Hindawi
Complexity
Volume 2017, Article ID 1232868, 12 pages
https://doi.org/10.1155/2017/1232868

WILEY | Hindawi

Research Article

HSimulator: Hybrid Stochastic/Deterministic Simulation of Biochemical Reaction Networks

Luca Marchetti,[1] Rosario Lombardo,[1] and Corrado Priami[1,2]

[1]*The Microsoft Research–University of Trento Centre for Computational and Systems Biology (COSBI), Piazza Manifattura, No. 1, 38068 Rovereto, Italy*
[2]*Department of Computer Science, University of Pisa, Pisa, Italy*

Correspondence should be addressed to Luca Marchetti; marchetti@cosbi.eu

Received 12 May 2017; Revised 18 August 2017; Accepted 19 November 2017; Published 13 December 2017

Academic Editor: Valeri Mladenov

HSimulator is a multithread simulator for mass-action biochemical reaction systems placed in a well-mixed environment. HSimulator provides optimized implementation of a set of widespread state-of-the-art stochastic, deterministic, and hybrid simulation strategies including the first publicly available implementation of the Hybrid Rejection-based Stochastic Simulation Algorithm (HRSSA). HRSSA, the fastest hybrid algorithm to date, allows for an efficient simulation of the models while ensuring the exact simulation of a subset of the reaction network modeling slow reactions. Benchmarks show that HSimulator is often considerably faster than the other considered simulators. The software, running on Java v6.0 or higher, offers a simulation GUI for modeling and visually exploring biological processes and a Javadoc-documented Java library to support the development of custom applications. HSimulator is released under the COSBI Shared Source license agreement (COSBI-SSLA).

1. Introduction

Computational systems biology is becoming a fundamental tool of life-science research, which aims at developing *models* representing biological phenomena and reliable *computational techniques* for their simulation [1–7].

We introduce *HSimulator*, a Java hybrid stochastic/deterministic simulator for mass-action biochemical reaction systems placed in a well-mixed environment, where position and speed of molecular species are randomized and therefore they do not affect reaction executions. Species $S_1, \ldots, S_n$ are represented in terms of *abundances* (number of molecules) and reactions $R_1, \ldots, R_m$ are defined as

$$R_j : \nu_{1j} S_1 + \cdots + \nu_{nj} S_n \longrightarrow \nu'_{1j} S_1 + \cdots + \nu'_{nj} S_n, \ c_j, \qquad (1)$$

where the species on the left of the arrow are *reactants* and the ones on the right are *products*. The stoichiometric coefficients ν_{ij} and ν'_{ij} indicate how many molecules of reactant are consumed and how many molecules of product are produced, respectively. The constant c_j at the end of the reaction is the *stochastic reaction constant* introduced by Gillespie [8] for computing reaction firing according to the definition of *mass-action propensities*.

From the milestone work of Gillespie [8], where the *Direct Method* (DM) has been defined, *exact stochastic simulation* is the most accurate simulation approach. Its drawback is the high computational complexity arising from the need of separately simulating each reaction firing. Several algorithms have been introduced to decrease simulation runtime, such as the *Next Reaction Method* (NRM, [9]) and later the *Rejection-based Stochastic Simulation Algorithm* (RSSA, [10]). The latter is tailored for complex biochemical reactions with time-consuming propensity functions and it constitutes the state of the art of exact stochastic simulation.

Despite the improvements, the problem of simulation complexity remains. In fact, the investigation of complex diseases or disorders is often concerned with extended biomolecular networks involving genes, proteins, metabolites, and signal transduction cascades. This justifies the introduction of *approximate* simulation strategies, which sacrifice accuracy to decrease runtime. Such strategies progressively reduce the stochasticity of the dynamics [11, 12] until reaching a *deterministic* simulation, where the model is translated to a

system of *ordinary differential equations* (ODEs, [7, 13]) and the dynamics provides an averaged behavior.

A drawback of an approximate simulation is that *all* reactions in the model are simulated according to the same approximate strategy. This is an issue when exact simulation is required at least for a small part of the reaction network, for example, when slow reactions are considered to model rare stochastic events. In such a case, a *hybrid* simulation strategy is able to partition reactions into subsets that are simulated by different algorithms [7, 14, 15]. Hybrid simulation therefore needs to *synchronize* the progress of different simulation strategies in order to guarantee the exactness of the simulation of slow reactions. This requires considering *time-varying transition propensities*, which in turn require the computation of integrals to calculate the exact firing time of slow reactions [7, 14, 16–20] (see also Section 2.5 and (19)). This step is computationally demanding and practical implementation necessarily approximates the integral computation by a numerical method that affects the accuracy of the simulation of slow reactions.

The *Hybrid Rejection-based Stochastic Simulation Algorithm* (HRSSA) has been recently introduced in [21] to avoid approximations in simulating slow reactions. HRSSA exploits some computational advantages introduced in RSSA to exactly simulate slow reactions and to apply an efficient and accurate dynamic partitioning of reactions, which constitute the two most significant bottlenecks of hybrid simulation.

HSimulator herein presented fills a gap in the current literature of stochastic and hybrid simulation by providing a suite of published state-of-the-art simulation algorithms including, in the same package, the exact algorithm RSSA and the first publicly available implementation of its hybrid version HRSSA. The benchmarks in the paper show that the HSimulator implementation is often faster than the state-of-the-art simulator COPASI [22]. HSimulator is released under the COSBI Shared Source license agreement (COSBI-SSLA) and it is available for download at the COSBI website at http://www.cosbi.eu/research/prototypes/hsimulator.

2. Materials and Methods

HSimulator provides an implementation of five state-of-the-art simulation strategies covering exact stochastic simulation (DM and RSSA), deterministic simulation (Forward Euler and the Runge-Kutta-Fehlberg RK45 adaptive algorithm), and hybrid simulation (HRSSA). For deterministic and hybrid simulations, the simulator automatically translates the mass-action reaction network into a set of ordinary differential equations (ODEs, see Section 2.3). In the following some insights about the implemented simulation algorithms are provided as well as their pseudocodes, which can be used as a reference to better understand the functionalities implemented in the simulator (for additional details, the reader is invited to refer to the user guide of HSimulator available at http://www.cosbi.eu/research/prototypes/hsimulator). Readers already familiar with the topic can skip this section and going directly to Section 3.

2.1. Direct Method (DM). The Direct Method constitutes the first practical implementation of an exact stochastic simulator [8]. The simulation of an exact algorithm is based on the concept of *reaction probability density function* (pdf):

$$P\left(\tau, j \mid \mathbf{x}, t\right) = a_j\left(\mathbf{x}\right) e^{-a_0(\mathbf{x})\tau}, \tag{2}$$

which provides the probability $P(\tau, j \mid \mathbf{x}, t)d\tau$ that the next reaction to apply will be R_j in the infinitesimal time interval $[t + \tau; t + \tau + d\tau]$, given the system state $\mathbf{x}$ at time t.

The *state of the system* at time t is represented by the column vector

$$\mathbf{X}\left(t\right) = \left(X_1\left(t\right), \ldots, X_n\left(t\right)\right)^T, \tag{3}$$

where $X_i(t)$ is the number of molecules of species S_i in the system at time t. In the following we will often write $\mathbf{x}$ for $\mathbf{X}(t)$ and x_i for $X_i(t)$ to improve the readability of formulas.

The term $a_j(\mathbf{x})$ in (2) is the *propensity* of R_j in the state $\mathbf{x}$, while the *total propensity* $a_0(\mathbf{x})$ is the sum of the propensities of all reactions:

$$a_0\left(\mathbf{x}\right) = \sum_{j=1}^{m} a_j\left(\mathbf{x}\right). \tag{4}$$

The reaction propensities $a_j(\mathbf{x})$ are computed from the stochastic reaction rate c_j:

$$a_j\left(\mathbf{x}\right) = \begin{cases} c_j, & \text{if } h_j\left(\mathbf{x}\right) = 0 \\ h_j\left(\mathbf{x}\right) c_j, & \text{otherwise,} \end{cases} \tag{5}$$

where $h_j(\mathbf{x})$ is the number of distinct reactant combinations for R_j in the state $\mathbf{x}$ [8]. For standard mass-action kinetics, as considered here, it is

$$h_j\left(\mathbf{x}\right) = \prod_i \binom{x_i}{v_{ij}} = \prod_i \frac{x_i!}{v_{ij}!\left(x_i - v_{ij}\right)!}, \tag{6}$$

where v_{ij} are the stoichiometric coefficients of R_j according to (1).

