IEEE Transactions on Systems, Man, and Cybernetics-Part A:Systems and Humans,
Vol.33, No.2, March 03, pp.179-193

Timing Constraint Workflow Nets for Workflow Analysis

JianQiang Li [a] YuShun Fan [a] MengChu Zhou [b]

[a]Department of Automation, Tsinghua University, Beijing 100084

[b]Department of ECE, New Jersey Institute of Technology, Newark, NJ07102-1982 USA

lijq99@mails.tsinghua.edu.cn, fan@cims.tsinghua.edu.cn, zhou@njit.edu

Abstract: The analysis of the correctness and rationality of a workflow model plays an important role in the research of workflow techniques and successful implementation of workflow management. This paper points out the relevant problems in the verification and analysis of a workflow model. It discusses two important properties: schedulability and boundedness of a workflow model considering timing constraints. To specify the timing constraints, WorkFlow net [1] is extended with time information, leading to Timing Constraint WorkFlow net (TCWF-net). This paper presents a model mapping method to convert a Directed Network Graph (DNG) based workflow model, which is built by a graphic process modeling language [2] extended with time information, into a TCWF-net. It then discusses the schedulability verification and synthesis of TCWF-nets. Due to the fact that there is no iteration in the TCWF-net obtained through model transformation, an algorithm to decompose the acyclic and free-choice TCWF-net into a set of *T*-components is presented. Then, to avoid the run-time congestion of a workflow model, which is the key to the performance management of the business process automation a boundedness verification method is derived. The usefulness of the research results is illustrated by an example.

Keywords: Workflow model, Petri nets, Boundedness, and Schedulability

1. Introduction

Workflow management is a key technology in supporting business process reengineering and an effective means realizing full or partial automation of a business process [3]. Despite the abundance of workflow management systems developed for different types of workflow based on different paradigms [4-7], the lack of rigorous theoretic foundation and then effective model verification and analysis methods has blocked workflow techniques' research and application.

Workflow specifications address many issues including process control, resource,

information, and function perspectives. Hence, we know from research that the rationality and correctness analysis should be carried out from four aspects that are relevant for workflow modeling and workflow execution: process control logic, timing constraint logic, resource dependency logic, and information dependency logic. The objective of correctness analysis of process control logic is to avoid the deadlocks or structural conflicts in the execution of a workflow model because of the errors in its process control. Some verification and conflict detection methods have been discussed in [2, 8-9]. Resource dependency logic verification focuses on the proof of correctness of the static or dynamic resource allocation rules and consistence with the process control logic. The information dependency logic, however, indicates the internal consistence of a workflow-related data and correct temporary relation among different workflow application data. The timing constraint verification and analysis deal with the temporal aspects of a workflow model such as deadlines, variable calendar windows, time scales, and alerting mechanisms for overdue actions. It also includs the schedulability analysis of the constituent activities of a workflow model and its boundedness verification in the case of multi-workflow-instances running concurrently, which is discussed in this paper. Note that the timing constraint verification and synthesis should be conducted after the process control verification is done, which means that the workflow model considered here is free of structural conflicts (deadlock).

In order to improve the efficiency and quality of a business process, each activity and the precedence relations between different constituent activities in a workflow model should have reasonable timing constraints according to the transaction instance arrival time, the required service quality, and efficient resource management. Our investigation shows that mainly two kinds of timing constraints should be considered: *external* and *internal timing constraints*. The former follows implicitly from control dependency of a workflow schema. It causes the buffer time that is handled by a workflow manager or workflow engine. For example, an activity can only start 5 minutes after its preceding activities have finished, or it must start 10 minutes after another specific activity ends. To specify this kind of timing constraint, the concepts of Lower Bound Constraint (*LBC*) and Upper Bound Constraint (*UBC*) [10] should be introduced and used in this paper. Internal timing constraint, which is managed by a resource agent (e.g. a software system or human) that is responsible for the enactment of an activity, is embedded in

the description of every individual activity. It includes the defined execution duration and the executable time span of this activity. Addressing the issues of how to verify the correctness of a workflow model from the time dimension and select rationally the time parameters at the build-time is important in realizing efficient workflow management.

When a workflow model is deployed in practice, the state of an activity in a workflow instance may include *initiated, enabled, ready, running, suspended, dead, error,* and *end*. However, in the time dimension verification and analysis of workflow models, only the temporal behavior of task execution, which is defined at the build time, is concerned. Thus we consider only the *enabled, ready, running*, and *end* states of activities in this paper.

The workflow process definition models the life-cycles of all the transaction instances. On one hand, the workflow engine interprets and maintains a workflow instance for each transaction instance in the running time environment. Each activity in the workflow instance must be allocated enough time to complete its execution with respect to imposed timing constraints. On the other hand, a workflow model deployed in practice has a stable transaction instance input rate, which means that there are many transaction instances handled concurrently according to the same workflow model. When the input rate of an activity is greater than its output rate, the congestion or overflow occurs. To avoid it in the running environment, the execution durations of the coordinated activities in the process should conform to certain constraints. Therefore, our time dimension workflow model verification and analysis method is addressed through two levels: (1) schedulability verification and synthesis of a workflow model; and (2) boundedness verification of a workflow model in the environment of multi-workflow instances running concurrently.

Petri Net (PN) originated from the early work of Carl Adam Petri [11] has found many applications in computer science. Because of their formal semantics, local state-based system description, and abundant analysis techniques [12], their use as a mathematical foundation for the formal analysis of workflow models is attractive to many researchers of workflow techniques. Since Zisman [30] used PN to model workflow processes for the first time in 1977, researchers have proposed their own techniques based on PN to model workflow. Some [3], [37] recognize the adaptability problem inherent to workflows, i.e., the frequently and/or radically changing character due to changing business rules. Thus, the work [1], [4], [14], [31], [32], [33],

[34] focuses on how process control or data flow is modeled by PN to improve the adaptability of a workflow model in a changing market environment. Table 1 highlights several proposed PN classes for workflow modeling. However, despite the popularity of using PN for business process specification, Han [31] warns that PN cannot apply directly for modeling workflows due to their fixed structure.

Table 1. Overview of the proposed PN classes for workflow modeling

Petri net class	Brief description
Information Control Nets (ICN) [4]	By adding a complementary data flow model, generalizing control flow primitives and simplifying semantics, ICN is a Colored Petri net variant intended to represent control and data flow of office workflow. A message-based exception handling mechanism for ICN is provided to improve the flexibility of a workflow system.
Workflow-nets (WF-nets) [1]	With one input place (ε) and one output place (θ), indicating the beginning and end of the modeled business procedure, WF-nets require that every transition (place) must be located on a path from ε to θ. They are suitable not only for the representation and valididation but also for the (process control logic) verification [8],[9] of workflow procedures.
Reconfigurable Nets [14]	As an extension of WF-nets, a Reconfigurable Nets consists of several Petri nets which constitute the different possible configurations for some mode of operation to support dynamic changes and realize self-modification of a workflow system.
Modular Process Nets [32]	Modular Process Nets are based on a hierarchical module concept and the constructs for synchronous and asynchronous communication between interpreted nets and their environment as a framework for flexible workflow modeling and enactment, and can be described as Element Net systems with minimal syntactic extension.
Element Net system [33]	By decomposing a WFMS into two basic components, namely a WF model and a WF Execution Model, only a subclass of Element Net System (EN-system) are needed to describe a WF model. Thus the WF Execution Module can be implemented with enhanced flexibility and adaptability.
Higher Order Object Nets (HOON) [31]	Based on Modular Process Nets, the structures of the organisation and resource configuration are explicitly embodied in HOON. The net model and its enviroment are arranged in a client/server manner.
Predicate Petri nets (PPN) [34]	Using inter-task dependencies to specify internal structure of a workflow, PPN captures the relationships among subtransactions within a flexible transaction model.

Other workflow researchers have noticed the importance of workflow model analysis in supporting business process reengineering and successful workflow management. Different

kinds of PNs are employed for correctness and rationality analysis of a workflow model. Because the control flow is at the heart of workflow specification, many researchers have addressed the correctness verification of the process control of a workflow model. Some transformation rules between WF-nets are proposed to allow workflow designers to modify workflow structures while preserving its correctness and consistency [9]. A WF-net-based verification method [38] for workflow Task Structures is presented in [38]. The process verification and consistency checking techniques are discussed in [13] based on ordinary P/T nets and the technique presented in [9]. In [46], an algebraic technique based on Petri nets is proposed to check the local consistency of a workflow model and, using a composition theorem, it can be used to verify the inter-organization workflow model. More work can be found in [39], [40], [41]. Colored Petri Net (CPN) is used for the formal specification of a workflow process, and simulation has been used to verify the coordination between workflow activities [45]. However, although the literature on workflow has consistently stressed the importance of time [3-5, 7, 10, 27, 42-43], only a few papers address the application of PN to temporal analysis of a workflow model. Through extending ordinary P/T nets with an interval function and a timestamp function to model absolute as well as relative time, existing PN verification approaches are employed [13] to test whether it is feasible to execute a workflow with specified temporal constraints. In [48], a set of linear reasoning algorithms for commonly used workflow patterns is presented to investigate the temporal properties of a workflow with timing constraints of deterministic intervals. Performance analysis of workflow is research topic yet to be given the importance it deserve [3, 7, 35-36, 42-50]. All the routing constructs of a workflow are mapped into a higher-level Stochastic PN (SPN), then throughput time of the process is analytically computed [47]. Based on four performance equivalent formulae, an approximate performance analysis method of a workflow is presented in [49]. These two techniques both assume the infinite availability of resources in the workflow configuration. Generalized Stochastic Petri Nets (GSPN) are used to model workflow [35, 44], and then a method based on a continuous time Markov chain (CTMC) is used to obtain upper bounds of the execution performance. A simple GSPN, which is a so-called *load equivalence aggregation* (LEA) model, has been developed in [36], and then the model is simulated using a Coloured GSPN (CGSPN) to obtain some performance related measures of human resources in a workflow. Commonly, the techniques applying PN in the domain of

workflow exploit the correspondence between the special kind of time-related PN and the dynamic behavior of workflow systems, and use existing PN analysis techniques for workflow analysis. For more literature on the application of PN to workflow modeling and analysis, readers can refer to the two workshops on workflow management [42], [43, 50].

Considering Han's statement as mentioned above and the fact that most of the commercial workflow products use different kinds of DNGs instead of PN for their model specification, the mapping from such DNG to PN are worthy to investigate. Also, to realize systematic time dimension verification and analysis of workflow models, it is necessary to incorporate all the relevant timing information into PN-based workflow models.

The next section introduces a graphic process modeling language [2] extended with timing information and the concept of TCWF-nets by incorporating timing information into WF-nets. Section 3 proposes a model mapping method from a DNG-based workflow model built by the extended basic process modeling language to TCWF-net. Section 4 presents the schedulability verification methods and heuristic rules for the timing synthesis of a workflow model. Based on that almost all workflow models have a free-choice characteristic [2], [9], Section 5 provides an effective algorithm decomposing free-choice and acyclic TCWF-net into a set of *T*—components and a boundedness verification method. Section 6 presents a case study. Finally, Section 7 makes conclusions.

2. Basic concepts

A basic process modeling language [2] based on a standard process definition notation, which is proposed by Workflow Management Coalition (WfMC) [15], can be used to represent the components of a workflow in a simple and direct way. In this language, processes are modeled using two types of objects: node and transition. A node is classified into two subclasses: task and choice/merge coordinator. The task, graphically represented by a rectangle, represents the work to be done to achieve some objectives. It can be used to build implicitly sequence, fork, and synchronous structures. The task is further classified into four types: activity, sub-process, block, and null task, which are necessary for the process modeling. However, for simplicity, all kinds of tasks are treated only as activities in our proposed model mapping method. The choice/merge coordinator, graphically represented by a circle, is used to build explicitly choice

and merge structures. A transition linking two nodes in a graph is graphically represented by a directed edge and used to specify the execution order and flow between its tail and head nodes. Figure 1 shows three modeling objects. Because the iteration structure is nested in a task that has an exit condition defined for iterative purposes, a DNG-based workflow model built by the basic process modeling language must be structurally acyclic. Refer to [2] for more details.

