PAPER Special Section on Concurrent/Real-time and Hybrid Systems: Theory and Applications

Performance Evaluation of Workflows Using Continuous Petri Nets with Interval Firing Speeds

Kunihiko HIRAISHI[†a)], ***Member***

SUMMARY In this paper, we study performance evaluation of workflow-based information systems. Because of state space explosion, analysis by stochastic models, such as stochastic Petri nets and queuing models, is not suitable for workflow systems in which a large number of flow instances run concurrently. We use fluid-flow approximation technique to overcome this difficulty. In the proposed method, GSPN (Generalized Stochastic Petri Nets) models representing workflows are approximated by a class of timed continuous Petri nets, called routing timed continuous Petri nets (RTCPN). In RTCPN models, each discrete set is approximated by a continuous region on a real-valued vector space, and variance in probability distribution is replaced with a real-valued interval. Next we derive piecewise linear systems from RTCPN models, and use interval methods to compute guaranteed enclosures for state variables. As a case study, we solve an optimal resource assignment problem for a paper review process.

key words: *fluidification, interval methods, workflow, performance evaluation, hybrid systems*

1. Introduction

How to make information systems safe, dependable and trustworthy is one of central concerns in the development of e-Society and e-Government. There are two kinds of correctnesses, qualitative correctness and quantitative correctness, where the former means that the system is proved to be logically correct, and the latter means that the system has sufficient performance for an expected amount of transactions. In this paper, we focus on quantitative correctness, and study performance evaluation of workflow-based information systems, particularly for those in which many instances of flows run concurrently. Such a situation often arises in web-based information systems for enterprises and local governments.

There are various results on modeling and verification of workflows (e.g., [8], [21]). In most of previous researches, some specific formalism, such as Petri nets, UML activity diagrams, and business process modeling languages (e.g., BPMN [22]), are used for describing workflows, and properties such as liveness, soundness and more general properties described by logical formulas are verified by using verification techniques developed for each formalism.

On the other hand, quantitative correctness was also studied as an important issue [1], [5], [15]. In actual workflow-based information systems, each workflow is nothing but a template, and many instances of the same workflow run concurrently. Therefore, guaranteeing correctness of an individual instance is insufficient for guaranteeing quantitative correctness of the entire system. In this paper, we study an optimal resource assignment problem as one of problems that particularly arise in workflow-based information systems. Recent workflow systems are often used for integrating various resources (software subsystems, databases, workers, machines, other organizations, etc.) that exist in enterprises. Therefore, we need to take care of quantity of resources necessary for performing workflows. Otherwise, many tasks may be assigned to the same resource, and as a result, the resource becomes a bottleneck of the system. To analyze this problem, we can use performance models such as stochastic Petri nets and queuing networks. However, state space explosion prevents us from dealing with a large number of flow instances.

Fluidification (or continuization) is a relaxation technique that tackles the state space explosion by removing the integrality constraints [20]. This idea is not new, and is found in various formalisms such as queuing networks (e.g., [14], [17]) and Petri nets. For Petri nets, fluidification was firstly introduced into the model called continuous Petri nets, and the formalism was extended to hybrid Petri nets [4]. Similar idea was also introduced into stochastic models such as fluid stochastic Petri nets [12]. Fluidification was also applied to analysis of a performance model based on process algebra [10].

In this paper, we introduce a class of timed continuous Petri nets, called routing timed continuous Petri nets (RTCPN), in order to approximate discrete state spaces of generalized stochastic Petri nets (GSPN). To deal with variance in probability distribution of each firing delay, interval firing speeds are introduced to timed transitions of RTCPN. In this sense, approximated models have uncertainty in their parameters.

There are several results on analysis of timed continuous Petri nets. In [9], [13], linear programming is used for computing performance measures at steady state. In contrast with previous works, we focus on transient analysis. Moreover, since our aim is to prove quantitative correctness, we would like to give some amount of guarantee to obtained results. It is known that the behavior of a timed continuous Petri net is represented by a piecewise linear (PWL) system. Based on interval methods for ordinary differential equations [7], [18], we derive a procedure to compute a guar-

Manuscript received April 10, 2008.
Manuscript revised May 19, 2008.

[†]The author is with the School of Information Science, Japan Advanced Institute of Science and Technology, Nomi-shi, 923-1292 Japan.
a) E-mail: hira@jaist.ac.jp

DOI: 10.1093/ietfec/e91–a.11.3219

anteed enclosure for state variables at each time step, where the guaranteed enclosure means that true value is surely in the enclosure. All the computation can be performed by linear programming solver and interval arithmetic.

As a case study, we consider the paper review process in an academic journal. The problem is to compute the minimum number of associate editors sufficient for handling an expected amount of submitted papers. We first show a result by GSPN. Next we compute transient behavior of the approximated RTCPN model, and compare the two results. Consequently, we claim that the proposed approach is scalable for the number of flow instances.

The paper is organized as follows. In Sect. 2, analysis by GSPN is presented. In Sect. 3, RTCPN is introduced, and the GSPN model built in Sect. 2 is approximated by RTCPN. Moreover, we derive a PWL system from the RTCPN model. In Sect. 4, an interval approximation algorithm for computing transient behavior of PWL systems is described. Numerical results of the algorithm are also shown. In Sect. 5, we discuss contribution and usefulness of the results in this paper. Section 6 is the conclusion.

2. Modeling Workflows by Stochastic Petri Nets

2.1 Example: Paper Review Process

We study the following workflow as an example. It is a workflow of the review process of an academic journal. The initial fragment of the workflow is shown in Fig. 1. By the editor in chief, each submitted paper is firstly assigned to one of associate editors responsible for research categories suitable for the paper. Then the associate editor finds two reviewers through some negotiation processes. There are three cases at the first review: acceptance, rejection, and conditional acceptance. If the decision is conditional acceptance, then the associate editor requests the authors to submit a revised manuscript toward the second review. The decision at the second review is the final one, and is either acceptance or rejection.

All the processes are supported by a web-based information system including electronic submission and automatic notification of due dates. Then the problem is how to decide the appropriate number of associate editors in each research category, considering load balancing.

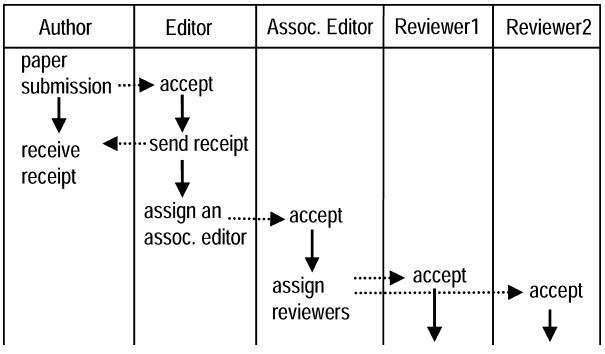

Fig. 1 The paper review process of an academic journal.

The problem is formally stated as follows:
Given

- A description of workflow,
- Statistics on paper submission,
- An upper bound of the number of papers each associate editor can handle,

Find

- The minimum number of associate editors such that the workflow runs stably. (The number of papers waiting for being processed should not become too large.)

