A Two-Stage Probabilistic Approach to Manage Personal Worklist in Workflow Management Systems*

Rui Han[1,2,3], Yingbo Liu[1,2,3], Lijie Wen[1,2,3], and Jianmin Wang[1,2,3]

[1] School of Software, Tsinghua University, Beijing, P.R. China, 100084
[2] Key Laboratory for Information System Security, Ministry of Education, P.R. China, 100084
[3] Tsinghua National Laboratory for Informativasron Science and Technology, P.R. China, 100084
{hanr07,lyb01,wenlj00}@mails.tsinghua.edu.cn, jimwang@tsinghua.edu.cn

Abstract. The application of workflow scheduling in managing individual actor's personal worklist is one area that can bring great improvement to business process. However, current deterministic work cannot adapt to the dynamics and uncertainties in the management of personal worklist. For such an issue, this paper proposes a two-stage probabilistic approach which aims at assisting actors to flexibly manage their personal worklists. To be specific, the approach analyzes every activity instance's continuous probability of satisfying deadline at the first stage. Based on this stochastic analysis result, at the second stage, an innovative scheduling strategy is proposed to minimize the overall deadline violation cost for an actor's personal worklist. Simultaneously, the strategy recommends the actor a feasible worklist of activity instances which meet the required bottom line of successful execution. The effectiveness of our approach is evaluated in a real-world workflow management system and with large scale simulation experiments.

Keywords: workflow management system, workflow scheduling, personal worklist management, probability.

1 Introduction

Workflow management systems (WFMSs) improve business processes by automating tasks and getting the right information to the right place for a specific job function in time [1]. One crucial area that WFMSs can bring great improvement to business processes is the application of scheduling in workflow [2, 3]. Existing techniques on workflow scheduling [2, 4, 5, 6, 7, 8] mainly concern about allocating tasks (activity instances) among multiple resources (mainly refers to actors, i.e., activity executors or

* Supported by the NSFC (90718010), National Basic Research Program (973 Plan) under grant No. 2009CB320700, National High-Tech Development Program (863 Plan) under grant No. 2008AA042301 and 2007AA040607, and Program for New Century Excellent Talents in University.

R. Meersman, T. Dillon, P. Herrero (Eds.): OTM 2009, Part I, LNCS 5870, pp. 24–41, 2009.

workflow participants). However, people are actually the driving force of workflow [9]. It is necessary to apply workflow scheduling to assist actors to manage their personal worklists, which contain activity instances (i.e., work items, this paper uniformly calls activity instances for convenience) that would be executed by the actors. Therefore, personal worklist management should be an important part of workflow scheduling and a necessary complement of current workflow scheduling techniques.

At present, there is little research on personal worklist management in WFMSs. In [10, 11], authors present preliminary work in this area. For an activity instance in the personal worklist, they analyze its several states of satisfying deadline together with discrete probabilities. Based on these deterministic analysis results, they sort activity instances in the personal worklist under rigorous assumptions (specified in Section 6).

Nevertheless, the dynamic nature of workflows causes highly uncertainties in the personal worklist management. From the aspect of process instance, two uncertainties, which are variations in activity (execution) durations and the unpredictability of execution paths, cause activity instances' continuous probabilities of satisfying deadlines. For example, in Figure 1, activity instance a_6's probability of satisfying deadline is 96.31%, and that probability is only 5~10% for activity instance a_5. From the aspect of personal worklist, new activity instances are continuously added to the personal worklist, and activity instances from multiple systems and process instances compete for execution during the same time interval. Hence some activity instances may have extremely low probabilities of satisfying the given deadlines, because these activity instances' executions are delayed by their preceding activity instances. For example, in Figure 1's personal worklist, probabilities of satisfying deadlines for activity instances a_1, a_2, a_4, a_7 to a_{10}, a_{12}, a_{14}, and a_{15} are all below 5%.

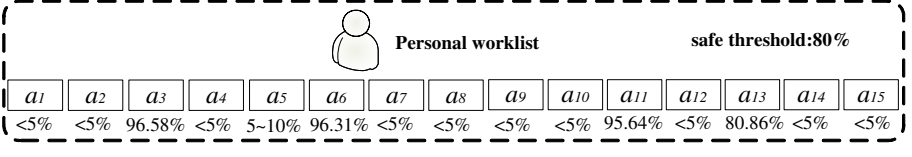


Fig. 1. An example of personal worklist

As stated above, serious deadline violations may occur in an actor's personal worklist, which cause considerable exception-handling (e.g., deadline-based escalation [12]) costs in the system. Moreover, the possibilities of violating deadline are stochastic due to the complex issues involved in the process instance and personal worklist. Therefore, traditional deterministic management of personal worklist is too rigorous.

In this paper, we proposed a two-stage probabilistic approach to provide stochastic and flexible management of personal worklist. Specifically, at the first process analysis stage, the approach analyzes the continuous probability of satisfying deadline for every activity instance. This probability is denoted as successful execution ratio and it provides a precise estimation to guide personal worklist scheduling. At the second worklist scheduling stage, an innovative scheduling strategy [13] is proposed to manage an actor's personal worklist. When new activity instances are added to the personal worklist, this strategy maintains a feasible worklist (named worklist *S*) and an infeasible worklist (named worklist *U*). The worklist *S* involves activity instances

whose joint successful execution ratio meets the required bottom line of successful execution, i.e., a probability value denoted as safe threshold (e.g., 80% for Figure 1's personal worklist). The worklist U is comprised of other deadline-violating activity instances. The objective of our scheduling strategy is to minimize the total deadline violation cost for activity instances in worklist U, thus achieving improved performance of WFMSs.

The rest of the paper is organized as follows. Section 2 gives the overview of our two-stage probabilistic approach. Section 3 explains the first process analysis stage and Section 4 introduces the second worklist scheduling stage. Section 5 presents the implementation of our approach in a real-world WFMS and utilizes large scale simulation experiments to evaluate the effectiveness of our scheduling strategy. Section 6 discusses related work. Finally, Section 7 presents the conclusions and future work.

2 A Two-Stage Probabilistic Worklist Management Approach

In this section, we present the overview of our two-stage probabilistic worklist management approach. As shown in Table 1, the approach includes two stages and each stage is comprised of two steps.