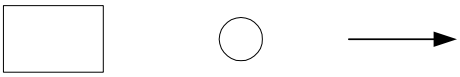


Figure 1. Graphic representation of three modeling objects

In order to incorporate the necessary timing information into a workflow model, we need to extend this basic process modeling language by considering its internal and external timing constraints.

As mentioned before, the internal timing constraint defined in an activity refers to execution duration and executable time span, where the executable time span managed by a responsible resource agent ranges from the time of task allocation to allowable latest completion time. Given a workflow model, designers can assign execution duration and executable time span (during which the activity can be executed) to every individual activity based on their experience and expectation from the past execution.

In order to specify external timing constraints, i.e., the temporal dependency relations between different activities, the concepts of *LBC* and *UBC* need to be introduced. Assuming that *A* and *B* represent two activities, *LBC*(*A*, *B*, *DL*) states that the duration between source event happened in running time of activity *A* and destination event happened in the running time of activity *B* must be greater than or equal to a lower bound time value *DL*. *UBC*(*A*, *B*, *DU*) demands that the time distance between source event happened in the running time of activity *A* and destination event happened in the running time of activity *B* must be smaller than or equal to an upper bound time value *DU*. Here, we call *A* and *B* as source and destination activities, respectively. For simplicity, *LBC*/*UBC* used in this paper refers only to the timing constraints between the *end execution event* of the source activity and *start enabled event* of the destination activity. Assuming that activity *A* completes its execution at time T_0 and the execution duration of activity *B* is *D*(*B*), and the timing constraints of *LBC*(*A*, *B*, *DL*) and *UBC*(*A*, *B*, *DU*) are imposed on *B*, we define the enabled time span of activity B as (T_0+*DL*, T_0+*D*(*B*)+*DU*). These

timing constraints are assigned according to the relevant organizational rules, laws, commitment, technical demands, and so on. They can also be selected based on the workflow performance requirement.

A merge node is only used for the description of the logic relation between its input and output nodes. Thus its enabled time span and execution duration are both set to zero. Then B with $LBC(A, B, DL)$ or $UBC(A, B, DU)$ cannot be a merge node. If $LBC(A, B, DL)$ or $UBC(A, B, DU)$ is needed, in which B is a merge node, a dummy activity is used to specify this situation. It means that the transaction instance routes out as soon as it reaches a merge node. However, the choice node may include actions such as the output path selection. Then it can be treated as an activity and the timing constraint mentioned above can be imposed on it. Also, the choice node's output path selection can be specified through assigning different timing constraints (executable time span) to different succeeding paths. For the sake of simplicity, we assume that all the time information is given in a same time unit.

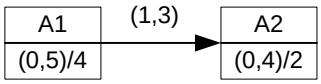


Figure 2. An example of the timing constraints in DNG-based workflow model

A simple example in Fig. 2 is used to show the specification of time information in a DNG-based workflow model. (*LBC*, *UBC*)=(1,3) on the edge from A_1 to A_2 specifies the temporal dependency relation between activities A_1 and A_2 as an external timing constraint. A_2's (0, 4)/2 specifies A_2's internal time constraints, i.e., its defined executable time span is $[T_1+0, T_1+4]$ if it starts enabling at time T_1; and its execution duration is 2. A_1's internal time constraints can be interpreted similarly. If A_1 completes its execution at T_0, the start enabled time span of A_2 is $[T_0+1, T_0+3]$, and A_2's enabled time span is $[T_0+1, T_0+2+3]$. In the run-time environment, because of the timing constraint imposed by the enabled time span, A_2's actual executable time span will be $[T_1+0, Min\{T_0+2+3, T_1+4\}]$, where $T_0+1 \le T_1 \le T_0+3$.

Petri nets as a design language for the specification of a complex workflow, as well as a powerful analysis technique for the correctness of workflow procedures are discussed in [1, 4]. Some basic concepts about PN [11], [17] are given as follows:

Definition 1: $PN = (P, T, F)$ is a free-choice PN iff $\forall\ t_1, t_2 \in T$, ${}^\bullet t_1 \cap {}^\bullet t_2 \neq \phi$ implies ${}^\bullet t_1 = {}^\bullet t_2$; and it is a marked graph iff each place p has exactly one input and one output transition.

Definition 2: $PN = (P, T, F)$ and $PN_1 = (P_1, T_1, F_1)$ are two PN. PN_1 is a subnet of PN iff $P_1 \subseteq P$, $T_1 \subseteq T$ and $F_1 = F \cap ((P_1 \times T_1) \cup (T_1 \times P_1))$. PN_1 is generated by T_1 iff $P_1 = {}^\bullet T_1 \cup T_1^\bullet$ (where the presets and postsets are taken w.r.t. F). It is called a T-component of PN iff PN_1 is the subnet generated by T_1 and, $\forall p \in P_1$: $|{}^\bullet p \cap T_1| \leq 1 \wedge |p^\bullet \cap T_1| \leq 1$.

A PN modeling the process aspect of workflow is called a WorkFlow net [1].

Definition 3: A Petri net PN is called WorkFlow net (WF-net) if and only if:

(1) PN has two special places: ε and θ. Place ε is a source place: ${}^\bullet \varepsilon = \phi$; Place θ is a sink place: $\theta^\bullet = \phi$.

(2) If we add a new transition t to PN which connects place θ with ε, namely, ${}^\bullet t = \{\theta\}$, $t^\bullet = \{\varepsilon\}$, then the resulting PN is strongly connected.

A WF-net has *proper termination* property if starting from the initial state (with only one token in place ε), it is always possible to reach the state with only one token in place θ. A WF-net with proper termination property is *sound* if it has no dead transitions, i.e., for each transition t, it is possible to reach (starting from initial state) a state where t is enabled. If there are no structural conflicts in a workflow model built by the basic process modeling language mentioned above, its corresponding WF-net must be sound [1].

Obviously, a WF-net gives only process control specification of a workflow model. To realize the time dimension verification and analysis of a workflow model, its temporal behavior should be specified, and then some extensions with time information to the WF-net are needed.

Different ways exist to introduce time into PN. Timed Petri nets treat a timing constraint as a single delay [18-20]. Time Petri nets treat a timing constraint as a delay pair consisting of lower and upper bounds [21-23]. Timing Constraint Petri Net (TCPN) [16], which extends PN by adding *minimum*, *maximum*, and *durational* timing constraints to places or transitions, synthesizes all the timing constraints considered in previous two cases. Considering the timing constraints to be specified in a workflow model, we propose to use TCPN for workflow modeling and analysis.

In TCPN, a time pair $[d_{min}(x), d_{max}(x)]$ is associated with node $x \in P \cup T$. $[d_{min}(p), d_{max}(p)]$ denotes the period that p can enable its output transition after a token arrives. The token enabled time of p is defined as $[K_a(p) + d_{min}(p), K_a(p) + d_{max}(p)]$, where $K_a(p)$ is the token arrival time at p. $[d_{min}(t), d_{max}(t)]$ represents the period that t is fireable after it is enabled. If p is the only input

place of t, t's fireable duration is determined collectively by $K_a(p)$, $d_{min}(p)/d_{max}(p)$ and $d_{min}(t)/d_{max}(t)$ [16]. A transition is schedulable means that it is firable and can complete its firing successfully. The formal semantics of TCPN can be used in a workflow model to specify naturally the situations such as the exception handling or selection of a succeeding routing path. In TCPN, all the tokens (denote as *TK*'s) used for enabling a transition will be preserved during the transition's firing. If all the transitions enabled by *TK*'s fail to complete their firing, *TK*'s will be trapped in their corresponding place. This kind of state evolving mechanism together with the relative and absolute time mode, as well as the weak firing rules make the TCPN particularly suitable for the specification about the execution of a workflow model in the run-time environment.

Based on the concept of a WF-net and TCPN, the definition of Timing Constraint WorkFlow net（TCWF-net）is given below:

Definition 4: A TCWF-net is a four tuple <*WF-net*, *C*, α, *M*>, where:

WF-net=(P, T, F) is a WorkFlow net;

$P=\{p_1, p_2, \cdots, p_m\}$ is a set of places representing the state of a transaction instance or the condition of its output transitions;

$T=\{t_1, t_2, \cdots, t_m\}$ is a set of transitions representing activities of workflow;

F is a set of directed arcs linking places and transitions, and used to describe precedence relations among activities;

C is a set of non-negative real number pairs [d_{min}, d_{max}] related to each transition or place, which is used to represent the imposed timing constraints of an activity or system state;

α is a set of firing delays associated with transitions, where $\alpha(t)$ represents the execution duration of transition t mapped from its corresponding activity;

M is a set of m-dimensional markings where $M(p)$ denotes the number of tokens representing the number of transaction instances in p.

What we should note here is that a transition in the TCWF-net corresponds to an activity of workflow; however, a transition defined in the basic process modeling language mentioned above is used to specify the execution order and flow between its tail and head nodes.

In addition, $E_F(t)$ ($L_F(t)$) denotes the earliest fireable beginning time (latest fireable ending time) of t; $E_E(t)$ ($L_E(t)$) denotes the earliest enable beginning time (latest enable ending time) of

t; $F_{begin}(t)$ ($F_{end}(t)$) denotes the time at which t begins (ends) firing.

It should be noted that a TCWF-net is used to specify the dynamic behavior of the life-cycle of transaction instances. For the verification and analysis of a workflow model, we have to consider the mapping from a workflow model built by the extended process modeling language to a TCWF-net.

3. Model mapping

According to the semantic properties of workflow models built by most of the workflow modeling tools (see e.g. [1], [5]), we know that most of them enjoy the free-choice characteristic. We use Fig. 3 to explain why a practical workflow model should exhibit this characteristic. Assuming that a transaction instance leads to two tokens (branches of the same instance) in nodes D and A respectively in Fig. 3(a). Because a transaction instance's routing depends only on the coordination between its attributes and the workflow control data, D can chooses R_1 as its output path and A chooses R_3. Then synchronous activity B will not take place and the instance will not terminate successfully. The same result follows from Fig. 3(b) if D chooses R_1 and A chooses R_3. Therefore, such non-free-choice structures as those in Fig. 3 should not appear in practical workflow models. Thus, this paper deals with only workflow models or TCWF-nets with free-choice semantics.

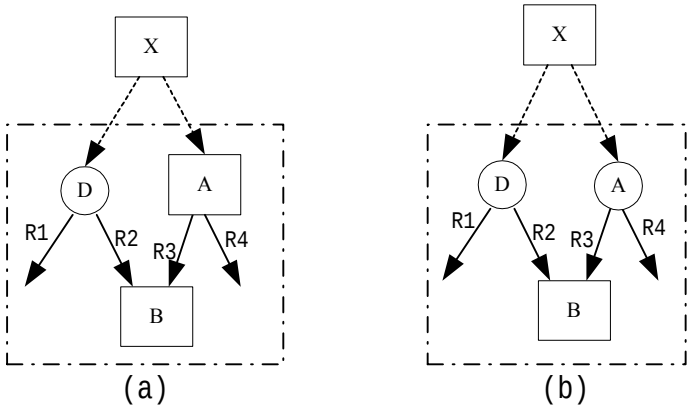


Figure 3. Two non-free-choice structures

Now we introduce a model mapping method from a DNG-based workflow model built by the extended basic process modeling language to a free-choice TCWF-net. Each choice/merge node D is mapped onto a place p_D. For each of D's output path i, transition t_D^i is created. Each activity A is mapped to a transition t_A. If it is in a sequence structure, p_A is created as t_A's input place. However, if A is a synchronous activity, for each of its input path j, a place p_A^j, whose output transition is t_A, is created. For the end activity E, a sink place θ is added as the output place of its mapped transition t_E. Finally, the transition in the graphic process modeling

language is mapped to the directed arc linking corresponding place and transition in the TCWF-net directly. We have so far completed the topology structure mapping and obtained a free-choice TCWF-net structure model of a workflow process. Because the workflow model considered here is free of structural conflicts (deadlock), the resulting TCWF-net must be sound [1].

