Proceedings of the 2018 Winter Simulation Conference
M. Rabe, A. A. Juan, N. Mustafee, A. Skoogh, S. Jain, and B. Johansson, eds.

AN IMPROVED SIMULATION OF HYBRID BIOLOGICAL MODELS WITH MANY STOCHASTIC EVENTS AND QUASI-DISJOINT SUBNETS

Mostafa Herajy

Department of Mathematics and Computer Science
Port Said University
Port Said, 42521, EGYPT

Monika Heiner

Computer Science Institute
Brandenburg University of Technology
Cottbus, 03013, GERMANY

ABSTRACT

Hybrid simulation, combining exact and approximate algorithms, provides an alternative to a completely stochastic simulation. However, one challenge for the efficient implementation of hybrid simulations is the additional overhead due to frequent switches between the two regimes. The amount of additional overhead considerably increases with the number of discrete events in the stochastic regime. However, reactions that take place rather frequently cannot completely be avoided due to the accuracy requirements. In this paper, we present an improved hybrid simulation method which takes advantage of the Hybrid Rejection-based Stochastic Simulation Algorithm (HRSSA), a variant of the hybrid simulation approach. To reduce the overhead on account of the switches from the stochastic to the deterministic regime, we analyse and record the dependencies of reactions as well as species between the stochastic and deterministic subnetworks. Comparing our technique with existing ones shows a clear improvement in terms of runtime, while preserving accuracy.

1 INTRODUCTION

The construction and execution of biological models has been recently attracting more and more attention. The Stochastic Simulation Algorithm (SSA) Gillespie (1977) is one of the well-known simulation approaches that is used to simulate a set of reactions in a well-mixed biochemical reaction system. Although there are many improvements of the key SSA idea (see, e.g., Gibson and Bruck (2000)), its drawback of being prohibitively slow when simulating reactions occurring too frequently may prevent its application for certain types of biological models (Herajy et al. 2018; Iwamoto et al. 2014). While the SSA procedure is rather simple, it includes two steps which consume the majority of the processing cycles (Thanh et al. 2014): the search for the next reaction to fire and the update of the state-dependent propensities each time a reaction takes place. For the former step, many methods have been proposed to reduce the search runtime, while the best known idea for the latter is to construct a dependency graph (Gibson and Bruck 2000), which is then deployed to update only affected propensities. As another (exact) approach to reduce the frequent updates of reaction propensities, a new algorithm has recently been proposed in Thanh et al. (2014) called Rejection-based Stochastic Simulation Algorithm (RSSA). The idea of the RSSA is to define a lower and upper bound of the system state called fluctuation interval. Reaction propensities are updated only when some or all of the system state entries are outside of the corresponding fluctuation interval. However, despite all improvements, the pure stochastic simulation is still known as a slow simulation approach which is not able to simulate larger biological models or models with frequent firing of reactions.

Hybrid simulation (Haseltine and Rawlings 2002; Herajy and Heiner 2012; Salis and Kaznessis 2005) may serve as an alternative to study the dynamics of models which cannot be executed in reasonable time via a pure SSA. Hybrid simulation involves partitioning the reaction network into two parts: slow and fast, and then executing the slow subnetwork applying the exact SSA, while approximating the fast subnetwork applying a deterministic or an approximate stochastic solver. The partitioning process can be done either

statically (just once before the simulation) or dynamically (repeatedly during the simulation). In this paper, we are interested in a hybrid simulation method where the deterministic part is simulated using an Ordinary Differential Equations (ODE) solver. A time transformation equation (also called jump equation (Salis and Kaznessis 2005) permits to accurately locate the exact time where a stochastic event is to occur (Gillespie 1991; Haseltine and Rawlings 2002). Nonetheless, one drawback of the hybrid simulation is the additional overhead due to frequent switches between the stochastic simulation and the ODE solver (Herajy and Heiner 2016).

The overhead considerably increases with the number of stochastic events occurring in the stochastic regime. However, abundant stochastic events in hybrid models cannot simply be avoided due to one of the following reasons. First, as soon as the number of slow reactions increases, the number of discrete events increases correspondingly. Second, a hybrid model may include fast reactions that cannot be assigned to the deterministic regime as they would violate the thermodynamic assumption which renders the deterministic simulation reasonable to execute such models (e.g., reactions with high rate constant, but including species with a few number of molecules).

Therefore, in Herajy and Heiner (2016) we have introduced a general approach that accelerates the performance of hybrid simulation by reducing the number of times the ODE solver is reinitialised. This involves the automatic identification of the stochastic reactions acting as interface reactions, i.e., their firings affect the system state of the deterministic regime. According to this approach, the ODE solver is reinitialised only when one of these interface reactions occurs. However, one problem of this method is the approximation of the jump equation used to locate the exact time when a stochastic reaction has to occur.

Avoiding the reinitialisation of the ODE solver can result in substantial improvements of the overall performance of the hybrid simulator for at least two reasons. First, to reduce the effects of switching from the stochastic to deterministic regime, hybrid simulation is usually used in combination with ODE solvers that employ single step approaches (e.g., Runge-Kutta family solvers). However, the step size should be recomputed after each firing of a stochastic reaction. This can result in a performance bottleneck for larger models (for an example see (Nagaiah et al. 2012)). The ODE solver cannot continue the numerical integration using the current step size because the solution state might have been considerably changed. Second, for a large class of models, multi-step ODE solvers that employ history information to advance the step size can result in a much better performance. However, combining such solvers with the hybrid simulator will considerably degrade the overall simulator performance, because the history information is cleared each time a stochastic event occurs.

