Hybrid Simulation of Biochemical Systems Using Hybrid Adaptive Petri Nets

Hongkun Yang
Department of Computer Science
Tsinghua University, China
yang.hongk@gmail.com

Chuang Lin
Department of Computer Science
Tsinghua University, China
clin@tsinghua.edu.cn

Quanlin Li
Department of Industrial Engineering
Tsinghua University, China
liquanlin@tsinghua.edu.cn

ABSTRACT

Due to the heterogeneity of many real biochemical systems, stochastic simulation methods do not scale well as systems become more complex and larger, whereas approximations provided by continuous models fail to capture the stochastic behavior of molecular species at very low numbers. A hybrid simulation method is a natural idea to resolve this dilemma. In this paper, we propose a novel notion of Petri net called hybrid adaptive Petri net (HAPN), which is a unified framework to conveniently incorporate ordinary differential equations (ODEs), stochastic models, *static* hybrid and *adaptive* hybrid models. By exploring the mutual dependence of transitions, we make an improvement on the hybrid simulation algorithm and achieve a substantial saving on computational cost. We implement an HAPN simulator on MATLAB and employ the improved algorithm in the simulator. Two numerical examples are used to evaluate the accuracy and efficiency of our improved algorithm.

Categories and Subject Descriptors

J.3 [**LIFE AND MEDICAL SCIENCES**]: Biology and genetics; D.2.2 [**Design Tools and Techniques**]: Petri nets; I.6.1 [**SIMULATION AND MODELING**]: Simulation Theory

Keywords

hybrid adaptive Petri nets, hybrid simulation, bacteriophage T7

1. INTRODUCTION

Over the past decades, the vast information gathered from large-scale biotechnologies requires effective computational models which are capable of integrating and analyzing the data concerning the behavior and relationships of various biological elements. There is sufficient evidence showing that randomness is the hallmark of biochemical processes which involve molecular species at very low numbers [3, 24, 11]. With these observations, stochastic chemical kinetics, which captures the stochastic nature of reactions involving dilute chemical species, has received a wide interest [11, 1, 21, 20, 9].

Many numerical algorithms and tools have been designed for the stochastic solution of well-mixed chemical kinetic systems. These methods can be traced back to the work of Gillespie [7], who outlined two variants of the stochastic simulation algorithm, the direct method and the first reaction method. Using two special data structures and efficient use of random numbers, Gibson and Bruck developed the next reaction method [6], another variant of the stochastic simulation algorithm.

However, many real biochemical systems are heterogeneous: a system (e.g. a genetic network) can have species with low copy numbers (e.g. a gene) and species with high copy numbers (e.g. proteins), and thus it includes reactions whose rates differ by several orders of magnitude. Owing to the heterogeneity of a biochemical system, the stochastic simulation algorithm is computationally expensive for the large populations, whereas approximations using deterministic models fail to capture the stochastic behavior that is essential to small populations. An intuitive idea to resolve this dilemma is using a hybrid model which differently treat reactions whose rates are of different orders of magnitude, in hopes that the hybrid model will speed up simulations and preserve the accuracy. This is one motivation of our work. In addition, we also note that a unified framework of different models (stochastic, deterministic and hybrid) is of great value to biological scientists, since in this framework, issues of little biological importance are removed and they can focus on modeling and analyzing biological systems.

In this paper, we propose a novel Petri nets formalism, hybrid adaptive Petri nets, to address these issues. We call our Petri nets formalism *hybrid adaptive Petri nets*, because in an HAPN, transitions have two firing modes: *continuous* and *discrete*, and they can adaptively change their behavior according to the marking. We furnish HAPNs with the partition function which indicates the behavior (*continuous* or *discrete*) of transitions. With the partition function, HAPNs can incorporate various types of models such as ODEs, stochastic, *static* hybrid and *adaptive* hybrid models.

The main contributions of this paper are twofold. First, we present the HAPN, a unified framework suitable for various types of models. Second, by exploring the mutual dependence of transitions in *discrete* mode, we make an improvement on the hybrid simulation algorithm and achieve a

VALUETOOLS 2009, October 20-22, Pisa, Italy

DOI 10.4108/ICST.VALUETOOLS2009.7753

substantial saving on computational cost. We implement a universal HAPN simulation platform on MATLAB and employ the improved algorithm in the platform. The platform is able to simulate not only biochemical networks but also general HAPN models.

Using HAPN, we show that when the biological system under study is heterogeneous, the adaptive hybrid modeling is more accurate than the static hybrid modeling, and more efficient than the exact stochastic modeling.

This paper is organized as follows. In the next section we give an overview on related work. Section 3 briefly introduces the necessary biochemical and mathematical background. In Section 4 we propose the concept and properties of HAPNs. In Section 5 we use HAPNs to model biological systems. Section 6 provides a simulation study of HAPNs, including the improvement of the hybrid simulation algorithm. Numerical examples are given in Section 7 to evaluate the accuracy and efficiency of our improved algorithm. The final section concludes the paper and prospects for future work.

2. RELATED WORK

In recent years, Petri nets have been widely used in the field of systems biology. According to the classification of [4], Petri nets [12, 16, 22] are categorized as computational models which qualitatively model and analyze biological systems. Moreover, stochastic Petri nets [9], timed Petri nets [17], colored Petri nets [5], hybrid functional Petri nets [13, 18] and our work, hybrid adaptive Petri nets serve as bridges between computational models and mathematical models. Compared to other Petri net formalisms, hybrid adaptive Petri nets are endowed with the capability to integrate various types of models.

In this paper, we make an improvement on the hybrid simulation algorithm. In the language of Petri nets, a hybrid simulation algorithm has three elements: the partition process, simulation of transitions in *continuous* mode, and simulation of transitions in *discrete* mode.

The partition scheme can be static [15] or adaptive [11, 1, 21, 20]. The criteria for partitioning is based on the marking, the rates of transitions, or a combination of the two [11, 1, 21, 20, 15]. Our algorithm uses an adaptive partition scheme with criteria based on both the marking and the rates of transitions.

As an alternative to ODEs [11, 1, 15], the chemical Langevin equation can be employed to simulate transitions in *continuous* mode [21, 20]. Theoretically, the chemical Langevin is more accurate, but numerical ODE solvers are well developed and easy to integrate into the simulation algorithm. Furthermore, in this paper, numerical examples show that ODEs receive acceptable levels of accuracy.

Transitions in *discrete* mode can be simulated with the direct hybrid method [11, 1], the next reaction hybrid method [1, 21] and the technique of thinning [20]. The technique of thinning is restricted to the circumstance where firing rates of transitions are bounded with tight bounds. The next reaction hybrid method allows a general distribution of transition delays [21], but the direct hybrid method is more computationally efficient [11, 1]. Our work is an improvement on the direct hybrid method, which significantly promotes computational efficiency and attains satisfactory accuracy.

Table 1: Propensities of some reaction forms

Reaction	Propensity
$R_i \ : \ X_j \xrightarrow{k_i} *$	$k_i x_j$
$R_i \ : \ X_j + X_k \xrightarrow{k_i} *$	$k_i x_j x_k$
$R_i \ : \ 2X_j \xrightarrow{k_i} *$	$k_i \frac{x_j(x_j-1)}{2}$

3. PRELIMINARIES

3.1 Theory of Mass-action Stochastic Kinetics

Sufficient evidence has shown that biological processes are inherently stochastic, and in this paper we give the formulation of a biochemical system in a stochastic perspective.