Equation (2) shows that the next reaction R_j fires with a discrete probability $a_j(\mathbf{x})/a_0(\mathbf{x})$ and its firing time τ has an exponential distribution with rate $a_0(\mathbf{x})$. Gillespie introduced the *Direct Method* (DM) for exactly sampling the pdf $P(\tau, j \mid \mathbf{x}, t)$ by applying the inverse transformation:

$$\begin{aligned} \tau &= -\frac{\ln\left(r_1\right)}{a_0\left(\mathbf{x}\right)}, \\ j &= \text{the smallest index s.t. } \sum_{i=1}^{j} \frac{a_i\left(\mathbf{x}\right)}{a_0\left(\mathbf{x}\right)} \geq r_2, \end{aligned} \tag{7}$$

where r_1 and r_2 are random numbers drawn from a uniform distribution $U(0, 1)$. The pseudocode of the DM algorithm is in Algorithm 1.

Input: a reaction system with n species and m reactions as the one in (1), with a stochastic reaction constant c_j associated with each reaction R_j; an initial state $\mathbf{X}(t_0)$ defining molecule abundances at time t_0; the last time instant t_{end} to be simulated.
Output: a time series of states $\mathbf{X}(t)$, $t \in [t_0; t_{\text{end}}]$, providing the dynamics of the system.
Pseudocode:

(0) $t \coloneqq t_0$;
(1) `while` $t < t_{\text{end}}$ `do`
  (2) For each reaction R_i, $i = 1, \ldots, m$, compute reaction propensities $a_i(\mathbf{x})$ according to (5);
  (3) Compute $a_0(\mathbf{x})$ according to (4);
  (4) Generate two random numbers r_1, r_2 in $U(0, 1)$;
  (5) $\tau \coloneqq -\ln(r_1)/a_0(\mathbf{x})$;
  (6) Select R_j such that j satisfies
$$\sum_{i=1}^{j-1} \frac{a_i(\mathbf{x})}{a_0(\mathbf{x})} < r_2 \leq \sum_{i=1}^{j} \frac{a_i(\mathbf{x})}{a_0(\mathbf{x})};$$
  (7) Compute $\mathbf{X}(t + \tau)$ by applying R_j to $\mathbf{x}$;
  (8) $t \coloneqq t + \tau$;
(9) `end while`

Algorithm 1: Gillespie's Direct Method (DM).

Input: The same as DM (Algorithm 1) together with a simulation parameter $0 < \delta < 1$ for calculating the fluctuation interval of the system state.
Output: a time series of states $\mathbf{X}(t)$, $t \in [t_0; t_{\text{end}}]$, providing the dynamics of the system.
Pseudocode:

(0) $t \coloneqq t_0$;
(1) `while` $t < t_{\text{end}}$ `do`
  (2) Define the fluctuation interval $\mathbb{X} = [\underline{\mathbf{x}}; \overline{\mathbf{x}}]$ according to (8);
  (3) For each reaction R_i, $i = 1, \ldots, m$, compute propensity bounds $a_i(\underline{\mathbf{x}})$, $a_i(\overline{\mathbf{x}})$ and the total propensity $a_0(\overline{\mathbf{x}})$;
  (4) `while` $(t < t_{\text{end}} \wedge \mathbf{x} \in \mathbb{X})$ `do`
    (5) Generate three random numbers r_1, r_2, r_3 in $U(0, 1)$;
    (6) $\tau \coloneqq -\ln(r_1)/a_0(\overline{\mathbf{x}})$;
    (7) Select R_j such that j satisfies
$$\sum_{i=1}^{j-1} \frac{a_i(\overline{\mathbf{x}})}{a_0(\overline{\mathbf{x}})} < r_2 \leq \sum_{i=1}^{j} \frac{a_i(\overline{\mathbf{x}})}{a_0(\overline{\mathbf{x}})};$$
    (8) accepted $\coloneqq$ false;
    (9) `if` $(r_3 \leq a_j(\underline{\mathbf{x}})/a_j(\overline{\mathbf{x}}))$ `then`
      (10) accepted $\coloneqq$ true;
    (11) `else`
      (12) Compute $a_j(\mathbf{x})$ according to (5);
      (13) `if` $r_3 \leq a_j(\mathbf{x})/a_j(\overline{\mathbf{x}})$ `then` accepted $\coloneqq$ true;
    (14) `end if`
    (15) `if` (accepted) `then`
      (16) Compute $\mathbf{X}(t + \tau)$ by applying R_j to $\mathbf{x}$;
    (17) `end if`
    (18) $t \coloneqq t + \tau$;
  (19) `end while`
(20) `end while`

Algorithm 2: Rejection-based Stochastic Simulation Algorithm (RSSA).

2.2. The Rejection-Based Stochastic Simulation Algorithm (RSSA). The Rejection-based Stochastic Simulation Algorithm (RSSA) is a novel exact stochastic simulation algorithm introduced in [10] and further improved in [7, 24–26]. RSSA constitutes the state of the art of exact stochastic simulation tailored for complex biochemical reactions with time-consuming propensity functions. The pseudocode of RSSA is provided in Algorithm 2.

RSSA computes for each reaction a lower bound and an upper bound of the propensity, which encompass all possible

values of reaction propensities over the fluctuation interval. Such bounds are derived by defining a *fluctuation interval* $\mathbb{X} = [\underline{\mathbf{x}}; \overline{\mathbf{x}}]$ for the system state $\mathbf{x}$ (step (2)), where

$$\begin{aligned} \underline{\mathbf{x}} &= \lfloor \mathbf{x} - \delta \mathbf{x} \rfloor, \\ \overline{\mathbf{x}} &= \lceil \mathbf{x} + \delta \mathbf{x} \rceil \end{aligned} \tag{8}$$

and $0 < \delta < 1$ is a simulation parameter whose value is usually between 0.1 and 0.2 (10–20% of the system state). Because reaction propensities with mass-action kinetics are monotonic functions, the propensity bounds of reaction R_i, for $i = 1, \ldots, m$, correspond to the interval $[a_i(\underline{\mathbf{x}}), a_i(\overline{\mathbf{x}})]$ (step (3)). These propensity bounds are recomputed only when the current system state is not anymore inside its fluctuation interval ($\mathbf{x} \notin \mathbb{X}$).

The selection of reaction firing in RSSA is composed of two steps. First, it selects a candidate reaction R_j with probability $a_j(\overline{\mathbf{x}})/a_0(\overline{\mathbf{x}})$ (step (7)), where $a_0(\overline{\mathbf{x}}) = \sum_{j=1}^{m} a_j(\overline{\mathbf{x}})$. The candidate reaction is then validated according to a rejection-based procedure that implements the toss of a biased coin with success probability $a_j(\mathbf{x})/a_j(\overline{\mathbf{x}})$ (steps (8)–(14)). To do this, the algorithm generates a random number r in $U(0, 1)$ and moves in one of the following three cases:

(1) If $r \leq a_j(\underline{\mathbf{x}})/a_j(\overline{\mathbf{x}})$, then R_j is accepted without computing its propensity $a_j(\mathbf{x})$ because $a_j(\underline{\mathbf{x}})/a_j(\overline{\mathbf{x}}) \leq a_j(\mathbf{x})/a_j(\overline{\mathbf{x}})$ (*quick accept*);

(2) If $r > a_j(\underline{\mathbf{x}})/a_j(\overline{\mathbf{x}})$, then reaction propensity $a_j(\mathbf{x})$ is computed and R_j is accepted if $r \leq a_j(\mathbf{x})/a_j(\overline{\mathbf{x}})$ (*slow accept*);

(3) If $r > a_j(\mathbf{x})/a_j(\overline{\mathbf{x}})$, then R_j is rejected (rejection).

After a reaction is accepted to fire, the state is updated and the algorithm moves to the next simulation iteration without updating the propensity bounds. Only for uncommon cases when state $\mathbf{x}$ moves out of its fluctuation interval does RSSA have to define a new fluctuation interval and update the propensity bounds. Numerical experiments show that propensity updates in RSSA are infrequent; hence their impact is very low during the computation [10]. This advantage is mitigated by the penalty paid to prepare a potential advancement of the system for a candidate reaction that is finally rejected to fire, but the overall performance is still considerably better than DM and other stochastic simulation algorithms.