Recently, a new hybrid simulation algorithm, called Hybrid Rejection-based Stochastic Simulation Algorithm (HRSSA), has been proposed in Marchetti et al. (2016), which does not rely on a jump equation to locate the occurrence time of the next stochastic event. Instead, HRSSA performs the jump when the fluctuation interval requires an update. Nevertheless, the HRSSA still requires frequent switches between the stochastic and deterministic regime as well as frequent checks of the system state for being inside of the fluctuation interval. The latter check is rather expensive and performed repeatedly.

The contribution of our paper is to combine the ideas from Herajy and Heiner (2016) and Marchetti et al. (2016). Additionally, we also consider the set of indirect interface reactions while simulating hybrid models. To improve the performance of the HRSSA, we collect a set of interface species similar to the set of interface reactions permitting the dynamic simulation of reaction networks by applying the accelerated hybrid simulation approach.

This paper is organised as follows: First, we recall some background information of stochastic simulation, (hybrid) rejection-based stochastic simulation, and accelerated hybrid simulation. Afterwards, we present our approach of improving the performance of the HRSSA by depicting and extracting the reaction dependency information. To evaluate the proposed approach, we present three case studies. Finally, we summarise the paper outcome and outline future work.

2 BACKGROUND

In this paper we consider simulating a system of M chemical reactions involving N species. The species (denoted by the set S) are assumed to interact in a well-mixed system of molecules (Gillespie 1977). Eq. 1 shows the form of the j-th reaction in the reaction set R.

$$\delta_{j1}s_1 + \delta_{j2}s_2 + \cdots + \delta_{jN}s_N \xrightarrow{k_j} \delta'_{j1}s_1 + \delta'_{j2}s_2 + \cdots + \delta'_{jN}s_N, \tag{1}$$

where $s_i \in S$ is the amount of the i-th species, δ_{ji} is the stoichiometric coefficient of s_i when participating in reaction $r_j \in R$ as a substrate, δ'_{ji} is the stoichiometric coefficient of species s_i when participating in reaction r_j as a product, k_j is the reaction rate constant of reaction r_j, and j takes the values from 1 to M.

The system state is characterised by a vector, usually denoted by $\mathbf{x}$, which gives the current number of molecules of each species. Moreover, each reaction r_j is associated with a state-change vector $\mathbf{v_j}$ specifying the change in each species when the reaction r_j takes place. That is $v_{ji} = \delta'_{ji} - \delta_{ji}$. Obviously, when r_j does not influence s_i, the value of v_{ji} will be zero. This can happen either when s_i does not participate in r_j or when $\delta'_{ji} = \delta_{ji}$. Using an appropriate kinetic rule (e.g., mass action), we can easily construct a system of ODEs that describes the evolution of species concentrations with respect to time (Herajy and Heiner 2018). However, this approach yields only an approximation of the system dynamics.

As an alternative and a more realistic simulation method, stochastic simulation can be employed to overcome the limitation of the deterministic simulation. The stochastic simulation algorithm (SSA) Gillespie (1977) produces exact numerical realisations of the Chemical Master Equation (CME). The latter is used to describe the system evolution in terms of the probability that the system will be in a certain state at time t. A fundamental premise to all SSA algorithms is the reaction propensity, $a_j(\mathbf{x})$, which can be deployed to calculate the probability that a reaction r_j will occur in the next instance of time (Gillespie 1977). However, when a chemical system contains many reactions, or even only a few reactions which do fire fairly frequently, the time steps between two successive reaction firings will be very small and the simulation procedure will consume a lot of runtime to reach the desired end of the simulation time.

A more recent approach, called Rejection-based Stochastic Simulation Algorithm (RSSA), is introduced in Thanh et al. (2014) to improve the SSA performance. The main goal of the RSSA is to minimise the frequent updates of reaction propensities, which is achieved by defining a fluctuation interval for the system state, $[\underline{\mathbf{x}}, \overline{\mathbf{x}}]$. The lower and upper bounds of the fluctuation interval ($\underline{\mathbf{x}}$ and $\overline{\mathbf{x}}$, respectively) are calculated by subtracting from and adding to each entry a percentage value pr, $0 < pr < 100\%$, of the same entry. A lower and upper bound propensity are defined for each reaction as $a_j(\underline{\mathbf{x}})$, $a_j(\overline{\mathbf{x}})$, respectively, based on the system state's fluctuation interval. Then the reaction propensities are updated only when the current state $\mathbf{x}$ leaves the fluctuation interval; i.e., when (2) becomes invalid.

$$\underline{\mathbf{x}} \leq \mathbf{x} \leq \overline{\mathbf{x}} \tag{2}$$

The upper bound propensities help to initially select a reaction to fire in a similar way as in the basic SSA. With other words, the index of the next reaction to fire is the minimum index μ satisfying (3)

$$\sum_{j=1}^{\mu} a_j(\overline{\mathbf{x}}) > p_1 a_0(\overline{\mathbf{x}}), \tag{3}$$

where p_1 is a random number from the uniform distribution $U(0,1)$, and $a_0(\overline{\mathbf{x}})$ is the cumulative propensity computed using the upper bound fluctuation interval. The initially selected reaction is first tested with the first part of Eq. (4), where p_2 is a random number from the uniform distribution $U(0,1)$. If it passes this test, the reaction will be fired. Otherwise, the second test is performed.

$$p_2 \leq \frac{a_\mu(\underline{\mathbf{x}})}{a_\mu(\overline{\mathbf{x}})} \qquad \text{or} \qquad p_2 \leq \frac{a_\mu(\mathbf{x})}{a_\mu(\overline{\mathbf{x}})} \tag{4}$$

Hence, a reaction propensity is only updated when the second test in (4) is required. According to the mathematical analysis in Thanh et al. (2014), this does not happen frequently. The time step until the next event is to occur is calculated by (5):

$$\tau = -\frac{ln(p_3)}{a_0(\overline{\mathbf{x}})}, \tag{5}$$

where p_3 is a random number from the uniform distribution $U(0,1)$. Although RSSA can substantially improve the performance of the basic SSA, it is still considered to be a pure stochastic simulation algorithm unable to deal with systems having a huge number of stochastic events.