The statistics on paper submission/handling is given as follows:

- Duration between submission and final judgment:
  - Acceptance at the first review: 2.4 months
  - Rejection at the first review: 3.9 months
  - Acceptance at the second review: 5.9 months
  - Rejection at the second review: 6.8 months
- Probabilities of acceptance and rejection:
  - Acceptance at the first review: 0.065
  - Rejection at the first review: 0.687
  - Acceptance at the second review: 0.238
  - Rejection at the second review: 0.010
- Average number of paper submissions: 16.9/month.

(This statistics is obtained from actual data of some academic journal.)

2.2 Stochastic Petri Nets

A generalized stochastic Petri net (GSPN) [2] is a 6-tuple $GSPN = (P, T, A, m_0, \lambda, w)$, where P is a set of places, T is a set of transitions, $A : P \times T \cup T \times P \to \mathbb{N}$ is the incidence function that specifies the weights of the arcs between places and transitions, and $m_0 : P \to \mathbb{N}$ is the initial marking. The incidence function is equivalently represented by two nonnegative integer matrices $\mathbf{A}^+ = [a^+_{ij}], \mathbf{A}^- = [a^-_{ij}] \in \mathbb{N}^{|P|\times|T|}$ by $a^+_{ij} = A(t_j, p_i)$ and $a^-_{ij} = A(p_i, t_j)$. Let $\mathbf{A} = \mathbf{A}^+ - \mathbf{A}^-$ be called *the incidence matrix*.

Transitions of GSPN are partitioned into two different classes: *immediate transitions* and *timed transitions*. Immediate transitions fire in zero time, and timed transitions fire after a random, exponentially distributed, enabling time. The function $\lambda : T_{exp} \to \mathbb{R}^+$ assigns a firing rate to each timed transition, where $\mathbb{R}^+$ is the set of nonnegative real numbers. The function $w : T_{im} \to \mathbb{R}^+$ assigns a firing weight to each immediate transition.

Firing semantics is described as follows. If the set of enabled transitions H comprises only timed transitions, then each transition $t_j \in H$ fires with probability $\lambda(t_j)/\sum_{t_k \in H} \lambda(t_k)$. If H comprises both immediate and timed transitions, then only immediate transitions can fire. If H

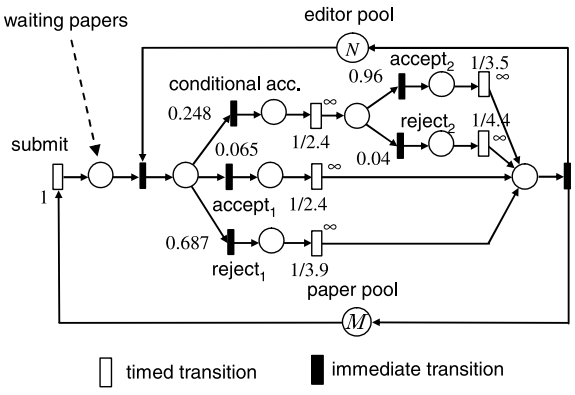

Fig. 2 GSPN model.

has several conflicting immediate transitions, then the firing weights determine probability that a particular immediate transition will fire. Let $C \subseteq H$ be the set of conflicting immediate transitions. Then the probability, called switching probability, that each transition $t_j \in C$ will fire is $w(t_j)/\sum_{t_k \in C} w(t_k)$.

The firing semantics described above is called *finite-server semantics*. There is another firing semantics, called *infinite-server semantics*. In infinite-server semantics, the same transition fires simultaneously. The multiplicity in the firing depends on the number of tokens in input places of each transition. As usual, we use the notation ${}^\bullet x := \{y \in P \cup T \mid A(y, x) > 0\}$ and $x^\bullet := \{y \in P \cup T \mid A(x, y) > 0\}$. For an infinite-server timed transition t_j and a marking m, let $enab(t_j, m) := min_{p_i \in {}^\bullet t_j}\{m(p_i)/A(p_i, t_j)\}$. Then the marking-dependent firing rate of t_j at marking m is $\lambda(t_j) \cdot enab(t_j, m)$. We will allow both types of timed transitions to exist in a model.

We first use GSPN to compute performance measures such as the average number of papers waiting for being processed. One of advantages of using Petri net models is that each instance of workflows is modeled by a token. Increasing the number of workflow instances corresponds to increasing the number of tokens. This means that modeling by GSPN is scalable in the number of workflow instances. Using analysis techniques on GSPN, we can compute stationary probability distribution of reachable states.

Figure 2 is the GSPN model of the workflow, where details of the processes (e.g., finding two reviewers, communication between editors and authors, research categories of associate editors etc.) are not considered for simplicity. The number associated with each transition is the firing delay (if the transition is timed) or the firing weight (if the transition is immediate). These numbers are determined from the statistics on paper submission/handling.

Once a paper is submitted, i.e., transition 'submit' fires, the paper is waiting for being processed. If an associate editor is available, then the result of the first review is decided according to the given probability. If the result is conditional acceptance, then the paper proceeds to the second review, and the final result is decided according to the given probability. These probabilities are specified by weights of immediate transitions. In the GSPN model, there are two sets of potentially conflicting immediate transitions: one corresponds to the decision at the first review, and the other corresponds to the decision at the second review.

Each duration τ in the paper review workflow is given as a firing rate $1/\tau$ of the corresponding timed transition. Since papers are processed in parallel by associate editors, these timed transitions are defined to be infinite-server (as indicated by symbol ∞ in the figure). Only transition 'submit' is a single-server timed transition. The firing rate '1' of transition 'submit' specifies the average number of papers added to place 'waiting papers' per one unit of time (= month). Since the average number of paper submissions is 16.9/month, one token corresponds to 16.9 papers.

The place 'paper pool' is necessary for the state space to be finite, and needs to be nonempty in order to get a correct result. Otherwise, transition 'submit' does not fire with the defined rate. As long as 'paper pool' is nonempty, we do not have to care about the number of tokens in it. The return arc to place 'submit' is introduced just for keeping the number of papers in the system constant. At the initial marking, we put a sufficient number of tokens in place 'paper pool.' In computer experiments, we check probability that the number of tokens in 'paper pool' is 0.

2.3 Computation Results: GSPN Model

We compute the expected number of waiting papers at steady state by a computer tool DSPNexpress-NG [16], [23]. Since exponential distribution is not appropriate for the delay of each transition, we use Erlangian distribution of order 2 as the probability distribution of each timed transition. It is known that Erlangian distribution of order n with average rate λ is simulated by a serial connection of n exponentially-distributed timed transitions with average rate $n\lambda$ [6]. Using this technique, we introduce Erlangian-distributed timed transitions in GSPN models. The drawback to using this technique is that it increases the size of the state space. Considering the actual probability distribution, order 2 may be insufficient. In computer experiments, however, it was hard to compute solutions for models with Erlangian distribution of order more than 2, because of state space explosion.