Table 1. A two-stage probabilistic worklist management approach

Overview	**Input**: Process instance, workflow event log. **Method**: Two-stage probabilistic worklist management approach. **Output**: Feasible worklist *S*, infeasible worklist *U*.
Stage 1: Process analysis	**Step 1.1**: Modeling process instance with PTCWF-nets.
	Step 1.2: Analyzing every activity instance's successful execution ratio.
Stage 2: Worklist scheduling	**Step 2.1**: Adding a new activity instance to an actor's personal worklist.
	Step 2.2: Scheduling worklist *S* to meet the safe threshold, while minimizing the total deadline violation cost in infeasible worklist *U*.

At the process analysis stage, when a new process instance is initiated for execution, step 1.1 models this process instance with Probabilistic Time Constraint WorkFlow Nets (PTCWF-nets). PTCWF-nets extend classical Petri nets and Workflow nets (WF-nets) [14] with time information. Based on PTCWF-nets, step 1.2 analyzes activity instances' successful execution ratios and then allocates these activity instances. Generally, an activity instance is directed to an actor for execution [10, 15].

At the worklist scheduling stage, when new activity instances are added to an actor's personal worklist, the approach schedules the worklist. To be specific, for a newly added activity instance, step 2.1 inserts it into the feasible worklist S. Then, step 2.1 recalculates the joint successful execution ratio for activity instances in worklist S. Next, step 2.2 schedules worklist S to make the ratio meet the safe threshold by eliminating some activity instances. The eliminated activity instances are added to infeasible worklist U for exception handling. Step 2.1 and 2.2 would iterate several times if multiple activity instances are added. The approach performs immediately when the actor is idle based on two assumptions: 1) the execution of activity instance

is non-preemptive [13]; 2) the scheduling takes a short time with respect to activity instances' execution time. Note that our approach recommends the actor a list of activity instances which ensure successful execution, while other deadline-violating activity instances' exception-handling costs are minimized. However, the real execution of activity instances is always up to the actors themselves.

3 Analysis of Process Instance

In this section, we introduce step 1.1 and 1.2 of our approach in subsection 3.1 and 3.2, respectively.

3.1 Modeling Process Instance with PTCWF-Nets

Inheriting and developing from Stochastic Petri nets (SPN) [16], PTCWF-nets model the process instance by adding time constraint C_P, time parameter C_T, and time parameter C_F to classical WF-nets' [14] set of places P, set of transitions T, and set of arcs F.

In PTCWF-nets, transitions represent activity instances. Therefore, time constraint C_P describes temporal relations between activity instances. Two time parameters separately represent two uncertainties in business processes (i.e., variations of activity execution durations and paths): C_T describes continuous distributed durations, and C_F represents probabilities of selecting execution paths. These time constraint and parameters can be obtained from workflow event log by statistical analysis [1, 17].

Definition 1. (Time constraint C_P). *Time constraint C_P is a finite set of pairs ($E_{\min}$, $E_{\max}$). For any place p ($p \in P$), c_p ($c_p \in C_P$) denotes token's available time for p's output transitions. To be specific, from the moment when p's all input transitions complete and a token arrives at p, $c_p.E_{\min}$ and $c_p.E_{\max}$ ($c_p.E_{\max} \geq c_p.E_{\min} \geq 0$) are separately token's earliest and latest available time for p's output transitions.*

For a place p, $c_p.E_{\min}$ represents the minimum time interval between the predecessor and successor activity instances. Similarly, $c_p.E_{max}$ represents the maximum interval. For example, in a process instance of altering contract, when activity instance *Countersign* is completed, its successor activity instance *Review* has to wait 2 hours (the minimum time interval) before being executable and should be finished within 58 hours (the maximum time interval).

Definition 2. (Time parameter C_T). *Time parameter C_T is a finite set of random variables which obey normal distribution $N(\mu, \sigma^2)$. For any transition t ($t \in T$), c_t ($c_t \in C_T$) denotes t's firing duration which is not instantaneous where $c_t \sim N(\mu_t, \sigma_t^2)$ and $\mu_t, \sigma_t^2 \in R^+$.*

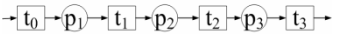
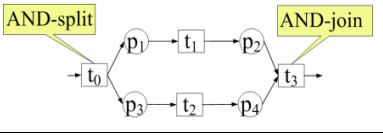
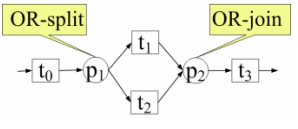
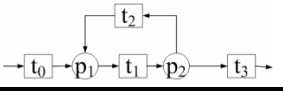
Time parameter C_T represents the probability distribution of activity duration. Without losing generality, we assume that all activity durations follow the normal distribution model $N(\mu, \sigma^2)$, where μ is the expected value, σ^2 is the variance, and σ is the standard deviation [17]. Those follow non-normal distribution models can be treated by normal transformation [18].

Definition 3. (Time parameter C_F). *Time parameter C_F is a finite set of probability values. For any arc f ($f \in F$), c_f ($c_f \in C_F$) denotes the probability of selecting f where $c_f \in R^+$. In addition, $c_f=1$ if f belongs to a sequential or parallel path which is certainly be selected. $0 \le c_f \le 1$ if f belongs to a selective or iterative path which is one of the n ($n \ge 1$) possible execution paths: $\{f_1, f_2, \ldots, f_n\}$, where $\sum_{i=1}^{n} c_{f_i} = 1$.*

Definition 4. (PTCWF-nets). *A PTCWF-net is a six tuple ($P, T, F; \theta_P, \theta_T, \theta_F$), where $<P, T, F>$ is the classical WF-net. θ_P is **place** function. It is defined from P into C_P such that: $\forall p \in P$, $[\theta_P(p) \in C_P]$. θ_T is **transition** function. It is defined from T into C_T such that: $\forall t \in T$, $[\theta_T(t) \in C_T]$. θ_F is **arc** function. It is defined from F into C_F such that: $\forall f \in F$, $[\theta_F(f) \in C_F]$.*

PTCWF-nets have the property of safe and free-choice [14]. Also, PTCWF-nets are isomorphic to continuous time Markov chains [16]. The number of states of the Markov chain corresponds to the number of reachable markings of the PTCWF-nets. In addition, PTCWF-nets include four types of workflow control structures: sequential, parallel, selective, and iterative (Table 2). These four structures can be combined to model majority of typical business processes in various domains, such as manufacture factory, hospital, bank, and so on [14].