Another approach to tackle the problem of handling a huge number of events is to use hybrid simulation (Haseltine and Rawlings 2002). Hybrid simulation algorithms usually partition the set of reactions into two subsets: G_{slow} and G_{fast}. Afterwards, G_{slow} are simulated stochastically, while G_{fast} are executed deterministically or using an approximate stochastic technique as in Salis and Kaznessis (2005). To exactly locate the time where a stochastic event is to occur, a jump equation is required. In Haseltine and Rawlings (2002) the authors propose the use of Eq. (6), based on the probability density function derived in Gillespie (1991), to locate the occurrence time of the stochastic event.

$$\int_t^{t+\tau} \sum_{j=0}^{M^{slow}} a_j(\mathbf{x})dt = -log(p)\,, \tag{6}$$

where p is a random number uniformly distributed from $U(0,1)$, j is the index of the j^{th} slow reaction, and M^{slow} is the number of slow reactions. The firing of a stochastic reaction may introduce a discontinuity in the solution of the system of ODEs. To overcome this problem, the ODE solver must restart the numerical integration from the time point where the discrete event occurred. However, frequent switches between the stochastic and deterministic regime introduce additional overheads. In Herajy and Heiner (2016), we have classified all reactions in the stochastic regime according to their relation with the deterministic regime into three groups: reactions with direct dependency, reactions with indirect dependency, and reactions without dependency. Fig. 1 gives an example for each type of relation. To simplify the discussion, we use Petri net notations (adopted from (Herajy and Heiner 2012; Herajy et al. 2013)) to illustrate the dependency type of two reactions.

A stochastic reaction is called a direct-dependent reaction if it shares at least one species with the deterministic regime. In other words, when a dependent reaction fires, it changes one or more entry values of the deterministic simulator's system state. For instance, in Fig. 1a, the two reactions $S_1 + S_2 \xrightarrow{k_1} P_1$ and $P_1 + S_3 \xrightarrow{k_2} P_2$ share a common place P_1. A stochastic reaction is called an indirect-dependent reaction if it does not share a common species with the deterministic regimes, however, one or more of its participating species are used to define the rate of deterministic reactions. Such indirect-dependent reactions do not directly affect the system state of the deterministic simulator, but they can indirectly influence it by manipulating the rate function. For instance, the two reactions in Fig. 1b: $S_1 + S_2 \xrightarrow{k_1} P_1$ and $S_3 + S_4 \xrightarrow{k_2 * P_1} P_2$ are in an indirect-dependent relation. Finally, a stochastic reaction is called *independent reaction*, if it does not share any species with the deterministic regime and its manipulated species are not involved in the definition of any reaction rates. For example the two reactions in Fig. 1c are completely independent.

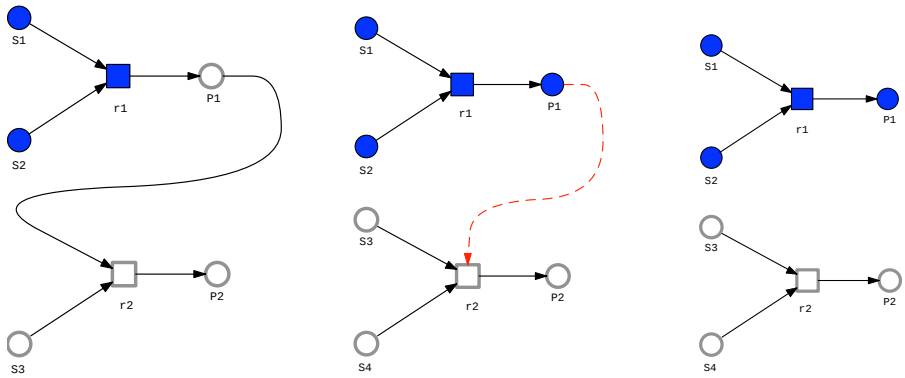

Figure 1: An example for the different dependency relations between reactions: (a) two reactions with a direct dependency, (b) two reactions with indirect dependency, and (c) two reactions with no dependency.

Such dependency information is important because we can skip the reinitialisation of the ODE solver when an independent stochastic reaction fires. In these cases, the switch from the stochastic to the deterministic regime will be much easier, since no special treatment is required.

Therefore, the overall performance of the ODE solver will be improved, when the number of dependent reactions is minimised during net partitioning. However, Eq. (6) also requires the reinitialisation of the ODE solver each time a stochastic event occurs (because it is simultaneously integrated with the system of ODEs). Thus, in Herajy and Heiner (2016), we have replaced (6) by (7), which can be evaluated independently of the numerical integration of the system of ODEs.

$$\sum_{j=0}^{M_{slow}} a_j(\mathbf{x}) \cdot \Delta\tau = -log(p)\,, \tag{7}$$

where $\Delta\tau$ is time from the firing of the previous event to the current one. Although the approximation error in (7) is of $O(h)$, where h is the step size, this can still result in a good approximation when there are many stochastic events in the simulated model. With other words, the time step between two successive discrete firings will be very small and correspondingly the error will also be very small. Nevertheless, we show in this paper how we can avoid the use of (7), while still exploiting the accelerated approach introduced in Herajy and Heiner (2016).

