_μ is the propensity of reaction μ and n_i is the population (molecule copy number) of S_i. The partial propensities of the three elementary reaction types are:

- Bimolecular reactions ($\mathrm{S}_i + \mathrm{S}_j \rightarrow$ Products): $a_\mu = n_i n_j c_\mu$ and $\pi_\mu^{(i)} = n_j c_\mu$, $\pi_\mu^{(j)} = n_i c_\mu$. If both reactants are of the same species, i.e., $\mathrm{S}_j = \mathrm{S}_i$, only one partial propensity exists, $\pi_\mu^{(i)} = \frac{1}{2}(n_i - 1)c_\mu$, because the reaction degeneracy is $\frac{1}{2}n_i(n_i - 1)$.
- Unimolecular reactions ($\mathrm{S}_i \rightarrow$ Products): $a_\mu = n_i c_\mu$ and $\pi_\mu^{(i)} = c_\mu$.
- Source reactions ($\emptyset \rightarrow$ Products): $a_\mu = c_\mu$ and $\pi_\mu^{(0)} = c_\mu$.

Here, the c_μ's are the specific probability rates. In the following, we consider only elementary reaction types. Any reaction with three or more reactants can be equivalently decomposed into a series of elementary subreactions.[1,15,16]

2. Partial-propensity SSAs

Partial-propensity SSAs reduce the computational cost of SSAs to at most $O(N)$ by sampling reaction *partners* instead of complete reactions. This is achieved by grouping the partial propensities of all reactions according to the index of the factored-out reactant,[7] resulting in at most $N + 1$ groups of size $O(N)$. Every reaction and its corresponding partial propensity are then identifiable by two indices: a *group index* and an *element index*. The group index identifies the partial-propensity group to which a reaction belongs (i.e., the first reaction partner) and the element index identifies the position of the reaction inside that group (i.e., the second reaction partner). Determining the index of the next reaction is done by first sampling its group index and then the element index.

After the selected reaction has fired and the populations of the involved species have changed, the partial propensities are updated using a dependency graph over species. This dependency graph points to all partial propensities that need to be updated due to a given change in population. Since any partial propensity is a function of the population of *at most* one species, the number of updates is at most $O(N)$. In weakly coupled reaction networks, the number of updates is $O(1)$, since the degree of coupling is bounded by a constant, by definition of a weakly coupled network.

B. Detailed description of the dPDM algorithm

Like in PDM, the partial propensities in dPDM are stored in a "partial-propensity structure" $\mathbf{\Pi} = \{\mathbf{\Pi}_i\}_{i=0}^N$ as a one-dimensional array of one-dimensional arrays.[7] Each array $\mathbf{\Pi}_i$ contains the partial propensities of the reactions belonging to group i, i.e., the partial propensities where n_i has been factored out. The partial-propensity structure only needs to be constructed once, at the beginning of a simulation. This is done automatically as described previously.[8] The reaction index μ corresponding to a certain entry in $\mathbf{\Pi}$ is stored in a look-up table $\mathbf{L} = \{\mathbf{L}_i\}_{i=0}^N$. Each reaction μ is then identified by its group index I and its element index J as $\mu = \mathrm{L}_{I,J}$. The "group-sum array" $\mathbf{\Lambda}$ stores the sums of the partial propensities in each group $\mathbf{\Pi}_i$, i.e., $\Lambda_i = \sum_j \Pi_{i,j}$. We also store the total propensity of each group in an array $\mathbf{\Sigma}$, computed as $\Sigma_i = n_i \Lambda_i, \quad i = 1, \ldots, N$, and $\Sigma_0 = \Lambda_0$.[7]

After each reaction event (reaction initiation or completion) the population $\mathbf{n}$, the partial propensities $\Pi_{i,j}$, the Λ_i's, and the Σ_i's need to be updated. Which values need to be updated depends on the type of event that happened (firing of a nondelay reaction, initiation of a nonconsuming delay

reaction, initiation of a consuming delay reaction, or completion of a delay reaction). We efficiently implement the updates using the following data structures:

- $\mathbf{U}^{(1)}$: an array of M arrays, where the i^{th} array contains the indices of all species involved in the i^{th} reaction.
- $\mathbf{U}^{(2)}$: an array of M arrays containing the corresponding stoichiometry (the change in population of each species upon reaction) of the species stored in $\mathbf{U}^{(1)}$.
- $\mathbf{U}^{(1)}_{(-)}$: an array of M arrays, where the i^{th} array contains the indices of all species that are reactants in the i^{th} reaction.
- $\mathbf{U}^{(2)}_{(-)}$: an array of M arrays containing the corresponding stoichiometry of the reactant species stored in $\mathbf{U}^{(1)}_{(-)}$.
- $\mathbf{U}^{(1)}_{(+)}$: an array of M arrays, where the i^{th} array contains the indices of all species that are products in the i^{th} reaction.
- $\mathbf{U}^{(2)}_{(+)}$: an array of M arrays containing the corresponding stoichiometry of the product species stored in $\mathbf{U}^{(1)}_{(+)}$.
- $\mathbf{U}^{(3)}$: an array of N arrays, where the i^{th} array contains the indices of all entries in $\mathbf{\Pi}$ that depend on n_i.

We also maintain a priority queue $\mathbf{T}$ that stores the global times ($T_i^{\text{g}},\ i = 0, \ldots, \Delta$) of the Δ pending reactions in ascending order. The corresponding indices and delay types (R_{D0}, R_{D1}, or R_{D2}) of the reactions are stored in the lists $\boldsymbol{\mu}^{(\text{D})}$ and $\mathbf{D}$, respectively.

In dPDM, the global time of firing (initiation) of the next reaction, τ^{g}, and the index of the next reaction are mutually dependent. First, the interval p is found according to Eq. (3) using linear search such that the global time of firing of the next reaction $\tau^{\text{g}} \in [T_p^{\text{g}}, T_{p+1}^{\text{g}})$. This tells us between which two reaction completion events the next firing or initiation event happens [see Fig. 1(a)]. The difference between dPDM and dDM in sampling p is the mechanism of updating the total propensity $a_0(T_i^{\text{g}})$ each time a pending reaction completes and is removed from the queue of pending reactions. In dPDM, we make use of the partial propensities $\mathbf{\Pi}$ and the associated data structures to update a_0. For instance, assume that $\tau^{\text{g}} \in [T_1^{\text{g}}, T_2^{\text{g}})$ and the reaction type associated with the global completion time T_1^{g} is R_{D2} (consuming delay reaction). In this case, we update $\mathbf{n}$ using $\mathbf{U}^{(1)}_{(+)}$ and $\mathbf{U}^{(2)}_{(+)}$. If the finishing reaction is of type R_{D1} (nonconsuming delay reaction), $\mathbf{n}$ is updated using $\mathbf{U}^{(1)}$ and $\mathbf{U}^{(2)}$. Subsequently, $\mathbf{\Pi}$ and the associated data structures are updated using $\mathbf{U}^{(3)}$, thereby obtaining δa_0 (the change in a_0) and hence the new a_0. All these updates are done at the completion time of each pending reaction until the interval containing the global time of firing (initiation) of the next reaction is reached and all p pending reactions that have completed are removed from the queue $\mathbf{T}$. Then, the global time of firing (initiation) of the next reaction, τ^{g}, within that interval is calculated according to Eq. (5).

For sampling the index μ of the next reaction, we use a single uniformly distributed random number $r_2 \in [0, 1)$ to (a) sample the group index I using linear search such that

$$I = \min\left[I' \ : \ r_1 a_0\left(T_p^{\text{g}}\right) < \sum_{i=0}^{I'} \Sigma_i\left(T_p^{\text{g}}\right) \right] \tag{7}$$