Table 2. Four types of workflow control structures in PTCWF-nets

Workflow control structure	Figure
Sequential	
Parallel	
Selective	
Iterative	

3.2 Analyzing Every Activity Instance's Successful Execution Ratio

After modeling the process instance with a PTCWF-net, the approach analyzes transition's successful firing ratio to represent activity instance's successful execution ratio. Figure 2 shows the pseudocode of the analyzing algorithm.

```
Input: The PTCWF-net with time constraints and time parameters
Output: Every transition's successful firing ratio
Method
1. a^i := c;    // a^i is the token's arrival time of source place
   i, and c is a time constant
2. t := T. getTransitionByBFT();// obtain the first transition
3. WHILE t≠ Ø DO
   BEGIN
4.    d^t_min := μ_t - 3σ_t;    // minimum firing duration
5.    d^t_max := μ_t + 3σ_t;    // maximum firing duration
6.    e^t_min := Max{a^p + θ_P(p).E_min| p∈ •t };   // earliest enabled time
7.    e^t_max := Min{a^p + θ_P(p).E_max| p∈ •t};    // latest enabled time
8.    s^t := e^t_min;    // start time
9.    r^t := P(d^t_min<θ_T(t)<e^t_max-s^t);    // successful firing ratio
10.   FOR EVERY p (p∈ t•) DO
      BEGIN
11.     p.calculateTokenArrivalTime();
       // calculate p's token's arrival time
      END
12.   t := T.getTransitionByBFT();    // obtain a new transition
  END
Function getTransitionByBFT()
// return a new transition through BFT of the PTCWF-net; return
Ø if there is no new transition in the PTCWF-net
Function calculateTokenArrivalTime()
// calculate place's token's arrival time
```

Fig. 2. Analyzing algorithm of successful firing ratio

Assumed that in the PTCWF-net, set T has n transitions, set P has m places, and set F has k arcs (n, m, $k \geq 1$). The algorithm in Figure 2 utilizes the basic breadth-first traversal (BFT) strategy to select transition from the PTCWF-net for analyzing its successful firing ratio (line 2 and line 12). The time complexity of this BFT strategy is $O(n+m+k)$ for directed graph PTCWF-net [19]. In addition, all operations in each analysis of a transition's successful firing ratio (line 4 to line 11) are linear time complexity. Therefore, the time complexity of the whole algorithm is $O(n+m+k)$.

As shown in Figure 2, to analyze a transition t's successful firing ratio (line 9), the algorithm first needs to calculate t's possible interval of duration ($d^t_{\min}, d^t_{\max}$) (line 4 and line 5), its enabled time interval ($e^t_{\min}, e^t_{\max}$) (line 6 and line 7), and its start time s^t (line 8).

We employ the "3σ" rule to define transition t's possible interval of duration ($d^t_{\min}, d^t_{\max}$) where t's duration $\theta_T(t) \sim N(\mu_t, \sigma_t^2)$. This rule describes that for any sample comes from normal distribution model, it has a probability of 99.73% to fall into

the range of [μ-3σ, μ+3σ] which is a systematic interval of 3 standard deviations around the expected value [17]. Therefore, t's minimum duration is defined as $d^t_{\min} = \mu_t - 3\sigma_t$, and the maximum duration is $d^t_{\max} = \mu_t + 3\sigma_t$.

Transition t is enabled if and only if all its input places have available tokens. For any place p ($p \in \bullet t$), its time constraint is $\theta_P(p)$ and its token's arrival time is a^p. According to definition 1, in place p, token's earliest available time for t is $a^p + \theta_P(p).E_{\min}$ and the latest time is $a^p + \theta_P(p).E_{max}$. Therefore, t's earliest enabled time $e^t_{\min}$ is the maximum one of token's earliest available time in t's input places: $e^t_{\min} = \mathrm{Max}\{a^p + \theta_P(p).E_{\min} \mid p \in \bullet t\}$. Similarly, t's latest enabled time e^t_{max} is the minimum one of token's latest available time in t's input places: $e^t_{max} = \mathrm{Min}\{a^p + \theta_P(p).E_{\max} \mid p \in \bullet t\}$. e^t_{max} is also t's deadline.

When transition t is enabled at time $e^t_{\min}$, it starts at time s^t after a time period Δt where $s^t = e^t_{\min} + \Delta t$. There are three situations for s^t: 1) t starts as soon as it is enabled if $s^t = e^t_{\min}$; 2) when t is enabled, it starts after a period of time if $s^t = e^t_{\min} + \Delta t$ and $e^t_{\min} < s^t \le e^t_{max}$; 3) t cannot start if $s^t > e^t_{max}$. In this paper, we discuss situation 1 and leave out situation 3. The corresponding discussion for situation 2 is similar as situation 1, because situation 2's definition style of start time ($s^t = e^t_{\min} + \Delta t$) is similar to that of situation 1 ($s^t = e^t_{\min}$).

As stated above, transition t's completion time is $(s^t + \theta_T(t)) \in i_1 = (s^t + d^t_{\min}, s^t + d^t_{\max})$. If t's enabled time interval is $i_2 = (e^t_{\min}, e^t_{max})$, the firing of t is considered as satisfying deadlines only if t completes before its maximum enabled time e^t_{max}, namely, $(s^t + \theta_T(t)) \in i_3 = (s^t + d^t_{\min}, e^t_{max})$, as shown in Figure 3. Therefore, t's successful firing ratio is the probability integral of random variable $(s^t + \theta_T(t))$ over the interval of i_3.

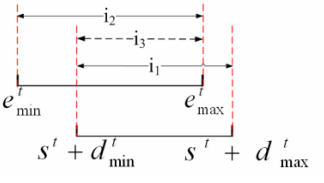


Fig. 3. Analyzing transition t's successful firing ratio

Definition 5. (Transition's successful firing ratio). *For a transition t in PTCWF-nets where* $\theta_T(t) \sim N(\mu_t, \sigma_t^2)$, *its successful firing ratio:*

$$r^t = P(s^t + d^t_{\min} < s^t + \theta_T(t) < e^t_{max}),$$
$$i.e.,\ r^t = P(d^t_{\min} < \theta_T(t) < e^t_{max} - s^t) \tag{1}$$

This ratio can be easily obtained if $\theta_T(t) \sim N(\mu_t, \sigma_t^2)$, because for a given random variable $x \sim N(\mu, \sigma^2)$, x's probability integral over interval (a, b) is [17]: $P(a<x<b)=$

$\Phi((b-\mu)/\sigma)$- $\Phi((a-\mu)/\sigma)$. Function $\Phi(y)$ is the standard normal distribution function where $y \sim N(0, 1)$.