2.3. The Forward Euler Algorithm. The forward Euler algorithm is the simplest example of *deterministic* simulation algorithm. Here the mass-action model is translated to a system of *ordinary differential equations* (ODEs, [7, 13]) and the dynamics provides an averaged behavior of the system.

Consider a biochemical reaction system with $S_1, \ldots, S_N$ species, $R_1, \ldots, R_M$ reactions defined according to (1). The mass-action deterministic rate constant k_j of R_j can be obtained from the stochastic one c_j as

$$k_j = \frac{c_j \left(N_A V\right)^{\text{Order}_j - 1}}{\prod_{i=1}^{N} \nu_{ij}!}, \tag{9}$$

where N_A indicates Avogadro's number, V is the volume where the reaction is taking place, and Order_j is the overall order of reaction R_j, which in mass-action is defined as the sum of the stoichiometric coefficients of reaction reactants. Finally, the ODE describing the evolution in terms of molar concentrations of species S_i, $i = 1, \ldots, N$, is

$$\frac{d\left[S_i\right]}{dt} = \sum_{j=1}^{M} \left(k_j \left(\nu'_{ij} - \nu_{ij} \right) \prod_{l=1}^{N} \left[S_l\right]^{\nu_{lj}} \right), \quad i = 1, \ldots, N, \tag{10}$$

where squared brackets are used to indicate molar concentration of species ($[S_i] = S_i/(N_A V)$, $i = 1, \ldots, N$). Once the model has been translated to a system of ODEs

$$\frac{d\left[\mathbf{X}\right]}{dt} = \mathbf{F}\left(t, \left[\mathbf{X}\right]\right), \tag{11}$$

a first-order approximation of the next system state can be computed by

$$\left[\mathbf{X}_{n+1}\right] = \left[\mathbf{X}_n\right] + h \cdot \mathbf{F}\left(t_n, \left[\mathbf{X}_n\right]\right), \tag{12}$$

where h is a user-defined discretization stepsize. The pseudocode of the Forward Euler algorithm is provided in Algorithm 3.

2.4. Runge-Kutta-Fehlberg Algorithm (RK45). The forward Euler algorithm introduced in the previous section is a first-order numerical method provided for didactic purposes. Several numerical methods have been introduced to increase the accuracy of deterministic simulations and to decrease simulation runtime. A popular algorithm is the *Runge-Kutta-Fehlberg algorithm (RK45)*. This algorithm represents the standard choice of several numerical integrators when the model does not exhibit stiffness. It is also often used in hybrid simulation strategies for the simulation of the fast part of the network. The reader can find a comprehensive guide to numerical methods of ODEs in [7, 13].

The Runge-Kutta Fehlberg method of fourth-order, also known as the RK45 method, updates the system state by

$$\left[\mathbf{X}_{n+1}\right] = \left[\mathbf{X}_n\right] + \frac{25}{216}K_1 + \frac{1408}{2565}K_3 + \frac{2197}{4104}K_4 - \frac{1}{5}K_5 \tag{13}$$

coupled with a fifth-order RK method

$$\left[\widetilde{\mathbf{X}}_{n+1}\right] = \left[\mathbf{X}_n\right] + \frac{16}{135}K_1 + \frac{6656}{12825}K_3 + \frac{28561}{56430}K_4 - \frac{9}{50}K_5 + \frac{2}{55}K_6. \tag{14}$$

Input: a system of ODEs $d[\mathbf{X}]/dt = \mathbf{F}(t, [\mathbf{X}])$ corresponding to a biochemical reaction system, the initial state $[\mathbf{X}(t_0)]$ of the system with species concentrations at time 0, the last time instant t_{end} to be simulated and the discretization stepsize h.
Output: a time series of states $[\mathbf{X}(t)]$, $t \in [t_0; t_{\text{end}}]$, providing the dynamics of the system in terms of molar concentrations with discretization stepsize h.
Pseudocode:
(0) initialize time $t = 0$ and state $[\mathbf{X}] = [\mathbf{X}(t_0)]$;
(1) `while` $t < t_{\text{end}}$ `do`
  (2) update $[\mathbf{X}] = [\mathbf{X}] + h \cdot \mathbf{F}(t, [\mathbf{X}])$;
  (3) update $t = t + h$;
(4) `end while`

Algorithm 3: Forward Euler method.

Given a system of ODEs as in (11) and a discretization stepsize h, the values of $K_1, \ldots, K_6$ are shared between (13) and (14) and they can be computed as

$$
\begin{aligned}
K_1 &= h \cdot \mathbf{F}\left(t_n, [\mathbf{X}_n]\right) \\
K_2 &= h \cdot \mathbf{F}\left(t_n + \frac{h}{4}, [\mathbf{X}_n] + \frac{1}{4}K_1\right) \\
K_3 &= h \cdot \mathbf{F}\left(t_n + \frac{3h}{8}, [\mathbf{X}_n] + \frac{3}{32}K_1 + \frac{9}{32}K_2\right) \\
K_4 &= h \cdot \mathbf{F}\left(t_n + \frac{12h}{13}, [\mathbf{X}_n] + \frac{1932}{2197}K_1 - \frac{7200}{2197}K_2 \right. \\
&\quad \left. + \frac{7296}{2197}K_3\right) \\
K_5 &= h \cdot \mathbf{F}\left(t_n + h, [\mathbf{X}_n] + \frac{439}{216}K_1 - 8K_2 + \frac{3680}{513}K_3 \right. \\
&\quad \left. - \frac{845}{4104}K_4\right) \\
K_6 &= h \cdot \mathbf{F}\left(t_n + \frac{h}{2}, [\mathbf{X}_n] - \frac{8}{27}K_1 + 2K_2 - \frac{3544}{2565}K_3 \right. \\
&\quad \left. + \frac{1859}{4104}K_4 - \frac{11}{40}K_5\right).
\end{aligned}
\tag{15}
$$

The fourth-order version of the algorithm provided in (13) is used to compute the dynamics of the system. The fifth-order scheme of (14), instead, is used to estimate the maximum local truncation error introduced in the simulated step:

$$
\Delta_{n+1} = \max\left(\frac{\left|\left[\widetilde{\mathbf{X}}_{n+1}\right] - \left[\mathbf{X}_{n+1}\right]\right|}{h}\right), \tag{16}
$$

where max provides the maximum value along the vector of truncation errors. The error estimate Δ_{n+1} is then compared to the error threshold ϵ_t specified by the user. When $\Delta_{n+1} \leq \epsilon_t$, the local truncation error is assumed to be smaller than the threshold, the state $[\mathbf{X}_{n+1}]$ is accepted, and the algorithm moves one step forward. Otherwise, the new state is not accepted and the next state is evaluated again using a different (smaller) value of h. In both cases, the value of h is updated as

$$
h_{n+1} = h_n \sigma \tag{17}
$$

$$
\sigma = \left(\frac{\epsilon_t}{2\Delta_{n+1}}\right)^{1/4} \approx 0.84\left(\frac{\epsilon_t}{\Delta_{n+1}}\right)^{1/4}. \tag{18}
$$

When the estimations $[\mathbf{X}_{n+1}]$ and $[\widetilde{\mathbf{X}}_{n+1}]$ agree to more significant digits than required, then σ becomes greater than 1 and h is increased. Equation (18) is derived from the general formula $\sigma = (\epsilon_t/\Delta_{n+1})^{1/p}$, which defines how to update the value of h of an adaptive one-step numerical method of order p, by considering the error estimate Δ_{n+1} and the user-defined threshold ϵ_t. The additional multiplicative factor of 0.84 is an empirical number commonly added in RK45 implementation to reduce the variability of h, because very high values of the stepsize increase the probability of repeating the next computed step.

The implementation of the algorithm is in Algorithm 4. The value of h is updated at each step starting from an user-provided initial value h_0. The next computed state of the system is accepted only when the estimate of the local truncation error Δ remains below the user-provided threshold ϵ_t. Steps (16)–(21) are additional steps added to the implementation in order to avoid very large modifications of h in a single step.

Even if the computation of a simulation iteration of the RK45 algorithm is more computational demanding than other nonadaptive Runge-Kutta implementation, the possibility of changing the value of h often dramatically decreases the simulation runtime.