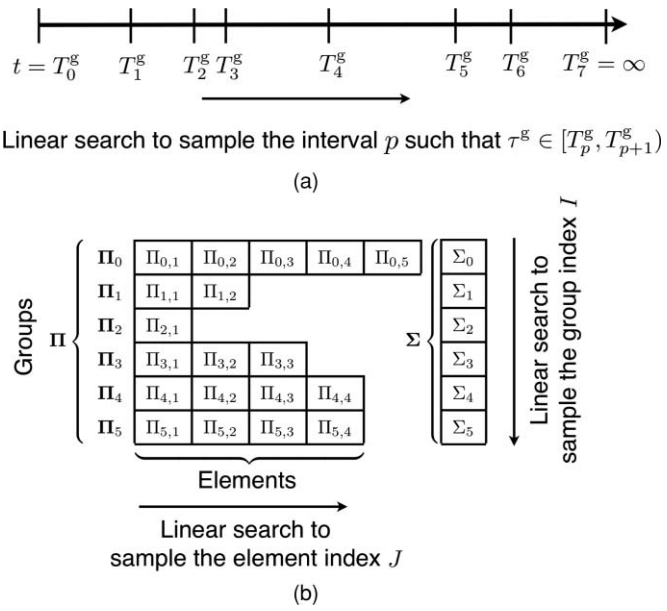

FIG. 1. Illustration of the main steps in dPDM. (a) Illustration of the linear search to find the interval p such that the global time of firing (initiation) of the next reaction $\tau^{\text{g}} \in [T_p^{\text{g}}, T_{p+1}^{\text{g}})$. In this figure, the number of pending reactions $\Delta = 6$. (b) Illustration of the partial-propensity structure $\mathbf{\Pi}$ and the grouping based on the index of the common factored-out reactant. The group index I of the next reaction is sampled using linear search over the total propensities of the groups, Σ_i. The element index J within the selected group is found using linear search over the partial propensities stored in group I.

and (b) sample the element index J in $\mathbf{\Pi}_I$ using linear search such that

$$J = \min\left[J' \ : \ r_1 a_0\left(T_p^{\text{g}}\right) < \sum_{j=1}^{J'} n_I \Pi_{I,j}\left(T_p^{\text{g}}\right) + \left(\sum_{i=0}^{I} \Sigma_i\left(T_p^{\text{g}}\right) \right) - \Sigma_I\left(T_p^{\text{g}}\right) \right] \tag{8}$$

if $\tau^{\text{g}} \in [T_p^{\text{g}}, T_{p+1}^{\text{g}})$ [see Fig. 1(b)]. The indices I and J are then translated back to the reaction index μ using the look-up table $\mathbf{L}$, thus $\mu = \text{L}_{I,J}$. It has been shown earlier that this sampling strategy over partial propensities is algebraically equivalent to the linear search over propensities used in Gillespie's DM.[7] Since dDM is a delay variant of DM, with the same sampling strategy as DM, this also makes the present sampling strategy equivalent to the linear search used in dDM.

Once the index of the next reaction is sampled, we ascertain the type of the reaction and initiate it. If μ is a nondelay (type R_{D0}) reaction, then the population $\mathbf{n}$ is immediately updated using $\mathbf{U}^{(1)}$ and $\mathbf{U}^{(2)}$. Subsequently, $\mathbf{\Pi}$ is updated using $\mathbf{U}^{(3)}$. If μ is a nonconsuming delay reaction (type R_{D1}), $\mathbf{n}$ and $\mathbf{\Pi}$ are not updated at the time of reaction initiation. Instead, the attributes of this delay reaction (its global time of completion, index, and type) are inserted into $\mathbf{T}$, $\boldsymbol{\mu}^{(\text{D})}$, and $\mathbf{D}$, respectively. We ensure that the global completion times in $\mathbf{T}$ are maintained in ascending order by inserting at the appropriate location, which is found using bisection search. If μ is a consuming delay reaction (type R_{D2}), $\mathbf{n}$ is immediately updated using $\mathbf{U}^{(1)}_{(-)}$ and $\mathbf{U}^{(2)}_{(-)}$. Subsequently, $\mathbf{\Pi}$ is updated

TABLE II. Detailed algorithm for the delay partial-propensity direct method (dPDM), explicitly describing all substeps.