During the model mapping, we must also consider the imposed timing constraints in a process model besides the structure mapping.

In a TCWF-net, the state of a transaction instance is specified by a place. Hence the timing dependency relations (*LBC* and *UBC*) between activities, which are managed by the workflow manager or workflow engine, are mapped to a time pair in a place. The time pair $[d_{\min}(p), d_{\max}(p)]$, which is used to specify the enabled time span of its output transition after a token arrives at p, includes the firing delay of its output transition. However, a *UBC*, as mentioned above, constrains the period ranging from the end of its source activity to the start enabling of its destination activity. It means that *DU* in *UBC*(*A*, *B*, *DU*) doesn't include the execution duration of activity *B*. Therefore, the actual $[d_{\min}(p), d_{\max}(p)]$ of p is the enabled time span of its corresponding activity. For a place p, which is created for the activity (choice node) *B* in the *LBC*(*A*, *B*, *DL*) and/or *UBC*(*A*, *B*, *DU*), the time pair $[d_{\min}(p), d_{\max}(p)]$ specifying the time dependency relation is set to [*DL*, *DU*+*D*(*B*)], [*0*, *DU*+*D*(*B*)], and [*DL*,∞] according to three cases: 1) *LBC* and *UBC* are both imposed, 2) only *UBC* imposed, and 3) only *LBC* imposed, respectively. If there is neither *LBC* nor *UBC*, the time pair for p is set to [0,∞]. As mentioned above, due to the fact that the workflow engine handles the routing in a merge node D, the time pair for p_D is set to [0, 0]. Hence, a transition in $p_D^{\bullet}$ is executable once it is enabled.

When a workflow engine is taking care of the control and execution of a workflow instance, the task is allocated to the resource agent that is responsible for its execution. The executable duration, i.e., the buffer time that is handled by the corresponding resource agent (executor), is ranged from the task allocation time to the allowed latest completion time. The time pair $[d_{\min}(t), d_{\max}(t)]$ is set to the corresponding executable time span defined for the activity t in the workflow model. If there is no executable time span imposed for the corresponding activity, the pair is set to [0,∞]. However, the executable time span of the transition created for the merge node is set to [0,0], implying that a token routes out once it reaches a merge place (the execution duration of its output transition must be zero).

The execution duration of each activity can be deterministic or stochastic. In the deterministic case, it is mapped to $\alpha(t)$ directly. In the stochastic case, $\alpha(t)$ mapped to a transition in TCWF-net is the average value of its corresponding random execution duration. In this situation, time dimension verification or analysis is conducted from the viewpoint of expectation.

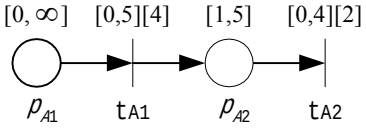


Figure 4. A TCWF-net fragment

Let's use a simple example to illustrate the time information mapping between the two kinds of workflow models. Fig. 4 is a TCWF-net fragment, which is mapped from the sub-workflow in Fig. 2. Activities A_1 and A_2 are mapped onto transitions t_{A1} and t_{A2}, and then p_{A1} and p_{A2} are created respectively as input places of t_{A1} and t_{A2}. A_1's internal time constraints (0, 5)/4 is mapped directly onto t_{A1}'s [0, 5][4], where [0, 5] and [4] correspond to $[d_{\min}(t_{A1}), d_{\max}(t_{A1})]$ and $\alpha(t_{A1})$ respectively. $[d_{\min}(t_{A1}), d_{\max}(t_{A1})]$=[0, 5] means that if t_{A1}, which is enabled at time T_1, is said to be firable during the time period from T_1+0 to T_1+5, which corresponds A_1's executable time span. t_{A2}'s timing constraints can be interpreted similarly. Because A_2's execution duration is 2, then the external timing constraint (*LBC*, *UBC*)=(1,3) between A_1 and A_2 is mapped onto $[d_{\min}(p_{A2}), d_{\max}(p_{A2})]$=[1, 3+2]=[1,5]. If t_{A1}'s firing is completed at T_0 (i.e., $K_a(p_{A2})=T_0$), p_{A2} can only enable its output transition t_{A2} during the time from T_0+1 to T_0+5, which corresponds to A_2's enabled time span. Since there are no external timing constraints for A_1, p_{A1}'s $[d_{\min}(p_{A1}), d_{\max}(p_{A1})]$ is set to [0,∞].

4. Workflow model schedulability verification and synthesis

As defined in [17], a marking M_n, is said to be reachable if there is a firing sequence $\sigma=(M_0 t_1 M_1 \cdots t_i M_i \cdots t_n M_n)$ or simply $(t_1, t_2, \cdots, t_n)$ that transforms M_0 to M_n. Due to timing constraints, to prove that M_n is reachable in TCWF-net, we have to prove that all the transitions in σ are schedulable with respect to M_0. In other words, let t_n be the final transition of σ from M_0 to M_n, M_n is reachable if and only if t_n and all the transitions that occurred prior to t_n are schedulable. $\delta_k(M_n)$, to be used below, denotes the collection of places and transitions except the first

transition in the k-th firing sequence from M_0 to M_n [16].

Using TCPN scheduability [16] as reference, we give the definition of the scheduability of a TCWF-net and then its corresponding verification method.

It is known that the arrival time of a transaction instance must be taken into account in the schedulability analysis of a TCWF-net model, which indicates a "strong" characteristic. Thus we do not distinguish the strong and weak schedulabilities. However, we differentiate local and global schedulabilities here.

Definition 5: A transition t in marking M of a TCWF-net is locally schedulable iff t's enabled marking M_t is reachable from M, $d_{\max}(t)$-$d_{\min}(t)\geq\alpha(t)$ and, $\forall p\in{}^\bullet t$: $d_{\max}(p)$-$d_{\min}(p)$-$d_{\min}(t)\geq\alpha(t)$. If t can complete its firing successfully in marking M of the TCWF-net, it is globally schedulable.

Obviously, the local schedulability is a necessary condition of the global one. A schedulable TCWF-net means that all its transitions are globally schedulable.

Assuming $p_i\in{}^\bullet t$, $E_E(t)/L_E(t)$ is constrained by token enabled times of all input places of t, i.e., $E_E(t)=\underset{i}{Max}\,(K_a(p_i)+d_{\min}(p_i))$ and $L_E(t)=\underset{i}{Min}\,(K_a(p_i)+d_{\max}(p_i))$. Therefore, assuming that t_s is mapped from the start activity we have the following theorem.

Theorem 1： A locally schedulable transition t in marking M of TCWF-net is globally schedulable iff $L_F(t)$ - $E_F(t)\geq\alpha(t)$, where

$$L_F(t)=L_E(t)$$

$$=\underset{i}{Min}\,(K_a(p_i)+d_{\max}(p_i))$$

$$=\underset{k}{Min}\,\{F_{end}(t_s)+\Sigma\, d_{\max}(p_{mk})\}$$

$p_i\in{}^\bullet t$, $p_{mk}\in\delta_k(M)$.

$$E_F(t)=E_E(t)+d_{\min}(t)$$

$$=\underset{i}{Max}\,(K_a(p_i)+d_{\min}(p_i))+d_{\min}(t)$$

$$=\underset{k}{Max}\,\{F_{end}(t_s)+\Sigma\, d_{\min}(t_{nk})+\Sigma\alpha(t_{nk})+\Sigma\, d_{\min}(p_{mk})\}+d_{\min}(t)$$

$p_i\in{}^\bullet t$, t_{nk}, $p_{mk}\in\delta_k(M)$.

Theorem 1 is derived directly from the TCPN theory. A trivial modification of the proof of related theorems in [16] can prove it. Its proof is omitted here. However, two points need to be noted here. First, our TCWF-net is sound. When the workflow model is instantiated to a

workflow instance by the arrival of a transaction instance, each input place of the enabled transition t has exactly one token during the model runtime. Then, it is unnecessary to compare the arrival time of different tokens in the same places. Second, according to the theorems in [16] p can only enable its output transition during the time span $[K_a(p)+ d_{min}(p), K_a(p)+d_{max}(p)]$. Therefore, if the enabled time of its output transition t is T_0, its fireable time-span is $[T_0+ d_{min}(t), Min\{T_0+d_{max}(t),K_a(p)+d_{max}(p)\}]$. If $d_{max}(p)>d_{max}(t)$, $K_a(p)+Min\{d_{max}(p), d_{max}(t)\}$ in formula (4.b) of [16] is equal to $K_a(p)+d_{max}(t)$. Obviously, it does not make sense. Thus, it should be replaced by $K_a(p)+d_{max}(p)$, the rationality of which can be found easily.

In the following discussions, "schedulable" refers to "globally schedulable". According to Theorem 1, a workflow modeled by a TCWF-net is schedulable if and only if all the transitions in the TCWF-net are schedulable. To facilitate the use of Theorem 1, Theorem 2 is deduced.

Theorem 2: A TCWF-net is schedulable iff all its synchronous transitions are globally schedulable and all the remaining transitions are locally schedulable.

Proof: A schedulable TCWF-net implies that every transition is globally schedulable. Thus the necessity is obvious. If all the synchronous transitions are globally schedulable, all the transitions preceding these synchronous transitions can complete firing successfully, i.e. globally schedulable. Thus we need to consider only the transitions behind all the synchronous transitions along all the possible paths from the source ε to the sink θ. Consider any non-synchronous transition t. The number of its input places must be exactly 1. Assuming $p_i \in {}^\bullet t$, we have $L_F(t) - E_F(t)$

$$= \underset{i}{Min}\,(K_a(p_i)+d_{max}(p_i))-(\underset{i}{Max}\,(K_a(p_i)+d_{min}(p_i))+ d_{min}(t))$$

$$= K_a(p_i)+d_{max}(p_i)-(K_a(p_i)+d_{min}(p_i)+ d_{min}(t))$$

$$= d_{max}(p_i)- d_{min}(p_i)- d_{min}(t).$$

Since t is a locally schedulable transition, $d_{max}(p_i)- d_{min}(p_i)- d_{min}(t) \geq \alpha(t)$. Then $L_F(t) - E_F(t) \geq \alpha(t)$. By Theorem 1, t is globally schedulable. Hence, all the transitions in a TCWF-net are globally schedulable, or the net is schedulable. □

Before the schedulability synthesis of a TCWF-net from the aspect of time dimension is discussed, two relevant concepts of the timing constraints of a transition should be introduced. In marking M_i of a TCWF-net, t's schedulable decision span is defined as $S(t)= [E_F(t), L_F(t)-\alpha(t)]$

and its schedulable buffer time $Y(t)=L_F(t)-\alpha(t)-E_F(t)$. A transition completing its firing successfully must start its firing during the absolute time span $S(t)$. $Y(t)$ represent the interval one can manage to start firing a transition. Assigning reasonable $S(t)$ for each transition t and then the corresponding activity is an important issue for leveraging the process performance and resource load. Based on the synthesis methods of Forward Computation and Backward Computation [16], we give some heuristic rules about how to assign $S(t)$ for each t considering its relative importance, the load of resources allocated for t's firing, and the influence of its firing on its succeeding transitions. According to different topological structures of a TCWF-net, several cases are discussed.

- For transitions in a sequence, OR_join or AND_split structure:

  If the resource satisfies the requirement, $S(t)$ and $Y(t)$ of the preceding transition should be scheduled early and reduced as much as possible respectively to expand the schedulable decision span of the succeeding transitions. The reason is that there is a total time limit for a workflow model and then the execution of preceding transitions has influence on schedulable decision span of all the succeeding transitions. If something goes wrong with the firing of a preceding transition, the larger $Y(t)$ and abundant $S(t)$ for the succeeding transitions give them enough time to handle the exception.

- For transitions in an OR_split structure:

  One can designate mutually exclusive fireable period $[d_{\min}(t), d_{\max}(t)]$ for the transitions corresponding to conflict activities, through which the different priorities are set. We can use it for the specification of the exception handling of time violation or the selection of an output path in a conflict structure. If transaction instance's routing in an OR_splict structure does not depend on its timing attributes, the transitions in this structure can be treated the same as in sequence structures.