A biochemical system can be viewed as a dynamical system which has m $(R_1, \ldots, R_m)$ coupled biochemical reactions involving n species $(X_1, \ldots, X_n)$ in a well-stirred volume. Generally speaking, a reaction R_i, $(i = 1, \ldots, m)$, can be written as

$$\sum_{j=1}^{n} \alpha_{i,j} X_j \xrightarrow{k_i} \sum_{j=1}^{n} \beta_{i,j} X_j \quad i = 1, 2, \ldots, m, \tag{1}$$

where $\alpha_{i,j}$ and $\beta_{i,j}$ are, respectively, the numbers of molecules of species X_j consumed and produced by reaction R_i, and k_i is the kinetic coefficient. In particular, if X_j is not involved in reaction R_i, then we will have $\alpha_{i,j} = \beta_{i,j} = 0$. The $m \times n$ stoichiometric matrix is $(v_{i,j})_{m \times n}$, where $v_{i,j} = \beta_{i,j} - \alpha_{i,j}$. We define $v_i = (v_{i,1}, \ldots, v_{i,n})$, which is an n-vector.

The state of the system at time t is given by $\boldsymbol{x}(t) = (x_1(t), \ldots, x_n(t))$, an n-vector of non-negative integers representing the number of molecules of each species. For the sake of convenience we use $\boldsymbol{x}$ in short of $\boldsymbol{x}(t)$, and denote by x_i the number of molecules of X_i.

The dynamics of the system are specified by the propensity of each reaction within it. The propensity of R_j, $\lambda_j(\boldsymbol{x})\mathrm{d}t$, is the probability that the reaction will occur during the interval $[t, t + \mathrm{d}t)$. $\lambda_j(\boldsymbol{x})\mathrm{d}t$ is a function of the state vector $\boldsymbol{x}$ by definition.

Although propensities of reactions can be computed using different rate laws, our work focuses on mass action kinetics. Under mass action kinetics Gillespie has shown that the propensity of a reaction is a function of the kinetic coefficient, the populations of the reactants and a combinatorial term capturing the number of reacting configurations [7]. Table 1 illustrates some examples of propensity functions, where $*$ indicates that products of a reaction are irrelevant to its propensity function.

Let $\{N_j(t), t \geq t_0\}$ be the stochastic process counting the occurrence of R_j during the time interval $[t_0, t]$, where t_0 is the initial time of the system and $N_j(t_0) = 0$. It can be shown that the time evolution equation for $\boldsymbol{x}(t)$ is [1]

$$\mathrm{d}\boldsymbol{x}(t) = \sum_{j=1}^{m} v_j \mathrm{d}N_j(t), \tag{2}$$

with some initial value $\boldsymbol{x}(t_0)$. Note that $N_j(t)$ is an inhomogeneous counting process such that

$$\mathbf{P}[N_j(t + \mathrm{d}t) - N_j(t) = 1 | \boldsymbol{x}(t)] = \lambda_j(\boldsymbol{x}(t))\mathrm{d}t$$

Equation (2) is seldom analytically tractable, but fortunately, there are exact algorithms to numerically simulate (2) [7, 6].

Digital Object Identifier: 10.4108/ICST.VALUETOOLS2009.7753
http://dx.doi.org/10.4108/ICST.VALUETOOLS2009.7753

3.2 Mathematics of the Hybrid Approximation Strategy

Under many practical circumstances, the exact simulation of a biochemical system turns out to be a formidable task since the number of reaction events is tremendously huge. To reduce the computational burden of stochastic simulations, many approximate simulation algorithms are proposed.

In one type of these approximations, a system of reactions is partitioned into subsets C and D (C and D are short for *continuous* and *discrete* respectively). This partition process can be either *static* or *adaptive*: the *static* partition means that the partition process is done before the simulation and subsets C and D are fixed during the simulation, while the *adaptive* partition means that the partition process will be invoked during the simulation when it is necessary and subsets C and D will change accordingly.

Reactions in subset C are approximated by the following system of ODEs

$$\mathrm{d}\boldsymbol{x} = \sum_{j:R_j \in C} v_j \lambda_j(\boldsymbol{x}) \mathrm{d}t. \tag{3}$$

This ODE approximation approach can be justified by the system entering a specific region of its state space or to a limiting process, which is called thermodynamic limit [1].

The remaining reactions, which constitute D, still follow (2). Due to the ODE approximation (Equation (3)) of the original stochastic system (Equation (2)), we now attain a hybrid system whose dynamics are specified by [1]

$$\mathrm{d}\boldsymbol{x}(t) = \sum_{j:R_j \in C} v_j \lambda_j(\boldsymbol{x}) \mathrm{d}t + \sum_{j:R_j \in D} v_j \mathrm{d}N_j(t). \tag{4}$$

If the partition rules meet certain criteria (this will be elaborated in later sections), we can gain significant improvement in computational efficiency as well as tolerable loss of accuracy when we use (4) to approximate (2). In fact, in the thermodynamic limit as the system size tends to infinity, (4) produces the same process as the solution of (2), and thus the approximation becomes exact.

Through this paper, we will tackle the problem of modeling and analyzing biochemical systems within this theoretical framework.

4. HYBRID ADAPTIVE PETRI NETS

In this section, we introduce the notation and study the dynamics of hybrid adaptive Petri nets. The concept of HAPNs is developed mainly grounded on Section 3. In later sections, we will find that the HAPN is quite an intuitive approach to model biological systems and is nicely integrated with the hybrid simulation algorithm of systems biology.

4.1 Notation of HAPNs

First of all, we give some notations. Let $N = \{0, 1, 2, \dots\}$, $R' = [0, +\infty)$.

Following the customary notation [19, 14, 13, 18] for defining Petri nets and their extensions, we define HAPNs as follows.

Definition 1. An HAPN is a 7-tuple $(P, T, A, W, M_0, F, Par)$ where:

1. $P = \{p_1, p_2, \dots, p_n\}$ is a finite set of places
2. $T = \{t_1, t_2, \dots, t_m\}$ is a finite set of transitions
3. $P \cap T = \emptyset$ and $P \cup T \neq \emptyset$
4. $A \subseteq (P \times T) \cup (T \times P)$ is a set of arcs
5. $W : A \to N$ is a weight function
6. $M_0 : P \to R'$ is the initial marking
7. $\mathcal{M}$ is the set of all reachable markings. $\forall M \in \mathcal{M}$, $M : P \to R'$.
8. $F : T \times \mathcal{M} \to R'$ is a firing rate function
9. $Par : T \times \mathcal{M} \to \{C, D\}$ is a partition function which specifies the firing mode for a given transition. Here C and D are short for *continuous* and *discrete*.