1. Initialization: set $t \leftarrow 0$, $\delta a_0 \leftarrow 0$, and the number of pending reactions $\Delta \leftarrow 0$; initialize the population vector $\mathbf{n}$, the partial propensities $\mathbf{\Pi}$ (see Appendix B of Ref. 8), the group sum array $\mathbf{\Lambda}$, $\mathbf{\Sigma}$, and the total propensity $a_0 \leftarrow \sum_{i=0}^{N} \Sigma_i$; initialize $\mathbf{T}$, $\mathbf{D}$, and $\boldsymbol{\mu}^{(\mathrm{D})}$ (these are empty at this step); initialize the update structures $\mathbf{U}^{(1)}$, $\mathbf{U}^{(2)}$, $\mathbf{U}^{(1)}_{(-)}$, $\mathbf{U}^{(2)}_{(-)}$, $\mathbf{U}^{(1)}_{(+)}$, and $\mathbf{U}^{(2)}_{(+)}$.
2. Sample the global time of firing of the next reaction, τ^{g}:
   2.1. Generate a uniform random number r_1 in [0,1).
   2.2. If $\Delta == 0$ (i.e., $\mathbf{T}$ is empty) then $t \leftarrow t - \log(r_1/a_0)$
   2.3. else
      2.3.1. $\lambda_1 \leftarrow t$; $\lambda_2 \leftarrow \mathrm{T}_1$; $a_t \leftarrow a_0(\lambda_2 - \lambda_1)$; $F \leftarrow 1 - \exp(-a_t)$
      2.3.2. While $F < r_1$
         2.3.2.1. Get current delay reaction and its type from $\mu_1^{(\mathrm{D})}$ and D_1, respectively. Update $\mathbf{n}$, $\mathbf{\Pi}$, $\mathbf{\Lambda}$, and $\mathbf{\Sigma}$ accordingly using the proper subset of update structures $\mathbf{U}^{(1)}$, $\mathbf{U}^{(2)}$, $\mathbf{U}^{(1)}_{(-)}$, $\mathbf{U}^{(2)}_{(-)}$, $\mathbf{U}^{(1)}_{(+)}$, and $\mathbf{U}^{(2)}_{(+)}$, $\mathbf{U}^{(3)}$ (see Sec. III B). Calculate δa_0 and set $a_0 \leftarrow a_0 + \delta a_0$.
         2.3.2.2. $\lambda_1 \leftarrow \mathrm{T}_1$. Remove T_1, $\mu_1^{(\mathrm{D})}$, and D_1 from the corresponding lists and decrement $\Delta \leftarrow \Delta - 1$.
         2.3.2.3. If $\Delta == 0$ then exit from the while loop 2.3.2.
         2.3.2.4. else $\lambda_2 \leftarrow \mathrm{T}_1$
         2.3.2.5. $a_t \leftarrow a_t + a_0(\lambda_2 - \lambda_1)$; $F \leftarrow 1 - \exp(-a_t)$
      2.3.3. if $\Delta == 0$ then $\tau^{\mathrm{g}} \leftarrow \lambda_1 + \frac{-\log(1-r_1)-a_t-a_0(\lambda_2-\lambda_1)}{a_0}$; set $t \leftarrow \tau^{\mathrm{g}}$
      2.3.4. else $\tau^{\mathrm{g}} \leftarrow \lambda_1 + \frac{-\log(1-r_1)-a_t}{a_0}$; set $t \leftarrow \tau^{\mathrm{g}}$.
3. Sample the index of the next reaction, μ: Using linear search, sample the group index I and element index J of the next reaction according to Eqs. (7) and (8), respectively. Look up the index of the next reaction as $\mu = \mathrm{L}_{I,J}$.
4. If μ is a delay reaction, increment $\Delta \leftarrow \Delta + 1$. Insert $t + d_\mu$ into $\mathbf{T}$, μ into $\boldsymbol{\mu}^{(\mathrm{D})}$, and the type of the delay reaction into $\mathbf{D}$. Use bisection search to ensure that the entries in $\mathbf{T}$ are in ascending order and maintain the correspondence between $\mathbf{T}$, $\boldsymbol{\mu}^{(\mathrm{D})}$, and $\mathbf{D}$.
4. Update $\mathbf{n}$ depending on reaction μ's type:
   4.1. If μ is $\mathrm{R}_{\mathrm{D}0}$, then update $\mathbf{n}$ using $\mathbf{U}^{(1)}$ and $\mathbf{U}^{(2)}$
   4.2. else if μ is $\mathrm{R}_{\mathrm{D}1}$, then do not update $\mathbf{n}$
   4.2. else if μ is $\mathrm{R}_{\mathrm{D}2}$, then update $\mathbf{n}$ using $\mathbf{U}^{(1)}_{(-)}$ and $\mathbf{U}^{(2)}_{(-)}$
5. Update $\mathbf{\Pi}$ using the update structure $\mathbf{U}^{(3)}$ and calculate the change in total propensity δa_0.
6. Update $a_0 \leftarrow a_0 + \delta a_0$.
7. Go to step 2.

using $\mathbf{U}^{(3)}$. In addition, the attributes of this reaction are inserted into $\mathbf{T}$, $\boldsymbol{\mu}^{(\mathrm{D})}$, and $\mathbf{D}$ at the appropriate location, again found by bisection search.

In summary, dPDM is an exact formulation of dSSA, generalizing PDM to handle reactions with delays according to the probability density functions of dSSA [Eqs. (1) and (2)]. The detailed algorithm of dPDM is given in Table II. The computational cost of dPDM is $O(pN + \log\Delta)$ for strongly coupled reaction networks and $O(p + N + \log\Delta)$ for weakly coupled ones, as shown in Appendix B.