The HRSSA (Marchetti et al. 2016) takes advantage of RSSA by switching from the deterministic to the stochastic simulator when one of the following two condition occurs: (i) the time of the next stochastic event is reached, (ii) any of the system state entries leaves the interval $[\underline{\mathbf{x}}, \overline{\mathbf{x}}]$ (see Eq.(2)). Condition (i) can be satisfied if there is no change in the time-dependent propensities of slow reactions due to the evolution of the deterministically simulated ones, while condition (ii) is met when the ODE solver affects one or more propensities of slow reactions. HRSSA does not require the integration of (6) suggesting its combination with the accelerated algorithm in Herajy and Heiner (2016).

3 IMPROVED HYBRID SIMULATION

In this section we present a more efficient hybrid simulation algorithm by carefully analysing the structure of the reaction networks. We consider first reaction dependency without assuming any reaction partitioning. Afterwards, we record the set of direct interface reactions (R^*) between the stochastic and deterministic regime based on the extracted dependencies. To improve the performance of the HRSSA, we also record the set of interface species (S^*). Moreover, the effect of indirect interface reaction firings is considered by extracting the set of monitored species (S^m).

3.1 Reaction Dependency

The idea of dependency graphs has been used in Gibson and Bruck (2000) to record dependent reactions such that when a reaction fires, all dependent reaction propensities are updated. For example, when the reaction r_1 in Fig. 1a fires, the propensity of reaction r_2 has to be updated (assuming the two reactions are simulated via SSA). In this paper, we collect the complementary dependency information which allows us to extend the method in Herajy and Heiner (2016) to easily support dynamic hybrid simulation. We record for each reaction r_j the set of other reactions which enforce r_j to update its propensity when any of them fires. For instance in Fig. 1a, instead of recording r_2 as a reaction dependent on r_1, we record r_1 as an updater reaction of r_2. In what follows we elaborate on how to compute and use this information, but first we formally define the following sets.

Definition 1. *Reactants and Products: let $r_\mu \in R$ and $s_i \in S$, then the sets of Reactants and Products of r_μ, denoted by $Reactants(r_\mu)$ and $Products(r_\mu)$, respectively, are defined as follows:*

- $Reactants(r_\mu) = \{s_i | s_i$ *appears as a substrate of the reaction* $r_\mu\}$
- $Products(r_\mu) = \{s_i | s_i$ *appears as a product of the reaction* $r_\mu\}$.

Definition 2. *Manipulated Species: let $r_\mu \in R$ and $s_i \in S$, then the set of manipulated species of r_μ, denoted by $Manipulated(r_\mu)$, is defined as follows:*

$$Manipulated(r_\mu) = \{s_i | s_i \in (Products(r_\mu) \cup Reactants(r_\mu)) \text{ and } v_{\mu i} \neq 0\}.$$

Definition 3. *Direct and Indirect Components: let $s_i \in S$, then the sets of direct and indirect components of the propensity of reaction $r_\mu \in R$, denoted by $dComponents(a_\mu)$ and $iComponents(a_\mu)$, respectively, are defined as follows:*

- $dComponents(a_\mu) = \{s_i | s_i$ *appears in* a_μ *and* $s_i \in Reactants(r_\mu)$ *and* $v_{\mu i} \neq 0\}$
- $iComponents(a_\mu) = \{s_i | s_i$ *appears in* a_μ *and* $s_i \notin dComponents(a_\mu)\}$.

Definition 4. *Direct and Indirect Updaters: let $r_j, r_\mu \in R$ and $s_i \in S$, then the sets of direct and indirect updaters of r_μ, denoted by $dUpdaters(r_\mu)$ and $iUpdaters(r_\mu)$, respectively, are defined as follows:*

- $dUpdaters(r_\mu) = \{r_j | Manipulated(r_\mu) \cap Manipulated(r_j) \neq \phi\}$
- $iUpdaters(r_\mu) = \{r_j | (iComponents(a_\mu) \cap Manipulated(r_j) \neq \phi)\}$.

Definition 5. *Interface Species: let $r_j, r_\mu \in R$, then the set of interface species between the reactions r_j and r_μ, denoted by $S^*_{\mu j}$, is defined as follows:*

$$S^*_{\mu j} = (dComponents(a_\mu) \cup iComponents(a_\mu)) \cap Manipulated(r_j).$$

Definition 6. *Monitored Species: For each $r_j \in R^{slow}$ and $r_\mu \in R^{fast}$, the set of monitored species is given by:*

$$S^m = Manipulated(r_j) \cup iComponents(r_\mu).$$

We can easily design an algorithm to extract the dependency of all reactions. The Reaction Dependency Extraction Algorithm (RDEA) in Fig. 2 takes the set of reactions R as input. For each reaction r_μ in R, we check all other reactions r_j. If the intersection of r_μ's manipulated species and the ones of r_j is not empty (step 3), r_j is added to the set of direct updaters of r_μ (step 4) and the set of interface species between the two reactions is recorded (step 5). If this is not the case, the RDEA checks the intersection between the indirect components of r_μ's propensity and the manipulated species of r_j (step 6). If the resulting set is not empty, r_j will be added to the set of indirect updaters of r_μ (step 7), and the set of interface species is recorded (step 8). At the end, the set of reactions together with the set of direct and indirect updaters of