2.5. Hybrid Rejection-Based Stochastic Simulation Algorithm (HRSSA). A drawback of approximate simulation is that *all* the model reactions are simulated according to the same approximate strategy. This is an issue when exact simulation is required at least for a small part of the reaction network, for example, when slow reactions are considered to model rare stochastic events. In such a case, a *hybrid* simulation strategy can be applied, which divides reactions into subsets that are simulated by different strategies at the same time [7, 14, 15].

An issue of this approach is the *synchronization* between the employed simulation strategies in order to guarantee the

Input: a system of ODEs $d[\mathbf{X}]/dt = \mathbf{F}(t, [\mathbf{X}])$ corresponding to a biochemical reaction system, the initial state $[\mathbf{X}(t_0)]$ of the system with species concentrations at time 0, the last time instant t_{end} to be simulated, an initial value for the discretization stepsize h_0 and a threshold for the maximum local truncation error ϵ_t.
Output: a time series of states $[\mathbf{X}(t)]$, $t \in [t_0; t_{\text{end}}]$, providing the dynamics of the system in terms of molar concentrations.
Pseudocode:

(0) initialize time $t = 0$, state $[\mathbf{X}] = [\mathbf{X}(t_0)]$ and discretization stepsize $h = h_0$;
(1) `while` $t < t_{\text{end}}$ `do`
  (2) compute $K_1 = h \cdot \mathbf{F}(t, [\mathbf{X}])$;
  (3) compute $K_2 = h \cdot \mathbf{F}(t + h/4, [\mathbf{X}] + (1/4)K_1)$;
  (4) compute $K_3 = h \cdot \mathbf{F}(t + 3h/8, [\mathbf{X}] + (3/32)K_1 + (9/32)K_2)$;
  (5) compute $K_4 = h \cdot \mathbf{F}(t + 12h/13, [\mathbf{X}] + (1932/2197)K_1 - (7200/2197)K_2 + (7296/2197)K_3)$;
  (6) compute $K_5 = h \cdot \mathbf{F}(t + h, [\mathbf{X}] + (439/216)K_1 - 8K_2 + (3680/513)K_3 - (845/4104)K_4)$;
  (7) compute $K_6 = h \cdot \mathbf{F}(t + h/2, [\mathbf{X}] - (8/27)\mathrm{K}_1 + 2K_2 - (3544/2565)K_3 + (1859/4104)K_4 - (11/40)K_5)$;
  (8) compute $[\mathbf{X}_{\text{new}}] = [\mathbf{X}] + (25/216)K_1 + (1408/2565)K_3 + (2197/4104)K_4 - (1/5)K_5$;
  (9) compute $[\widetilde{\mathbf{X}}_{\text{new}}] = [\mathbf{X}] + (16/135)K_1 + (6656/12825)K_3 + (28561/56430)K_4 - (9/50)K_5 + (2/55)K_6$;
  (10) compute $\Delta = \max(|[\widetilde{\mathbf{X}}_{\text{new}}] - [\mathbf{X}_{\text{new}}]|/h)$ according to (16);
  (11) `if` $(\Delta \leq \epsilon_t)$ `then`
    (12) update $[\mathbf{X}] = [\mathbf{X}_{\text{new}}]$;
    (13) update $t = t + h$;
  (14) `end if`
  (15) compute $\sigma = 0.84(\epsilon_t/\Delta)^{1/4}$;
  (16) `if` $(\sigma < 0.1)$ `then`
    (17) update $\sigma = 0.1$;
  (18) `end if`
  (19) `if` $(\sigma > 4)$ `then`
    (20) update $\sigma = 4$;
  (21) `end if`
  (22) update $h = h \cdot \sigma$;
(23) `end while`

Algorithm 4: The Runge-Kutta Fehlberg (RK45) algorithm.

exactness of the simulation of slow reactions. This is a fundamental aspect of hybrid simulation relying on the definition of the probability density function (pdf) of slow reactions. In fact, even though fast reactions can be safely simulated across slow reaction events, it is not always the case that slow reactions can be simulated regardless of what fast reactions are changing in the system. Actually the reaction propensity a_j of a slow reaction $R_j \in \mathscr{R}^{\text{slow}}$ may depend on species whose quantities are changed by fast reactions. For this reason, the reaction probability density function of (2) is not suitable to simulate the reaction subnetwork made of only the slow reactions of the system and it has to be extended to consider *time-varying transition propensities* [7, 14], which account for the fact that reaction propensities of slow reactions may change over time by the simulation of fast reactions.

According to [7, 14, 16–20], the pdf of the next firing of a slow reaction $R_j \in \mathscr{R}^{\text{slow}}$ finally becomes

$$P^{\text{slow}}(\tau, j \mid \mathbf{x}, t) = a_j(\mathbf{X}(t+\tau))\, e^{-\int_t^{t+\tau} a_0^{\text{slow}}(\mathbf{X}(t'))dt'}, \quad (19)$$

where

$$a_0^{\text{slow}}(\mathbf{x}) = \sum_{R_j \in \mathscr{R}^{\text{slow}}} a_j(\mathbf{x}). \quad (20)$$

The firing time τ of the next slow reaction R_j is thus obtained by solving the equation

$$\int_t^{t+\tau} a_0^{\text{slow}}\left(\mathbf{X}\left(t'\right)\right) dt' = -\ln(r), \quad (21)$$

where r is a random number from $U(0, 1)$. Solving (21) is computationally challenging because the state $X(t)$ is changed by fast reactions during time interval $[t, t + \tau]$. Therefore, the hybrid simulation has to evaluate the integral simultaneously with the simulation of fast reactions in order to correctly generate the next slow reaction event. Moreover, in practical implementation this step has to be necessarily approximated by a numerical method relying on an error threshold that introduces an additional approximation in the simulation of slow reactions. This step is so critical that the standard choice of several implementation available in literature, such as the hybrid algorithms implemented in COPASI [22], intentionally avoids the computation of the integral during the simulation by accepting some approximation also in the simulation of slow reactions.

The hybrid algorithm HRSSA is a novel hybrid simulation algorithm introduced in [21] to solve this problem. HRSSA is built on top of RSSA introduced in Section 2.2. In particular, the RSSA concept of fluctuation interval of the system

Input: The same as RSSA (Algorithm 2) together with the time granularity τ_{fast} used for the (approximate) simulation of fast reactions; the minimum amount $\gamma \in \mathbb{N}$ of molecules that has to be available for fast reactions; the minimum number of times $\theta \in \mathbb{N}$ that a fast reaction has to be applied, in average, within the time range of size τ_{fast}.
Output: the sets $\mathscr{R}^{\text{slow}}$ and $\mathscr{R}^{\text{fast}}$ of slow and fast reactions, respectively.
Pseudocode:

(0) $\mathscr{R}^{\text{slow}} := \emptyset$; $\mathscr{R}^{\text{fast}} := \emptyset$;
(1) `for each` reaction R_j
  (2) `if` $(a_j(\underline{\mathbf{x}})\tau_{\text{fast}} < \theta)$ `then`
    (3) Put reaction R_j in the set $\mathscr{R}^{\text{slow}}$;
  (4) `else if` ($\exists S_i$, modified by R_j with stoichiometric coefficient ν_{ij}, such that $\underline{x}_i < \gamma \cdot \nu_{ij}$) `then`
    (5) Put reaction R_j in the set $\mathscr{R}^{\text{slow}}$;
  (6) `else`
    (7) Put reaction R_j in the set $\mathscr{R}^{\text{fast}}$;
  (8) `end if`
(9) `end for`

Algorithm 5: The dynamic reaction partitioning of HRSSA (please refer to [21] for details).

state is widely used to provide an important computational advantage. HRSSA synchronizes the simulation of slow and fast reactions by avoiding the computation of the integral of (21) and applies an accurate dynamic partitioning of reactions without updating reaction classification at each simulation step.

HRSSA updates reaction partitioning only when the current system state does not fit anymore in its fluctuation interval (see (8)). This permits reducing the computational overhead without losing the accuracy of the classification. HRSSA considers both reaction propensities and the number of transformed molecules for reaction partitioning, by replacing real propensities with their lower bounds. The adoption of bounds instead of real propensities does not affect the accuracy of the classification. Conversely, the usage of lower bounds imposes more stringent constraints that tend to increase the number of reactions that are classified as slow (and therefore simulated without approximations). The pseudocode of the dynamic reaction partitioning of HRSSA is in Algorithm 5. The if clauses of steps (2) and (4) implement the two conditions on reaction propensities and number of transformed molecules, respectively.