The computation result is shown in Table 1. In the initial state, we put N tokens in place 'editor pool.' For each value of N, the number of tangible states, CPU time, the expected number of waiting papers at steady state, and probability that the paper pool is empty are shown in Table 1. The computer environment used for the computation is a Linux high-performance computer with Intel Itanium2, 1.6 GHz/9 MB Cache CPU, 16 GB main memory. It is observed that $N \geq 6$ gives a desirable level. The amount of resources $N = 6$ means that $6 \times 16.9 = 101.4$ papers can be processed in parallel.

A high probability of $p(\#paperpool = 0)$ means that the amount of resources is insufficient for handling papers. If the probability is high, most of tokens initially given in the

Table 1 Numerical results of GSPN analysis.

N	#states	CPU Time (sec.)	#waiting papers	$p(\#paperpool = 0)$
3	2926	0.31	10.18	0.30
4	8866	0.7	5.94	0.094
5	23023	2.3	1.99	0.013
6	53053	6.2	0.63	0.0021
7	110968	15	0.21	0.00049
8	213928	29	0.08	0.00020
9	384098	58	0.03	0.00010

place 'paper pool' are stuck in place 'waiting papers.' We also observe that CPU time and the size of the state space increase very rapidly.

3. Modeling by a Class of Timed Continuous Petri Nets

In the GSPN model (Fig. 2), the number of reachable states increases exponentially in the number N. As a result, we will not be able to compute the steady state for larger models. We will use fluidification technique to overcome this difficulty. For this purpose, we introduce a class of timed continuous Petri nets as formalism to deal with continuous dynamics.

3.1 Routing Timed Continuous Petri Nets

We define a class of timed continuous Petri nets that will be used for approximating the GSPN model. *A routing timed continuous Petri nets* (RTCPN) is a 6-tuple $RTCPN = (P, T, A, m_0, \lambda, w)$, where P, T, and A are the same as those in GSPN except that the range of A is the set of real numbers, and $m_0 : P \to \mathbb{R}^+$ is the initial marking.

Transitions of RTCPN are partitioned into *timed transitions* and *routing transitions*. Timed transitions correspond to infinite-server timed transitions of GSPN, and routing transitions correspond to immediate transitions of GSPN.

The function $\lambda : T_{time} \to \mathbb{R}^+$, where T_{time} is the set of timed transitions, assigns a firing speed to each timed transition. A timed transition t_j fires at speed $\lambda(t_j) \cdot enab(t_j, m)$, similarly to an infinite-server timed transition of GSPN.

Routing transitions together with the routing rate function $w : T_{route} \to \mathbb{R}^+$, where T_{route} is the set of routing transitions, are used for determining routing of flows.

In RTCPN, we put the following restriction on the Petri net structure.

- If a place p_i has an output routing transition, then every output transition of p_i is a routing transition.
- There is no cycle consisting only of routing transitions.
- For any two routing transition t_i and t_j, if ${}^\bullet t_i \cap {}^\bullet t_j \neq \emptyset$ then $|{}^\bullet t_i| = |{}^\bullet t_j| = 1$, i.e., every potentially conflicting routing transition has *a free-choice structure*.

Remark 3.1: The first and the second items are natural assumptions. The third one is introduced just for simplifying the discussion. Under the third assumption, the routing defined by routing transitions becomes static like in STAR-CPN [9], and is easy to handle. In fact, for any RTCPN model, there exists an equivalent RTCPN model without any routing transitions. We omit the detail because of lack of space. We note that RTCPN can still represent FORK-JOIN type structure of workflows under this restriction.

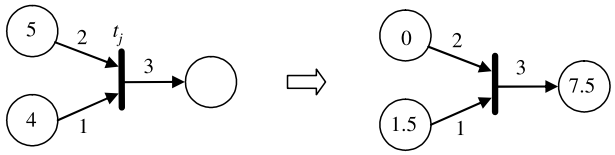

Fig. 3 Type (i) firing of a routing transition.

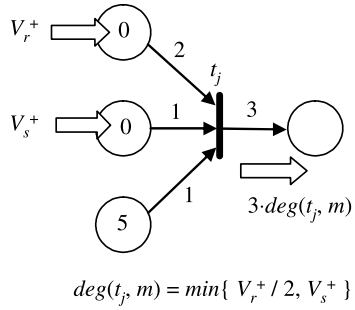

Fig. 4 Type (ii) firing of a routing transition (non-conflicting case).

Let t_j be a non-conflicting routing transition. There are two types in firing:

(i) When $enab(t_j, m) > 0$, a quantity $A(p_i, t_j) \cdot enab(t_j, m)$ is removed from each input place p_i of t_j, and a quantity $A(t_j, p_i) \cdot enab(t_j, m)$ is added to each output place p_i of t_j. These actions are instantaneous, and as a result, at least one of input transitions of t_j becomes empty (see Fig. 3). This case may occur only at the initial marking, or at the time when external quantity is put on a place.

(ii) Suppose that $enab(t_j, m) = 0$ and every empty input place p_i is fed, i.e., a positive quantity V_i^+ flows into p_i. Let E be the set of empty input places of p_i, and we define the firing degree by $deg(t_j, m) := min_{p_i \in E}\{V_i^+ / A(p_i, t_j)\}$. Then a quantity $A(p_i, t_j) \cdot deg(t_j, m)$ flows from each input place p_i of t_j, and a quantity $A(t_j, p_k) \cdot deg(t_j, m)$ flows into each output place p_k of t_j (see Fig. 4). We remark that nonempty input places are irrelevant to the firing degree.

For simplicity, we assume that in the initial marking m_0, $enab(t_j, m_0) = 0$ holds for every routing transition t_j, and ignore the type (i) firing. In the type (ii) firing, which one of input places determines the firing degree may be switched during execution.

The firing rule for conflicting routing transitions is as follows. Let C be the set of routing transitions that have a common input place p_i. Then a fraction $\rho(t_j) := w(t_j) / \sum_{t_k \in C} w(t_k)$ of flow V_i^+ is used exclusively for firing of each $t_j \in C$. Let $deg(t_j, m) := \rho(t_j) \cdot V_i^+ / A(p_i, t_j)$. Then, similarly to the non-conflicting case, a quantity $A(p_i, t_j) \cdot$

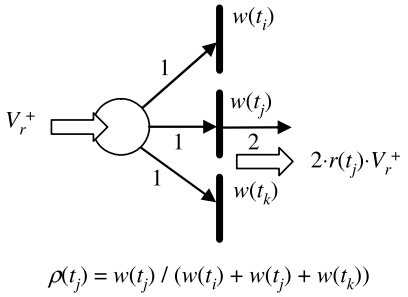

Fig. 5 Type (ii) firing of routing transitions (conflicting case).

$deg(t_j, m)$ flows from each input place p_i of t_j, and a quantity $A(t_j, p_k) \cdot deg(t_j, m)$ flows into each output place p_k of t_j (see Fig. 5).