Moreover, in Figure 2, the calculations of time variables $e^t_{\min}$, e^t_{max}, and s^t (line 6 to line 8) are related to token's arrival time of place p ($p \in \bullet t$). Given that, in the PTCWF-net, token's arrival time of the source place is a time constant (line 1). The algorithm calculates token's arrival time of other places by means of token's arrival time of their preceding places (line 11). In four workflow control structures, token's arrival time is calculated differently. In Table 3, we take place p_2 as our target to explain the calculation of token's arrival time.

Table 3. Token's arrival time of place p_2 in four workflow control structures

Workflow control structure	Token's arrival time of place p_2:
Sequential or Parallel	$a^{p_1} + \theta_P(p_1).E_{min} + \mu_{t_1}$
Selective	$\theta_F(f_1) \times (a^{p_1} + \theta_P(p_1).E_{min} + \mu_{t_1}) +$ $\theta_F(f_2) \times (a^{p_1} + \theta_P(p_1).E_{min} + \mu_{t_2})$
Iterative	$a^{p_1} + \theta_P(p_1).E_{min} + \mu_{t_1}$ for transition t_2, $a^{p_1} + (\theta_P(p_1).E_{min} + \mu_{t_1}) \times (\theta_F(f_1)/\theta_F(f_2)+1)$ $+(\theta_P(p_2).E_{min} + \mu_{t_2}) \times (\theta_F(f_1)/\theta_F(f_2))$ for transition t_3

– In **sequential** or **parallel** control structure, a^{p_2} is transition t_1's expected completion time ($s^{t_1} + \mu_{t_1}$) where $s^{t_1} = e^{t_1}_{\min} = a^{p_1} + \theta_P(p_1).E_{\min}$.

– In **selective** control structure, a^{p_2} is the probability synthesis of transition t_1 and t_2's expected completion time. t_1's probability of firing is $\theta_F(f_1)$ and its expected completion time is ($a^{p_1} + \theta_P(p_1).E_{\min} + \mu_{t_1}$). t_2's probability of firing is $\theta_F(f_2)$ and its expected completion time is ($a^{p_1} + \theta_P(p_1).E_{\min} + \mu_{t_2}$).

– In **iterative** control structure, a^{p_2} is different for p_2's output transition t_2 which is inside the loop body and t_3 which is outside the loop body. Statistically, each time t_3 fires, t_1 fires $(\theta_F(f_1)/\theta_F(f_2)+1)$ times and t_2 fires $(\theta_F(f_1)/\theta_F(f_2))$ times. Besides, in each t_1's firing, t_1's holding time includes its waiting time before firing $\theta_P(p_1).E_{\min}$ and its expected firing duration μ_{t_1}. Similarly, t_2's holding time is $(\theta_P(p_2).E_{\min} + \mu_{t_2})$ per firing. For t_2, a^{p_2} is t_1's first expected completion time: $a^{p_1} + \theta_P(p_1).E_{\min} + \mu_{t_1}$. For t_3, a^{p_2} is the time when the iterative firings of t_1 and t_2 complete.

4 Scheduling of Personal Worklist

In this section, we introduce step 2.1 and 2.2 of our approach in subsection 4.1 and 4.2, respectively.

4.1 Adding a New Activity Instance to an Actor's Personal Worklist

To perform scheduling on an actor's personal worklist, the newly added activity instance a is associated with six time-related variables: start time s^a, expected duration μ_a, deadline $e^a_{\max}$, successful execution ratio r^a, fixed deadline violation cost c^a_f, and probabilistic deadline violation cost c^a_p. The first four time variables can be obtained from the process analysis stage of the approach. c^a_f is the specified exception-handling cost of activity instance a when a violates its deadline. c^a_f is set as a constant here with consideration of various conditions in systems (e.g., context, resource utilization) and process instances (e.g., importance, urgency). c^a_f can also be a cost function such as exponential function. Detail discussion about the cost function is beyond the scope of this paper. c^a_p is the estimated deadline violation cost of activity instance a, which is the probability synthesis of (1- r^a) and $c^a_p := c^a_f \times (1- r^a)$.

Assumed that, before the insertion of activity instance a, worklist S contains (n-1) activity instances: $\{a_1, a_2, \ldots, a_{n-1}\}$ (n>1). After insertion, the execution order of activity instances in worklist S are rearranged according to activity instances' start time. Given that after rearrangement, activity instance a is the ith one in worklist S, thus there are (i-1) activity instances $\{a_1, a_2, \ldots, a_{i-1}\}$ ($1<i<n$) whose execution orders are before a, and (n-i) ones whose execution orders are later.

For any activity instance a_j ($i \le j \le n$) which is sorted equal or after the newly inserted activity instance in worklist S, a_j's start time and successful execution ratio should be recalculated. Given that, a_j's previous activity instance is a_{j-1}. a_{j-1} starts at time s^{a-1}, its expected duration is μ_{a-1}, its deadline is $e^{a-1}_{\max}$, so it completes before or equal to its deadline: $\text{Min}\{s^{a-1}+\mu_{a-1}, e^{a-1}_{\max}\}$. a_j's start time is decided by the later one of its the original start time and a_{j-1}'s completion time: $s^a=\text{Max}\{s^a, \text{Min}\{s^{a-1}+\mu_{a-1}, e^{a-1}_{\max}\}$ ($i \le j \le n$). Besides, a_j's successful execution ratio r^{a_j} should be updated along with a_j's new start time by means of formula 1 (definition 5, Section 3.2).

After updating activity instances' successful execution ratios in worklist S, the approach recomputes the joint successful execution ratio of activity instances in worklist S.