Require: R: the set of reactions;
1: **for each** r_μ in R **do**
2: **for each** r_j in $R - \{r_\mu\}$ **do**
3: **if** $Manipulated(r_\mu) \cap Manipulated(r_j) \neq \phi$ **then**
4: Add r_j to the set of $dUpdaters(r_\mu)$
5: $S^*_{\mu j} = dComponents(a_\mu) \cap Manipulated(r_j)$
6: **else if** $iComponents(a_\mu) \cap Manipulated(r_j) \neq \phi$ **then**
7: Add r_j to the set of $iUpdaters(r_\mu)$
8: $S^*_{\mu j} = iComponents(a_\mu) \cap Manipulated(r_j)$
9: **end if**
10: **end for**
11: **end for**
12: **return** R with the dependency information;

Figure 2: Reaction Dependency Extraction Algorithm (RDEA).

each reaction are returned. Please note that a reaction can either be marked as a direct or indirect updater of the same reaction. Moreover, a reaction cannot be marked as direct or indirect updater to itself.

The time complexity of the RDEA is of the order $O(M^2)$, where M is the number of reactions. This will rapidly slow down the simulation's initialisation for bigger models. However, the RDEA can easily be simplified to be of order $O(M)$ by performing the calculations in two rounds. In the first round, we record for each species the set of reactions that manipulate it. This can easily be obtained through the local information provided in the reactions' species. In the second round, for each species $s_i \in Reactants(r_\mu)$, we add all reactions that manipulate s_i to $dUpdaters(r_\mu)$. Similarly, we add the set of reactions that manipulate $s_i \in iComponents(a_\mu)$ to the set $iUpdaters(r_\mu)$ (assuming they were added to $dUpdaters(r_\mu)$).

3.2 Extracting Interface Reactions

Using the dependencies collected by RDEA as input, the Direct Interface Reactions Extraction Algorithm (DIREA) (Fig. 3) detects the set of direct interface reactions (R^*). The reactions in R^* reset the numerical integration as soon as they occur, since they directly change the system state of the ODE solver (Herajy and Heiner 2016). A reaction r_j is classified as a *direct interface reaction* if the following two conditions hold: (i) r_j belongs to the set of slow reactions and (ii) r_j is listed as an updater of one of the fast reactions. Hence, we will not be able to determine R^* unless the reaction network is partitioned into slow and fast subnets. For instance, r_1 is a direct interface reaction in the network in Fig. 1a. The inputs to DIREA are the set of slow reactions G_{slow}, the set of fast reactions G_{fast}, and the dependency information extracted by RDEA.

DIREA iterates over all reactions in the fast group (step 2). It marks the slow reactions in the direct updaters of r_μ as direct interface reactions (steps 3-7). Finally, it returns the set of direct interface reactions.

There are two other types of reactions which are not classified by this algorithm: indirect interface reactions and independent reactions. Indirect interface reactions are those which indirectly affect the system state of the deterministic solver when they fire. They are discussed in more detail in the next section.

Indirect Interface Reactions and Monitored Species Indirect interface reactions are intricate to deal with as they do not influence the system state when they fire. Instead, they may cause discontinuity in the fast regime when they impose abrupt changes in the system state of the ODE solver. Therefore, we have to consider their effects when such an abrupt change is to occur. A reaction is classified as an *indirect interface reaction*, if (i) it belongs to the set of slow reactions, and (ii) it belongs to one of the indirect updaters of the fast reactions. In Herajy and Heiner (2016) we have ignored indirect interface reactions and relied on the ODE solver to overcome the discontinuities due to their firing. For instance, some ODE solvers (e.g., CVODE (Hindmarsh et al. 2005)) return with a special error code when they are not able to

Require: G_{slow} and G_{fast} the sets of slow and fast reactions, respectively;
Require: $dUpdaters(r_j)$, $\forall r_j \in G_{fast}$ (see Figure 2);
1: Set $R^* = \phi$; {The set of direct interface reaction is initially empty}
2: **for each** r_μ in G_{fast} **do**
3: **for each** $r_j \in dUpdaters(r_\mu)$ **do**
4: **if** $r_j \in G_{slow}$ **then**
5: Add r_j to R^*; {Mark r_j as a direct interface reaction}
6: **end if**
7: **end for**
8: **end for**
9: **return** R^*;

Figure 3: Direct Interface Reactions Extraction Algorithm (DIREA).

deal with a discontinuity. In this case we can make use of such an opportunity and restart the numerical integration from this time point, when such an error code is returned. However, this strategy cannot easily be applied for all kinds of ODE solvers. Therefore an efficient solution to this problem will be of paramount importance to improve the accuracy of the accelerated approach.

One simple solution is to record all indirect interface reactions similar to the direct ones. During the simulation we check if any of the indirect interface reactions will cause an abrupt change in the system state of the ODE solver by examining the propensities of the affected reactions before and after the firing of r_j. However, this will complicate the implementation and introduce additional overhead due to the repeated updating and checking of the reaction propensities.

In this section we propose a different solution by recording the set of intersection species between the manipulated species of the indirect interface reactions and the indirect components of the corresponding affected fast reactions. We call this set *monitored species* (S^m). Later during the simulation, if any of such monitored species involves an abrupt change, the ODE solver will be reinitialised. Fortunately, we do not need to do that for each reaction firing, instead we can make such check while updating the fluctuation interval (see Section 2). The reason for this assumption is the following. If any of the system state entries exhibits an abrupt change, the fluctuation interval will need to be updated. The Monitored Species Extraction Algorithm (MSEA) in Fig. 4 extracts the set of monitored species by taking the partitioned net as well as the extracted dependency information as inputs. The set of monitored species is initially set to empty (step 1). Afterwards, MSEA iterates over each reaction r_μ in the group of fast reactions (step 2). For each slow reaction in the set of indirect updaters of r_μ, we add the intersection of the manipulated set of r_j and the indirect components of r_μ to the set of monitored species (step 5). At the end, the set S^m is returned.