3.2 Approximating Probability Distributions

From the GSPN model, we build an approximated RTCPN model based on the following idea:

- An infinite-server timed transition t_j of GSPN (with mean firing rate $\lambda(t_j)$) is approximated by a timed transition with a marking dependent firing speed $\lambda(t_j) \cdot enab(t_j, m)$.
- The variance of the probability distribution for firing rate $\lambda(t_j)$ is approximated by an interval firing speed $[\lambda(t_j)] = [\underline{\lambda}(t_j), \overline{\lambda}(t_j)]$. We assume that the actual firing delay is within the interval with a high probability.
- Switching probability among conflicting immediate transitions is approximated by routing rates of routing transitions.

Remark 3.2: (1) Approximation by mean firing rates preserves the expected number of tokens in each place of GSPN. We consider the following simple situation. Let t_j be an infinite-server timed transition with firing rate $\lambda(t_j)$ in GSPN. Suppose that t_j is non-conflicting and has a unique input place p_i with arc weight 1. Let W_i denote the random variable representing firing delay of t_j. Suppose that at time 0, there are k tokens in p_i. Some amount of time $t > 0$ has elapsed and the remaining tokens has decreased to $k' < k$ by firing of t_j. Then the expected value of k' is $E[m(p_i)] = kP[W_j > t] = ke^{-\lambda(t_j)t}$ since t_j is exponentially distributed in GSPN. In the RTCPN model, the firing speed of the corresponding transition is $\lambda(t_j) \cdot x_i$, where x_i is the variable representing $m(p_i)$. Solving the differential equation with initial condition $x_i(0) = k$, we obtain $x_i(t) = ke^{-\lambda(t_j)t}$. Both values coincide. This holds in general cases by the linearity of the system.

(2) For "congested" systems, this approximation is valid for any probabilistic distribution [20]. This is a result of the central limit theorem. Suppose that in some stochastic Petri net, delay for single firing has any probabilistic density function with mean τ and variance σ^2. Then mean delay for firing with multiplicity n is approximately normally distributed with the same mean τ and variance σ^2/n, provided

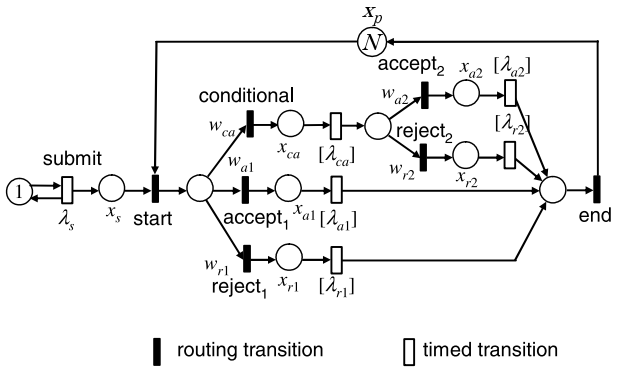

Fig. 6 RTCPN model.

that the time instant at which each of n firings becomes enabled is randomly selected. Considering this fact, the size of the interval $[\underline{\lambda}(t_j), \overline{\lambda}(t_j)]$ can be narrowed to one proportional to $1/\sqrt{enab(t_j, m)}$, when the system is approaching to a steady state.

From the GSPN model in Fig. 2, we obtain an RTCPN model shown in Fig. 6, where λ_s is the firing speed of timed transition 'submit,' $[\lambda_j]$ denotes the interval of firing speed for each timed transition, and w_k denotes the routing rate of each routing transition. There is no 'paper pool' in the RTCPN model since we do not have to make the state space finite any more. Since RTCPN has no timed transitions corresponding to single-server transitions, a single-server transition 'submit' is simulated by a timed transition with a self-loop.

3.3 From RTCPN Model to Differential Equations

We can derive a set of differential equations from the RTCPN model. We first define the flow for arc $A(p_i, t_j)$ by $V(p_i, t_j) := A(p_i, t_j) \cdot \lambda(t_j) \cdot enab(t_j, m)$ if t_j is a timed transition; $V(p_i, t_j) := A(p_i, t_j) \cdot deg(t_j, m)$ if t_j is a routing transition. Similarly, the flow of arc $A(t_j, p_i)$ is $V(t_j, p_i) := A(t_j, p_i) \cdot \lambda(t_j) \cdot enab(t_j, m)$ if t_j is a timed transition; $V(t_j, p_i) := A(t_j, p_i) \cdot deg(t_j, m)$ if t_j is a routing transition.

The differential equation with respect to place p_i is given as follows:

$$\dot{m}(p_i) = V_i^+ - V_i^- \tag{1}$$

where

$$V_i^+ = \sum_{t_j \in {}^\bullet p_i} V(t_j, p_i), \quad V_i^- = \sum_{t_j \in p_i^\bullet} V(p_i, t_j) \tag{2}$$

The differential Eq. (1) is not linear in general because $enab(t_j, m)$ and $deg(t_j, m)$ may contain 'min' operator. However, we can rewrite the set of differential equations as a piecewise linear (PWL) system.

Theorem 3.3: For the set of differential Eq. (1) derived from RTCPN, there exists an equivalent PWL system.

Mode I:$\{x_p = 0, x_s > 0\}$	**Mode II**:$\{x_p > 0, x_s = 0\}$
$\dot{x}_s = \lambda_s - R(x)$	$\dot{x}_s = 0$
$\dot{x}_p = 0$	$\dot{x}_p = R(x) - \lambda_s$
$\dot{x}_{ca} = \rho_{ca}R(x) - [\lambda_{ca}]x_{ca}$	$\dot{x}_{ca} = \rho_{ca}\lambda_s - [\lambda_{ca}]x_{ca}$
$\dot{x}_{a1} = \rho_{a1}R(x) - [\lambda_{a1}]x_{a1}$	$\dot{x}_{a1} = \rho_{a1}\lambda_s - [\lambda_{a1}]x_{a1}$
$\dot{x}_{r1} = \rho_{r1}R(x) - [\lambda_{r1}]x_{r1}$	$\dot{x}_{r1} = \rho_{r1}\lambda_s - [\lambda_{r1}]x_{r1}$
$\dot{x}_{a2} = \rho_{a2}[\lambda_{ca}]x_{ca} - [\lambda_{a2}]x_{a2}$	$\dot{x}_{a2} = \rho_{a2}[\lambda_{ca}]x_{ca} - [\lambda_{a2}]x_{a2}$
$\dot{x}_{r2} = \rho_{r2}[\lambda_{ca}]x_{ca} - [\lambda_{r2}]x_{r2}$	$\dot{x}_{r2} = \rho_{r2}[\lambda_{ca}]x_{ca} - [\lambda_{r2}]x_{r2}$

$R(x) = [\lambda_{a1}]x_{a1} + [\lambda_{r1}]x_{r1} + [\lambda_{a2}]x_{a2} + [\lambda_{r2}]x_{r2}$.
$\rho_j = w_j/(w_{ca} + w_{a1} + w_{r1})$ $(j \in \{ca, a1, r1\})$,
$\rho_j = w_j/(w_{a2} + w_{r2})$ $(j \in \{a2, r2\})$.

Fig. 7 The PWL system derived from the RTCPN model.