We define the marking M of an HAPN as a mapping: $M : P \to R'$. However we usually denote by $M(t)$ the marking of an HAPN at time t for convenience. Graphical representation of an HAPN follows the conventional way [19]: places are drawn as circles, transitions as bars or boxes. Arcs are labeled with their weights (nonnegative integers) while labels for unity weight are usually omitted. We use dotted arcs to represent test arcs[13, 18]. An example can be found in Section 5.

According to the definition, in a specific HAPN, P, T, A, W specify its basic topology, M_0 designate the initial state, and F, Par are related to its dynamics.

4.2 Dynamics of HAPNs

Here we describe the way in which HAPNs evolve over time. We mainly focus on features of HAPNs that make them distinct from ordinary Petri nets. The most distinguishing feature of HAPNs is their adaptability, which means that transitions can adaptively choose its behavior (*continuous* or *discrete*) according to the partition function during the evolution of HAPNs. This adaptability is fulfilled by the dual firing mode of HAPN transitions and the partition function.

4.2.1 Dual Firing Mode of HAPN Transitions

In an HAPN, each transition has two firing modes: *continuous* mode and *discrete* mode. Assume that p is a place of an HAPN. Arc a_i ($i = 1, 2, \dots$) connects transition t_{a_i} to p, and arc b_i ($i = 1, 2, \dots$) connects p to transition t_{b_i}. Figure 1 is an example. Let $C(t)$ denote the set of transitions in *continuous* mode at time t, and $D(t)$ the set of transitions in *discrete* mode.

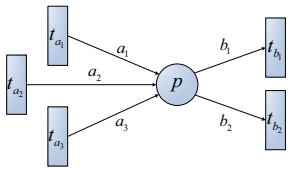


Figure 1: A place and transitions connected to it.

If a transition is in *continuous* mode, it fires continuously like its peers in hybrid functional Petri nets (HFPNs) [13, 18] and its firing speed is given by the firing rate function F.

Digital Object Identifier: 10.4108/ICST.VALUETOOLS2009.7753
http://dx.doi.org/10.4108/ICST.VALUETOOLS2009.7753

If a transition is in *discrete* mode, it fires discretely after a random delay. When it fires, it behaves like transitions of ordinary Petri nets [19]. The distribution of firing delays is determined by the rate function F. The firing process of a transition in *discrete* mode can be regarded as a counting process. Let $\{N_{t_i}(t)\}$ denote the stochastic process which counts the number of discrete firings of t_i until time t. $N_{t_i}(t)$ is a marking-dependent counting process such that

$$\mathbf{P}[N_{t_i}(t+\mathrm{d}t) - N_{t_i}(t) = 1 | M(t)] = F(t_i, M(t))\mathrm{d}t.$$

Thus, the variation of contents in place p can be described by

$$\begin{aligned}&\frac{\mathrm{d}M(p)}{\mathrm{d}t}\\ &= \sum_{t_{a_i}\in D(t)} W(a_i)\cdot \mathrm{d}N_{t_{a_i}}(t) + \sum_{t_{a_i}\in C(t)} W(a_i)\cdot F(t_{a_i}, M(t))\\ &- \sum_{t_{b_i}\in D(t)} W(b_i)\cdot \mathrm{d}N_{t_{b_i}}(t) - \sum_{t_{b_i}\in C(t)} W(b_i)\cdot F(t_{b_i}, M(t))\end{aligned}$$

where W is the weight function.

4.2.2 Partition Function

The role of a partition function in an HAPN is somewhat like an indicator: it shows whether a transition should fire in *continuous* mode or *discrete* mode. The motivation of introducing the partition function is to integrate the adaptive hybrid simulation algorithm into HAPNs: when certain conditions are satisfied, transitions will change their firing mode in order to keep a balance between computational efficiency and simulation accuracy. However it is also feasible to represent other types of models. The partition function takes the marking and one transition as its input, and outputs C or D, which indicates the mode (*continuous* or *discrete*) that the transition should follow.

The type of the model that an HAPN represents relies on the property of the partition function. If the partition of transitions depends on the marking of the HAPN, which evolves with time, we obtain an *adaptive* hybrid model. However, if the partition function is independent of the marking, we attain a *static* hybrid model. In particular, if the partition function takes value C on all the transitions, we arrive at an ODE model. And if the partition function takes value D on all the transitions, we obtain a stochastic model.

4.2.3 Firing Rate Function

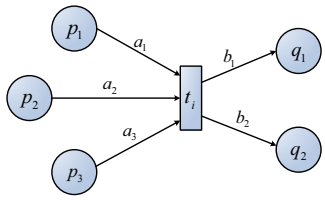


Figure 2: A transition and places connected to it.

The firing rate of a transition in either *continuous* or *discrete* mode is specified by the firing rate function. Since both the partition function and the firing rate function are marking-dependent, a transition can have different forms of rate function in different firing modes.

We do not impose many restrictions on the form of the firing rate function, but add one requirement to it. Consider the scenario in Figure 2 and we require that

$$\begin{cases} F(t_i, M(t)) = 0 \quad \text{if } Par(t_i, M(t)) = D \text{ and} \\ \qquad\qquad \exists j \text{ such that } M(p_j) < W(a_j); \\ F(t_i, M(t)) = 0 \quad \text{if } Par(t_i, M(t)) = C \text{ and} \\ \qquad\qquad \exists j \text{ such that } M(p_j) = 0 \text{ and } W(a_j) > 0. \end{cases} \tag{5}$$

where the notations are the same as in Definition 1. In fact, (5) is concerned with the firability of t_i. In other words, t_i is enabled if $F(t_i, M(t)) > 0$.

4.2.4 Evolution Equation of HAPNs

Let $H = (P, T, A, W, M_0, F, Par)$. H has n places. Assume that at time t, the marking of H is $M(t)$, and Par separates transitions of H into two groups: $C(t)$ and $D(t)$, i.e. transitions in *continuous* mode and transitions in *discrete* mode.

$$\begin{cases} C(t) = \{t_i \in T | Par(t_i, M(t)) = C\} \\ D(t) = \{t_i \in T | Par(t_i, M(t)) = D\} \end{cases} \tag{6}$$

Let $N_{t_i}(t)$ denote the counting process which enumerates the number of firings of transition t_i, if t_i is in *discrete* mode. Let v_{t_i} denote the row in the incidence matrix of H that corresponds to t_i. v_{t_i} is an $n-$vector $(v_{t_i}^1, \ldots, v_{t_i}^n)$ such that $v_{t_i}^j = w(t_i, p_j) - w(p_j, t_i), j = 1, \ldots, n$.

$$w(t_i, p_j) = \begin{cases} W(\alpha), & \text{if } \alpha = (t_i, p_j) \in A, \\ 0, & \text{otherwise.} \end{cases}$$

where A is the set of arcs and W is the weight function. $w(p_j, t_i)$ has a similar meaning.

The evolution of H can be described as

$$\mathrm{d}M(t) = \sum_{t_i\in C(t)} v_{t_i}\cdot F(t_i, M(t))\mathrm{d}t + \sum_{t_i\in D(t)} v_{t_i}\cdot \mathrm{d}N_{t_i}(t) \tag{7}$$

Note that (7) is essentially the same as (4), which is the theoretical basis of HAPNs' integration with the adaptive hybrid simulation algorithm of systems biology.