Definition 6. (Joint successful execution ratio of activity instances in worklist *S*). *There are n activity instances: {a_1, a_2, …, a_n} (n> 0) in worklist S after the insertion of a new activity instance. The joint successful execution ratio of these activity instances is the product of their successful execution ratios:*

$$r^s = \prod_{i=1}^{n} r^{a_j} \tag{2}$$

4.2 Scheduling of Worklist S

After recalculation of the joint successful execution ratio in worklist S, the approach compares this ratio with the safe threshold. If the ratio is greater than or equal to the

safe threshold, the worklist *S* is maintained for execution without any scheduling. Otherwise, a scheduling strategy is proposed to eliminate some activity instances from worklist *S* until the ratio meets the safe threshold. This safe threshold also represents the required stability of actor's execution.

The safe threshold is set through a win-win negotiation between the system manager and actor, which considers both requirements of systems (e.g., performance, quality of service, stability, and so on) and conditions of actor (e.g., proficiency level, payment rate, workload, project experience, and so on).

The proposed scheduling strategy first selects the activity instance with the smallest successful execution ratio to eliminate, so this strategy is named Smallest Ratio First (SRF) strategy. Figure 4 shows the pseudocode of our strategy, whose time complexity is $O(n^2)$ given that worklist S contains *n* activity instances.

There are other four fundamental scheduling strategies [13]: First In First Out (FIFO), Latest Deadline First (LDF), Smallest Cost First (SCF), and Longest Excepted Duration First (LEDF). Strategy FIFO first removes the activity instance which is lastly added to the personal worklist, and Strategies LDF, SCF, and LEDF first remove the activity instance with the latest deadline, smallest fixed deadline violation cost, and longest expected duration, respectively. The effectiveness of our strategy with respect to other four strategies is evaluated in the next section.

In the scheduling of worklist *S*, the removed activity instances are added to infeasible worklist *U* for exception handling, such as adjusting deadline or reemploying other actors. How exception handling should be triggered is outside the scope of this paper and it is often related to some other overall aspects of the WFMSs such as quality of service [12].

```
Input: feasible worklist S, infeasible worklist U, safe
threshold r^Safe
Output: Scheduled Worklist S and U
Method
1. num := n;    // initial worklist S has n activity instances
2. While worklist S's joint successful execution ratio r^S < r^Safe
   BEGIN
3.   FOR i := 1 TO num DO
     BEGIN
4.     Find the activity instance a_i with smallest successful
       firing ratio r^(a_i);
     END
5.   S := S \ {a_i};    // remove activity a_i from worklist S
6.   num= num-1;
     // feasible worklist S has (num-1) activity instances
7.   U := U ∪{a_i};  // add activity a_i to end of worklist U
8.   IF r^S ≥ r^Safe THEN RETURN;    //exit the scheduling strategy
   END
```

Fig. 4. Smallest Ratio First (SRF) scheduling strategy

After scheduling of worklist *S*, the approach recalculates the total deadline violation cost of activity instances in worklist *U*. The approach takes the probabilistic deadline violation cost as the measure of violation cost, because for an activity instance a_i, $c_f^{a_i}$ comprehensively considers both a_i's possibility of violating deadline $(1-r^{a_i})$ and a_i's fixed deadline violation cost $c_f^{a_i}$.

Definition 7. (Total deadline violation cost of activity instances in worklist *U*). *There are m activity instances: $\{a_1, a_2, \ldots, a_m\}$ $(m>1)$ in worklist U after the scheduling strategy. The total deadline violation cost of activity instances in worklist U is the sum of their probabilistic violations costs:*

$$c^U = \sum_{i=1}^{m} (c_f^{a_i} \times (1 - r^{a_i})). \tag{3}$$

5 System Implementation and Simulation Experiment

In this section, we implement our approach as a prototype system in a WFMS called Tsinghua InfoTech Product Lifecycle Management (TiPLM) workflow [20], and demonstrate two stages of our approach by means of concrete examples. The referenced time constraints and time parameters in these examples are extracted from TiPLM event log through applying statistical analysis [17], or set by consulting related workflow documents. Moreover, we evaluate the effectiveness of our scheduling strategy through large scale simulation experiments whose results are independent of specific platforms.

The first process analysis stage of the approach is realized by revising the TiPLM workflow model editor. In the model editor, a workflow process is described by a directed graph that has four different kinds of nodes: activity nodes, route nodes, start node, and end node. Nodes are connected by directed links to define different kinds of control flows. Figure 5 shows a screenshot of the TiPLM workflow model editor.

From TiPLM workflow, we extract a process instance of alteration of contract (Figure 6) to demonstrate step 1.1 and 1.2 of our approach. Step 1.1 converts the example process instance into a PTCWF-net, as depicted in Figure 7. The congruent relationship of elements between the example process instance and PTCWF-net is shown in Table 4. In addition, Table 5 presents time constraints and parameters for

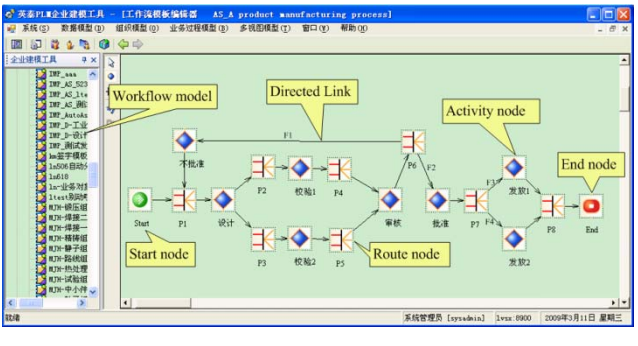


Fig. 5. TiPLM workflow model editor

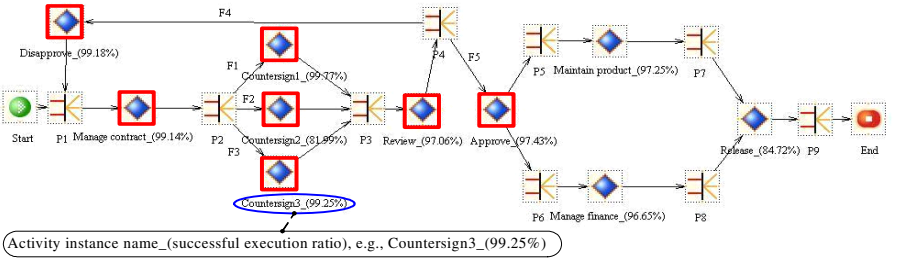


Fig. 6. Example process instance of alteration of contract

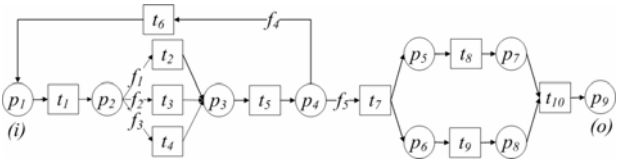


Fig. 7. Modelling the example process instance with a PTCWF-net

Table 4. Congruent relationship of elements between the example process and PTCWF-net