Extracting Interface Species One drawback of the hybrid approach presented in Marchetti et al. (2016) is the check of Eq. (2). As this check is performed several times for every call of the ODE solver, it will consume a substantial amount of time, particularly for larger models. However, we do not need to check all the species since many of them are not affected by the ODE solver (e.g., all discrete species). Therefore, we select only those species which change the propensities of the slow reactions. We call them *interface species*. Once we have extracted the set of interface species, the check in Eq. (2) can be simplified to (8)

$$\underline{\mathbf{x}^*} \leq \mathbf{x}^* \leq \overline{\mathbf{x}^*}, \tag{8}$$

where $\mathbf{x}^* \subseteq \mathbf{x}$ is a subset of the state vector which only involves the values of interface species, and $\underline{\mathbf{x}^*}$ and $\overline{\mathbf{x}^*}$ are the corresponding lower and upper bounds of the fluctuation vector corresponding to the set of interface species. The Interface Species Extraction Algorithm (ISEA) (Fig. 5) extracts all interface species between the fast and slow regime. It iterates over the sets of direct and indirect updaters of slow reactions (steps 2-7). If any of them belongs to the fast group, the set of interface species between the two reactions will be added to S^* (step 5).

Require: G_{slow} and G_{fast} the sets of slow and fast reactions, respectively;
Require: $iUpdaters(r_j)$, $\forall r_j \in G_{fast}$ (see Figure 2);
1: $S^m = \phi$; {the set of monitored species is initially empty}
2: **for each** r_μ in G_{fast} **do**
3: **for each** $r_j \in iUpdaters(r_\mu)$ **do**
4: **if** $r_j \in G_{slow}$ **then**
5: Add $Manipulated(r_j) \cup iComponents(r_\mu)$ to S^m ;
6: **end if**
7: **end for**
8: **end for**
9: **return** S^m;

Figure 4: Monitored Species Extraction Algorithm (MSEA).

Improved Hybrid Simulation Algorithm Once interface reactions and species have been extracted, they can be fed to the HRSSA algorithm. In Fig. 6 is an accelerated version of the HRSSA given in Marchetti et al. (2016). The proposed improvements include introducing constraints when reinitialising the deterministic solver and the use of a subset of the system state vector to implement the jump equation (2).

The improved HRSSA takes as inputs the sets of slow and fast reactions, the set of direct interface reactions, and the sets of interface and monitored species. Initially, the simulation time is set to zero and the ODE solver is initialised with the initial state vector. Afterwards, the algorithm repeats steps 3-25 until the end simulation time is reached.

Each time, when the fluctuation interval requires an update, the interval as well as the lower and upper propensity bounds are calculated in steps 4-5. The propensity bounds are then used to fire reactions until they become no longer valid (i.e., the state vector leaves the interval $[\underline{\mathbf{x}}, \overline{\mathbf{x}}]$). This may happen in one of the following circumstances: (i) when a stochastic event fires causing a larger change in the state vector (step 14), (ii) Eq. (8) becomes invalid (step 19), or (iii) it is not possible to select a stochastic reaction according to Eq. (3)- (4).

Between the updates of the fluctuation interval, the algorithm calculates the time for the next reaction to occur (step 8). Afterwards, the ODE solver is called to solve the system of ODEs that corresponds to the fast reactions (step 9). The ODE solver stops the integration when either the time of the stochastic event is reached or Eq. 8 is violated. In the former case, a reaction is selected to fire (steps 12-13), while in the latter the fluctuation interval is updated (step 19). In both cases, the simulation time is advanced by a time step taken by the ODE solver. Unlike the original algorithm in Marchetti et al. (2016), the deterministic solver does not need to be reset at every occurrence of stochastic reactions. Instead, only a few cases will generally require such a heavy reinitialisation. The first case is when the reaction selected to fire belongs to the set of direct interface reactions (steps 15-17). The second case is when this reaction belongs to the set of indirect interface reactions and the firing will have a considerable effect on the rate of one ore more reactions in the deterministic regime (steps 23-25). This check is done only during the update of the fluctuation interval. Furthermore, in some models, an abrupt change of the monitored species does not imply the same for the corresponding reactions. For instance, if a reaction propensity is defined using Michaels-Menten kinetics, then the change of a substrate from one to zero does not imply the same change for the propensity. Thus, a pre-analysis of the reaction propensities can preclude some of the species which have previously been marked as monitored species.

4 EVALUATION

In this section we evaluate our approach of improving the hybrid simulation using three case studies as benchmarks. These comprise two models devoted to yeast (62 species and 194 reactions) and eukaryotic cell cycles (26 species and 51 reactions), and another model to study calcium dynamics (10,252 species and 57,401 reactions). The yeast cell cycle model is based on the stochastic kinetics from Barik et al. (2010).