Proof. We first remark that each marking $m : P \to \mathbb{R}$ can be identified with a real vector $m \in \mathbb{R}^{|P|}$. For $enab(t_j, m)$, where t_j is a timed transition, we define a polytope

$$\xi_{time}[t_j, p_k] := \{m \in \mathbb{R}^{|P|} \mid \forall p_i \in {}^\bullet t_j - \{p_k\}. \\ m(p_k)/A(p_k, t_j) \le m(p_i)/A(p_i, t_j)\}.$$

Then $enab(t_j, m)$ is replaced with $m(p_k)/A(p_k, t_j)$ under the condition $m \in \xi_{time}[t_j, p_k]$.

Similarly for a non-conflicting routing transition t_j, we define the following set, which is not necessarily a polytope:

$$\xi_{route}[t_j, p_k, E] := \\ \{m \in \mathbb{R}^{|P|} \mid [\forall p_i \in E.\, m(p_i) = 0] \wedge \\ [\forall p_i \in {}^\bullet t_j - E.\, m(p_i) > 0] \wedge \\ [\forall p_i \in E - \{p_k\}. \\ V_k^+(m)/A(p_k, t_j) \le V_i^+(m)/A(p_i, t_j)]\},$$

where $E \subseteq^\bullet t_j$ and $V_i^+(m)$ is the total flow to place p_i determined by the current marking m. Suppose that for any place $p_i \in E$, all transitions in ${}^\bullet p_i$ are timed transitions. Such a case exists since there is no cycle consisting only of routing transitions, as we have assumed. Then, $V_i^+(m)$ is represented as a piecewise linear function of m. This is a result of the partition by $\xi_{time}[\,]$'s shown above. Therefore, there exists a finite set of polytopes $\{\xi^l_{route}[t_j, p_k, E]\}$ such that we can replace $deg(t_j, m)$ with a linear formula $V_k^+(m)/A(p_k, t_j)$ when $m \in \xi^l_{route}[t_j, p_k, E]$. Moreover, if such a finite set of polytopes is obtained for every input transition, which is not necessarily a timed transition, of place p_i, then $V_i^+(m)$ is represented by a piecewise linear function. Repeating this procedure, we can obtain the above finite set of polytopes $\{\xi^l_{route}[t_j, p_k, E]\}$ for every routing transition t_j.

Using the finite partition of $\mathbb{R}^{|P|}$ obtained by the above polytopes, we can have an equivalent PWL system. ∎

In the RTCPN model in Fig. 6, we assign a state variable x_i to each place, as indicated in the figure. The quantities in the place after 'start' and the place before 'end' are always 0, and we do not assign variables to these places. Then we obtain a PWL system with two modes shown in Fig. 7. At lease one of x_p and x_s is empty at every time instant.

4. Guaranteed Transient Analysis by Interval Method

In this section, we show a method for transient analysis of PWL systems derived from RTCPN models. The result includes guaranteed enclosures of state variables. The method is based on interval methods for ordinary differential equations. All the computation can be performed by interval arithmetic and linear programming.

4.1 Interval Method

Interval methods for ordinary differential equation (ODE) systems were introduced by Moore [18]. These methods provide numerically reliable enclosures of the exact solution at discrete time points $t_0, t_1, \cdots, t_k$.

We consider the following nonlinear continuous-time system:

$$\dot{x}(t) = f(x(t), \theta),\ x(t_0) = x_0 \tag{3}$$

where $x \in \mathbb{R}^n$ and θ is the vector of uncertain system parameters. Each parameter θ_i is assumed to be bounded by an interval $[\underline{\theta_i}, \overline{\theta_i}]$. In what follows, we will write $[v]$ to denote a vector of intervals that give the range of each component of a vector v. First the continuous-time state Eq. (3) is discretized by Taylor series expansion with respect to time:

$$x(t_{k+1}) \\ = x(t_k) + \sum_{r=1}^{\gamma} \frac{h^r}{r!} f^{(r-1)}(x(t_k), \theta) + e(x(\eta), \theta) \tag{4}$$

with $h = t_{k+1} - t_k$ and $t_k \le \eta \le t_{k+1}$. The guaranteed bound for the time discretization error is calculated by

$$e(x(\eta), \theta) \subseteq [e_k] = \frac{h^{\gamma+1}}{(\gamma+1)!} F^{(\gamma)}([B_k], [\theta]) \tag{5}$$

where $F^{(\gamma)}$ is the interval extension of the γ-th order derivative $f^{(\gamma)}$ of f, and $[B_k]$ is the bounding box for the range of state variables $x(t) \in [B_k],\ \forall t \in [t_k, t_{k+1}]$. The bounding box $[B_k]$ is computed by applying the Picard operator

$$\Phi([B_k]) := [x_k] + [0, h] \cdot F([B_k], [\theta]) \subseteq [B_k] \tag{6}$$

where F is the interval extension of the function f. Calculation is usually performed by interval arithmetic. The bounding box $[B_k]$ is initialized with the state vector $[x(t_k)]$. If $\Phi([B_k]) \not\subset [B_k]$, then the bounding box $[B_k]$ has to be enlarged. If $\Phi([B_k]) \subseteq [B_k]$, then (6) is evaluated recursively until the deviation between $\Phi([B_k])$ and $[B_k]$ is smaller than a specified value. In the case that this algorithm does not converge or that the interval of the discretization error $[e_k]$ is unacceptably large, the step size h has to be reduced.

There are several ways to compute a guaranteed bound $[x(t_{k+1})]$ by Eqs. (4) and (5) [7]. The direct interval technique is to compute $[x(t_{k+1})]$ by interval arithmetic, i.e.,

$$[x(t_{k+1})] = [x(t_k)] + \sum_{r=1}^{\gamma} \frac{h^r}{r!} F^{(r-1)}([x(t_k)], [\theta]) + [e_k] \tag{7}$$

The direct interval techniques propagate entire boxes through interval solutions. As a consequence errors may tend to accumulate as computations proceed. Let $v|_i$ denote the i-th component of a vector v. In the piecewise interval technique, the solution is computed by

$$[x(t_{k+1})]|_i = \left[\inf_{x\in[x(t_k)],\theta\in[\theta]} \psi(x,\theta)|_i, \sup_{x\in[x(t_k)],\theta\in[\theta]} \psi(x,\theta)|_i \right] + [e_k]|_i \quad (8)$$

where $\psi(x,\theta) := x + \sum_{r=1}^{\gamma} (h^r/r!) f^{(r-1)}(x,\theta)$, i.e., intervals are computed by solving optimization problems for each component. The main idea of the piecewise interval technique is to propagate small boxes, and works effectively for reducing the accumulation of error.