After reaction partitioning in the two sets of slow and fast reactions, HRSSA computes the sum of upper propensity bounds $a_0^{\text{slow}}(\overline{\mathbf{x}})$ of slow reactions as defined in (20). The firing time of a candidate slow reaction is then computed as

$$\tau = -\frac{\ln(r)}{a_0^{\text{slow}}(\overline{\mathbf{x}})}, \tag{22}$$

where r is a random number in $U(0, 1)$.

Under the hypothesis that the system state will remain inside its fluctuation interval in $[t, t + \tau]$, we can consider $a_0^{\text{slow}}(\overline{\mathbf{x}})$ not dependent on time over $[t, t+\tau]$ and this allows us to simulate fast reactions over this interval without taking any side effect on the application of slow reactions into account. This is because (22) remains valid, regardless of the action of fast reactions, as long as the current system state respects its bounds.

After the simulation of fast reactions for the time interval $[t, t + \tau]$, a slow reaction is chosen and validated to fire by a rejection test, according to the RSSA simulation strategy. To guarantee the exact simulation of slow reactions (the proof is in [21]), the simulation of fast reactions is required to happen in a feasible system state. Therefore, every time the system state exits from its bounds the simulation is stopped and the fluctuation interval is updated. The pseudocode of HRSSA is provided in Algorithm 6.

3. Results and Discussion

HSimulator is a cross-platform simulation software written in Java, offering a suite of state-of-the-art stochastic, deterministic, and hybrid simulation strategies for mass-action well-stirred biochemical reaction systems. Multi-compartmental modeling is natively supported allowing the definition of nested reaction volumes via a customary text-based representation of reactions and compartments. The multithreading implementation allows scaling very well when computing averaged model dynamics obtained by running multiple stochastic simulations in parallel. The software has been designed to run in three user-scenarios: (i) via command-line, for example, as batch jobs or as part of wider modeling terminal scripts; (ii) programmatically embedded via its API in custom applications [27] or within MATLAB/Octave/R/Mathematica projects; (iii) as a self-standing simulation environment with a graphical user interface (GUI) for biomedical system modeling. For the three usage scenarios extensive documentation is available at the software web page (http://www.cosbi.eu/research/prototypes/hsimulator.). The GUI assists the modeler while defining the reactions with syntax highlighting of the typed entities and quick graphical access to simulation parameters. The simulation environment

Input: The same as RSSA (Algorithm 2) together with the time granularity τ_{fast} used for the (approximate) simulation of fast reactions; the minimum amount $\gamma \in \mathbb{N}$ of molecules that has to be available for fast reactions; the minimum number of times $\theta \in \mathbb{N}$ that a fast reaction has to be applied, in average, within the time range of size τ_{fast}.
Output: a time series of states $\mathbf{X}(t)$, $t \in [t_0; t_{\text{end}}]$, providing the dynamics of the system.
Pseudocode:

(0) $t \coloneqq t_0$;
(1) `while` $t < t_{\text{end}}$ `do`
  (2) Define the fluctuation interval $\mathbb{X} = [\underline{\mathbf{x}}; \overline{\mathbf{x}}]$ according to (8);
  (3) For each reaction R_i, $i = 1, \ldots, m$, compute propensity bounds $a_i(\underline{\mathbf{x}})$ and $a_i(\overline{\mathbf{x}})$;
  (4) Update reaction partitioning (sets $\mathscr{R}^{\text{slow}}$ and $\mathscr{R}^{\text{fast}}$) by applying the algorithm in Algorithm 5 according to input parameters γ, θ and τ_{fast};
  (5) Compute $a_0^{\text{slow}}(\overline{\mathbf{x}})$ according to (20);
  (6) updateNeeded $\coloneqq$ false;
  (7) `while` $(t < t_{\text{end}} \wedge \neg$updateNeeded$)$ `do`
    (8) $\tau \coloneqq -\ln(r)/a_0^{\text{slow}}(\overline{\mathbf{x}})$, where r is a random number in $U(0, 1)$;
    (9) Compute $\mathbf{X}(t + \tau')$ by simulating fast reactions ($\mathscr{R}^{\text{fast}}$), at time steps of maximum length τ_{fast}, according to an approximate algorithm (either stochastic or deterministic), where τ' is a value $\tau' \leq \tau$ such that $\mathbf{X}(t + \tau') \in \mathbb{X}$;
    (10) `if` $(\tau' = \tau)$ `then`
      (11) Select a candidate slow reaction $R_j \in \mathscr{R}^{\text{slow}}$ by applying RSSA steps (7)–(14) in Algorithm 2;
      (12) `if` R_j is accepted by RSSA `then` update $\mathbf{X}(t + \tau')$ computed at step (9) by applying R_j;
      (13) `if` $(\mathbf{X}(t + \tau') \notin \mathbb{X})$ `then` updateNeeded $\coloneqq$ true;
    (14) `else`
      (15) updateNeeded $\coloneqq$ true;
    (16) `end if`
    (17) $t \coloneqq t + \tau'$;
  (18) `end while`
(19) `end while`

Algorithm 6: The hybrid rejection-based stochastic simulation algorithm (HRSSA, please refer to [21] for details). In HSimulator, step (9) is implemented by a deterministic Runge-Kutta numerical method.

is completed by an interactive viewport to explore and understand the modeled system dynamics (see Figure 1) which can be then exported as Excel spreadsheets.

To evaluate the performance of the HSimulator implementation, a set of benchmarks have been carried out considering the state-of-the-art simulator COPASI [22], version 4.19 (build 140). All the simulations have been run in the same conditions on a 64 bit macOS Sierra MacBook Pro, with 16 GB of RAM. Six models have been simulated according to 6 different algorithms covering all the simulation strategies (stochastic, deterministic, and hybrid). Namely, we considered the Direct Method (implemented in both COPASI and HSimulator) and RSSA (only implemented in HSimulator) for exact stochastic simulation, LSODA (implemented in COPASI) and RK45 (implemented in HSimulator) for deterministic simulation, and Hybrid Runge-Kutta (implemented in COPASI) and HRSSA (implemented in HSimulator) for hybrid simulation. Benchmarks have been computed by averaging the runtime of 20 simulations. The simulation parameters for each algorithm are in Table 1 and they have been chosen to preserve as much as possible the reliability of comparisons.

The Hybrid Runge-Kutta algorithm provided by COPASI combines two fast simulation strategies available in literature to simulate slow and fast reactions (NRM combined with the 4th order Runge-Kutta method [7, 13]). Reaction partitioning is computed dynamically during the simulation according to a threshold which provides the maximum number of reactant molecules of a slow reaction. Moreover, the simulation of slow and fast reactions is synchronized without computing the integral of (21) by approximating reaction probabilities of slow reactions as constant during one stochastic step. This solution introduces an error in the simulation of slow reactions, but makes the computation faster. HRSSA resulted to be faster than this algorithm in all the simulation cases we considered, even if HRSSA does not apply such approximation in simulating slow reactions.