Route node	Place	Activity Node	Transition	Directed link	Arc
P_1	p_1	Manage contract	t_1	F_1	f_1
P_2	p_2	Countersign1	t_2	F_2	f_2
P_3	p_3	Countersign2	t_3	F_3	f_3
P_4	p_4	Countersign3	t_4	F_4	f_4
P_5	p_5	Review	t_5	F_5	f_5
P_6	p_6	Approve	t_6		
P_7	p_7	Disapprove	t_7		
P_8	p_8	Maintain product	t_8		
P_9	p_9	Manage finance	t_9		
		Release	t_{10}		

Table 5. Time constraints and parameters in the example PTCWF-net

p	$\theta_F(f).E_{\min}$	$\theta_F(f).E_{\max}$	T	μ_t	σ_t,	f	$\theta_F(f)$
p_1	1.02	6.98	t_1	5.52	0.18	f_1	0.33
p_2	5.11	68.27	t_2	57.11	1.97	f_2	0.24
p_3	1.47	57.86	t_3	61.26	2.06	f_3	0.43
p_4	2.74	12.85	t_4	58.12	2.02	f_4	0.36
p_5	3.17	17.34	t_5	52.73	1.92	f_5	0.64
p_6	3.24	15.72	t_6	9.52	0.24		
p_7	1.12	36.05	t_7	9.52	0.32		
p_8	1.72	37.27	t_8	13.24	0.48		
p_9	-	-	t_9	11.72	0.41		
			t_{10}	33.49	1.17		

Table 6. Example 15 activity instances and their related time variables

Activity instance name	Start time	Expected duration	Deadline	Successful execution ratio (%)	Fixed deadline violation cost	Probabilistic deadline violation cost
a_1	30.79	5.34	36.46	96.69	70.63	2.34
a_2	28.35	6.25	35.07	98.54	30.96	0.45
a_3	61.12	6.63	68.16	96.58	85.05	2.91
a_4	5.74	9.44	15.93	98.81	60.11	0.72
a_5	15.37	7.00	22.95	99.11	26.82	0.24
a_6	7.91	7.78	16.33	99.17	17.61	0.15
a_7	80.07	11.59	92.76	99.58	95.96	0.41
a_8	33.29	8.42	42.33	98.23	72.12	1.27
a_9	69.24	14.71	84.82	96.31	13.39	0.49
a_{10}	3.59	5.81	9.90	99.43	45.98	0.26
a_{11}	2.41	7.85	10.71	95.64	17.73	0.77
a_{12}	86.42	9.48	96.53	97.38	54.82	1.24
a_{13}	22.53	14.89	38.28	95.66	66.29	2.88
a_{14}	82.48	11.03	94.29	98.22	62.16	1.11
a_{15}	12.10	8.02	20.60	95.83	16.16	0.67

the example PTCWF-net, and the time unit is hour (referring to transitions and places). For arcs, there are only five time parameters of arc f_1 to f_5 are given because other arcs' time parameters are 1. Step 1.2 analyzes every activity instance's successful execution ratio in the example process instance, as shown in Figure 6.

Next, we use Figure 1's personal worklist (Section 1) to illustrate the second worklist scheduling stage of our approach. Table 6 shows 15 activity instances in this worklist and their six related time variables. Time unit for 'Start time', 'Expected duration', and 'Deadline' is a same time unit, for 'Successful execution ratio' is percentage, and for the last two variables is a same cost unit. For the convenience of focusing on the scheduling results of five strategies, in Table 6, these 15 activity instances' start time, expected durations, and successful execution ratios are assumed to randomly range from 0 to 100, 5 to 15, and 95% to 100%, respectively (these assumptions are still applicable in the following simulation experiments).

Given that, in the example the safe threshold is 80%. The initial personal worklist is empty and 15 activity instances are added to the worklist from a_1 to a_{15}.

Figure 8 shows the scheduling results of five scheduling strategies after adding the first two activity instances a_1 and a_2. When a_1 is inserted to worklist S, worklist S only contains one activity instance whose successful execution ratio is 96.69% and this ratio is above the safe threshold. When a_2 is inserted to worklist S, the joint successful execution ratio of a_1 and a_2 is below 5%, so worklist S needs scheduling. In the scheduling, strategy SRF and LDF eliminate a_1 and the other three strategies remove a_2. Hence in strategy SRF and LDF, total deadline violation cost of activity instances in worklist U is a_1's probabilistic violations cost: 2.34. Similarly, in the other three strategies, this cost is a_2's probabilistic violations cost: 0.45.

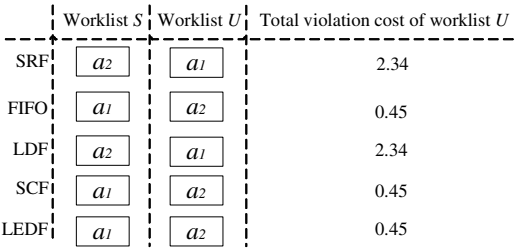


Fig. 8. Example scheduling results after adding the first two activity instances

Figure 9 shows the scheduling results of five strategies after adding all 15 activity instances. Among five scheduling strategies, worklist *S* in SRF strategy has the largest number of activity instances (sorted according to their start time), hence activity instances in worklist *U* have the smallest total deadline violation cost.