We have introduced the notation and discussed dynamics of HAPNs. It is obvious that HAPNs is a natural extension of stochastic Petri nets (SPNs). Moreover, we can see that HAPNs represent a certain kind of stochastic fluid models. And through a further analysis, we can prove that a subset of fluid stochastic petri nets (FSPNs) [14], which contain only timed transitions, can be regarded as special cases of HAPNs [26].

5. MODELING BIOLOGICAL SYSTEMS USING HAPNS

In this section, we prove Theorem 1, which guarantees that the HAPN formalism is sufficient to model general biochemical systems. Let

$$\mathcal{H} = \{H | H = (P, T, A, W, M_0, F, Par)\}$$

be the set of all HAPNs. According to Section 3, a biochemical system can be expressed as a quintuple

$$B = (\boldsymbol{X}, \boldsymbol{R}, \boldsymbol{x}(0), \Lambda, Type)$$

Digital Object Identifier: 10.4108/ICST.VALUETOOLS2009.7753
http://dx.doi.org/10.4108/ICST.VALUETOOLS2009.7753

where $\boldsymbol{X} = \{X_1, \ldots, X_n\}$ is the set of molecular species involved in system B, $\boldsymbol{R} = \{R_1, \ldots, R_m\}$ is the set of reactions in B, $\boldsymbol{x}(0) = (x_1(0), \ldots, x_n(0))$ is the initial state of B, $\Lambda = \{\lambda_1(\boldsymbol{x}), \ldots, \lambda_m(\boldsymbol{x})\}$ is the set of propensities of reactions in B, and $Type$ indicates the type of the biochemical model (ODEs, *static* hybrid, *adaptive* hybrid or stochastic). Let $\mathcal{B} = \{B | B = (\boldsymbol{X}, \boldsymbol{R}, \boldsymbol{x}(0), \Lambda, Type)\}$ denote the set of all biochemical systems. We use the notation $\mathcal{X} \subseteq \mathcal{Y}$ to indicate that the processes that arise from $\mathcal{X}$ are contained in those arising from $\mathcal{Y}$.

THEOREM 1. $\mathcal{B} \subseteq \mathcal{H}$.

PROOF. $\forall B \in \mathcal{B}$, $B = (\boldsymbol{X}, \boldsymbol{R}, \boldsymbol{x}(0), \Lambda, Type)$. To prove this theorem, it is sufficient to show that $B \in \mathcal{H}$. To fulfill this task, we construct an HAPN which is equivalent to B.

Let $H = (P, T, A, W, M_0, F, Par)$. For every $X_i \in \boldsymbol{X}$, construct a place $p_i \in P$ corresponding to it. Similarly, for every $R_i \in \boldsymbol{R}$, construct a transition $t_i \in T$ corresponding to it. Thus, $P = \{p_1, \ldots, p_n\}$ and $T = \{t_1, \ldots, t_m\}$. So the state of B at time t, $\boldsymbol{x}(t)$, can be regarded as the marking of H, $M(t)$. In particular, the initial marking $M_0 = \boldsymbol{x}(0)$.

$A \subseteq (P \times T) \cup (T \times P)$ is constructed as follows:

$$\begin{cases} (p_i, t_j) \in A \iff X_i \text{ is a reactant of } R_j, \\ (t_j, p_i) \in A \iff X_i \text{ is a product of } R_j. \end{cases}$$

Correspondingly, $W : A \to N$ is constructed as follows:

$$W(a) = \begin{cases} \alpha_{j,i}, & \text{if } a = (p_i, t_j) \in A, \\ \beta_{j,i}, & \text{if } a = (t_j, p_i) \in A \end{cases}$$

where $\alpha_{j,i}$ and $\beta_{j,i}$ can be found in (1). The firing rate function F is defined as follows:

$$F(t_j, M) = \lambda_j(\boldsymbol{x}) \quad t_j \in T, \lambda_j(\boldsymbol{x}) \in \Lambda$$

where M is the marking of H and $\boldsymbol{x}$ is the state of B.

The specification of Par depends on a case by case analysis of $Type$ of B. If B is an ODE model, Par takes value C on all the transitions. if B is a stochastic model, Par takes value D on all the transitions. If B is a *static* hybrid model, then $\boldsymbol{X}$ is previously divided into two groups: $Continuous$ and $Discrete$. At this time,

$$Par(t_j, M) = \begin{cases} C, & \text{if } R_j \in Continuous, \\ D, & \text{if } R_j \in Discrete. \end{cases}$$

If B is an *adaptive* hybrid model, then the partition scheme is dependent on the state of B. Given that Par takes the marking of H as its input, it is quite straightforward to express the partition scheme as Par. A typical *adaptive* partition scheme can be found in 6.1.

Equation (4) and (7) ensure that B and H share the same dynamics, which implies that $B \in \mathcal{H}$. □

With Theorem 1, we can convert a biochemical system to an HAPN. The bacteriophage T7 model illustrates the usage of Theorem 1.

Bacteriophage T7 Model. The bacteriophage T7 model (Figure 3) describes the intracellular growth of bacteriophage T7 [1, 23]. The components of the model are the viral nucleic acids, a viral structural protein (*struct*) and progeny virus (*virus*). The viral nucleic acids are classified as genomic (*gen*) and template (*tem*). The biological process is modeled by six transitions (Table 2). The model specification is not complete, since the partition function has not been designated. In Section 7, we will analyze and compare different types of models for bacteriophage T7.

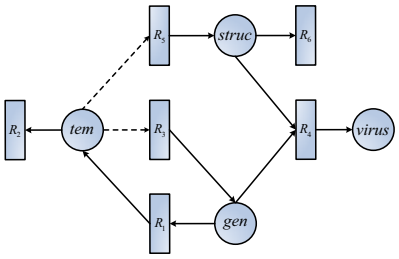


Figure 3: The bacteriophage T7 model. The dotted arcs represent test arcs.

6. A SIMULATION STUDY OF HAPNS

In this section, we propose a simulation study of HAPNs. Although an analytic-numeric study of HAPNs is desirable, the analytic solution becomes a formidable task from the computational point of view in many practical situations. Thus the simulation approach might be the unique or the more convenient way to study dynamics of HAPNs.

Although the HAPN is capable of representing different types of models, here we focus on the hybrid simulation algorithm, since other models, such as ODEs and stochastic models, can be regarded as special cases of hybrid models. From (7) we see that simulation of an HAPN is essentially applying the hybrid simulation algorithm to a given biological system within the framework of HAPNs. A general scheme of simulation algorithms of HAPNs is given in Figure 4. The algorithm has three main components: the partition process, simulation of transitions in $C(t)$ and simulation of transitions in $D(t)$. From Figure 4 we can see that the last two components are intertwined.

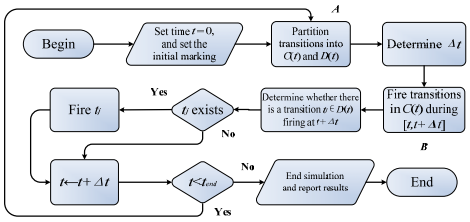


Figure 4: A general scheme of the HAPN simulation algorithm. t is the simulation time and t_{end} is the end time of simulation. $C(t)$ and $D(t)$ are the results of the partition process at time t: $C(t)$ is the group of transitions in *Continuous* mode and $D(t)$ is the group of transitions in *Discrete* mode. The specification of Δt is discussed in 6.3. In later subsections, step A is discussed in 6.1 and step B is discussed in 6.2.