- For transitions in an AND_join structure:

  In this synchronous structure, the execution of a succeeding transition cannot begin until all its preceding transitions complete their firing. Therefore, in order to reduce the mutual waiting time of individual execution paths, the time constraints for the preceding transitions must be selected to make them as accordant (according to their succeeding synchronous transition's $S(t)$ and $Y(t)$ constrained by different input paths) as possible.

5. Boundedness verification of a workflow model

A workflow model is instantiated by multiple transaction instances. Boundedness verification of a workflow model concerns whether congestion or overflow may occur in the environment of multi-instances running concurrently. In a TCWF-net, source place ε with a tokens (transaction instances) arrival rate λ can be viewed as a structure of a transition (with no input arcs) with firing rate λ connecting to ε. This semantics cause many tokens of multiple instances reside in the same TCWF-net. A TCWF-net is bounded if every place is bounded. Then, workflow's boundedness verification corresponds to verify if there is a place that can have an infinite number of tokens.

To realize the boundedness verification of a TCWF-net, an extended net $\overline{PN}=(\overline{P},\overline{T},\overline{F})$ [1] is defined. $\overline{PN}$ is the PN obtained by adding an extra transition t which connects place θ and ε to a TCWF-net *PN*, where $\overline{P}=P$, $\overline{T}=T\cup\{t\}$, and $\overline{F}=F\cup\{(\theta,t),(t,\varepsilon)\}$. Based on the TCWF-net mapped from the DNG-based workflow model, the extended free-choice TCWF-net is obtained by adding a new transition *t* between the intial and end places, i.e., ${}^{\bullet}t=\{\theta\}$, $t^{\bullet}=\{\varepsilon\}$.

We know each workflow model describes the life-cycles of several kinds of transaction instances. Corresponding to each kind of transaction instances, there is a routing path in a workflow model. Because the original DNG-based workflow model built by the basic process modeling language is structurally acyclic, there is no iteration structure in the TCWF-net obtained through the model transformation. Heuristically, we decompose an extended free-choice TCWF-net specifying a workflow model to a set of *T*-components representing the routing path of each kind of transaction instance. Based on the resulting set of *T*-components, a boundedness verification method of the corresponding TCWF-net can be deduced.

Note that the algorithm discussed below focuses only on a TCWF-net's structure decomposition, the corresponding time information of every transition or place in its subnets is unchanged. Also, for simplicity, we use $PN=(P, T, F)$ to represent an extended TCWF-net. Before the decomposition algorithm is introduced, some relevant concepts are given below.

Definition 6: An elementary path in $PN=(P, T, F)$, called a path for short, is $(x_1, x_2, \ldots, x_k)$ such that arc (x_i, x_{i+1}) exists $1\le i\le k$-1, and $x_i=x_j$ implies $i=j$, $1\le i, j\le$ k where $x_i\in P\cup T$. It is called an (elementary) circuit if $x_i=x_j$, $1\le i, j\le$ k implies i=1 and $j=k$. $PN_1=(P_1, T_1, F_1)$ is a subset of $PN=(P, T, F)$, $path=\{t_0, p_1, t_1\cdots p_m, t_m\}$ in *PN* is a transition path of PN_1iff:

- It is a path
- $t_0, t_m\in T_1$

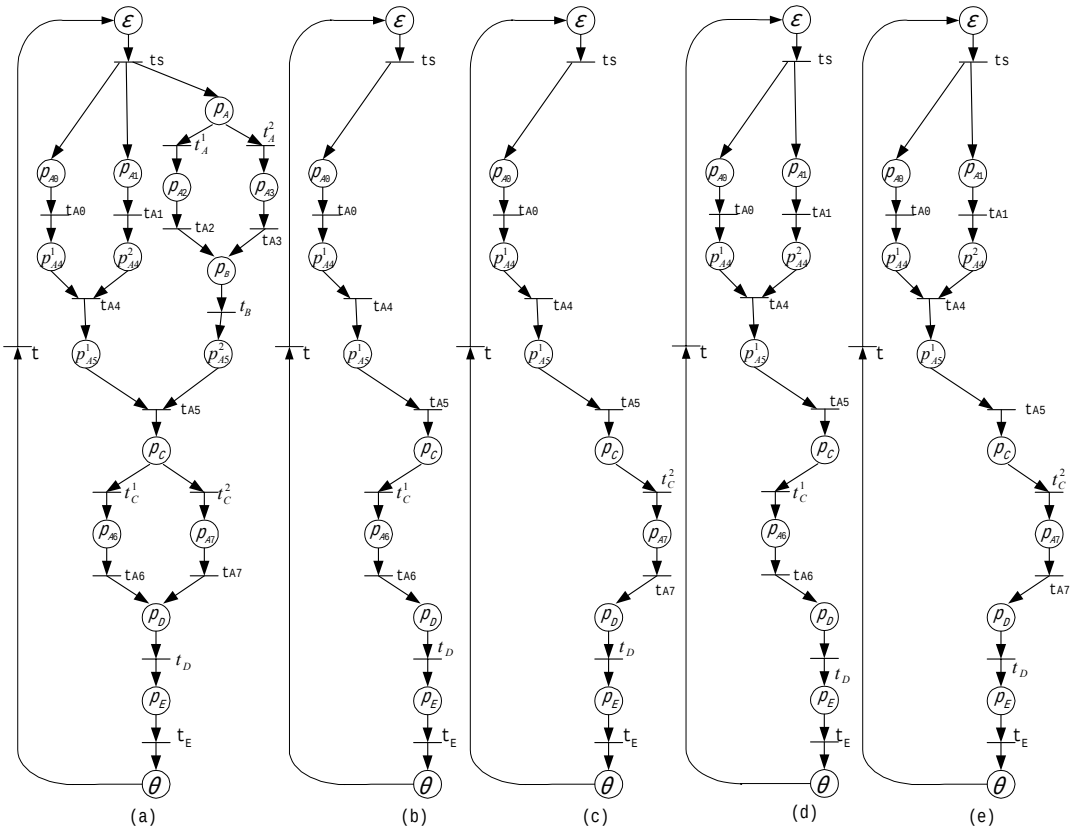
- $p_j \notin P_1$, $1 \le j \le m$, and $t_j \notin T_1$, $1 \le j < m$

Symmetrically, $path = \{p_0, t_1, p_1 \cdots t_m, p_m\}$ in PN is a place path of PN_1 iff:

- It is a path
- $p_0, p_m \in P_1$
- $t_j \notin T_1$, $1 \le j \le m$, and $p_j \notin P_1$, $1 \le j < m$

A transition path defined here has the same semantic as a *nice path* in [24].

Definition 7: A place p in $PN = (P, T, F)$ is a choice place iff $|p^{\bullet}| \ge 2$. Suppose that a path or circuit $PN_1 = (P_1, T_1, F_1)$ is a subnet of $PN = (P, T, F)$ and p' is the first or source place. Given a choice place $p \in P_1$, its choice degree is defined as the number of choice places in the path from place p' to p in PN_1.



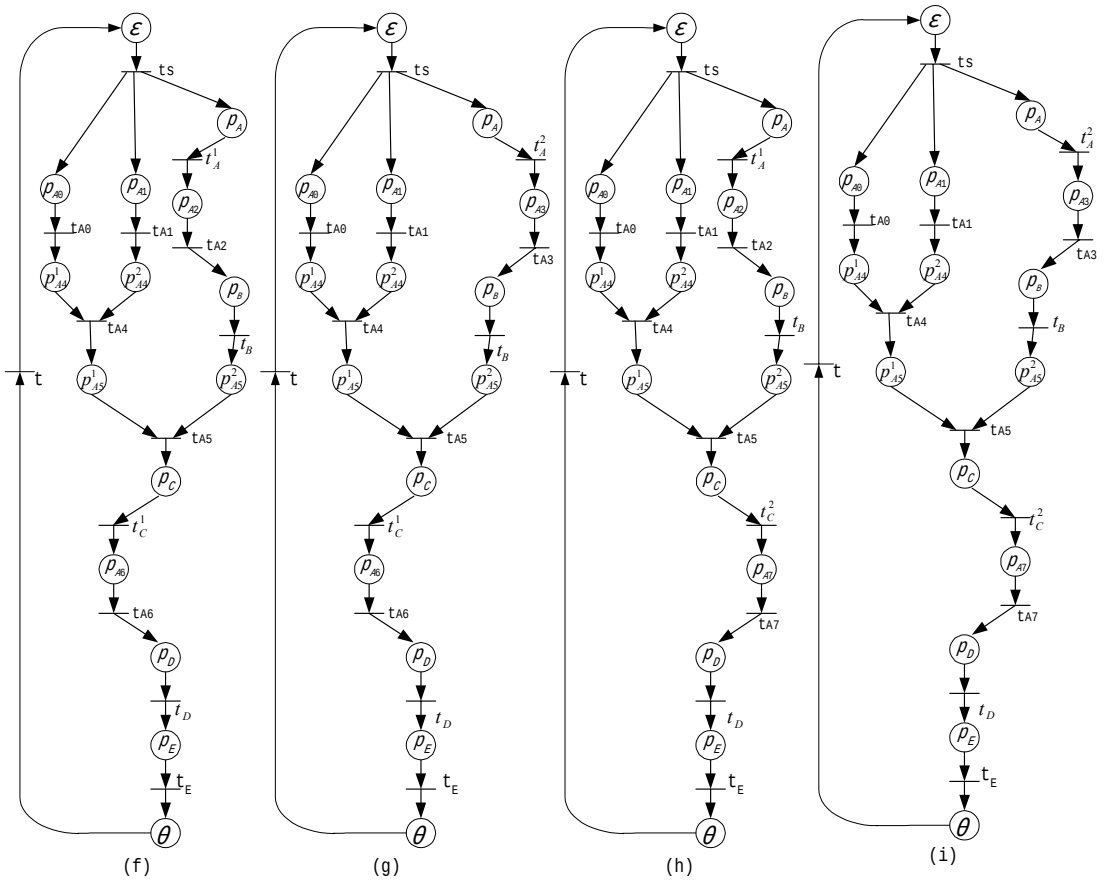


Figure 5. The extended TCWF-net and its decomposition result

Definition 8: Suppose that R is a circuit (path) set. A choice place p is proper iff ∀circuit (path) in R, p is not on it or p's choice degree is the least.

Let's use an example to explain Definitions 6-8. $PN_1 = (P_1, T_1, F_1)$ in Fig. 5(b) is a circuit subnet of $PN = (P, T, F)$ in Fig. 5(a). For path $\{t_s, p_{A1}, t_{A1}, p^2_{A4}, t_{A4}\}$ in PN, t_s, $t_{A4} \in T_1$ and its other elements do not belong to PN_1. Thus it is a transition path of PN_1. However, because p^1_{A5} and t_{A5} belong to PN_1, path $\{t_s, p_{A1}, t_{A1}, p^2_{A4}, t_{A4}, p^1_{A5}, t_{A5}\}$ is not a transition path of PN_1. Similarly, $\{p_C, t^2_C, p_{A7}, t_{A7}, p_D\}$ is a place path of PN_1.