4.2 Piecewise Interval Method for Systems with Multiple Modes

For systems with multiple modes like PWL systems, we need to take account of mode changes during time interval $[t_k, t_{k+1}]$. Such systems with switching are studied in [19]. We briefly describe the interval method for a PWL system

$$\dot{x}(t) = f_j(x(t),\theta) \text{ if } x \in \xi_j \quad (9)$$

where $x \in \mathbb{R}^n$, each f_j represents a linear continuous-time dynamic with a vector of interval parameters θ, and each ξ_j is a polytope in $\mathbb{R}^n$. The idea is to replace function f in Eq. (3) with the following function f_a that represents the union of f_j's for all active modes during time interval $[t_k, t_{k+1}]$, i.e.,

$$F_{f_a} \supseteq \bigcup_{j\in M([B_k])} F_{f_j} \quad (10)$$

where $[B_k]$ is a bounding box for time interval $[t_k, t_{k+1}]$, $M([B_k])$ denotes the set of all mode j such that $\xi_j \cap [B_k] \neq \emptyset$, and F_f represents the exact value set $F_f := \{y \mid y = f([B_k],[\theta])\}$ of function f under consideration of all interval arguments. Since the bounding box $[B_k]$ and function f_a are mutually dependent, we need to perform an iterative procedure to compute them.

We use here a piecewise interval technique which was not used in [19]. This is possible because we are considering only linear systems. Considering that the system is piecewise linear, we will use $\gamma = 1$ in Eqs. (4) and (5).

The result by applying the Picard operator is computed by the following piecewise computation:

$$\Phi(B_k)|_i := \left[\inf_{\substack{(j,x,y,d,\theta)\\ \in S_1^k}} \phi_j(x,y,d,\theta)|_i, \sup_{\substack{(j,x,y,d,\theta)\\ \in S_1^k}} \phi_j(x,y,d,\theta)|_i \right] \quad (11)$$

where

$$S_1^k := \{(j,x,y,d,\theta) \mid j \in M([B_k]), x \in [x(t_k)] \cap \Theta, y \in [B_k] \cap \Theta, d \in [0,h], \theta \in [\theta]\} \quad (12)$$

and $\phi_j(x,y,d,\theta) := x + d \cdot f_j(y,\theta)$ for each region ξ_j. Θ is the additional linear constraints that we can assume on the state space. As the constraints Θ, we will use P-invariants derived from RTCPN models later. Let $BoundingBox([x(t_k)],[\theta],\Theta,h)$ denote the above procedure to compute the bounding box.

The piecewise interval method to compute guaranteed bounds for $x(t_k)$, $k = 0, 1, \cdots, k_f$ is shown in Fig. 8, where piecewise computation is used at the step (*) for computing $[x(t_{k+1})]|_i$.

Theorem 4.1: Suppose that the PWL system (9) satisfies the following condition: for any bounded set $X \subseteq \mathbb{R}^n$, the number of regions ξ_j such that $X \cap \xi_j \neq \emptyset$ is finite. Then the computation of (11) and step (*) in procedure *solve*() can be performed by solving a finite number of linear programming (LP) problems.

Proof. We first prove the theorem for (11). Constraints for $x \in [x(t_k)] \cap \Theta$, $y \in [B_k] \cap \Theta$, and $\theta \in [\theta]$ are all linear. Let $x = [x_1, \cdots, x_n]^T$ and $y = [y_1, \cdots, y_n]^T$. Each component of $\phi_j(x,y,d,\theta) = [\phi_{j,1}, \cdots, \phi_{j,n}]^T$ has the form $\phi_{j,l} = x_l + d \cdot \sum_i \alpha_i y_i$, where $d \in [0,h]$ and $\alpha_i \in [\underline{\alpha_i}, \overline{\alpha_i}]$. By introducing new variables z_i, $i = 1, \cdots, n$, $\phi_{j,l}$ can be rewritten as $\phi_{j,l} = x_l + d \cdot \sum_i z_i$ with linear constraints $\underline{\alpha_i} y_i \le z_i \le \overline{\alpha_i} y_i$ if $\underline{\alpha_i} \ge 0$ and $y_i \ge 0$, and constraints for other cases are similarly obtained. Since scalar variable d is constrained only by interval $[0,h]$, $\phi_{j,l}$ gives its optimal value when $d = 0$ or $d = 1$. Therefore, we can compute the interval of $\phi_{j,l}$ by solving LP problems for all $j \in M([B_k])$ and $d \in \{0,h\}$, where $M([B_k])$ is finite by the assumption. The situation is the same in optimization problems of (*) except that d is fixed to h. ∎

The PWL system obtained from RTCPN satisfies the condition in the above theorem, because the system consists of a finite number of regions. Therefore, the procedures for computing (11) and (*) in *solve*() can be implemented on a constraint solver that can solve liner programming problems. On the other hand, the computation $[e_k]$ includes Jacobian $f^{(1)}$, and we need solve nonlinear constraints in order to compute piecewise interval solutions. By this reason, interval arithmetic is used for computing $[e_k]$.

4.3 Using P-Invariants As Constraints of LP Problems

A P-invariant is a nonnegative integer solution $y \in \mathbb{N}^{|P|}$ of a linear homogeneous equation $y^T \mathbf{A} = 0$, where $\mathbf{A}$ is the incidence matrix of the Petri net structure. As well known, $y^T m = y^T m_0$ holds for any marking m reachable from the initial marking m_0. Since the marking of RTCPN is defined in a real vector space, we define a P-invariant of RTCPN as any real-valued solution $y \in \mathbb{R}^{|P|}$ of equation $y^T \mathbf{A} = 0$. When we solve optimization problems in (11) and (*), we can add an equation $y^T x = y^T x_0$, where y is a P-invariant and x_0 is the initial state, as one of linear constraints Θ. This will works effectively in reducing the sizes of intervals. In the RTCPN model in Fig. 6, the following P-invariant is

$solve([x(t_0)], [\theta], \Theta, t_0, h, k_f)$::

for $k = 0$ **to** $k_f - 1$ **do**

$[B_k] := BoundingBox([x(t_k)], [\theta], \Theta, h)$;

$$[e_k] := \frac{h^2}{2!} F^{(1)}([B_k], [\theta]);$$

$$[x(t_{k+1})]|_i := \left\lfloor \inf_{(j,x,\theta)\in S_2^k} \psi_j(x,\theta)|_i, \quad \sup_{(j,x,\theta)\in S_2^k} \psi_j(x,\theta)|_i \right\rfloor + [e_k]|_i \ (*)$$

where
$S_2^k := \{(j, x, \theta) \mid j \in M([B_k]), x \in [x(t_k)] \cap \Theta, \theta \in [\theta]\}$,
$\psi_j(x, \theta) := x + h \cdot f_j(x, \theta)$;

endfor.

Fig. 8 Procedure to compute guaranteed enclosures.

obtained: $x_p + x_{ca} + x_{a1} + x_{r1} + x_{a2} + x_{r2} = N$. Moreover, the following inequalities also hold in all reachable states and, are added as constraints of the LP problems: $x_s \geq 0, x_p \geq 0, x_{ca} \geq 0, x_{a1} \geq 0, x_{r1} \geq 0, x_{a2} \geq 0, x_{r2} \geq 0$.