IV. BENCHMARKS

We benchmark the computational performance of dPDM on both a weakly coupled and a strongly coupled prototypical reaction network. We choose the cyclic chain model[4] and the colloidal aggregation model[17] as representative networks for which we compare the performance of dPDM with that of dDM.[13] The cyclic chain model is the most weakly coupled network possible, the colloidal aggregation model is strongly coupled. The performance for other networks with intermediate degrees of coupling will lie between these two extremes. In the benchmarks, we only consider consuming delay reactions since they require updates at both the time of reaction initiation as well as completion.

All tested SSA formulations are implemented in C++ using the random number generator of the GSL library and compiled using the Intel C++ compiler version 11.1 with the O3 optimization flag. All timings are measured on a Linux 2.6 workstation with a 2.8 GHz quad-core Intel Xeon E5462 processor and 8 GB of memory. For all test cases, we simulate the reaction network until 10^7 reactions have been initiated, and we report the average CPU time Θ per reaction initiation (i.e., the average time to execute steps 2–7 in Table II for dPDM and Table I for dDM).

A. A strongly coupled reaction network: Colloidal aggregation model

The colloidal aggregation model is given by

$$\begin{aligned} &\mathrm{S}_n + \mathrm{S}_m \xrightarrow{c_{n,m}} \mathrm{S}_{n+m}, \qquad n+m = 2,\ldots,N, \\ &\mathrm{S}_{n+m} \xrightarrow{\bar{c}_{n,m}} \mathrm{S}_n + \mathrm{S}_m, \qquad n+m = 2,\ldots,N. \end{aligned} \tag{9}$$

For N chemical species, the number of reactions is $M = \lfloor N^2/2 \rfloor$. The degree of coupling (maximum out-degree of the dependency graph) of this reaction network is $3N - 7$ and hence scales with system size.

At time $t = 0$, we set all $n_i = 1$ and all specific probability rates $c_\mu = 1$. We set all reactions with an even index to be consuming delay reactions ($\mathrm{R}_{\mathrm{D}2}$), each with a delay of $d_\mu = 0.1$. The rest of the reactions are nondelay reactions ($\mathrm{R}_{\mathrm{D}0}$). The benchmarks confirm that the search depth p to sample the global time of firing (initiation) of the next reaction is $O(1)$, and that the logarithm of the number of pend-

ing delay reactions, $\log\Delta$, is $O(\log N)$. Hence, the computational cost of this simulation is $O(N)$ for dPDM and $O(N^2)$ for dDM. This is shown in Fig. 2(a), where $\Theta(N)$ for dPDM and dDM are compared.

Figure 2(b) shows the results for larger networks on a linear scale. Here, we consider networks of up to $N = 2000$ species and $M = 2$ million reactions in order to reveal memory contention effects. Around $N = 1000$ species, the slope of the cost curve increases, while remaining $O(N)$. This is probably due to the partial-propensity structure not fitting into cache any more. The machine used for the benchmark has a 4 MB L2 cache. At $N = 1000$ the partial-propensity structure for this network contains 500 000 double-precision floating-point numbers of 8 bytes each, amounting to exactly 4 MB.

In summary, for a strongly coupled reaction network, the computational cost of dPDM is $O(pN + \log\Delta)$ as predicted by the theoretical analysis.

B. A weakly coupled reaction network: Cyclic chain model

The cyclic chain model is given by the reaction network

$$\begin{aligned} &S_i \xrightarrow{c_i} S_{i+1}, \qquad i = 1, \ldots, N-1, \\ &S_N \xrightarrow{c_N} S_1. \end{aligned} \tag{10}$$

For N chemical species, this network has the smallest possible number of $M = N$ reactions. The degree of coupling of this reaction network is 2, independent of system size.

At time $t = 0$, we set all $n_i = 1$ and all specific probability rates $c_\mu = 1$. We set all reactions with an even index to be consuming delay reactions (R_{D2}), each with a delay $d_\mu = 0.1$. The rest of the reactions are nondelay reactions (R_{D0}). The benchmarks confirm that the search depth p to sample τ^g is $O(1)$ and that $\log\Delta$ is $O(\log N)$. Hence, the computational cost of this simulation is $O(N)$ for dPDM as well as for dDM. The corresponding $\Theta(N)$ for dPDM and dDM are shown in Fig. 2(c).