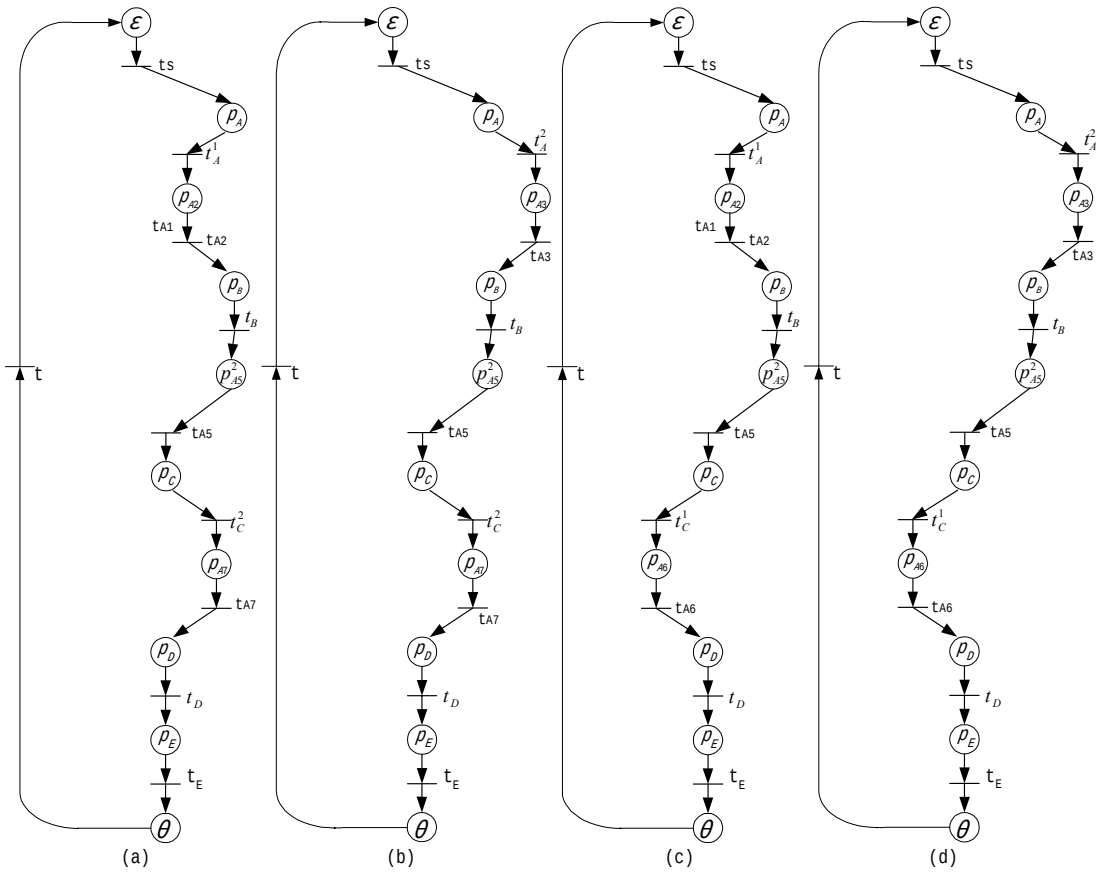


Figure 6. A different decomposing process of the TCWF-net in Fig. 5(a)

If a token reaches a choice place, it must choose an output path (transition). In Fig. 5(a), p_A and p_C are two choice places. To decide their choice degree, let's consider circuit PN_1 in Fig.5(b). Since only p_C in the path from ε to p_C is a choice place, p_C's choice degree in PN_1 is 1. Now consider PN_2= (P_2, T_2, F_2) of Fig. 6(a). Since there are two choice places (p_A and p_C) in the path from ε to p_C, p_C's choice degree in PN_2 becomes 2.

Suppose that the circuit in Fig. 5(b) is the only element of R. Then p_C is a proper choice place. However, if the circuit of Fig. 6(a) is also in R, the proper choice place is not p_C but p_A. The reason is that although p_C's choice degree is the minimal in the circuit of Fig. 5(b), it is not the minimal in the circuit of Fig. 6(a). Since p_A belongs to only the circuit of Fig. 6(a), and its choice degree is the minimal in that circuit, it is a proper choice place in R.

Based on these definitions, we discuss in detail about the workflow model decomposition. It has been proved in [24] that each live and safe free-choice PN is covered by a set of strongly connected T-components. Also, an algorithm (Algorithm 7.4 in [24]) is defined to construct a T-component from an arbitrarily selected transition. Obviously, each T-component of a TCWF-net is corresponding to the routing path of a specific transaction instance. As mentioned before, the resulting extended TCWF-net considered in this paper is sound. Hence, one can

assure its liveness and safeness. Next, we extend the algorithm in [24] to decompose a sound free-choice extended TCWF-net PN= (P, T, F) into a set of T—component as follows.

Algorithm 1 (Decomposition)

Step 1: Corresponding to an arbitrarily selected transition t in $p^{1\bullet}=\{t_1, t_2, \cdots, t_n\}$, where $p^1 \in P_C=\{p| |p^\bullet| \geq 2\}$, an circuit $PN_1 = (P_1, T_1, F_1)$ passing t is constructed. In addition, $P_S = \phi$ and $R_1 = \{PN_1\}$, where P_S is a choice place set and R_1 is a subnet set.

Step 2: Repeat the following steps until $\forall\ PN_j=(P_j, T_j, F_j)\in R_1,\ \Psi_j=\{p| p\in P_j \wedge |p^\bullet| \geq 2 \wedge p\notin P_S\}$ is empty. Denote the resulting subnet set as $R=R_1=\{PN_1, PN_2,\cdots, PN_n\}$.

2.1: $R_2= R_1$, $PL=\phi$;

2.2: For every $PN_j= (P_j, T_j, F_j)\in R_2$, if $\Psi_j=\{p| p\in P_j \wedge |p^\bullet| \geq 2 \wedge p\notin P_S\}$ is nonempty, choose the choice place $p\in \Psi_j$ whose choice degree is the least one in Ψ_j and $PL= PL\cup\{p\}$;

2.3: For every place $p_k\in PL$, if p_k generated from Ψ_k of $PN_k\in R_2$ belongs to Ψ_m ($m\neq k$) of $PN_m\in R_2$ and the choice degree of p_k is not the least one in Ψ_m, then $PL= PL\backslash\{p_k\}$;

2.4: $P_S= P_S \cup PL$, and for every place $p_l\in PL$ do

For every $PN_j = (P_j, T_j, F_j)\in R_2$, if $p_l\in P_j$, there must be a nonempty transition set $\eta=\{t|t\in p_l^\bullet \wedge t\notin T_j\}$. Then for every transition t in η, an arbitrary circuit $PN_t=\{P_t, T_t, F_t\}$ which goes down from source ε to p_l along PN_j and then passes t is constructed and added to R_1;

Step3: For every $PN_k=\{P_k, T_k, F_k\}\in R$, repeat the following exhaustively:

If there is $t''\in T_k$ with a nonempty set $\{p''| p''\in t''^\bullet \wedge p''\notin P_k\}$, then for every $p_1\in t''^\bullet\backslash P_k$, an arbitrary transition path $path = \{t_0, p_1, t_1\cdots p_m, t_m\}$ is constructed and merged into PN_k, where $t''= t_0$, $t_m\in T_k$. However, if there are choice places in the merged $path$ for the new PN_k, $R'_1=\{PN_k\}$, the following steps are repeated until $\forall\ PN_j= (P_j, T_j, F_j)\in R'_1$, the set $\Psi'_j=\{p'||p'^\bullet|\geq 2 \wedge p'\in P_j \wedge p'\notin P_S\}$ is empty. Then the result R'_1 is merged into R.

3.1: $R'_2= R'_1$, $PL'=\phi$;

3.2: For every $PN_j= (P_j, T_j, F_j)\in R'_2$, if $\Psi'_j=\{p'| p'\in P_j \wedge |p'^\bullet| \geq 2 \wedge p'\notin P_S\}$ is nonempty, each p' must belong to a path from t_0 to t_m. Choose the choice place $p'\in \Psi'_j$ whose choice degree in the path from t_0 to t_m is the least one in Ψ'_j, and let $PL'= PL'\cup\{p'\}$;

3.3: For every place $p'_r\in PL'$, if p'_r generated from Ψ'_r of $PN_r\in R'_2$ belongs to Ψ'_s ($r\neq s$) of $PN_s\in R'_2$ and the choice degree of p'_r is not the least one in the path from t_0 to t_m

(in Ψ'_s), then $PL' = PL' \setminus \{p'_r\}$;

3.4: $P_S = P_S \cup PL'$, and for every place $p'_l \in PL'$ do

For every $PN_j = (P_j, T_j, F_j) \in R'_2$, if $p'_l \in P_j$, a nonempty post transition set $\eta' = \{t' | t' \in p'^{\bullet}_l (\text{in } PN) \wedge t' \notin T_j\}$ exists. Then for every transition $t' \in \eta'$, there must be a place path $path1 = \{p'_l, t', \cdots p'''\}$ in which only p'_l and p''' belong to the transition path from t_0 to t_m in PN_j. Use $path1$ to replace the corresponding part from p'_l to p''' of the transition path from t_0 to t_m in PN_j to obtain and add a new subset $PN_{t'}$ to R'_1;

In Algorithm 1, Step 1 is used to construct a circuit PN_1 (subnet of PN) passing one arbitrary output transition of an arbitrary choice place in PN. Obviously, it must contain source and sink places. For example, Fig. 5(b) and Fig. 6(a) can be viewed as two circuits constructed respectively from choice place p_C's output transition t_C^1 and choice place p_A's output transition t_A^1. The choice place set P_C contains all the choice places in PN. In the TCWF-net of Fig. 5(a), $P_C = \{p_A, p_C\}$. P_S represents the choice place set in which all the choice places have been investigated and selected to propagate new subnets.

Step 2 finds all the circuits generated from PN_1. In each iteration of Step 2, some proper choice places of $P_C \setminus P_S$ are investigated and selected to be merged into P_S. Steps 2.2 and 2.3 select all the proper choice places in $P_C \setminus P_S$ into PL. Next, for each circuit (of current R_2) in which one (only one) proper choice place $p \in PL$ is located, Step 2.4 propagates $|p^{\bullet}|$-1 ($p^{\bullet}$ represents p's output transitions set in PN) circuits corresponding to $|p^{\bullet}|$-1 new place paths of p and merges them into R_1 (R_2 of next iteration). Obviously, each choice place can only be selected to be a proper choice once, which guarantees that each arbitrarily constructed circuit in Step 2.4 is new for R_1.

During each iteration of Step 2, all the proper choice places of $P_C \setminus P_S$ in current R_2 are merged into P_S. They are put into P_S in an ascending sequence of their choice degrees. Then, when a choice place p is put into P_S, the choice places with less choice degree in PN_j must have been in P_S and $|p^{\bullet}|$-1 circuits are created for each circuit PN_j containing p. Therefore, if p carries the maximal choice degree in every $PN_j \in R_1$, then all the circuits passing p and its preceding choice places are included in R_1. Otherwise, Step 2 will continue its iteration.

Based on Algorithm 7.4 [24] to grow a T-component from a given single transition, Step 3 extends every subnet in R from its *fork* transitions (each of which has multiple outputs in PN), and obtains the corresponding T-components. Its validity is demonstrated in [24]. However, it is possible that the new merged transition paths in Step 3 include new choice places that do not belong to P_S. In this situation, there must be new subnets that are derived from the choice places in the new merged transition paths of PN_j. Then Steps 3.1-3.4 (the proper choice place is in path)

similar to Steps 2.1-2.4 (the proper choice place is in circuit) are needed to construct and merge all these new subnets into R.

Panagos and Rabinovich [24] demonstrated the existence of a transition path *path* corresponding to place p_1 in Step 3. In fact, according to the characteristics of a sound free-choice extended TCWF-net (every closed loop must cover source place ε and sink place θ), every path starting from $t \in PN_i$ can return to PN_i and the *join* node must be a transition. Otherwise, there must be a *join* place and it is not safe, then the TCWF-net is not sound. Symmetrically, it can also be proved in Step 3.4 (Step 2.4 is similar to Step 3.4) that the place path starting from p'_l must return to another place p''' in *path*. The reason is that if a *join* place is not in *path*, then the return node of *path*1 to PN_i is a place or a transition distinct from t_m. These two situations will destroy the soundness of an extended TCWF-net. If the return node is in *path* but not a place, it also conflicts with the soundness of the TCWF-net. Hence, the conclusion of the existence of the place path *path*1 is derived.

In this algorithm, each place in every transition generating net is treated only once, i.e., either in a circuit or transition path. Therefore, we can assure that each place has exactly one input and one output arc, and each element in the resulting set R is a marked graph, then a T-component.

Algorithm 7.4 in [24] focuses on how to generate a T-component from an arbitrarily selected transition. Algorithm 1 demonstrates how to obtain all the T-components contained in a TCWF-net. From the analysis mentioned above we can draw the conclusion: The algorithm can decompose every free-choice extended TCWF-net $PN=\{P, T, F\}$ to R, a set of all T-components that can be derived from PN. Obviously, each T—component net in R corresponds to a processing procedure of a kind of transaction instance, i.e., a routing path of a specific transaction instance in a workflow model.