Require: G_{slow} and G_{fast} the sets of slow and fast reactions, respectively;
Require: $iUpdaters(r_j)$ and $iUpdaters(r_j)$, $\forall r_j \in G_{slow}$ (see Figure 2);
1: $S^* = \phi$; {the set of marked species is initially empty}
2: **for each** $r_\mu \in G_{slow}$ **do**
3: **for each** $r_j \in dUpdaters(r_\mu) \cup iUpdaters(r_\mu)$ **do**
4: **if** $r_j \in G_{fast}$ **then**
5: Add $S^*_{\mu j}$ to S^*;
6: **end if**
7: **end for**
8: **end for**
9: **return** S^*;

Figure 5: Interface Species Extraction Algorithm (ISEA).

In this model reactions with high rates (e.g., protein interactions) are modelled as deterministic processes, while reactions with low rates are considered as stochastic events (e.g., gene expression). For the purpose of our paper, we use a hybrid model constructed in Herajy et al. (2013), providing the model structure, initial values, and kinetic rate constants. are also available in Herajy et al. (2013). We perform only one hybrid run for this experiment. The ODE numerical integration library CVODE (Hindmarsh et al. 2005) has been used to solve the deterministic part. The second case study is the Eukaryotic cell cycle model based on the hybrid model presented in Herajy et al. (2018), giving also the initial state as well as the kinetic rate constants. Table 1 compares the runtimes of our method with two other hybrid simulation algorithms. The calcium model is applied in Nagaiah et al. (2012) to study calcium flow from the endoplasmic reticulum to the cytoplasm when channels are open. Diffusion reactions and the binding/unbinding of calcium with buffers are simulated deterministically, while channel state transitions are simulated stochastically. In this paper we consider a simple discretisation of the grid into 50×50 and 25 channels arranged in one cluster. The model description is available at Nagaiah et al. (2012). The benchmarks are performed using snoopy Herajy et al. (2017) - a Petri net tool that supports, among others, the simulation of hybrid models. Snoopy implements Algorithms 2- 6 in addition to the exact version of the Haseltine &Rawlings approach in Haseltine and Rawlings (2002). The runtimes in Table 1 demonstrate that the accelerated approach in combination with the HRSSA indeed improves the hybrid simulation performance. The performance gain increases with a decrease in the number of interface reactions. For instance, our saving in runtime when simulating the Yeast Cell Cycle model is better than for the Eukaryotic Cell Cycle, since the former has less interface reactions. For larger models (e.g., the calcium model), the ODE solver will need more time to be reinitialised. This is a great opportunity to reduce the effects of such a step using the presented improvements. Moreover, the large number of variables in a model considerably affects the performance of the HRSSA algorithm (as demonstrated by the calcium model). Therefore, our suggestion to use interface species (see Eq. (8)) improves the performance of the HRSSA. The accuracy of our proposed approach is equivalent to the original HRSSA, since our improvements do not alter the way in which HRSSA simulates hybrid models. In Marchetti et al. (2016), the accuracy of the HRSSA has been compared with many other simulation algorithms including Gillespie's direct method. It has been confirmed that the accuracy of the HRSSA is close to the direct method's results. Moreover, compared with the accelerated algorithm in Herajy and Heiner (2016), the improved HRSSA removes the need for approximating the jump equation.

5 CONCLUSIONS AND FUTURE WORK

In this paper we have introduced an improved approach to speed up the simulation of hybrid biological models that require the combination of stochastic and deterministic approaches. Our ideas may best be used in combination with the static partitioning approach and can be extended to support the dynamic one.

For the simulation of statically partitioned nets, the algorithms outlined in this paper can be considerably simplified to minimise the initialisation time, particularly for larger nets consisting of hundreds of thousands

Require: G_{slow} and G_{fast}: the sets of slow and fast reactions, respectively;
Require: R^*: the set of direct interface reactions;
Require: S^*, S^m: the sets of interface and monitored species, respectively;
Initialise the ODE solver with the initial system state, $\mathbf{x_0}$;
Set the current time $t = 0$;
while $t < t_{end}$ **do**
  Calculate the fluctuation intervals, $\underline{\mathbf{x}}$, and $\overline{\mathbf{x}}$;
  Calculate the propensity upper and lower bounds, $\mathbf{a}(\underline{\mathbf{x}})$ and $\mathbf{a}(\overline{\mathbf{x}})$;
  UpdateFluctuationInterval:=false;
  while *UpdateFluctuationInterval*=false **and** $t < t_{end}$ **do**
    Calculate τ according to (5);
    Integrate the system of ODEs until time $t + \tau$ or (8) is violated;
    let Δt= the time step taken by the ODE solver;
    if Eq. (8) has not been violated **then**
      Select a reaction r_μ to fire according to (3) - (4);
      Fire r_μ and update the system state accordingly;
      if $\mathbf{x}$ leaves $[\underline{\mathbf{x}},\overline{\mathbf{x}}]$ **then** *UpdateFluctuationInterval*=true;
      if $r_\mu \in R^*$ **then**
        Restart the integration from $t + \Delta t$ using the current system state $\mathbf{x}$;
      end if
    else
      UpdateFluctuationInterval=true;
    end if
    Update the current time, $t = t + \Delta t$;
  end while
  if $\exists s_i \in S^m$ **such that** there is an abrupt change in x_i **then**
    Restart the integration from $t + \Delta t$ using the current system state $\mathbf{x}$;
  end if
end while

Figure 6: Improved HRSSA (iHRSSA).

Table 1: Runtimes (in seconds) for the three case studies using different hybrid algorithms performed on a Mac Pro. with 3 GHz Core i7 processor and 8GB memory.