In summary, for a weakly coupled reaction network, the computational cost of dPDM is $O(p + N + \log\Delta)$ as predicted by the theoretical analysis.

V. CONCLUSIONS AND DISCUSSION

We have introduced the delay partial-propensity direct method (dPDM), a partial-propensity formulation of the delay stochastic simulation algorithm (dSSA)[13] to simulate chemical reaction networks with delays. dPDM uses partial propensities and reaction groups in order to improve computational efficiency. For reaction networks with no delays, dPDM becomes identical to the partial-propensity direct method (PDM).[7]

The presented dPDM is an exact dSSA formulation with a computational cost of $O(pN + \log\Delta)$ for strongly coupled reaction networks and $O(p + N + \log\Delta)$ for weakly coupled networks. Here, N is the number of chemical species, p is the search depth to sample the time to the next reaction, and Δ is the number of pending delay reactions at a given time. We have presented a theoretical cost analysis of

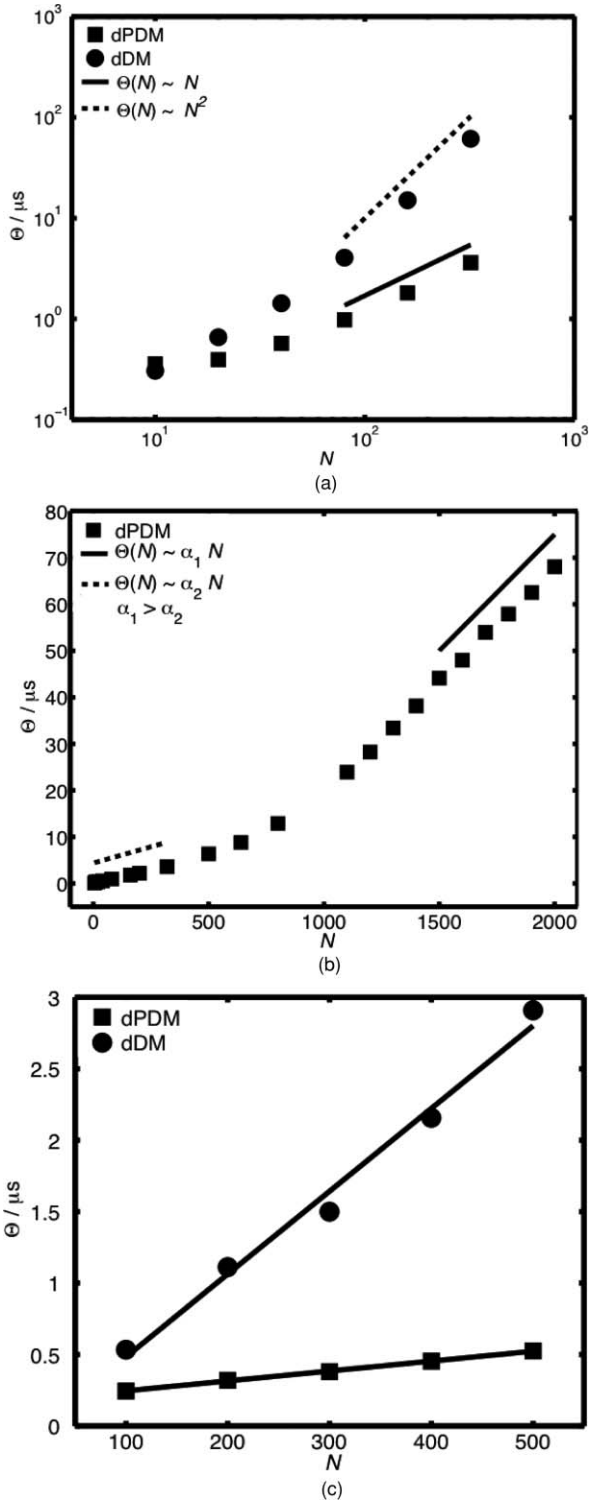

FIG. 2. Computational cost of dPDM (squares) and dDM (circles). The average (over 100 independent runs) CPU time Θ per reaction initiation (i.e., the average time to execute steps 2–7 in Table II for dPDM, and Table I for dDM) is shown as a function of the number of species N in the reaction network. (a) Logarithmic plot of $\Theta(N)$ for the strongly coupled colloidal aggregation model, considering systems of size up to $N = 320$. Θ is $O(N)$ for dPDM and $O(M) = O(N^2)$ for dDM. (b) Linear plot of $\Theta(N)$ for the strongly coupled colloidal aggregation model, considering systems of size up to $N = 2000$ (2 million reactions). While the scaling of the computational cost remains linear for all system sizes tested, the slope increases around $N = 1000$. This is the system size beyond which the partial-propensity structure does not fit into the computer's cache memory any more. (c) Linear plot of $\Theta(N)$ for the weakly coupled cyclic chain model. The solid lines are linear least square fits. Θ is $O(N)$ for both dPDM and dDM, but with a smaller slope for dPDM.