4.4 Computation Results: RTCPN Model

As shown in Theorem 4.1, we need to solve LP problems a number of times. Moreover, interval arithmetic is used in the computation of $[e_k]$. All processes of the algorithm are implemented on a single constraint solver called KCLP-HS, which is a rapid prototyping tool for algorithms on hybrid systems developed by the author's group [11]. KCLP-HS is a constraint logic programming language equipped with a linear constraint solver, quadratic programming solver, manipulation of convex polyhedra, and interval arithmetic. Disjunctions of linear constraints are handled by backtracking mechanism. We compute at each time step guaranteed intervals for state variables under the following parameters:

- Initial state: $x_p = N$, $(N = 50, 60, \cdots, 150)$. Values of other state variables are all 0.
- Interval of firing speeds: $[(1 - \delta) \cdot \lambda(t_j), (1 + \delta) \cdot \lambda(t_j)]$ for each timed transition t_j, where δ is a parameter determining the width of intervals.
- Step size: $h = 0.1$ month.
- Duration: 12 months. This means that the number of iterations is 120.

Figure 9 shows the upper bound of x_s, i.e., the number of waiting papers, for each N, when $\delta = 0.2$. Figure 10 shows the guaranteed enclosures of state variables in case of $N = 100$, where two enclosures are depicted for each state variable, one is obtained by a fixed $\delta = 0.2$, and the other is obtained by $\delta = 0.2/\sqrt{enab(t_j, m)}$ as discussed in Remark 3.2. Table 2 shows CPU times and upper bounds of x_s after 12 months. CPU times are measured by the same computer environment as that used for the GSPN analysis.

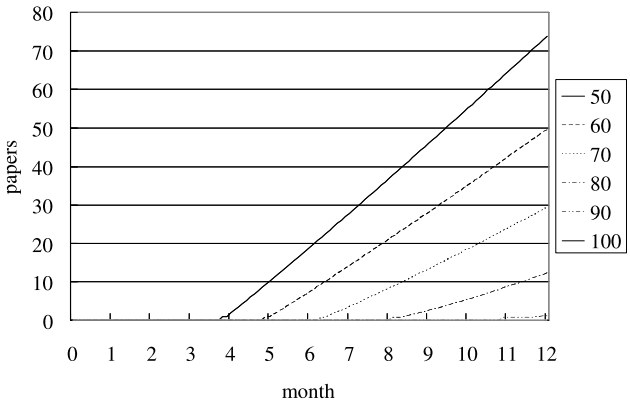

Fig. 9 Upper bounds of x_s ($\delta = 0.2$).

Table 2 CPU times and the upper bound of x_s after 12 months ($\delta = 0.2$).

N	CPU Time (sec.)	upper bound of x_s
50	24.0	73.8
60	22.6	49.6
70	19.9	29.3
80	16.4	12.2
90	11.2	1.1
100	7.7	0
110	6.2	0
120	6.2	0
130	6.2	0
140	6.2	0
150	6.2	0

Note that the value of N does not depend on the computation time. This shows that the method is scalable for the amount of resources.

From these results, we observe that $N > 100$ is sufficient for keeping the number of waiting papers at a low level. Looking at the transient behavior in Fig. 10, the quantity x_p, which corresponds to the residual resource, is nearly converged after 12 months. Notice that the number $N = 100$ is almost the same as the number computed by the GSPN model.

5. Discussion

In this section, we discuss the contribution of this paper both in application side and theoretical side.

From the application point of view, the proposed method based on fluidification gives a way to evaluate performance of workflow-based information systems with a large number of workflow instances. Comparing with conventional approaches such as stochastic performance modeling, one of the advantages of the proposed method is the scalability for the number of workflow instances. As a result the proposed method will be applicable to large systems intractable by methods working on discrete states. Moreover, we will be able to use the transient analysis for simulating various situations such as sudden change of performance by system faults.

From the theoretical point of view, contribution of this

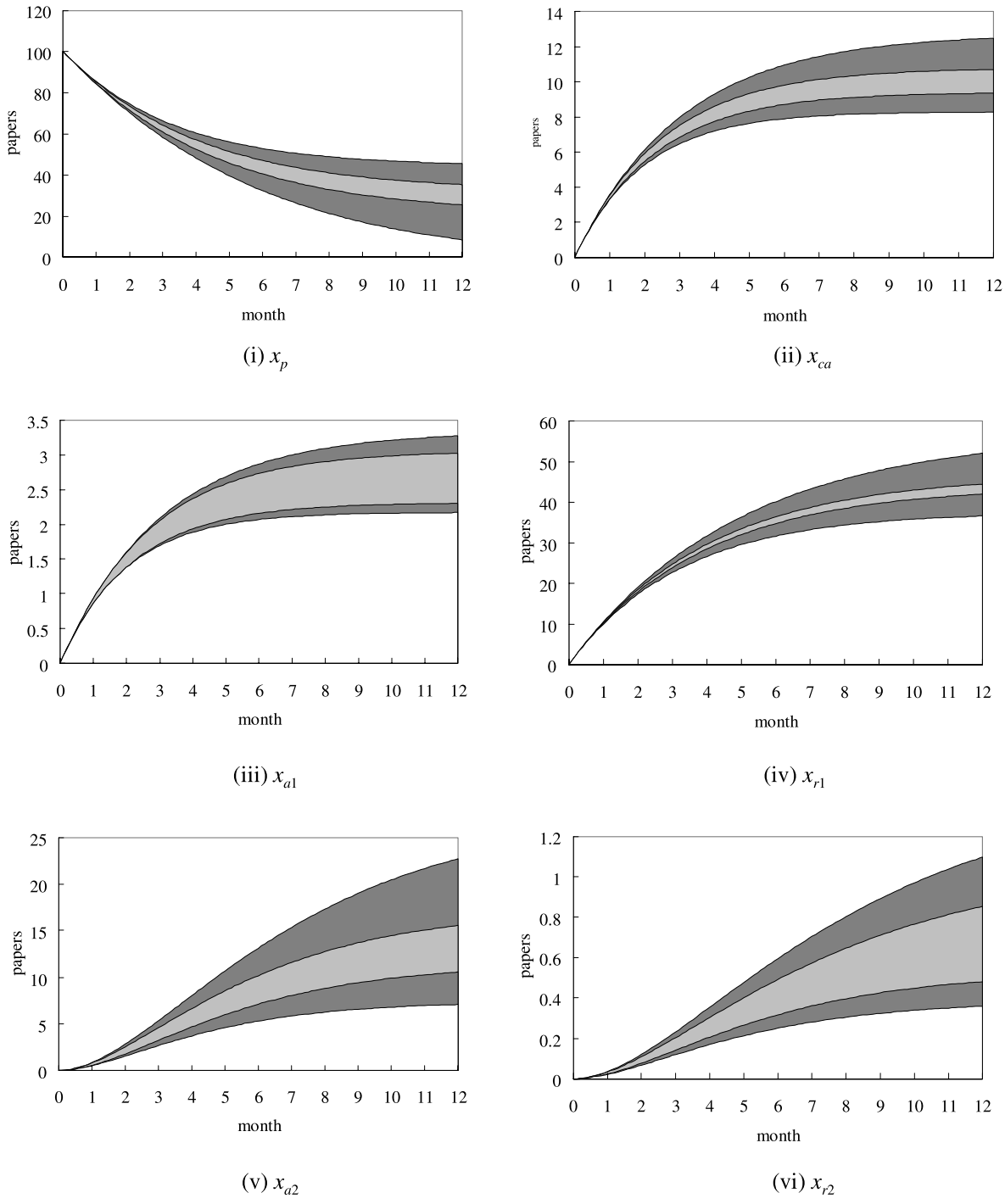

Fig. 10 Guaranteed enclosures of state variables ($N = 100$) (Gray area: $\delta = 0.2$, Light gray area: $\delta = 0.2/\sqrt{enab(t_j, m)}$).

paper is summarized as follows. Firstly, we have proposed a new class of continuous Petri net RTCPN, which can be used for fluidification of GSPN models. In fact, for any RTCPN model, there exists an equivalent RTCPN without any routing transitions. This can be proved by a similar way as that discussed in [9]. The reason why routing transitions are introduced is that we want to show the fluidification process in an intuitive way.