Models/ Algorithms	Haseltine &Rawlings	Accelerated	HRSSA	Improved HRSSA	Simulation Time
Yeast Cell Cycle	984.5	423	435.2	163.3	10,000 min.
Eukaryotic Cell Cycle	780	276.2	240	164.2	1,000 min.
Calcium model	22,620	14,460	30,980	10,800	10 sec.

or even millions of species and reactions. This kind of models usually comprise a few stochastic reactions compared with the number of deterministic ones. In this scenario, the set of direct interface reactions can be extracted by following the manipulated species of each stochastic transition and checking their connections with deterministic reactions. One interesting direction to extend the work presented in this paper is to develop an algorithm to determine the best partitioning of the net that minimises the number interface reactions, interface species, and monitored species, while partitioning the set of reactions into fast and slow ones. Moreover, extending our algorithms to support dynamic partitioning is also of paramount importance.

REFERENCES

Barik, D., W. T. Baumann, M. R. Paul, B. Novak, and J. J. Tyson. 2010. "A Model of Yeast Cell-Cycle Regulation based on Multisite Phosphorylation". *Molecular Systems Biology* 6(1):405.

Gibson, M., and J. Bruck. 2000. "Exact Stochastic Simulation of Chemical Systems with many Species and Many Channels". *J. Phys. Chem.* 105:1876 – 89.

Gillespie, D. 1977. "Exact Stochastic Simulation of Coupled Chemical Reactions". *J. Phys. Chem.* 81(25):2340–2361.

Gillespie, D. 1991. *Markov Processes: An Introduction for Physical Scientists*. Academic Press.

Haseltine, E., and J. Rawlings. 2002. "Approximate Simulation of Coupled Fast and Slow Reactions for Stochastic Chemical Kinetics". *J. Chem. Phys.* 117(15):6959–6969.

Herajy, M., and M. Heiner. 2012, November. "Hybrid Representation and Simulation of Stiff Biochemical Networks". *J. Nonlinear Analysis: Hybrid Systems* 6(4):942–959.

Herajy, M., and M. Heiner. 2016. "Accelerated Simulation of Hybrid Biological Models with Quasi-Disjoint Deterministic and Stochastic Subnets". In *Hybrid Systems Biology: 5th International Workshop, HSB 2016*, edited by E. Cinquemani and A. Donzé, LNBI, 20–38. Springer.

Herajy, M., and M. Heiner. 2018. "Adaptive and Bio-semantics of Continuous Petri Nets: Choosing the Appropriate Interpretation". *Fundamenta Informaticae* 160(1-2):53–80.

Herajy, M., F. Liu, and M. Heiner. 2018. "Efficient Modelling of Yeast Cell Cycles based on Multisite Phosphorylation using coloured Hybrid Petri Nets with Marking-Dependent Arc Weights". *Nonlinear Analysis: Hybrid Systems* 27:191 – 212.

Herajy, M., F. Liu, C. Rohr, and M. Heiner. 2017, Jul. "Snoopy's Hybrid Simulator: A Tool to Construct and Simulate Hybrid Biological Models". *BMC Systems Biology* 11(1):71.

Herajy, M., M. Schwarick, and M. Heiner. 2013. *Hybrid Petri Nets for Modelling the Eukaryotic Cell Cycle*, Chapter ToPNoC, 123–141. Springer Berlin Heidelberg.

Hindmarsh, A., P. Brown, K. Grant, S. Lee, R. Serban, D. Shumaker, and C. Woodward. 2005. "SUNDIALS: Suite of Nonlinear and Differential/Algebraic Equation Solvers". *ACM Trans. Math. Softw.* 31:363–396.

Iwamoto, K., H. Hamada, Y. Eguchi, and M. Okamoto. 2014, 07. "Stochasticity of Intranuclear Biochemical Reaction Processes Controls the Final Decision of Cell Fate Associated with DNA Damage". *PLoS ONE* 9:1–12.

Marchetti, L., C. Priami, and V. H. Thanh. 2016. "HRSSA – Efficient Hybrid Stochastic Simulation for Spatially Homogeneous Biochemical Reaction Networks". *J. Comput. Phys.* 317:301 – 317.

Nagaiah, C., S. Rüdiger, G. Warnecke, and M. Falcke. 2012. "Adaptive Space and Time Numerical Simulation of Reaction-Diffusion models for Intracellular Calcium Dynamics". *Applied Mathematics and Computation* 218(20):10194 – 10210.

Salis, H., and Y. Kaznessis. 2005. "Accurate Hybrid Stochastic Simulation of a System of Coupled Chemical or Biochemical Reactions". *J. Chem. Phys* 122(5):054103.

Thanh, V. H., C. Priami, and R. Zunino. 2014. "Efficient Rejection-Based Simulation of Biochemical Reactions with Stochastic Noise and Delays". *J. Chem. Phys* 141(13):134116.

AUTHOR BIOGRAPHIES

MOSTAFA HERAJY is an assistant professor of computer science at Port said university, Egypt. He got his Ph.D in computer science from Brandenburg University of Technology, Germany at 2013. His research interest is systems biology and hybrid simulation. He published some journal papers about hybrid Petri nets and their applications to model multi-timescale biological systems. His email is: mheraajy@sci.psu.ed.eg.

MONIKA HEINER received her Ph.D degree from TU-Dresden. Since 1994, she holds a professorship for data structures and software dependability at BTU-Cottbus. She has a long history of expertise in Petri nets, shifting her application focus from technical to natural systems in the late 1990s. Methodology oriented aspects of her work include a unifying framework of computational models and BioModel engineering. Over the last 20 years, her group at Cottbus has developed a sophisticated toolkit to support the methodology, comprising Snoopy, Charlie, and Marcie. Her email is: monika.heiner@b-tu.de.