We note that the latest version of COPASI (version 4.19 build 140) considered in our benchmarks provides other two implementation cases of hybrid simulation algorithms (Hybrid LSODA and Hybrid RK45). Here we considered Hybrid Runge-Kutta because its implementation is closer to the one of HRSSA. However, we obtained very similar simulation runtimes also by running our benchmarks with

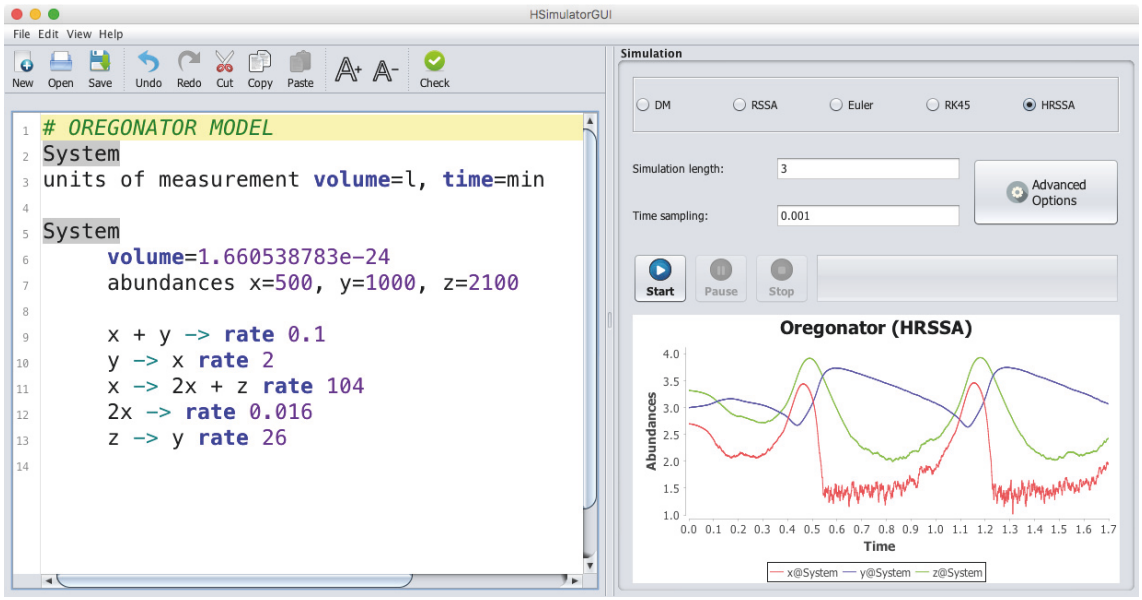

Figure 1: The HSimulator graphical user interface allows easily defining, even complex, biological models in terms of customary text-based reactions. The appropriate simulation strategy can be selected and parameterised as well according to the model, although the HRSSA method offers the best hybrid simulator. The figure shows the Oregonator model [8] simulated by the HRSSA algorithm [21]. The plot shows variable abundances in logarithmic scale to illustrate the switch between the deterministic simulation of fast reactions (high abundances) and the exact stochastic simulation of slow reactions (low abundances).

Table 1: Simulation parameters used for the benchmarks of the compared tools.

Algorithm	Software	Parameters
DM	HSimulator	No parameters needed
DM	COPASI	No parameters needed
RSSA	HSimulator	$\delta = 0.1$
LSODA	COPASI	Relative and absolute thresholds: 10^{-3}
RK45	HSimulator	Initial stepsize: 10^{-2} Error threshold: 10^{-3}
Hybrid Runge-Kutta	COPASI	Maximum number of reactant molecules for slow reactions: 100 RK4 stepsize: 10^{-2}
HRSSA	HSimulator	$\delta = 0.1$; $\tau_{\text{fast}} = 10^{-2}$; $\gamma = 100$; $\theta = 10$

Hybrid LSODA (data not shown). Hybrid RK45 has been not considered here, because this simulation strategy uses a static reaction partitioning whereas HSimulator and the compared methods use a dynamical approach allowing for a fair benchmark between strategies. We also have to acknowledge that COPASI is not the only simulation software available in the literature. In the present work we considered COPASI due to its high popularity and to allow, as much as possible, a fair comparison with HSimulator. In fact, COPASI provides a way of specifying the biochemical system that is very close to the one of HSimulator. Snoopy [28] is another promising simulation software that provides different simulation strategies for hybrid models, including a reinterpretation of HRSSA.

Simulation benchmarks are provided in Table 2. To allow the reproducibility of the presented results, all the considered models are provided with HSimulator both implemented for HSimulator and for COPASI. Simulation benchmarks comprehend two biological models (the MAPK cascade and the Gemcitabine mechanism of action) to test simulation strategies in real modeling applications plus two theoretical models (the fully connected model and the multiscale model) considered with two different parameterisations each, to evaluate the performance of simulation algorithms under specific conditions. The fully connected model has been considered to evaluate the performance of simulation algorithms when a mass-action biochemical reaction network of reactions that are all slow (MCM $i = 20$) or fast (MCM $i = 2000$) is simulated. Conversely, the multiscale model allowed testing the intermediate condition, where the biochemical network can be divided into two subnetworks working at different time scales. This scenario has been specifically considered to test hybrid simulation strategies. The biochemical network of the multiscale model has been generated two times to test the scalability of simulation algorithms (MSM $(10, 50)$, network of 60 species and 140 reactions; MSM $(20, 100)$, network of 120 species and 480 reactions, see Section 3.4

Table 2: Simulation running times of HSimulator and of the state-of-the-art simulator COPASI. Except for the deterministic simulation, HSimulator is demonstrated to be faster in all the considered scenarios (FCM indicates the fully connected model; MSM indicates the multiscaled model). The considered models were simulated for 150 time units (MAPK and fully connected model), 12 time units (Gemcitabine), and 10 time units (multiscaled model).

Algorithm	MAPK	Gemcitabine	FCM $i = 20$	FCM $i = 2000$	MSM (10, 50)	MSM (20, 100)
COPASI - DM	2.39 sec.	0.88 sec.	5.09 sec.	415.04 sec.	28.87 sec.	262.1 sec.
HSimulator - DM	2.23 sec. (−6%)	0.7 sec. (−12%)	1.61 sec. (−68%)	168.75 sec. (−59%)	11.0 sec. (−62%)	120.84 sec. (−54%)
HSimulator - RSSA	2.03 sec. (−15%)	0.66 sec. (−25%)	1.10 sec. (−78%)	77.46 sec. (−81%)	4.25 sec. (−85%)	37.23 sec. (−86%)
COPASI - LSODA	0.06 sec.	0.02 sec.	0.56 sec.	0.08 sec.	0.08 sec.	0.21 sec.
HSimulator - RK45	0.06 sec. (+0%)	0.04 sec. (+100%)	0.001 sec. (−99%)	0.001 sec. (−99%)	0.13 sec. (+63%)	0.52 sec. (+148%)
COPASI - Hyb. RK	0.21 sec	0.54 sec.	26.74 sec.	14.67 sec.	1.93 sec.	35.49 sec.
HSimulator - HRSSA	0.1 sec. (−52%)	0.38 sec. (−30%)	1.10 sec. (−96%)	0.8 sec. (−95%)	2.53 sec. (−31%)	5.89 sec. (−83%)

for details). The provided benchmarks seem to indicate that HSimulator is more scalable than COPASI with respect to the growing complexity of the model. This result is interesting, especially from an implementer's point of view, showing how a recent Java-based implementation (HSimulator), in the given conditions, may compare favourably with respect to C++ (COPASI).

3.1. The MAPK Cascade. The *Mitogen-Activated Protein Kinase cascade (MAPK cascade)* is one of the most important and intensively studied signaling pathways [29]. The MAPK cascade is at the heart of a molecular-signaling network that governs the growth, proliferation, differentiation, and survival of many cell types. Moreover, the MAPK pathway is deregulated in various diseases, ranging from cancer to immunological, inflammatory, and degenerative syndromes, and thus represents also an important drug target [30].

The MAPK cascade is a series of three protein kinases that are responsible for cell response to growth factors. The signal $E1$ activates MAPKKK by phosphorylation, which in turn activates MAPKK. Once activated, MAPKK activates MAPK. When $E1$ is added to the system, the output of activated MAPK increases rapidly. By removing the signal $E1$, the output level of activated MAPK reverts back to zero. Reverse reactions are triggered by the signal $E2$ and by the MAPK phosphatases KKpase and Kpase. The reaction-based representation is

$$\begin{aligned}
R_1 &: \mathrm{KKK} + E1 \longrightarrow \mathrm{KKKp} + E1\\
R_2 &: \mathrm{KKKp} + E2 \longrightarrow \mathrm{KKK} + E2\\
R_3 &: \mathrm{KK} + \mathrm{KKKp} \longrightarrow \mathrm{KKp} + \mathrm{KKKp}\\
R_4 &: \mathrm{KKp} + \mathrm{KKpase} \longrightarrow \mathrm{KK} + \mathrm{KKpase}\\
R_5 &: \mathrm{KKp} + \mathrm{KKKp} \longrightarrow \mathrm{KKpp} + \mathrm{KKKp}\\
R_6 &: \mathrm{KKpp} + \mathrm{KKpase} \longrightarrow \mathrm{KKp} + \mathrm{KKpase}\\
R_7 &: \mathrm{K} + \mathrm{KKpp} \longrightarrow \mathrm{Kp} + \mathrm{KKpp}\\
R_8 &: \mathrm{Kp} + \mathrm{Kpase} \longrightarrow \mathrm{K} + \mathrm{Kpase}\\
R_9 &: \mathrm{Kp} + \mathrm{KKpp} \longrightarrow \mathrm{Kpp} + \mathrm{KKpp}\\
R_{10} &: \mathrm{Kpp} + \mathrm{Kpase} \longrightarrow \mathrm{Kp} + \mathrm{Kpase},
\end{aligned} \tag{23}$$

where KKK denotes MAPKKK, KK denotes MAPKK, and K denotes MAPK and tags p and pp indicate phosphorylation and double phosphorylation, respectively.