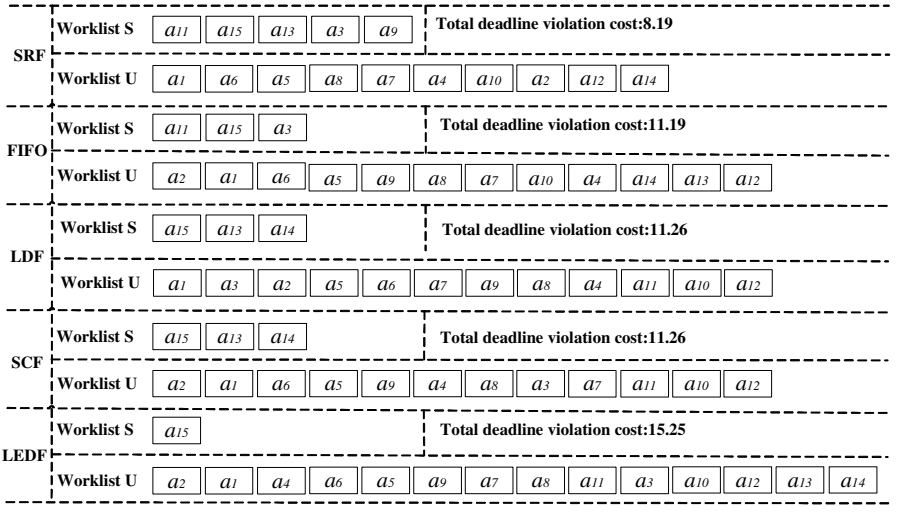


Fig. 9. Example scheduling results after adding all 15 activity instances

In the following, we compare our SRF strategy with the other four strategies through large scale simulation experiments. Each experiment includes 15 steps, and every step adds an activity instance to the personal worklist. Hence from step 1 to step 15, the personal worklist contains 1 to 15 number of activity instances, which reflect the workload of actor from light to heavy. In each step, five strategies are applied to schedule worklist *S* to meet the given safe threshold. Then the total deadline violation cost of activity instances in worklist *U* is recalculated. In addition, we set four safe thresholds: 60%, 70%, 80%, and 90%. These safe thresholds represent the stability requirements of actor's execution from low to high. To investigate the statistic performance, for each safe threshold, the experiment is executed for 1000 times. Therefore, in our simulation, a large scale of 4000 independent experiments have been

executed. For fairness, the total deadline violation cost in each step of the experiment is specified as the average value of all 1000 ones.

Figure 10 (a) to Figure 10 (d) show the experiment results for safe threshold 60% to 90%, respectively. Experiment results show that for all four safe thresholds, the total deadline violation cost of our strategy is smaller than that of the other four strategies. Besides, when the worklist becomes larger by inserting more activity instances, the gap of total violation cost also becomes larger between our strategy and the other four strategies. Furthermore, our strategy performs even better when safe threshold becomes smaller (i.e., from 90% to 60%), which means our strategy can reduce more deadline violation costs than other strategies when actors are required lower stability of execution. To conclude, our scheduling strategy is almost always advantageous to minimize deadline violation cost than other commonly used scheduling strategies, especially for worklists with heavy workload and low stability.

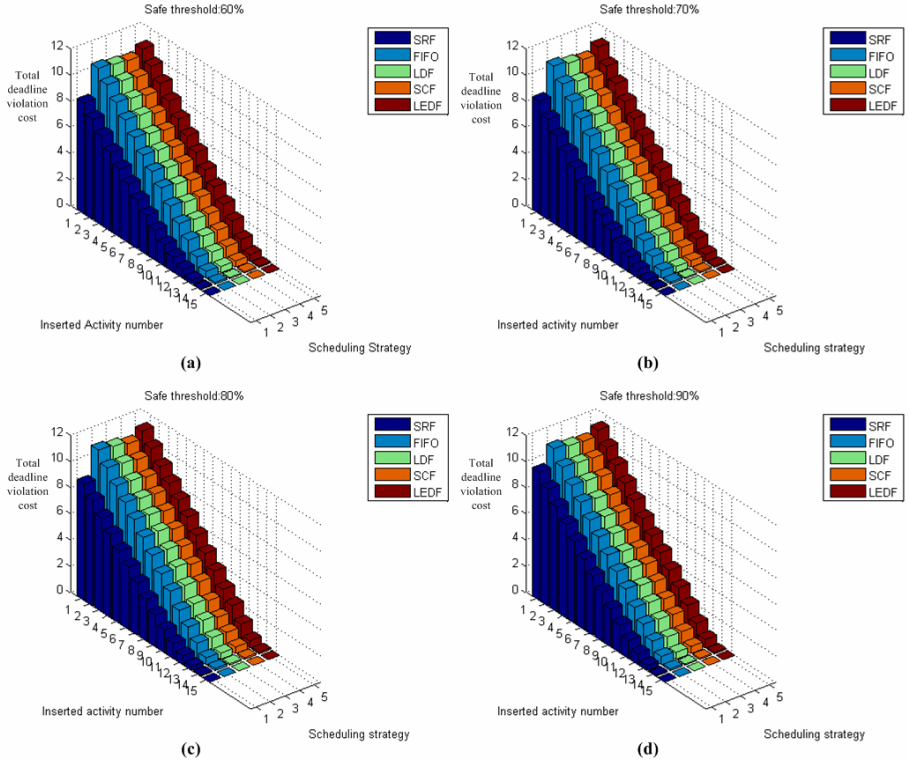


Fig. 10. Simulation experiments of five scheduling strategies

6 Related Work

In this section, we review some related work on workflow scheduling and personal worklist management. To the best of our knowledge, the work that is most similar to ours is presented in [10, 11, 21]. In [10], Eder et al. discuss main problems in the

management of personal worklist. Their approach first employs duration histograms [21] to model workflow process and generate time plan for activity instance. Based on time plan, it performs a simple Earliest Deadline First (EDF) scheduling strategy to sort activity instances in the personal worklist. Furthermore, in [11], they envision two new scheduling strategies to sort activity instances without providing much detail.

Our work differs from Eder et al.'s work [10, 11, 21] mainly in two aspects. First, their work introduces the notion of time histograms to describe a number of activity durations together with their probabilities. Hence, they present deterministic analysis result as several satisfaction states and their probabilities. In contrast, our work considers continuous distributed activity durations and analyzes activity instance's stochastic probabilities of satisfying deadline. Secondly, Eder et al.'s scheduling strategy makes two rigorous assumptions: 1) worklist is safe only if its probability of satisfying deadlines is 1.0, i.e., the safe threshold is 100%; 2) execution intervals of activity instances cannot overlap in the worklist. In comparison, our approach relaxes the first assumption by setting the safe threshold through a negotiating process between the system manager and actor. We also consider execution intervals' overlap in our scheduling strategy. Consequently, our approach can provide flexible management of personal worklist which adapts to dynamic workflow environment.