The distinguishing feature of our adaptive hybrid simulation algorithm is that we separate transitions in *Discrete* mode into two different groups and simulate the two groups with different methods. Compared to previous work [11, 1], this discriminatory approach achieves a considerable saving of computational time. This will be elaborated in 6.3.

6.1 Partition Process

Digital Object Identifier: 10.4108/ICST.VALUETOOLS2009.7753
http://dx.doi.org/10.4108/ICST.VALUETOOLS2009.7753

Table 2: Specifications of transitions in Figure 3

No.	Description	Firing rate (day^{-1})	Rate constant (c_i)
R_1	Switch from genome to template	$c_1 \cdot M(gen)$	$c_1 = 0.025$
R_2	Degradation of template	$c_2 \cdot M(tem)$	$c_2 = 0.25$
R_3	Synthesis of genome	$c_3 \cdot M(tem)$	$c_3 = 1.0$
R_4	Production of progeny virus	$c_4 \cdot M(gen) \cdot M(struc)$	$c_4 = 7.5 \times 10^{-6}$
R_5	Synthesis of structural protein	$c_5 \cdot M(tem)$	$c_5 = 1000$
R_6	Degradation or structural protein	$c_6 \cdot M(struc)$	$c_6 = 1.99$

Assume that an HAPN $H = (P, T, A, W, M_0, F, Par)$. The partition process (step A in Figure 4) is performed by Par and $C(t)$ and $D(t)$ are obtained from (6).

The role of the partition process in the algorithm is to make a trade-off between computational efficiency and simulation accuracy. It determines whether it is appropriate for a transition to approximate *Discrete* mode with *Continuous* mode, or return to *Discrete* mode from *Continuous* mode. If the partition function is independent of the marking, then we attain a static partition scheme for H, which indicates that the partition of H is given before simulation, and is fixed for the entire evolution. However, the static partition scheme can not accommodate to the dynamic variation of biological systems [11, 15], so an adaptive partition scheme is more desirable.

As is stated in [1, 21], an adaptive partition function can be specified by two constants, τ and μ. If $\forall t_i \in T$, the following two conditions are satisfied:

$$F(t_i, M(t)) \geq \tau \tag{8}$$

and $\forall p_j$ which is the input place of t_i,

$$M(p_j) \geq \mu \cdot (W(\alpha) - w(t_i, p_j)), \tag{9}$$

where $\alpha = (p_j, t_i)$ is an arc of H and

$$w(t_i, p_j) = \begin{cases} W(\beta), & \text{if } \beta = (t_i, p_j) \text{ is an arc of } H, \\ 0, & \text{otherwise.} \end{cases}$$

then

$$Par(t_i, M(t)) = C.$$

If (8) or (9) is not met,

$$Par(t_i, M(t)) = D.$$

In the algorithm, (8) and (9) are used to partition the set of transitions during the simulation when it is necessary.

It can be shown that if τ and μ are sufficiently large, the approximation given by the partition process can attain accurate results. However, larger τ and μ will incur heavier computational burden. The theory behind the partitioning of transitions is related to the approximation of the chemical Master equation [21].

6.2 Simulation of Transitions in $C(t)$

Simulating transitions in $C(t)$ is essentially numerically solving the following system of ODE:

$$\mathrm{d}M(t) = \sum_{t_i \in C(t)} v_{t_i} \cdot F(t_i, M(t))\mathrm{d}t \tag{10}$$

where v_{t_i} is t_i's corresponding row in the incidence matrix of H (Equation (7)).

Numerical methods of ODEs, such as the Euler method and the Runge-Kutta methods, are well developed and fairly simple computationally. A myriad of numerical methods are already implemented and can be easily incorporated into the simulation algorithm.

6.3 Simulation of Transitions in $D(t)$

Simulating transitions in $D(t)$ is to find which transition will fire next and the time when that transition will fire. If a transition $t' \in D(t)$ will fire next at $t + \Delta t$, our task is to determine t' and Δt.

In [11, 1], all the reactions (equivalent to transitions in Petri nets) in $D(t)$ are simulated using the time-scale transformation. However, the mutual dependence of transitions can be explored to improve computational efficiency. According to the mutual dependence of transitions, we classify $D(t)$ into two subsets: $D_1(t)$ and $D_2(t)$. Unlike [11, 1], transitions in different subsets are simulated with different methods.

$D_1(t)$ and $D_2(t)$ are constructed as follows. If the firing rate of a transition in $D(t)$ is dependent of the firing of some transition in $C(t)$, we say that this transition in $D(t)$ is influenced by some transition in $C(t)$. Otherwise, this transition in $D(t)$ is not influenced by any transitions in $C(t)$. $D_1(t)$ is the set of transitions in $D(t)$ which are not influenced by any transitions in $C(t)$. On the contrary, $D_2(t)$ is the set of transitions in $D(t)$ which are influenced by some transition in $C(t)$.

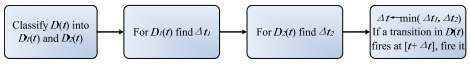


Figure 5: Firing transitions in $D(t)$. This figure further explains step C and step D in Figure 4. Details of Δt_1 and Δt_2 can be found in Algorithm 2.

The process of simulating transitions in $D(t)$ (Figure 5) has 3 main components: classifying $D(t)$, simulating $D_1(t)$ and simulating $D_2(t)$. We detail them as follows.

6.3.1 *Classification of* $D(t)$

In the context of mass action kinetics (Table 1), it is fairly simple to determine whether a transition is influenced by other transitions. If one of t_i's ($t_i \in D(t)$) input place is varied by some transition $t_j \in C(t)$, t_i is influenced by t_j and then $t_i \in D_2(t)$. If t_i is not influenced by any transition in $C(t)$, then $t_i \in D_1(t)$. A place is varied by a transition if and only if it is an output place of the transition or an input place of the transition with a positive weighted arc connecting them. An example is given in Figure 6. The process of classifying $D(t)$ is summarized in Algorithm 1.

Digital Object Identifier: 10.4108/ICST.VALUETOOLS2009.7753
http://dx.doi.org/10.4108/ICST.VALUETOOLS2009.7753

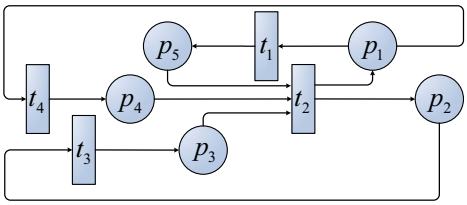


Figure 6: All arcs have positive weights. $F(t_1, M(t)) = c_1 \cdot M(p_1), F(t_2, M(t)) = c_2 \cdot M(p_3) \cdot M(p_4) \cdot M(p_5), F(t_3, M(t)) = c_3 \cdot M(p_2)$, and $F(t_4, M(t)) = c_4 \cdot M(p_1)$. p_1 and p_4 are varied by t_4. $C(t) = \{t_4\}, D(t) = \{t_1, t_2, t_3\}$. Thus, t_1 and t_2 are influenced by t_4. $D_1(t) = \{t_3\}, D_2(t) = \{t_1, t_2\}$.