A simulation benchmark of the MAPK cascade is in Table 2 (first column). In the computed benchmark, the model has been simulated for 150 time units with all stochastic reaction constants equal to 0.001. The initial abundances have been set to $E1 = 2000$, $E2 = 2000$, $\mathrm{K} = 200000$, $\mathrm{KK} = 200000$, and $\mathrm{KKK} = 20000$. All the other species are initialized with 0 molecules.

3.2. The Gemcitabine Model. Gemcitabine is a drug for non-small-cell lung cancer, pancreatic cancer, bladder cancer, and breast cancer. Here we will consider a simplified model of its mechanism of action according to [23].

Gemcitabine (dFdC, see Box 1) is transported from plasma into the cell through the cell membrane ($\mathrm{dFdC_{out}} \rightarrow$ dFdC). Gemcitabine can be deaminated by CDA in the cytoplasm and in the extracellular environment leading to dFdU. Both dFdC and dFdU can be phosphorylated by dCK. Monophosphorylated Gemcitabine can be deaminated by dCMPD, whereas dCMPD is inhibited by $\mathrm{dFdC_{TP}}$. Alternatively, it is further phosphorylated. The Gemcitabine triphosphates $\mathrm{dFdC_{TP}}$ and $\mathrm{dFdU_{TP}}$ compete with the natural nucleoside triphosphate dCTP for incorporation into nascent DNA chain and inhibit DNA synthesis, thus blocking cell proliferation in the early DNA synthesis phase.

A simulation benchmark of the Gemcitabine model is in Table 2 (second column, simulation length 12 time units). In the computed benchmark, stochastic reaction constants are [23]: $c_1 = 9.97$, $c_2 = 2.61e^{-4}$, $c_3 = 4.72e^{-6}$, $c_4 = 0.05$, $c_5 = 0$, $c_6 = 0.001$, $c_7 = 0.087$, $c_8 = 2.37$, $c_9 = 0.21$, $c_{10} = 2.52$, $c_{11} = 1.45$, $c_{12} = 9.68e^{-4}$, $c_{13} = 5.6e^{-5}$, $c_{14} = 0.078$, $c_{15} = 0.004$, $c_{16} = 0.16$, $c_{17} = 9.05e^{-5}$, $c_{18} = 4.76e^{-4}$, $c_{19} = 4.559e^{-6}$, $c_{20} = 1000$, $c_{21} = 0.05$, $c_{22} = 25.20$, $c_{23} = 1e^{-5}$, $c_{24} = 0.1$, $c_{25} = 1e^{-5}$, $c_{26} = 0.1$, $c_{27} = 1e^{-7}$, $c_{28} = 0.1$, $c_{29} = 0.05$, $c_{30} = 7.37e^{-4}$, and $c_{31} = 0.5$. Initially all the variables have been set to 0 except for $\mathrm{dFdC_{out}} = 100000$, $\mathrm{dCK} = 1000$, $\mathrm{RR} = 1000$, $\mathrm{dCMPD} = 1000$, and $\mathrm{CDP} = 2000$.

3.3. The Fully Connected Model. The fully connected model is a theoretical model that consists of n species $S_1, S_2, \ldots, S_n$

R_1 : $dFdC_{out} \longrightarrow dFdC$
R_2 : $dFdC \longrightarrow dFdC_{out}$
R_3 : $dFdC_{out} \longrightarrow dFdU$
R_4 : $dFdU \longrightarrow dFdU_{out}$
R_5 : $dFdC \longrightarrow dFdU$
R_6 : $dFdC + dCK \longrightarrow dFdC_{MP} + dCK$
R_7 : $dFdC_{MP} \longrightarrow dFdC$
R_8 : $dFdC_{MP} \longrightarrow dFdC_{DP}$
R_9 : $dFdC_{DP} \longrightarrow dFdC_{MP}$
R_{10} : $dFdC_{DP} \longrightarrow dFdC_{TP}$
R_{11} : $dFdC_{TP} \longrightarrow dFdC_{DP}$
R_{12} : $dFdU + dCK \longrightarrow dFdU_{MP} + dCK$
R_{13} : $dFdU_{MP} \longrightarrow dFdU$
R_{14} : $dFdU_{MP} \longrightarrow dFdU_{DP}$
R_{15} : $dFdU_{DP} \longrightarrow dFdU_{MP}$
R_{16} : $dFdU_{DP} \longrightarrow dFdU_{TP}$
R_{17} : $dFdU_{TP} \longrightarrow dFdU_{DP}$
R_{18} : $dFdC \longrightarrow dFdU$
R_{19} : $dFdC_{MP} + dCMPD \longrightarrow dFdU_{MP} + dCMPD$
R_{20} : $\emptyset \longrightarrow CDP$
R_{21} : $CDP + RR \longrightarrow dCDP + RR$
R_{22} : $dCDP \longrightarrow dCTP$
R_{23} : $dFdC_{DP} + RR \longrightarrow dFdC_{DP} : RR$
R_{24} : $dFdC_{DP} : RR \longrightarrow dFdC_{DP} + RR$
R_{25} : $dCTP + dCK \longrightarrow dCTP : dCK$
R_{26} : $dCTP : dCK \longrightarrow dCTP + dCK$
R_{27} : $dFdC_{TP} + dCMPD \longrightarrow dFdC_{TP} : dCMPD$
R_{28} : $dFdC_{TP} : dCMPD \longrightarrow dFdC_{TP} + dCMPD$
R_{29} : $dFdC_{TP} \longrightarrow dFdC_{TP} : DNA$
R_{30} : $dFdU_{TP} \longrightarrow dFdU_{TP} : DNA$
R_{31} : $dCTP \longrightarrow dCTP : DNA$

Box 1: Reactions of the Gemcitabine model [23].

and a set of reactions that reversibly convert each species into each other of the form

$$S_i \longleftrightarrow S_j, \quad i, j = 1, \ldots, n, \; i \neq j. \tag{24}$$

The fully connected model is useful to evaluate the scalability of algorithms (variation in runtime with growing values of n). Moreover, if we fix initial values of the species and stochastic reaction constants, then we can evaluate the performance of a simulation algorithm when it simulates a system that remains in a steady state condition for the entire length of the simulation.

Here we use the fully connected model starting from two different steady state conditions to quantify simulation runtime of a reaction network composed by only slow or fast reactions. These benchmarks are provided in Table 2, third and fourth columns. In both cases, the model has been generated with $n = 20$, that is, with 20 variables and 380 reactions. All stochastic reaction constants have been set to 1. In the first benchmark (MCM $i = 20$), the initial state of all model variables has been set to 20 in order to create a network of slow reactions. In the second one (MCM $i = 2000$), instead, the initial state of all model variables has been set to 2000 in order to create a network of fast reactions. In both conditions, simulation algorithms implemented in HSimulator were the fastest (simulation length 150 time units).

3.4. The Multiscaled Model. The multiscaled model is a theoretical model with a reaction network that can be divided into two subnetworks working at different time scales. It has been specifically considered to test hybrid simulation strategies because it allows testing the intermediate condition (network made of both fast and slow reactions) between the two scenarios previously considered with the fully connected model (network made of only slow or fast reactions). The first subnetwork of the model is given by a fully connected model of n_{fast} species $S_1, \ldots, S_{n_{\text{fast}}}$ modified by a set of fast reactions. The model is then extended with n_{slow} species $s_1, \ldots, s_{n_{\text{slow}}}$ modified by a set of n_{slow} reactions of the form

$$S_i + s_j \longrightarrow s_k, \quad j = 1, 2, \ldots, n_{\text{slow}}, \tag{25}$$

where S_i and s_k are random species such that $S_i \in \{S_1, \ldots, S_{n_{\text{fast}}}\}$ and $s_k \in \{s_1, \ldots, s_{n_{\text{slow}}}\}$. The stochastic constant rates of fast reactions are some order of magnitude bigger than the rates of the slow reactions to emphasize the difference of speed between the two subnetworks.