Distinguishing from our work which views workflow scheduling from individual workflow actors' viewpoint, recent research mainly concerns macro-level scheduling from workflow system's point of view, i.e., they concentrate on assigning tasks among multiple actors [22]. In [2, 4], Baggio et al. discuss problems of applying existing scheduling techniques in WFMSs. They use a "Guess and Solve" scheduling approach and other basic scheduling strategies to minimize the late process instances in WFMSs. In [5, 6, 7], Combi et al. focus on temporalities in the conceptual organizational model and task assignment policies. They propose a temporal organizational model, which extends traditional organizational models, to describe different constraints of resources such as availability constraints and deadline constraints. Based on these constraints, they design a scheduling algorithm to evaluate the priority of tasks according to their expected deadlines. In addition, in [8], Senkul et al. mention workflow scheduling in a single process instance. They propose an architecture to model resource information and resource allocation constraints. Then, they employ a scheduler model to incorporate a constraint solver, so as to find proper resource assignments and to schedule workflows with resource allocation constraints.

7 Conclusions

In this paper, we have proposed a two-stage probabilistic approach, which provides stochastic scheduling to flexibly manage individual actor's personal worklist. To be specific, at the first stage, the approach analyzes every activity instance's continuous probability of satisfying deadline. This probability acts as an accurate guidance for personal worklist scheduling. At the second stage, an innovative scheduling strategy is proposed. Every time new activity instances are added to an actor's personal worklist, this strategy maintains a feasible worklist S according to the negotiated safe threshold, and an infeasible worklist U which is comprised of deadline-violating activity instances. The strategy targets at minimizing the overall deadline violation cost for activity instances in worklist U, thus achieving improved system performance.

Moreover, we use concrete examples and large scale simulation experiments to verify the practicability of our work in facilitating personal worklist management under dynamic workflow environment.

In our future work, based on the probabilistic approach proposed in this paper, we plan to investigate the impact of managing individual actor's worklist on other actors' worklists, or even the overall system performance. Besides deadline violation cost, we will discuss other performance measures, such as throughput, turnaround time, and so on.

References

1. Eder, J., Panagos, E.: Managing Time in Workflow Systems. In: Workflow Handbook 2001, Future Strategies INC, in association with Workflow Management Coalition (2000)
2. Baggio, G., Wainer, J., Clarence, E.: Applying scheduling techniques to minimize the number of late jobs in workflow systems. In: Proceedings of the 2004 ACM symposium on Applied computing, pp. 1396–1409. ACM Press, Nicosia (2004)
3. van der Aalst, W.M.P., van Hee, K.M.: Workflow Management: Models, Methods, and Systems. The MIT Press, Cambridge (2002)
4. Tramontina, G.B., Wainer, J.: Modeling the behavior of dispatching rules in workflow systems: A statistical approach. Groupware: Design, Implementation, and Use, 208–215 (2005)
5. Combi, C., Pozzi, G.: Temporal conceptual modelling of workflows. In: Song, I.-Y., Liddle, S.W., Ling, T.-W., Scheuermann, P. (eds.) ER 2003. LNCS, vol. 2813, pp. 59–76. Springer, Heidelberg (2003)
6. Combi, C., Pozzi, G.: Towards temporal information in workflow systems. In: Olivé, À., Yoshikawa, M., Yu, E.S.K. (eds.) ER 2003. LNCS, vol. 2784, pp. 13–25. Springer, Heidelberg (2003)
7. Combi, C., Pozzi, G.: Task scheduling for a temporal workflow management system. In: Proceedings of the Thirteenth International Symposium on Temporal Representation and Reasoning (TIME 2006), pp. 61–68. IEEE Press, Budapest (2006)
8. Senkul, P., Toroslu, I.H.: An architecture for workflow scheduling under resource allocation constraints. Information Systems 30, 399–422 (2005)
9. Moore, C.: Common Mistakes in Workflow Implementations. Giga Information Group, Cambridge (2002)
10. Eder, J., Pichler, H., Gruber, W., Ninaus, M.: Personal schedules for workflow systems. In: van der Aalst, W.M.P., ter Hofstede, A.H.M., Weske, M. (eds.) BPM 2003. LNCS, vol. 2678, pp. 216–231. Springer, Heidelberg (2003)
11. Eder, J., Eichner, H., Pichler, H.: A probabilistic approach to reduce the number of deadline violations and the tardiness of workflows. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM 2006 Workshops. LNCS, vol. 4277, pp. 5–7. Springer, Heidelberg (2006)
12. van der Aalst, W.M.P., Rosemann, M., Dumas, M.: Deadline-based escalation in process-aware information systems. Decision Support Systems 43, 492–511 (2007)
13. Brucker, P.: Scheduling algorithms. Springer, Heidelberg (2007)
14. van der Aalst, W.M.P.: The application of Petri nets to workflow management. Journal of Circuits Systems and Computers 8, 21–66 (1998)
15. Russell, N., Ter Hofstede, A., Edmond, D., van der Aalst, W.M.P.: Workflow resource patterns. Technical Report, Eindhoven Univ. of Technology (2004)
16. van der Aalst, W.M.P., van Hee, K.M., Reijers, H.: Analysis of discrete-time stochastic Petri nets. Statistica Neerlandica 54, 237–255 (2000)

17. Stroud, K.A.: Engineering Mathematics, 6th edn. Palgrave Macmillan, New York (2007)
18. Law, A.M., Kelton, W.D.: Simulation Modelling and Analysis, 4th edn. McGraw-Hill, New York (2007)
19. Zhou, R., Hansen, E.A.: Breadth-first heuristic search. Artificial Intelligence 170, 385–408 (2006)
20. Introduction to Infotech Product Lifecycle Management Solution (in Chinese), http://www.thit.com.cn/chanpinshijie/TiPLM.htm
21. Eder, J., Pichler, H.: Duration histograms for workflow systems. In: Proc. of the Conf. on Engineering Information Systems in the Internet Context 2002, pp. 239–253. Kluwer Academic Publishers, Dordrecht (2002)
22. Yingbo, L., Jianmin, W., Jiaguang, S.: Using Decision Tree Learning to Predict Workflow Activity Time Consumption. In: Proceedings of the Ninth International Conference on Enterprise Information Systems (ICEIS 2007), pp. 69–75 (2007)