Algorithm 1 Classifying $D(t)$

```
1: input C(t), D(t)
2: D_1(t) ← ∅, D_2(t) ← ∅, P ← ∅
3: for all t_c ∈ C(t) do
4:    for all p which is an output place of t_c do
5:       P ← P ∪ {p}
6:    end for
7:    for all p which is an input place of t_c do
8:       if W(α) > 0 then
9:          P ← P ∪ {p} // α = (p, t_c) and W(α) is the
            weight of α
10:      end if
11:   end for
12: end for
13: for all p ∈ P do
14:   for all t' of which p is an input place do
15:      if t' ∈ D(t) then
16:         D_2(t) ← D_2(t) ∪ {t'}
17:      end if
18:   end for
19: end for
20: D_1(t) ← D(t)\D_2(t)
21: return D_1(t) and D_2(t)
```

It is not hard to see that $D_1(t)$ is corresponding to the set of transitions with fluid independent firing rates in [2, 10], and that $D_2(t)$ is corresponding to the set of transitions with fluid dependent firing rates in [2, 10]. This means that methods described in [2, 10] are also applicable to simulating $D_1(t)$ and $D_2(t)$.

In [2, 10], the classification of fluid independent and fluid dependent firing rates is static, and cannot be changed during simulation. However, our classification scheme is adaptive, and $D_1(t)$ and $D_2(t)$ can be adjusted if necessary.

6.3.2 *Simulating* $D_1(t)$

For transitions of $D_1(t)$, firing time instant can be determined in advance. The direct method, the first reaction method [7] and the next reaction method [6] can be employed to simulate transitions in $D(t)$. However, compared to the first reaction method, the direct method is preferable since it is more efficient than the first reaction method [25]. In addition, evidence [11, 1] shows that the next reaction method is more time-consuming than the direct method in the hybrid environment. So we choose the direct method to simulate $D_1(t)$ for our algorithm.

6.3.3 *Simulating* $D_2(t)$

Transitions in $D_2(t)$ can be simulated using two methods: thinning and time-scale transformation [2, 10]. Time-scale transformation can tackle general cases while thinning requires that firing rates of transitions should be bounded with tight bounds [10]. However, under mass action laws, it is not easy to find proper bounds for transitions in $D_2(t)$. So, we use time-scale transformation in our algorithm.

Since $F(t_i, M(t))$ for $t_i \in D_2(t)$ is influenced by firing of transitions in $C(t)$, firing of transitions in $D_2(t)$ can be considered as a nonhomogeneous Markov process. It can be shown [11, 8] that the probability density function $f(\Delta t_2)$ is

$$f(\Delta t_2) = F_2^*(t + \Delta t_2) \exp(-\int_t^{t+\Delta t_2} F_2^*(\tau) \mathrm{d}\tau) \qquad (11)$$

where

$$F_2^*(t) = \sum_{t_i \in D_2(t)} F(t_i, M(t))$$

is the sum of firing rates of transitions in $D_2(t)$. And the conditional probability that transition $t_i \in D_2(t)$ fires at time $t + \Delta t_2$ is

$$\mathbf{P}(t_i \text{ fires at } t + \Delta t_2 | \Delta t_2) = \frac{F(t_i, M(t + \Delta t_2))}{F_2^*(t + \Delta t_2)}. \qquad (12)$$

With the help of (11) and (12), we simulate transitions in $D_2(t)$ with time-scale transformation, which can also be regarded as the direct hybrid method [11, 1].

The algorithm of simulating $D(t)$ is summarized as Algorithm 2. Note that Algorithm 2 generates less random numbers than the corresponding algorithm presented in [10]. In [10], for every enabled fluid dependent transition t_i, one should generate an random number Δ_i, solve the equation $\int_t^{t+\tau_i} F(\xi, M(\xi)) \mathrm{d}\xi = \Delta_i$ for τ, and then $\Delta t = \min(\Delta_i, t_i \in D(t))$. However, in Algorithm 2, to determine Δt, one only needs to generate in total two random numbers and solve one similar equation.

In practice, it is possible that $\Delta t = \infty$, which indicates that transitions in $D(t)$ will not fire in the current marking. In this instance, we set Δt to a predefined constant.

Digital Object Identifier: 10.4108/ICST.VALUETOOLS2009.7753
http://dx.doi.org/10.4108/ICST.VALUETOOLS2009.7753

Compared to [11, 1], we separate transitions ($D_1(t)$) whose rates are not influenced by firing of transitions in $C(t)$ from those ($D_2(t)$) whose rates are influenced by firing of transitions in $C(t)$, and simulate the two groups of transitions in different ways. In Section 7 we show that this manner of differential treatment brings about a considerable reduction of computing time.

Algorithm 2 Simulating $D(t)$

input $D_1(t)$ and $D_2(t)$
$\Delta t_1 \leftarrow \infty, F_1^* \leftarrow 0$
for all $t_i \in D_1(t)$ **do**
  $F_1^* \leftarrow F_1^* + F(t_i, M(t))$
end for
if $F_1^* \neq 0$ **then**
  generate $\Delta \sim \text{Expo}(F_1^*)$ // Δ follows the exponential distribution with parameter F_1^*
  $\Delta t_1 \leftarrow \Delta$
end if
generate $\Delta \sim \text{Expo}(1)$ // Δ follows the exponential distribution with parameter 1
find $\tau > 0$ such that $\int_t^{t+\tau} F_2^*(\xi)\, \mathrm{d}\xi = \Delta$ // this calculation is accompanied by simulating transitions in $C(t)$
$\Delta t_2 \leftarrow \tau$
if $\Delta t_1 < \Delta t_2$ **then**
  generate $\Lambda \sim \text{Unif}(0, F_1^*)$ // Λ follows the uniform distribution which takes parameters 0 and F_1^*
  find $t_j \in D_1(t)$ such that $\sum_{u=1}^{j-1} F(t_u, M(t)) \leq \Lambda < \sum_{u=1}^{j} F(t_u, M(t))$
  return Δt_1 and t_j
else
  generate $\Lambda \sim \text{Unif}(0, F_2^*(t + \Delta t_2))$ // Λ follows the uniform distribution which takes parameters 0 and $F_2^*(t + \Delta t_2)$
  find $t_j \in D_2(t)$ such that $\sum_{u=1}^{j-1} F(t_u, M(t + \Delta t_2)) \leq \Lambda < \sum_{u=1}^{j} F(t_u, M(t + \Delta t_2))$
  return Δt_2 and t_j
end if

6.4 Implementation

We implement a universal HAPN simulation platform on MATLAB, and incorporate our hybrid simulation algorithm in the platform. This platform can deal with not only biochemical networks but also general HAPN models. In the simulation algorithm, we use an adaptive partition scheme and set $\tau = \mu = 10$. For transitions in $C(t)$, we use the Runge-Kutta 4th Order Method to numerically solve (10) with time step 0.01. Although true dynamism [20], which involves repartitioning immediately when it is needed, is more adaptive to set the behavior of transitions according to the marking of the HAPN, it requires a real-time checking mechanism which incurs costly computational effort. In our algorithm, we check whether a repartition is necessary after every Δt is obtained. Numerical examples (Section 7) are proposed to show that the setting of our hybrid algorithm receives satisfactory results.