Secondly, we have proposed to use interval methods for transient analysis of RTCPN models. The method is based on piecewise interval methods for multi-mode systems, and computes guaranteed enclosures of state variables. All computations can be performed by solving a finite number of linear programming problems together with interval arithmetic. Using place invariants in the computation of intervals is also an original idea.

As discussed in a survey paper by Silva [20], one possible way to compute the transient behavior of a fluidified model is to use numerical methods implemented on a computer, for example using MATLAB. However, ordinary numerical methods do not reflect probabilistic deviations in system parameters, which are mandatory in performance evaluation. Using the proposed method, we can compute guaranteed enclosures of state variables for systems having deviations in their systems parameters.

6. Concluding Remarks

For performance evaluation of workflows, we have tried two approaches, analysis by GSPN and approximation by a class of timed continuous Petri nets, called RTCPN. A PWL system is derived from the RTCPN model. Using piecewise interval method, guaranteed enclosures for state variables at each time step have been computed. The difference between two approaches is in scalability for the number of workflow instances. Computation times in RTCPN analysis do not depend on the number of workflow instances. Moreover, since state variables in RTCPN are almost decoupled, we expect that interval methods can be applied to larger models including hundreds of variables. Experiments for larger models remain as future work.

Transient analysis may correspond to bounded model checking for discrete-state systems [3], where the objective of bounded model checking is to find errors by simulation for finite time steps. The models we have studied in this paper are autonomous. We expect that the proposed approach is applicable to performance models with external logical control. Hybrid system models such as hybrid Petri nets can be used for the modeling. In addition, the proposed method is able to check properties of systems with uncertainty in their parameters. Such systems are often found in medical and biological systems. These are also targets of our approach.

Acknowledgments

The research is partly supported by the Grant-in-Aid for Scientific Research of the Ministry of Education, Science, Sports and Culture of Japan, under Grant No. 17560386, and also the 21st Century COE program "Verifiable and Evolvable E-Society" at JAIST. The author thanks to Prof. Hofer and Mr. Rauh, University of Ulm, Germany, for valuable suggestion about interval methods. The author also thanks to Prof. Lindemann, University of Dortmund, for allowing us to use DSPNexpress-NG.

References

[1] A.F. Abate, A. Esposito, N. Grieco, and G. Nota, "Workflow performance evaluation through WPQL," Proc. SEKE'02, pp.489–495, 2002.
[2] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis, Modeling with Generalized Stochastic Petri Nets, Wiley Series in Parallel Computing, John Wiley & Sons, 1995.
[3] A Biere, et al., Bounded model checking, Book chapter: Advances in Computers, vol.58, Academic Press, 2003.
[4] R. David and H. Alla, "On hybrid petri nets," Discrete Event Dynamic Systems: Theory and Applications, vol.11, pp.9–40, 2001.
[5] J. Dehnert, J. Freiheit, and A. Zimmermann, "Modeling and performance evaluation of workflow systems," Proc. 4th. World Multiconference on Systems, Cybernetics and Informatics, vol.VIII, pp.632–637, 2000.
[6] A.A. Desrochers and R.Y. Al-Jaar, Applications of Petri Nets in Manufacturing Systems, IEEE Press, 1995.
[7] Y. Deville, M. Janssen, and P.V. Hentenryck, "Consistency techniques in ordinary differential equations," Proc. 4th Int. Conf. Principles and Practice of Constraint Programming, LNCS 1520, pp.162–176, 1998.
[8] R. Eshuis and R. Wieringa, "Verification support for workflow design with UML activity graphs," Proc. ICSE02, pp.166–176, 2002.
[9] B. Gaujal and A. Guia, "Optimal stationary behavior for a class of timed continuous petri nets," Automatica, vol.40, pp.1505–1516, 2004.
[10] J. Hillston, "Fluid flow approximation of PEPA models," Proc. 2nd Int. Conf. Quantitative Evaluation Systems, pp.33–42, 2005.
[11] K. Hiraishi, "KCLP-HS: A rapid prototyping tool for implementing algorithms on hybrid systems," JAIST Research Report IS-RR-2006-012, Aug. 2006.
[12] G. Horton, V.G. Kulkarni, D.M. Nicol, and K.S. Trivedi, "Fluid stochastic petri nets: Theory, applications, and solution techniques," NASA Contractor Report, no.198274, 1996.
[13] J. Júlvez, L. Recald, and M. Silva, "Steady-state performance evaluation of continuous mono-t-semiflow petri nets," Automatica, vol.41, pp.605–616, 2005.
[14] L. Kleinrock, Queuing Systems, Volume II: Computer Applications, vol.2, Wiley, 1976.
[15] J. Li, Y.S. Fan, and M.C. Zhou, "Performance modeling and analysis of workflow," IEEE Trans. Syst. Man Cybern. A, vol.34, no.2, pp.229–242, 2004.
[16] C. Lindermann, Performance Modeling with Deterministic and Stochastic Petri Nets, Wiley, 1998.
[17] A. Mandelbaum and H. Chen, "Discrete flow networks: Bottlenecks analysis and fluid approximations," Mathematics of Operations Research, vol.16, pp.408–446, 1991.
[18] R, Moore, Methods and Applications of Interval Analysis, SIAM, Philadelphia, 1979.
[19] A. Rauh, M. Kletting, H. Aschemann, and E.P. Hofer, "Interval methods for simulation of dynamical systems with state-dependent switching charactaristics," Proc. IEEE Int. Conf. Control Applications, pp.355–360, 2006.
[20] M. Silva and L. Recade, "Continuization of timed petri nets: From performance evaluation to observation and control," Proc. ICATPN2005, LNCS 3536, pp.26–47, 2005.
[21] W.M.P. van der Aalst, "Verification of workflow nets," Proc. ATPN 1997, LNCS 1248, pp.407–426, 1997.
[22] http://www.bpmn.org/
[23] http://rvs.informatik.uni-leipzig.de/de/software/index.php

Kunihiko Hiraishi received from the Tokyo Institute of Technology the B.E. degree in 1983, the M.E. degree in 1985, and D.E. degree in 1990. He is currently a professor at School of Information Science, Japan Advanced Institute of Science and Technology. His research interests include discrete event systems and formal verification. He is a member of the IEEE, IPSJ, and SICE.