dPDM and confirmed its results in two benchmark cases prototypical of strongly and weakly coupled reaction networks. Since p and Δ are properties of the chemical reaction network alone, and the only other variable that the computational cost depends on is linear in N, dPDM is especially efficient for strongly coupled reaction networks with delays. This is because in these networks the number of chemical species N grows much slower with network size than the number of chemical reactions M.

However, dPDM inherits the limitations of partial-propensity methods.[7,8] It is limited to chemical reaction networks composed of elementary reactions involving at most two reactants. Nonelementary reactions can be treated by decomposing them into elementary reactions.[1,15] This, however, increases the network size proportionally to the order (the number of reactant species) of the nonelementary reaction. For small networks, dPDM is outperformed by other methods due to the overhead of the additional data structures. Other dSSA formulations, such as the delay direct method (dDM),[13] might be more efficient there. In addition, the computational cost of dPDM could be further reduced to $O(p + \log\Delta)$ for weakly coupled reaction networks by using composition-rejection sampling[6,8,18] instead of linear search[2,7] to sample the index of the next reaction. On multi-scale (stiff) reaction networks, prototypical of biochemical networks where the propensities span several orders of magnitude, dynamic sorting[5,7] could further reduce the computational cost, even though its scaling with N would remain the same. Such a formulation would be analogous to SPDM, the sorting variant of PDM.

A C++ implementation of dPDM at the time of writing is available as supplementary material to this article.[19] A constantly updated version is available free of charge on the web page of the authors.

ACKNOWLEDGMENTS

R.R. was financed by a grant from the Swiss SystemsX.ch initiative, grant WingX, evaluated by the Swiss National Science Foundation. This project was also supported with a grant from the Swiss SystemsX.ch initiative, grant LipidX-2008/011 to I.F.S.

APPENDIX A: COMPUTATIONAL COST OF dDM

The algorithm of the delay direct method (dDM)[13] is summarized in Table I. It is built around a list of global completion times of the pending delay reactions, maintained in ascending order. The computational cost of this algorithm is determined by the following steps:

Update step: For a strongly coupled reaction network, firing of one reaction can potentially affect all propensities. Hence, the computational cost of updating the reaction propensities is $O(M)$, where M is the number of reactions in the network. For a weakly coupled reaction network the update step is $O(1)$ since the number of propensities affected by a reaction is (by definition of a weakly coupled network) bounded by a constant.

Sampling the global time of the next reaction: The computational cost of sampling the global time of firing (initiation) of the next reaction, τ^{g}, is $O(pM)$ for a strongly coupled reaction network. Here, p is the search depth to locate τ^{g} according to Eq. (3). This is because the number of times the propensities need to be updated due to pending reactions finishing is p when $\tau^{\mathrm{g}} \in [T_p^{\mathrm{g}}, T_{p+1}^{\mathrm{g}})$. In each of these p updates, $O(M)$ propensities need to be updated. Similarly, for a weakly coupled reaction network, the computational cost of sampling τ^{g} is $O(p)$.

Sampling the index of the next reaction: The index of the next reaction is found by linear search across the M propensities. The computational cost of this operation is $O(M)$. If the sampled reaction is a delay reaction, it is added to the list of pending reactions, along with its global completion time. Using bisection search to maintain the list of global completion times in ascending order upon inserting a new reaction is $O(\log\Delta)$, where Δ is the number of pending reactions currently in the list.

In summary, the computational cost of dDM is $O(pM + M + \log\Delta)$ for strongly coupled reaction networks. This is equivalent to $O(pM + \log\Delta)$ for $p > 0$. For weakly coupled reaction networks, the computational cost is $O(p + M + \log\Delta)$. Note that when there are no delay reactions, the computational cost of dDM is $O(M)$, as for Gillespie's original direct method.