7. NUMERICAL EXAMPLES

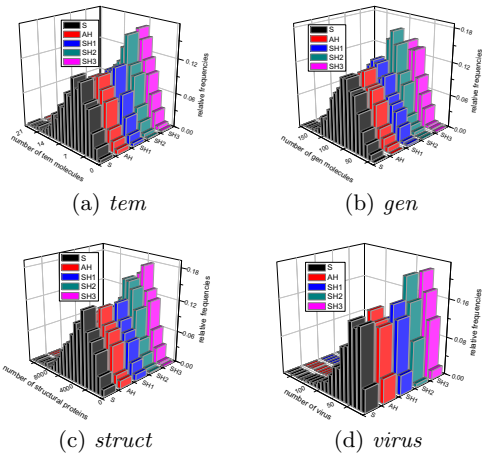
In this section, we present two numerical examples to confirm our theory. The first example shows that adaptive hybrid models are more accurate than static hybrid models for heterogeneous biological systems. The second example demonstrates that Algorithm 1 can speed up simulation without loss of accuracy.

Table 3: Partition functions of SH1, SH2, and SH3

Par	R_1	R_2	R_3	R_4	R_5	R_6
SH1	D	D	D	C	C	C
SH2	D	C	D	D	C	C
SH3	C	D	D	D	C	C

7.1 Bacteriophage T7 Model

The bacteriophage T7 model has been extensively studied by [1, 23]. Here, we use this model to assess the accuracy of the *adaptive* hybrid simulation algorithm. We consider 5 models: S, AH, SH1, SH2, and SH3. S is the stochastic model, AH is the *adaptive* hybrid model which uses the algorithm stated in Section 6, and SHi ($i = 1, 2, 3$) are *static* hybrid models whose partition functions are given in Table 3. The selection of *static* hybrid models is representative, since numerical results show that other choices of the partition function receive no better results than those obtained by the partition functions in Table 3. We set the initial marking $M_0 = (tem, gen, stru, virus) = (3, 20, 100, 4)$ and simulate 5 models for 30 days of virtual time. We obtain the empirical distribution function of the number of each species after 30 days (Figure 7).



(a) *tem* (b) *gen*

(c) *struct* (d) *virus*

Figure 7: The empirical distribution function of the number of each species after 30 days. Results are based on 1000 trials.

The stochastic model S serves as the standard, and the other four models are approximations of S. From Figure 7, we see that the results of *adaptive* hybrid algorithm, AH, conform well with those of S. However, for *static* hybrid models, things are different. SH1 produces fairly good approximations of S, but SH2 and SH3 incur tangible deviations from S. Thus, it can be implied that R_1 and R_2 might play a key role in the stochastic behavior of T7, since making R_1 or R_2 fire *continuously* significantly modifies the overall dynamical behavior of T7.

Digital Object Identifier: 10.4108/ICST.VALUETOOLS2009.7753
http://dx.doi.org/10.4108/ICST.VALUETOOLS2009.7753

From the above discussion, we find that *static* hybrid models might provide accurate approximations if before simulation, one could identify those reactions that occur most frequently. Nevertheless, this is not necessarily the case, and it is entirely possible that the set of fast transitions would vary according to time during simulation. On the other hand, *adaptive* hybrid models, although they are comparatively intricate, can accommodate to the dynamical behavior of the biochemical system and attain reliable results.

7.2 A Benchmark Model

In our simulation algorithm, we classify $D(t)$ into two groups, and simulate them in different ways. In this example, we give a quantitative study of the computational saving brought by this practice. We utilize an artificial biochemical network (Figure 8) as a benchmark to assess the efficiency of 5 algorithms: S, AH1, AH2, SH1, and SH2. S is the fully stochastic model which serves as the standard. AH1 is the *adaptive* hybrid algorithm which classifies $D(t)$, and AH2 is the *adaptive* one which does not classify $D(t)$, and simulates D(t) with the direct hybrid method. SH1 is the *static* hybrid algorithm classifying $D(t)$, and SH2 is the *static* one not classifying $D(t)$. SH1 and SH2 share the same partition function:

$$Par(t_{ij}, M) = \begin{cases} C & \text{for } i = 1,\ j = 0, \ldots, k, \\ D & \text{for } i = 2, 3,\ j = 0, \ldots, k. \end{cases}$$

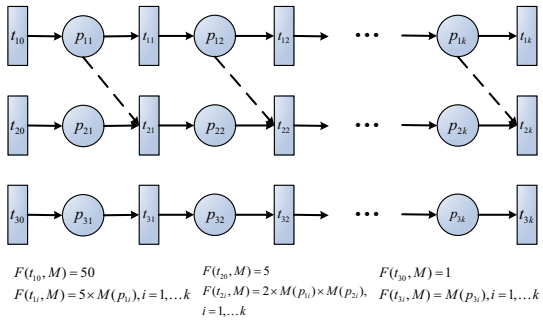


Figure 8: The benchmark model. The firing rates of transitions is shown in the figure. The number of places per line, k, indicates the size of the model. The benchmark model has $3k$ places and $3k+3$ transitions.

From Figure 9, we can see that with respect to one type of hybrid algorithms (*adaptive* or *static*), the algorithm which divides $D(t)$ into two groups (AH1 or SH1) is more efficient than the one which does not divide $D(t)$ (AH2 or SH2). Thus, the practice of classifying $D(t)$ brings to the hybrid simulation algorithms a considerable saving on computing time. Compared to the *adaptive* hybrid algorithm, the *static* hybrid algorithm need not monitor the state of the system or update $C(t)$ and $D(t)$ accordingly. However, AH1 is slightly faster than SH2, which is the contribution of classifying $D(t)$ into two groups.

From Subsection 7.1, we show that *adaptive* hybrid models are more accurate than *static* hybrid ones since adaptive models can accommodate to the dynamical behavior of the biochemical system and attain reliable results. Thus, although the *adaptive* hybrid algorithm consumes more time, it obtains more accurate results.

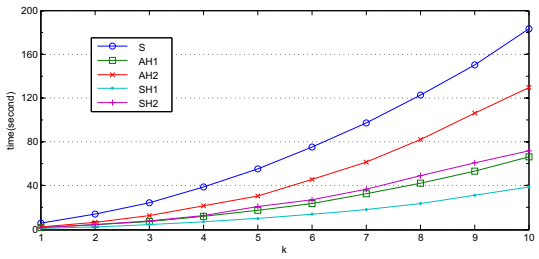


Figure 9: Average computing time of 5 algorithms as the size of the benchmark increases. k is the number of places per line in Figure 8. In the beginning, every place has one token. The simulation time is 5 seconds. Results are based on 100 trials. The simulation is run on a 2.2GHz processor and the version of MATLAB is R2008b.