Obviously, the worst case is that the number of possible T–components could grow exponentially as the number of choice places increases in a TCWF-net. Then the decomposition algorithm must be of exponential time complexity. However, the sum of the iteration number in Steps 2 and 3 is at most $|P_C|$, then the algorithm can terminate in a polynomial number of arithmetic operations. On the other hand, Steps 2 and 3 in our brute force algorithm can be carried out concurrently, which means each circuit generated in Step 2 can be processed by Step 3 immediately. If multi-processors are used, this property can improve the time performance of the algorithm.

Now we demonstrate how to verify the boundedness of a TCWF-net PN specifying a workflow process based on the resulting T-components. Let's suppose that PN including k transitions is mapped from a workflow model built for the processing of n kinds of transaction instances $\{I_1, I_2, \cdots, I_n\}$, and each kind of transaction instance I_i accounts for a certain proportion

of all the transaction instances processed by this workflow model. So we have vector $\beta = \{\beta_1, \beta_2, \cdots, \beta_n\}$ satisfying $\beta_1+\beta_2+\cdots+\beta_n=1$, where $\beta_i \geq 0$ is the percentage of I_i in the total number of transaction instances. To describe distinctly the constituent transitions of a *T*-component, we construct a transition vs. *T*-component matrix $B_{n\times k}$:

$$b_{ij} = \begin{cases} 1 & \text{if } T\text{-component } PN_i \text{ includes transition } t_j \\ 0 & \text{if } T\text{-component } PN_i \text{ doesn't include transition } t_j \end{cases}$$

We assume that the arrival of transaction instances is a stochastic process, and the total arrival rate is λ. Then, the arrival rate of transaction instance I_i, $\lambda_i = \lambda * \beta_i$. To complete I_i's processing, each transition in *T*-component PN_i should fire exactly once. Let $\lambda_I=\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$, the transaction instance input rate of each transition in PN can be obtained by calculating $\lambda_t = \lambda_I * B_{n\times k}$, where the *i*th item λ_{ti} of λ_t represents t_i's token arrival rate. Then the necessary and sufficient condition of the boundedness of the TCWF-net-based workflow model can be deduced as below.

Theorem 3: A sound TCWF-net $PN=\{P, T, F\}$ is bounded iff $\forall t_i \in T$, $\alpha(t_i)*\lambda_{ti}<1$.

This theorem can be easily proved and thus its proof is omitted here. However, it should be noted here that "≤" cannot be used in the inequation. The time span of any activity's execution is deterministic or stochastic. If transition t_i satisfing $\alpha(t_i)*\lambda_{ti}=1$, its boundedness depends on the distribution functions associated with t_i. For instance, for deterministic timing, the number of tokens in input place *p* of transition t_i remains bounded. However, for exponentially distributed random time, the marking of *p* is a null-recurrent state, thus destroying the boundedness.

When a workflow model is deployed in practice, its arrival rate of transaction instances is only an estimated value. Even if its boundedness has been verified, it is interesting to know how far away this model is from its unboundedness. Using Theorem 3, one can compute $Max\{\alpha(t_i)*\lambda_{ti}\}$. Then the value 1- $Max\{\alpha(t_i)*\lambda_{ti}\}$ can be viewed as the boundedness distance of a bounded workflow model. On the other hand, the transition with the maximal value of $\alpha(t_i)*\lambda_{ti}$ must be the potential bottleneck of the workflow model, which is valuable information for system managers.

6. A case study

Now we use an example to illustrate the application of our approach. Fig. 7(a) is a DNG-based workflow model built by the extended basic process definition language. Through the application of the model mapping method, the TCWF-net in Fig. 7(b) is obtained. During the topology structure mapping, a transition t_{Ai} is created for each activity *Ai*, and the label of its input place in a sequence structure is p_{Ai}. The synchronous transition t_{A4}'s two input places are labeled as p^1_{A4} and p^2_{A4}, respectively. The choice/merge nodes *A*, *B*, *C*, and *D* are mapped as places p_A, p_B, p_C, and p_D respectively. The labels of p_A's (p_C's) two output transitions are t^1_A and t^2_A (t^1_C and t^2_C). Similarly, the output transitions of p_B and p_D are labeled as t_B and t_D respectively. The start activity *S* is mapped to the source ε and transition t_s, and a sink place θ is added as the output place of transition t_E. In order to simplify the figure, some labels mentioned above are omitted in Fig. 7. Also, the specification of imposed timing constraints is transformed from the DNG in Fig. 7(a) into the TCWF-net in Fig. 7(b).

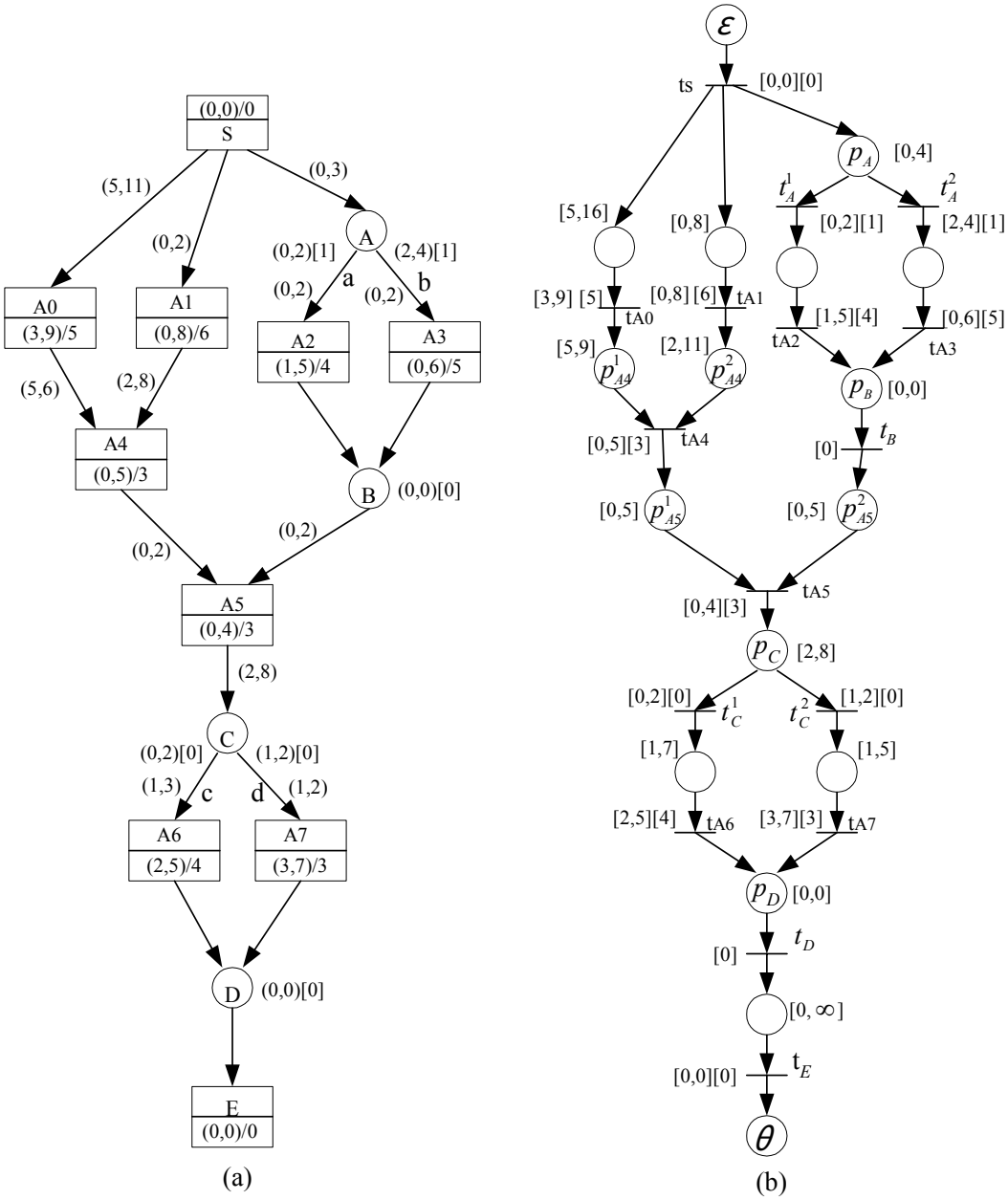


Figure 7. Model mapping from the DNG-based workflow model to TCWF-net

In Fig. 7(b), one can find easily that $d_{\max}(t_{A6})$-$d_{\min}(t_{A6})$=5–2 < 4 =$\alpha(t_{A6})$ and $d_{\max}(p_{A7})$-$d_{\min}(p_{A7})$- $d_{\min}(t_{A7})$ =5– 1– 3 = 1 < 3=$\alpha(t_{A7})$. Hence, neither of t_{A6} and t_{A7} is locally schedulable and then globally schedulable. For synchronous transition t_{A4}, one can calculate $d_{\max}(p_{A4}^1)$-$d_{\min}(p_{A4}^1)$- $d_{\min}(t_{A4})$ =9 – 5 – 0 = 4, $d_{\max}(p_{A4}^2)$- $d_{\min}(p_{A4}^2)$-$d_{\min}(t_{A4})$= 11 – 2 – 0= 9, and $d_{\max}(t_{A4})$-$d_{\min}(t_{A4})$= 5 - 0 = 5. Their results are all greater than $\alpha(t_{A4})$= 3. Thus t_{A4} is locally schedulable. Now, let's consider t_{A4}'s global schedulability. There are two paths δ_1= (t_s, p_{A0}, t_{A0}, p_{A4}^1, t_{A4}) and δ_2= (t_s, p_{A1}, t_{A1}, p_{A4}^2, t_{A4}) from t_s to t_{A4}, then

$$E_F(t_{A4}) = Max\{[F_{end}(t_s) + d_{\min}(p_{A0}) + d_{\min}(t_{A0}) + \alpha(t_{A0}) + d_{\min}(p^1_{A4})],$$

$$[F_{end}(t_s) + d_{\min}(p_{A1}) + d_{\min}(t_{A1}) + \alpha(t_{A1}) + d_{\min}(p^2_{A4})]\} + d_{\min}(t_{A4})$$

$$= Max\{[F_{end}(t_s) + 5 + 3 + 5 + 5], [F_{end}(t_s) + 0 + 0 + 6 + 2]\} + 0$$

$$= Max\{[F_{end}(t_s) + 18], [F_{end}(t_s) + 8]\}$$

$$= F_{end}(t_s) + 18$$

$$L_F(t_{A4}) = Min\{[F_{end}(t_s) + d_{\max}(p_{A0}) + d_{\max}(p^1_{A4})],$$

$$[F_{end}(t_s) + d_{\max}(p_{A1}) + d_{\max}(p^2_{A4})]\}$$

$$= Min\{[F_{end}(t_s) + 16 + 9], [F_{end}(t_s) + 8 + 11]\}$$

$$= F_{end}(t_s) + 19$$

$L_F(t_{A4}) - E_F(t_{A4}) = F_{end}(t_s) + 19 - F_{end}(t_s) - 18 = 1 < \alpha(t_{A4}) = 3$. From Theorem 1, we know t_{A4} is not globally schedulable. Then, all the transitions behind t_{A4} along all the possible paths from the source place ε to the sink place θ are unschedulable. Similarly, we can verify that the remaining transitions t^1_A, t^2_A, t_{A2}, and t_{A3} are locally schedulable.

However, if p_{A0}'s timing constraint $[d_{\min}(p_{A0}), d_{\max}(p_{A0})] = [5,16]$ is replaced with [3,14] (corresponding to the external timing constraints between start activity and activity A_0 is changed from [5,11] to [3, 9]), it can be verified easily now that transition t_{A4} is globally schedulable. To make t_{A6} and t_{A7} locally schedulable, their $[d_{\min}(t), d_{\max}(t)]$ are relaxed and set be [1,6] and [1,7], respectively. Synchronous transitions t_{A4} and t_{A5} are both globally schedulable, and all other transitions are locally schedulable, and then from Theorem 3 we know that the TCWF-net in Fig. 7(b) is now schedulable.