APPENDIX B: COMPUTATIONAL COST OF dPDM

The algorithm of the delay partial-propensity direct method (dPDM) as presented in this paper is detailed in Table II. Its computational cost is determined by the following steps:

Update step: The computational cost of the update step is $O(N)$ and $O(1)$ for strongly and weakly coupled reaction networks, respectively, where N is the number of species in the network. Assuming that the number of species involved in any chemical reaction is $O(1)$ (i.e., does not increase beyond a constant bound as the number of species in the network increases), the number of entries in $\boldsymbol{\Pi}$ that need to be updated after any reaction has fired scales at most linearly with N.[7,8] This is the case for strongly coupled reaction networks. For weakly coupled ones, the number of partial propensities that need to be updated after any reaction is $O(1)$, by definition of a weakly coupled network.[7,8]

Sampling the global time of the next reaction: The computational cost of sampling the global time of firing (initiation) of the next reaction, τ^{g}, is $O(pN)$ and $O(p)$ for strongly and weakly coupled reaction networks, respectively. This is because the number of times the partial propensities need to be updated due to a finishing pending reaction is p, where p is search depth to locate τ^{g}. During each of these p updates, the number of partial propensities that need to be recomputed is $O(N)$ and $O(1)$ for strongly and weakly coupled reaction networks, respectively.

Sampling the index of the next reaction: Sampling the group index is performed using linear search across the $N + 1$ groups. Subsequently, the element index is sampled using linear search across the $O(N)$ partial propensities within the

selected group. The computational cost of sampling the index of the next reaction hence is $O(N)$. If the sampled reaction is a delay reaction, it is added to the list of pending reactions, along with its attributes. Using bisection search to maintain the list of global completion times in ascending order upon inserting a new reaction is $O(\log\Delta)$, where Δ is the number of pending reactions currently in the list.

In summary, the computational cost of dPDM is $O(pN + N + \log\Delta)$ for strongly coupled reaction networks. This is equivalent to $O(pN + \log\Delta)$ for $p > 0$. For weakly coupled reaction networks, the computational cost is $O(p + N + \log\Delta)$. Note that when there are no delay reactions, the computational cost of dPDM is $O(N)$, as for PDM.[7]

[1]D. T. Gillespie, Physica A **188**, 404 (1992).
[2]D. T. Gillespie, J. Comput. Phys. **22**, 403 (1976).
[3]M. A. Gibson and J. Bruck, J. Phys. Chem. A **104**, 1876 (2000).
[4]Y. Cao, H. Li, and L. Petzold, J. Chem. Phys. **121**, 4059 (2004).
[5]J. M. McCollum, G. D. Peterson, C. D. Cox, M. L. Simpson, and N. F. Samatova, Comput. Biol. Chem. **30**, 39 (2006).
[6]A. Slepoy, A. P. Thompson, and S. J. Plimpton, J. Chem. Phys. **128**, 205101 (2008).
[7]R. Ramaswamy, N. González-Segredo, and I. F. Sbalzarini, J. Chem. Phys. **130**, 244104 (2009).
[8]R. Ramaswamy and I. F. Sbalzarini, J. Chem. Phys. **132**, 044102 (2010).
[9]B. Alberts, D. Bray, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter, *Essential Cell Biology* (Garland Publicati[illegible]nc., New York, 1997).
[10]X. Cai and Z. Xu, J. Chem. Phys. **126**, 074102 (2007).
[11]Q. Li and X. Lang, Biophys. J. **94**, 1983 (2008).
[12]Z. Xu and X. Cai, EURASIP J. Bioinform. Syst. Biol. **2009**, 386853 (2009).
[13]X. Cai, J. Chem. Phys. **126**, 074102 (2007).
[14]R. Ramaswamy and I. F. Sbalzarini, Fast exact stochastic simulation algorithms using partial propensities, in Proc. *ICNAAM, Numerical Analysis and Applied Mathematics, International Conference* (AIP, New York, 2010), pp. 1338–1341.
[15]K. R. Schneider and T. Wilhelm, J. Math. Biol. **40**, 443 (2000).
[16]T. Wilhelm, J. Math. Chem. **27**, 71 (2000).
[17]P. Meakin, Ann. Rev. Phys. Chem. **39**, 237 (1988).
[18]L. Devroye, *Non-Uniform Random Variate Generation* (Springer-Verlag, New York, 1986).
[19]See supplementary material at http://dx.doi.org/10.1063/1.3521496 for a C++ implementation of dPDM at the time of writing. A constantly updated version is available free of charge on the web page of the authors.