8. CONCLUSION

In this paper, we introduce the hybrid adaptive Petri net, a unified framework which is capable of integrating various types of models. This modeling framework is of great value to biological scientists, since within this framework, issues of little biological importance are removed and they can focus on modeling and analyzing biological systems. Our formulation of biological systems applies to the scenario of well-mixed systems. We are now interested in adding the spatial attribute to the HAPN, which will further enhance its modeling ability.

We also make an improvement on the hybrid simulation algorithm. By exploring the mutual dependency of transitions in *discrete* mode, we classify transitions in *discrete* mode into two subsets and simulate them with different methods. We implement an HAPN simulator on MATLAB, and incorporate the improved algorithm in the simulator. Numerical examples show that the improved algorithm significantly promotes computational efficiency and attains satisfactory accuracy.

When we design and implement the hybrid simulation algorithm, two important issues should be concerned: efficiency and accuracy. The hybrid simulation algorithm is a compromise between the two issues. Our work is a substantial contribution to the first issue. On the other hand, with a precise analysis of the approximation error of the hybrid model compared to the fully stochastic one, we can develop reliable criteria for partitioning the set of reactions, which attain expected accuracy as well as minimized computational effort. [1, 21] present some attempts at the mathematical analysis of the error caused by the hybrid simulation algorithm. However, a full analysis of this issue remains to be done. Our paper focuses on the first issue, and we hope to contribute to the second issue in the future work.

9. ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (No. 60673187 and 60736028), and the National Grand Fundamental Research 973 Program of China (No. 2009CB320504 and 2006CB805901).

10. ADDITIONAL AUTHORS

Digital Object Identifier: 10.4108/ICST.VALUETOOLS2009.7753
http://dx.doi.org/10.4108/ICST.VALUETOOLS2009.7753

Additional authors: Hao Wen(Department of Computer Science, Tsinghua University, Beijing, China).

11. REFERENCES

[1] Aurélien Alfonsi, Eric Cancès, Gabriel Turinici, Barbara Di Ventura, and Wilhelm Huisinga. Adaptive simulation of hybrid stochastic and deterministic models for biochemical systems. *ESAIM: Proceedings*, 14:1–13, 2005.

[2] G. Ciardo, D. M. Nicol, and K. S. Trivedi. Discrete-event simulation of fluid stochastic petri nets. *IEEE Trans. Softw. Eng.*, 25(2):207–217, 1999.

[3] B. Di Ventura, C. Lemerle, K. Michalodimitrakis, and L. Serrano. From in vivo to in silico biology and back. *Nature*, 443(7111):527–533, OCT 5 2006.

[4] J. Fisher and T. A. Henzinger. Executable cell biology. *Nature Biotechnology*, 25(11):1239–1249, NOV 2007.

[5] H. Genrich, R. Küffner, and K. Voss. Executable Petri net models for the analysis of metabolic pathways. *International Journal on Software Tools for Technology Transfer*, 3(4):394–404, 2001.

[6] M. A. Gibson and J. Bruck. Efficient Exact Stochastic Simulation of Chemical Systems with Many Species and Many Channels. *J. Phys. Chem. A*, 104(9):1876–1889, March 2000.

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

[8] D. T. Gillespie. *Markov Processes: An Introduction for Physical Scientists*. Academic Press, OCT 22 1991.

[9] P. J. E. Goss and J. Peccoud. Quantitative modeling of stochastic systems in molecular biology by using stochastic Petri nets. *Proceedings of the National Academy of Sciences of the United States of America*, 95(12):6750–6755, 1998.

[10] M. Gribaudo and M. Sereno. Simulation of fluid stochastic petri nets. In *Proc. MASCOTS 2000*, page 231, Washington, DC, USA, 2000. IEEE Computer Society.

[11] M. Griffith, T. Courtney, J. Peccoud, and W. H. Sanders. Dynamic partitioning for hybrid simulation of the bistable hiv-1 transactivation network. *Bioinformatics*, 22(22):2782–2789, 2006.

[12] M. Heiner and I. Koch. Petri net based model validation in systems biology. In *Proc. ICATPN 2004*, volume 3099 of *LNCS*, pages 216–237, 2004.

[13] Hiroshi Matsuno, Yukiko Tanaka, Hitoshi Aoshima, Atsushi Doi, Mika Matsui, and Satoru Miyano. Biopathways representation and simulation on hybrid functional petri net. *In Silico Biology*, 3(0032), 2003.

[14] G. Horton, V. G. Kulkarni, D. M. Nicol, and K. S. Trivedi. Fluid stochastic petri nets: Theory, applications, and solution techniques. *European Journal of Operational Research*, 105(1):184 – 201, 1998.

[15] T. R. Kiehl, R. M. Mattheyses, and M. K. Simmons. Hybrid simulation of cellular behavior. *Bioinformatics*, 20(3):316–322, 2004.

[16] I. Koch, B. H. Junker, and M. Heiner. Application of Petri net theory for modelling and validation of the sucrose breakdown pathway in the potato tuber. *Bioinformatics*, 21(7):1219–1226, 2005.

[17] C. Li, Q.-W. Ge, M. Nakata, H. Matsuno, and S. Miyano. Modelling and simulation of signal transductions in an apoptosis pathway by using timed Petri nets. *Journal of Biosciences*, 32(1, Sp. Iss. SI):113–127, JAN 2007.

[18] H. Matsuno, S. Fujita, A. Doi, M. Nagasaki, and S. Miyano. Towards biopathway modeling and simulation. In *Proc. ICATPN 2003*, volume 2679 of *LNCS*, pages 3–22, 2003.

[19] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, Apr 1989.

[20] N. Neogi. Dynamic partitioning of large discrete event biological systems for hybrid simulation and analysis. In *Proc. HSCC 2004*, volume 2993 of *LNCS*, pages 463–476, 2004.

[21] H. Salis and Y. Kaznessis. Accurate hybrid stochastic simulation of a system of coupled chemical or biochemical reactions. *Journal of Chemical Physics*, 122(5), FEB 1 2005.

[22] E. Simao, E. Remy, D. Thieffry, and C. Chaouiya. Qualitative modelling of regulated metabolic pathways: application to the tryptophan biosynthesis in E.Coli. *Bioinformatics*, 21(suppl_2):ii190–196, 2005.

[23] R. Srivastave, L. You, J. Summers, and J. Yin. Stochastic vs. deterministic modeling of intracellular viral kinetics. *Journal of Theoretical Biology*, 218(3):309 – 321, 2002.

[24] F. St-Pierre and D. Endy. Determination of cell fate selection during phage lambda infection. *Proceedings of the National Academy of Sciences of the United States of America*, 105(52):20705–20710, 2008.

[25] D. Wilkinson. *Stochastic Modelling for Systems Biology*. Mathematical and Computational Biology. Chapman & Hall/CRC, 2006.

[26] H. Yang, C. Lin, Q. Li, and H. Zhou. Hybrid Adaptive Petri Nets for Modeling and Analyzing Biochemical Systems. In *International Workshop on Timing and Stochasticity in Petri nets and other models of concurrency*, 2009.

Digital Object Identifier: 10.4108/ICST.VALUETOOLS2009.7753
http://dx.doi.org/10.4108/ICST.VALUETOOLS2009.7753