To show schedulability synthesis of the synchronous structure in this TCWF-net, we consider input paths $\delta_1 = (t_s, p_{A0}, t_{A0}, p^1_{A4}, t_{A4})$ and $\delta_2 = (t_s, p_{A1}, t_{A1}, p^2_{A4}, t_{A4})$ of synchronous transition t_{A4} independently. If only δ_1 (δ_2) exists, the schedulable decision span of t_{A4} is $S(t_{A4}) = [F_{end}(t_s)+16, F_{end}(t_s)+21]$ ($S(t_{A4}) = [F_{end}(t_s)+8, F_{end}(t_s)+17]$). It means that, to complete its firing successfully, t_{A4} should start firing in $[F_{end}(t_s)+16, F_{end}(t_s)+21]$($[F_{end}(t_s)+8, F_{end}(t_s)+17]$). Although t_{A4} is schedulable now, if the sub-transaction instance in δ_2 is usable for t_{A4} at $F_{end}(t_s)+8$, it must await the sub-transaction instance in δ_1 at least 8 time units. Obviously, to avoid this unreasonable situation, one must adjust the timing constraints of δ_1 or δ_2 to make them

accordant with each other. For example, after modifying the timing constraints of p_{A0} and p^1_{A4} respectively to [0, 12] and [0,7], $S(t_{A4})$ imposed by the timing constraints in paths δ_1 and δ_2 are both $[F_{end}(t_s)+8, F_{end}(t_s)+17]$. Then the average mutual waiting time of different execution paths is reduced to the least.

As for the OR_split structure, we should notice that the imposed timing constraints of choice nodes *A*'s (*C*'s) different output paths *a* and *b* (*c* and *d*) are different. For p_A's two output transitions t^1_A and t^2_A, their timing constraints $[d_{\min}(t), d_{\max}(t)]$ are mutually exclusive. t^2_A can only be firable under the situations that t^1_A is un-firable or t^1_A's firing is failure. Then specification of the exception handling of time violation or the selection of an output path in a conflict structure can be realized using a weakly firing mode [16]. Similarly, for p_C's two output transitions t^1_C and t^2_C, if p_C starts enabling at T_0, its output transition t^2_C cannot be selected in (T_0+0, T_0+1), which means that path *c* has a higher priority than path *d*.

For the boundedness verification, we should decompose an extended TCWT-net to a set of *T*–components. Fig. 5(a) is the extended TCWF-net model.

After Step 1 of Algorithm 1, in which $p^1=p_C$ and $t=t^1_C$, the circuit PN_1 in Fig. 6(b) is obtained. Initially, $P_C=\{p_A, p_C\}$, $P_S=\phi$, and $R_2=R_1=\{PN_1\}$. After Step 2.2, $PL=\{p_C\}$. There is only one choice place p_C in *PL*, then it must be *proper* and merged to P_S. p_C has two output transitions t^1_C and t^2_C, and a circuit generated from t^1_C has been in R_2. Then after Step 2.4, only the circuit PN_2 in Fig. 6(c) are constructed and added to R_1. Step 2 stops after one interation. Now, $P_S=\{p_C\}$, and $R=R_1=\{PN_1, PN_2\}$.

Next, let's show how to apply Step 3 to PN_1 (the case of PN_2 is similar to that of PN_1) of *R*. Clearly, t_s's output places p_{A1} and p_A do not belong to PN_1. For p_{A1}, transition path $\{t_s, p_{A1}, t_{A1}, p^2_{A4}, t_{A4}\}$ is constructed and merged into PN_1, and then PN_1 in *R* is replaced with the subset in Fig. 5(d). Similarly, transition path $\{t_s, p_A, t^1_A, p_{A2}, t_{A2}, p_B, t_B, p^2_{A5}, t_{A5}\}$ is constructed for p_A and merged into the subnet in Fig. 5(d). The subnet in Fig. 5(f) is finally obtained.

However, there is a choice place p_A in the new merged transition path. Then it is necessary now to enter the sub-steps of Step 3 to obtain all the subnets propagated from the newly

appearing choice place. In Steps 3.1-3.4, similar to Steps 2.1-2.4, a place path $\{p_A, t_A^2, p_{A3}, t_{A3}, p_B\}$ is constructed and used to replace place path $\{p_A, t_A^1, p_{A2}, t_{A2}, p_B\}$ of Fig. 5(f). Then a new subnet in Fig. 5(g) is generated from the subnet in Fig. 5(f). In the end, there are two subnets in Figs. 5(f-g) derived out from PN_1. Similarly, the other two subnets in Figs. 5(h-i) are obtained from PN_2. Then, the resulting set R contains four T–components as shown in Figs. 5(f-i).

However, if $p^1=p_A$ and $t=t_A^1$ are selected in Step 1, the initially constructed circuit must be one of the two circuits shown respectively in Figs 6(a-b) (In fact, under the condition of $p^1=p_A$ and $t=t_C^1$ as mentioned formerly, Fig. 6(b) can be constructed). Obviously there are two choice places p_A and p_C in any of them. At this time, Step 2 needs two iterations to find all the circuits generated from the initial circuit (for example the one in Fig. 6(a)). In the first iteration, the proper choice place is p_A. Based on p_A's two output transitions t_A^1 and t_A^2, two circuits are constructed as shown in Figs. 6(a-b). However, p_C is a proper choice place in the second iteration, and two circuits shown in Figs. 6(c-d) are generated from p_C's two output transitions t_C^1 and t_C^2. Because $P_S=P_C$ after Step 2, meaning that no new choice place exists in the newly merged transition path, Step 3 extends directly each of them to a T-component without entering sub-steps of Step 3. Obviously, the results are also the four T-components in Figs. 5(f-i).

After the model decomposition, we know that the workflow model corresponding to this TCWF-net specifies the process of four kinds of transaction instances. Suppose that the four kinds of transaction instances are I_1, I_2, I_3, and I_4, and the T–components of Figs. 5(f-i) describes their routing paths in Fig. 5(a) respectively. Also, assuming that the average input rate of transaction instances to the workflow model is λ=0.2, and different kinds of transaction instances account for the same percentage in the total arrival rate, i.e., $\beta=\{\beta_1, \beta_2, \beta_3, \beta_4\}=\{0.25, 0.25, 0.25, 0.25\}$. Then $\lambda_i=\lambda*\beta=\{0.05,0.05, 0.05, 0.05\}$, whose ith item represents the input rate of I_i. Based on the resulting set R (its four elements are shown in Figs. 5(f-i)), the transition vs. T-component matrix $B_{n\times k}$ of the workflow model are constructed as below:

$$t_s \quad t_{A0} \quad t_{A1} \quad t_A^1 \quad t_A^2 \quad t_{A2} \quad t_{A3} \quad t_{A4} \quad t_B \quad t_{A5} \quad t_C^1 \quad t_C^2 \quad t_{A6} \quad t_{A7} \quad t_D \quad t_E$$

$$B_{4\times16}=\begin{bmatrix}1&1&1&1&0&1&0&1&1&1&1&0&1&0&1&1\\1&1&1&0&1&0&1&1&1&1&1&0&1&0&1&1\\1&1&1&1&0&1&0&1&1&1&0&1&0&1&1&1\\1&1&1&0&1&0&1&1&1&1&0&1&0&1&1&1\end{bmatrix}\begin{matrix}I_1\\I_2\\I_3\\I_4\end{matrix}.$$

Then, the transaction instance arrival rate of each transition located in different *T*-components can be obtained by calculating

$$\begin{matrix}t_s & t_{A0} & t_{A1} & t_A^1 & t_A^2 & t_{A2} & t_{A3} & t_{A4} & t_B & t_{A5} & t_C^1 & t_C^2 & t_{A6} & t_{A7} & t_D & t_E\end{matrix}$$

$$\lambda_t=\lambda_1*B_{4\times16}=\begin{bmatrix}0.2 & 0.2 & 0.2 & 0.1 & 0.1 & 0.1 & 0.1 & 0.2 & 0.2 & 0.2 & 0.1 & 0.1 & 0.1 & 0.1 & 0.2 & 0.2\end{bmatrix}$$

where each item λ_{ti} represents the corresponding transition's transaction instance input rate.

Now, let's use Theorem 3 to verify this TCWF-net's boundedness. Through computing $\alpha(t)*\lambda_t=\begin{bmatrix}0 & 1 & 1.2 & 0.2 & 0.2 & 0.6 & 0.8 & 0.6 & 0 & 0.6 & 0 & 0 & 0.4 & 0.3 & 0 & 0\end{bmatrix}$ where $\alpha(t)$ represents the vector of $\alpha(t_i)$, $\alpha(t_i)*\lambda_{ti}$ for each transition is obtained. Clearly, t_{A0} and t_{A1} do not satisfy the condition of Theorem 3, i.e., their input places are unbounded. To make the input places of t_{A0} and t_{A1} bounded (then to avoid the run-time overflow or congestion), we need to shorten (through allocating more resources, modifying the static or dynamic resource allocation rules, or applying new technology) their firing delays to 3 and 4. Then, the $\alpha(t_i)*\lambda_{ti}$ for transitions t_{A0} and t_{A1} are 0.6 and 0.8, i.e., they satisfy Theorem 3 now. Obviously, the boundedness distance of this workflow model is 1- $Max\{\alpha(t_i)*\lambda_{ti}\}$=0.2, and its bottleneck occurs at transitions t_{A1} and t_{A3} (their boundedness distances are both 0.2) .

7. Conclusion

Existing workflow systems offer limited support for handling time-related issues and detecting potential time-related problems at build, instantiation, and run times. However, they are received more and more attentions in the workflow research.

Static data, statistical data, and run-time information are used to estimate the remaining execution time for workflow instances, and then adjust the activity deadlines [25][26]. Using the integration of workflow systems with project management tools to realize the time management is proposed in [27]. The focus of their work is the time management in running environment. Because these techniques do not allow the specification of explicit time

constraints, they are not applicable to build-time model analysis. A framework is presented for computing internal deadlines so that the overall process deadline and external timing constraints are all satisfied [10]. But it can only be used for the *well-structured* acyclic timed workflow graph in which no choice structures are allowed. In order to provide process analysts with estimates of the best, worst, and median execution times of an activity, the PERT-net technique is used to formulate timing constraints in a workflow to compute internal deadlines in the presence of sequential, alternative, concurrent structures [28]. The technique is extended to handle optional structures [29]. These approaches can only deal with a single workflow instance, which restricts their application (in most cases, there are multi-workflow-instances running concurrently).

These existing modeling and analysis techniques used to resolve different level of time-related problems in a workflow model lack a common rigorous theoretic foundation, which restricts their application in practice. The PN researchers [13], [35], [36] try to exploit the correspondence between some special time-related PN enabling and firing rules and the dynamic behavior of a workflow system, and then apply mechanically the state-of-the-art PN-based techniques to analyze workflow. Since they cannot deal with different kinds of timing constraints from practice, most of them are not immediately applicable to analysis of practical workflow, such as the presented example in Fig. 7(a).

This paper considers the timing constraints of a workflow model from the requirement in its actual running environment. Since a DNG-based workflow model is widely used in many workflow products, we adopt it. Furthermore we incorporate all the necessary information, including explicit timing constraints, structural timing constraints, and internal activity deadlines [10, 28-29], into it. On the other hand, to provide a formal framework for modeling and analyzing workflow, we define TCWF-net, which is constructed particularly for specification of imposed timing constraints of a workflow. Then, a method mapping a DNG-based workflow model built by the extended basic process modeling language onto a TCWF-net is discussed. Based on the definitions of local and global schedulabilities, a verification technique of the schedulability of a workflow model is presented. In order to verify the boundedness of the resulting TCWF-net, we present a decomposition algorithm from an acyclic and free-choice TCWF-net to a set of *T*-components. Using the computed result of the

transition instance input rate of each transition, we derive the boundedness verification method of a workflow TCWF-net model. Obviously, the time-related issues addressed in this paper include not only overall deadline meeting (one goal of the schedulability verification) but also the system optimization and potential problem detection.