The fully connected model has been considered to evaluate the performance of simulation algorithms when a mass-action biochemical reaction network of reactions that are all slow (MCM $i = 20$) or fast (MCM $i = 2000$) is simulated. Conversely, the multiscale model allowed testing the intermediate condition, where the biochemical network can be divided into two subnetworks working at different time scales.

In Table 2 two benchmarks related to this model are provided. The first one (fifth column, MSM (10, 50)) is related to a model with $n_{\text{fast}} = 10$ and $n_{\text{slow}} = 50$ (60 species and 140 reactions), stochastic reaction constants equal to 10.0 (fast reactions) and 0.001 (slow reactions), initial values of fast species set to 2000, and initial values of slow species set to 20. The second benchmark (last column of the table, MSM (20, 100)) uses the same parameters of the first one except for $n_{\text{fast}} = 20$ and $n_{\text{slow}} = 100$ (120 species and 480 reactions). The provided benchmarks show that the gain in simulation runtime provided by HSimulator is scalable with respect to the complexity of the model (simulation length 10 time units).

4. Conclusions

We presented HSimulator, a state-of-the-art multithread Java simulator for mass-action well-stirred biochemical reaction systems. HSimulator provides a suite of published state-of-the-art simulation algorithms including, in the same package, the exact algorithm RSSA and the first publicly available implementation of its hybrid version HRSSA. The benchmarks in the paper show that the HSimulator implementation is often faster than the state-of-the-art simulator COPASI [22]. This could open new perspectives in computational systems biology, where often scientists have to balance the accuracy of their simulations with the need of considering large reaction networks modeling complex diseases or disorders.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors acknowledge Alessandro Pozzer for technical assistance and Dr. Giuditta Franco (Department of Computer Science, University of Verona, Italy) for her helpful suggestions and discussions. This work was partially supported by the PAT (Provincia Autonoma di Trento) Project S503/2015/664730 Zoomer.

References

[1] H. Kitano, "Computational systems biology," *Nature*, vol. 420, no. 6912, pp. 206–210, 2002.

[2] H. Kitano, "Systems biology: a brief overview," *Science*, vol. 295, no. 5560, pp. 1662–1664, 2002.

[3] L. Hood and D. Galas, "The digital code of DNA," *Nature*, vol. 421, no. 6921, pp. 444–448, 2003.

[4] A. P. Heath and L. E. Kavraki, "Computational challenges in systems biology," *Computer Science Review*, vol. 3, no. 1, pp. 1–17, 2009.

[5] C. Priami, "Algorithmic systems biology," *Communications of the ACM*, vol. 52, no. 5, pp. 80–88, 2009.

[6] C. Priami and M. J. Morine, *Analysis of Biological Systems*, Imperial College Press, London, UK, 2015.

[7] L. Marchetti, C. Priami, and V. H. Thanh, *Simulation Algorithms for Computational Systems Biology*, Springer International Publishing, Berlin, Germany, 2017.

[8] D. T. Gillespie, "Exact stochastic simulation of coupled chemical reactions," *The Journal of Physical Chemistry C*, vol. 81, no. 25, pp. 2340–2361, 1977.

[9] M. A. Gibson and J. Bruck, "Efficient exact stochastic simulation of chemical systems with many species and many channels," *The Journal of Physical Chemistry A*, vol. 104, no. 9, pp. 1876–1889, 2000.

[10] V. H. Thanh, C. Priami, and R. Zunino, "Efficient rejection-based simulation of biochemical reactions with stochastic noise and delays," *The Journal of Chemical Physics*, vol. 141, no. 13, Article ID 134116, 2014.

[11] D. T. Gillespie, "The Chemical Langevin equation," *The Journal of Chemical Physics*, vol. 113, no. 1, pp. 297–306, 2000.

[12] D. T. Gillespie, "Approximate accelerated stochastic simulation of chemically reacting systems," *The Journal of Chemical Physics*, vol. 115, no. 4, pp. 1716–1733, 2001.

[13] J. C. Butcher, *Numerical Methods for Ordinary Differential Equations*, John Wiley & Sons, New York, NY, USA, 2nd edition, 2008.

[14] J. Pahle, "Biochemical simulations: stochastic, approximate stochastic and hybrid approaches," *Briefings in Bioinformatics*, vol. 10, no. 1, pp. 53–64, 2009.

[15] M. Bentele and R. Eils, "General stochastic hybrid method for the simulation of chemical reaction processes in cells," in *Computational Methods in Systems Biology*, V. Danos and V. Schachter, Eds., vol. 3082 of *Lecture Notes in Computer Science*, Springer, Berlin, Germany, 2005.

[16] M. Herajy and M. Heiner, "Hybrid representation and simulation of stiff biochemical networks," *Nonlinear Analysis: Hybrid Systems*, vol. 6, no. 4, pp. 942–959, 2012.

[17] R. Irizarry, "Stochastic simulation of population balance models with disparate time scales: hybrid strategies," *Chemical Engineering Science*, vol. 66, no. 18, pp. 4059–4069, 2011.

[18] M. Griffith, T. Courtney, J. Peccoud, and W. H. Sanders, "Dynamic partitioning for hybrid simulation of the bistable HIV-1 transactivation network," *Bioinformatics*, vol. 22, no. 22, pp. 2782–2789, 2006.

[19] H. Salis and Y. Kaznessis, "Accurate hybrid stochastic simulation of a system of coupled chemical or biochemical reactions," *The Journal of Chemical Physics*, vol. 122, no. 5, Article ID 054103, 2005.

[20] E. L. Haseltine and J. B. Rawlings, "On the origins of approximations for stochastic chemical kinetics," *The Journal of Chemical Physics*, vol. 123, no. 16, Article ID 164115, 2005.

[21] L. Marchetti, C. Priami, and V. H. Thanh, "HRSSA—efficient hybrid stochastic simulation for spatially homogeneous biochemical reaction networks," *Journal of Computational Physics*, vol. 317, pp. 301–317, 2016.

[22] S. Hoops, R. Gauges, C. Lee et al., "COPASI—a COmplex PAthway SImulator," *Bioinformatics*, vol. 22, no. 24, pp. 3067–3074, 2006.

[23] O. Kahramanoğullari, G. Fantaccini, P. Lecca, D. Morpurgo, and C. Priami, "Algorithmic modeling quantifies the complementary contribution of metabolic inhibitions to gemcitabine efficacy," *PLoS ONE*, vol. 7, no. 12, Article ID e50176, 2012.

[24] V. H. Thanh, R. Zunino, and C. Priami, "On the rejection-based algorithm for simulation and analysis of large-scale reaction networks," *The Journal of Chemical Physics*, vol. 142, no. 24, Article ID 244106, 2015.

[25] V. H. Thanh and C. Priami, "Simulation of biochemical reactions with time-dependent rates by the rejection-based algorithm," *The Journal of Chemical Physics*, vol. 143, no. 5, Article ID 054104, 2015.

[26] V. H. Thanh, R. Zunino, and C. Priami, "Efficient constant-time complexity algorithm for stochastic simulation of large reaction networks," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 14, no. 3, pp. 657–667, 2017.

[27] R. Lombardo and C. Priami, "Graphical modeling meets systems pharmacology," *Gene Regulation and Systems Biology*, vol. 11, pp. 1–10, 2017.

[28] M. Herajy, F. Liu, C. Rohr, and M. Heiner, "Snoopy's hybrid simulator: a tool to construct and simulate hybrid biological models," *BMC Systems Biology*, vol. 11, no. 71, 2017.

[29] L. Chang and M. Karin, "Mammalian MAP kinase signalling cascades," *Nature*, vol. 410, no. 6824, pp. 37–40, 2001.

[30] R. J. Orton, O. E. Sturm, V. Vyshemirsky, M. Calder, D. R. Gilbert, and W. Kolch, "Computational modelling of the receptor-tyrosine-kinase-activated MAPK pathway," *Biochemical Journal*, vol. 392, no. 2, pp. 249–261, 2005.