Mapping an extended DNG-based workflow model onto a TCWF-net links the practical application and theoretic analysis techniques of PN theory. If an appropriate mapping method is provided, which is just like an interface, the timing constraints of all kinds of workflow models with the free-choice characteristics defined by different fashionable modeling tools can be analyzed using the proposed methods.

This paper mainly focused on the time dimension verification of the workflow model at the build-time. The future work should deal with process execution monitoring and management in the enactment environment. Finding some appropriate coordination mechanisms to schedule the execution of the constituent activities of a workflow system is an important and interesting problem especially in the environment where multi-processes or multi-instances run concurrently. Verifying a workflow model from the resource and information perspectives is also a challenge.

References

[1] W. M. P. van der Aalst. The Application of Petri Nets to Workflow Management. *The Journal of Circuits*, *Systems and Computers* 8 (1): 21-66, (1998).

[2] W. Sadiq and M. E. Orlowska. Analyzing Process Models Using Graph Reduction Techniques, *Information System*s, Vol. 25, No. 2, pp. 117-134, 2000.

[3] Y. S. Fan. Fundamentals of Workflow Management Technology. Tsinghua University Press, Springer, 2001(in Chinese).

[4] C. A. Ellis and G. J. Nutt. Modeling and Enactment of Workflow Systems. In *Proceedings of the 14th International Conference* Chicago, Illinois, USA, June 1993, M. A. Marsan, editor, *Application and Theory of Petri Nets 1993*, volume 691 of *Lecture Notes in Computer Science*, Springer-Verlag, Berlin, pp. 1-16 (1993).

[5] D. Georgakopoulos, M. Hornick and A. Sheth. An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. *Distributed and Parallel Databases*, 3: 119-153 (1995).

[6] S. Jablonski and C. Bussler. Workflow Management: Modeling Concepts, Architecture, and Implementation. International Thomson Computer Press (1996).

[7] P. Lawrence, editor. Workflow Handbook 1997, Workflow Management Coliation. John Wiley and Sons, New York (1997).

[8] J. Q. Li and Y. S. Fan. Research on the Petri Net Reduction Method Based Workflow Model Verification, *Information and Control* (in Chinese) forthcoming

[9] W. M. P. van der Aalst and Arthur H. M. ter Hofstede. Verification of Workflow Task Structures: A Petri-Net-Based Approach, *Information systems* Vol. 25. No. 1, pp. 43-69 2000.

[10] J. Eder, E. Panagos and M. Rabinovich. Time Constraints in Workflow Systems. *Advanced Information Syatem Engineering* Vol.1626 of *Lecture Notes in Computer Science*, pp. 286-300. Springer 1999.

[11] C. A. Petri. Kommunikation mit Automation. PhD thesis, University of Bonn, Bonn, Germany 1962(In German).

[12] W. M. P. van der Aalst. Chapter 10: Three Good Reasons for Using a Petri-net-based Workflow Management System. In T. Wakayama et al., editor, *Information and Process Integration in Enterprise: Rethinking docmuments*, The Kluwer International Series in Engineering and Computer Science, Kluwer Academic Publishers, Norwell pp. 161-182 (1998).

[13] N. R. Adam, V. Atluri and W. K. Huang, Modeling and Analysis of Workflow Using Petri Nets. *Journal of Intelligent Information Systems: Special Issue on Workflow and Process Management*, M. Rusinkiewicz and S. H. Abdelsalam, editor, Vol. 10, No. 2, pp. 1-29, 1998.

[14] E. Badouel and J. Oliver, Reconfigurable Nets, a Class of High Level Petri Nets Supporting Dynamic Changes within Workflow Systems, Publication Interne IRISA PI 1163, 1998.

[15] Workflow Management Coalition. *Interface 1:Process Defition Interchange,Process Model.* Document Number WfMC TC-1016-P (1998).

[16] J. P. Tsai and S. J Yang. Timing Constraint Petri Nets and Their Application to Schedulalibility Analysis of Real-Time System Specification *IEEE Trans Software Engineering*, 1995, 21 (1): 32-49.

[17] Murata. Petri Nets: Properties, Analysis and Applications. *Proceeding of the IEEE*, Vol. 79(4), Appril 1989.

[18] M. A. Holliday and M. K. Vernon, A Generalized Timed Petri Net Model for Performance Analysis. *IEEE Trans. Software Eng.*, Vol. SE-13, pp. 1297-1310, Dec. 1987.

[19] B. Liu and A. Robbi, TipNet: a Graphical Tool for Timed Petri Nets. *International Workshop on Petri Nets and Performance Models*, 1995 212-213.

[20] C. V. Ramamoorthy and G. S. Ho, Performance Evaluation of Asynchronous Concurrent Systems Using Petri Nets. *IEEE Trans. Software Eng.*, Vol. SE-6, pp. 440-516, Apr. 1987.

[21] N. G.Leveson and J. L. Stolzy, Safety Analysis Using Petri Nets. *IEEE Trans. Software Eng.*, Vol. SE-13, No. 3, pp. 386-397, Mar. 1987.

[22] P. M. Merlin and D. J. Farber, Recoverability of Communication Protocols Implications of a Theoretical Study. *IEEE Trans.*

Commun., Vol. COM-24, pp. 1036-1043, Sept. 1976.

[23] B. Serthomieu and M. Diaz. Modeling and Verification of Time Dependent System Using Time Petri nets *IEEE Trans. Software Engineering*, 1991, 17 (3): 259-273.

[24] E. Best. Structure theory of Petri nets: the free choice hiatus. In W. Brauer, W. Reisig and G. Rozenberg, editors, *Advances in Petri nets 1986 Part I:Petri Nets,Central Models and Their Properties*,Vol. 254 of *Lecture Notes in Computer Science,* pp. 168-206. Springer-Verlag, Berlin, 1987.

[25] E. Panagos and M. Rabinovich. Predictive workflow management. In *Proceedings of the 3rd International Workshop on Next Generation Information Technologies and Systems*, Neve Ilan, ISRAEL, June 1997.

[26] E. Panagos and M. Rabinovich. Reducing escalation-related costs in WFMSs. In A. Dogac et al., editors, *NATO Advanced Study Institute on Workflow Management Systems and Interoperability*. Springer, Istanbul, Turkey, August 1997.

[27] C. Bussler. Workflow Instance Scheduling with Project Management Tools. In *9th Workshop on Database and Expert System Applications DEXA'98*, Vienna, Austria, 1998. IEEE Computer Society Press.

[28] H. Pozewauning, J. Eder and W. Liebhart. EPERT: Extending PERT for Workflow Management Systems. In *First EastEurapean Symposium on Advances in Database and Information Systems ADBIS'97*, St. Pertersburg, Russia, Sept. 1997.

[29] J. Eder, E. Panagos, H. Pozewaunig and M. Rabinovich. Time Management in Workflow Systems, in *Proceeding of International Conference on Business Information Systems*, April 1999, 266-280.

[30] M. D. Zisman, Representation, specification and Automation of Office Procedures, University of Pennsylvania Wharton School of Business, PhD Thesis, 1997.

[31] Y. Han, HOON-A Formalism Supporting Adaptive Workflows, *Technical Report #UGA-CS-TR-97-005*, Department of Computer Science, University of Georgia, November 1997.

[32] D. Wikarski, An Introduction to Modular Process Nets, *Technical Report TR-96-019*, International Computer Science Institute (ICSI) Berkley, CA, USA, 1996.

[33] A. Agostini, G. De Michelis and K. Petruni, Keeping Workflow Models as Simple as Possible, In *Proceedings of the Workshop on Computer-Supported Cooperative Work*, *Petri nets and Related Formalisms within the 15th International Conference on application and Theory of Petri Nets*, Zaragoza, Spain, June 21st, pp. 11-29, 1994.

[34] Elmagarmid, A. K., Len, Y., Litwin, W., and Businkiewicz, M. (1990). A Multidatabase Transaction Model for InterBase. In *Proceeding of 16th International Conference On Verg Large Data Bases*, pp. 507-518, Briabane, Australia.

[35] A. Ferscha, Qualitative and Quantitative Analysis of Business Workflows using Generalized Stochastic Petri Nets, In *Proceeding of CON,94: Workflow Management- Challenges, Paradigms and Products*, Linz, Australia, October 19-21, 1994, G. Chroust, A. Benczur (Eds.), pp. 222-234, Olderbourg Verlag, 1994.

[36] A. K. Schomig and H. Rau, A Petri Net Approach for the Performance Analysis of Business Process, University of Wurzburg, Report 116 Semiar at IBFI, Schloss Dagstuhl, May 22-26, 1995.

[37] K. Gerrit, Jan Verelst, and Bart Weyn, Techniques for modeling Workflows and Their Support of Resuse, *Business Process Management, LNCS 1806,* W. van der Aalst et al. (Eds) pp. 1-15, 2000.

[38] W. M. P. van der Aslst, and Arthur H. M. Ter Hofstede. Verification of Workflow Task Structures: A Petri-Net-Based Approach. *Information Systems* Vol. 25, No. 1, pp. 43-69, 2000.

[39] W. M. P. van der Aslst. Finding Errors in the Design of a Workflow Process: A Petri-net-based Approach. In W. M. P. van der Aslst, G. De Michelis, and C. A. Ellis, editors, *Proceedings of Workflow Management: Net-based Concepts, Models, Techniques and Tools (WFM'98)*, Vol. 98/7 of Computing Science Reports Eindhoven University of Technology, Eindhoven, pp. 60-81, Lisbon, Portugal (1998).

[40] T. Basten. In Terms of Nets, Systems Design with Petri Nets and Process Algebra. Ph. d thesis, Eindhoven University of Technology, Eindhoven, The Netherlands (1998).

[41] M. Voorhoeve. Modeling and Verification of Workflow Nets. In W. M. P. van der Aslst. and C. A. Ellis, editors, *Proceedings of Workflow Management: Net-based Concepts, Models, Techniques and Tools (WFM'98)*, Vol. 98/7 of Computing Science Reports Eindhoven University of Technology, Eindhoven, pp. 96-108, Lisbon, Portugal (1998).

[42] W. M. P. van der Aslst.and C. A. Ellis, editors, *Proceedings of Workflow Management: Net-based Concepts, Models, Techniques and Tools (WFM'98)*, Lisbon, Portugal, UNINOVA, Lisbon (1998).

[43] G. De Michelis. C. A. Ellis, and G.Memmi, editors. *Proceedings of the second Workshop on Computer-Supported Cooperative Work, Petri nets and related formalisms* Zaragoza, Spain (1994).

[44] A. Ferscha, Business workflow analysis using Generalized Stochastic Petri Nets, *Nine Austrian -Hungarian Informatics Conference*, 1994, pp. 222-234.

[45] M. Merz, D. Moldt, K. Muller, S., Donateli, W. Lamerdorf, Workflow Modeling and Execution with Coloured Petri Nets in COSM, *Proceedings of the Workshop on Applications of Petri Nets to Protocols within the 16th International Conference on Application and Theory of Petri Nets*, 1995.

[46] E. Kindler, A. Martens, W. Reisig, Inter-operability of Workflow Applications: Local Criteria for Global Soundness, in *Business Process Management, LNCS 1806,* W. van der Aalst et al. (Eds), pp. 235-253, 2000.

[47] K. M. van Hee, H.A. Reijers, Using Formal Analysis Techniques in Business Process Redesign, in *Business Process Management, LNCS 1806,* W. van der Aalst et al. (Eds), pp. 142-160, 2000.

[48] Yang Qu, Chuang Lin, and Jiye Wang Linear Temporal Inference of Workflow Management Systems Based on Timed Petri Nets Models, *Lecture Notes in Computer Science 2480*, pp. 30 –44, 2002

[49] Chuang Lin, Yang Qu, Fengyuan Ren, and Dan C. Marinescu**,** Performance Equivalent Analysis of Workflow Systems Based on Stochastic Petri Net Models, *Lecture Notes in Computer Science 2480*, pp. 64 –79, 2002.

[50] K. Salimifard, M. Wright, Petri net-base Modeling of Workflow Systems: An Overview, *European Journal of Operational Research* 134 (2001), pp. 664-676.