PAPER

Time Performance Optimization and Resource Conflicts Resolution for Multiple Project Management

Cong LIU[†], Jiujun CHENG[††a)], Yirui WANG[†††], *Nonmembers*, *and* Shangce GAO[††††b)], *Member*

SUMMARY Time performance optimization and resource conflict resolution are two important challenges in multiple project management contexts. Compared with traditional project management, multi-project management usually suffers limited and insufficient resources, and a tight and urgent deadline to finish all concurrent projects. In this case, time performance optimization of the global project management is badly needed. To our best knowledge, existing work seldom pays attention to the formal modeling and analyzing of multi-project management in an effort to eliminate resource conflicts and optimizing the project execution time. This work proposes such a method based on PRT-Net, which is a Petri net-based formulism tailored for a kind of project constrained by resource and time. The detailed modeling approaches based on PRT-Net are first presented. Then, resource conflict detection method with corresponding algorithm is proposed. Next, the priority criteria including a key-activity priority strategy and a waiting-short priority strategy are presented to resolve resource conflicts. Finally, we show how to construct a conflict-free PRT-Net by designing resource conflict resolution controllers. By experiments, we prove that our proposed priority strategy can ensure the execution time of global multiple projects much shorter than those without using any strategies.

key words: *multi-project management, resource conflict resolution, time performance optimization, priority criteria, petri nets, resolution controller design*

1. Introduction

The simultaneous management of multiple projects by one organization (or team) is an everyday situation. Generally speaking, these projects do not have the luxury of dedicated resources, but have to share at least some public resources with others [1]. Therefore, the simultaneous management of multiple projects using shared resources is highly desired in terms of efficient resource allocation and scheduling as well as time performance optimization. Resource utilization has long been a critical issue in the practices of project management [1]–[17]. The issue is concerned with the assignment of limited resources to activities and the scheduling of activities as a result of the assignment, i.e. arranging the execution order of activities according to resource assignment. As has been indicated by Payne [1], this issue is more commonly seen when resources are shared among multiple projects which run simultaneously. Without proper coordination, a reasonable resource assignment in one project may cause a serious delay in another project due to the lockups of those commonly required resources. Hence, to help people in understanding the impact on the project execution time of different resource assignments, it is necessary to have a formal model that is capable of modeling and simulating various resource assignment strategies as well as calculating the entire time needed to accomplish all involved projects which may interact with each other [13].

Traditional models, such as Zero-one Programming [2], [3], Branch and Bound [4], Critical Path [7], [18], Tabu Search [5], [6], X-pass [19], Simulated Annealing [20], [21] and Genetic Algorithms [8], [9], have been extensively used to graphically represent, monitor, and analyze projects executing over a long period of time. Using traditional models, quantities of studies have been carried out to optimally schedule the activities in a resource constrained environment. These studies have definitely facilitated in analyzing the scheduling associated with a project and in taking corrective measures. However, these methods focus on project scheduling without considering whether the resources are available or not. Moreover, a number of factors such as activities interdependencies, criticality, conflicting priorities (criteria for prioritization of activities while scheduling limited resources) and project time performance, which affect the control over project execution have not received the attention they deserve. This oversight may be due to the inadequacy (or lacking the capability) of the traditional models to represent and analyze these factors effectively and efficiently.

As a tool to model and analyze dynamic discrete event systems, Petri nets [22]–[28] are well-known for their great power to describe concurrencies and conflicts. There are at least three main reasons for using them to model and analyze process-oriented projects: (1) Graphical nature and formal semantics; (2) the explicit model of a case state; and (3) the availability of many analysis techniques. By extending Petri nets, some research has modeled and simulated resource sharing and activity dependence in projects [10]–[14]. In [10], Liu and Horowitz first introduce syntax to mark the resources needed by activities. Kim et al. [11] use a timed Petri net to describe the resources needed for each activity in the model directly. Kumar and Ganesh [12] deal with the resources needed and consumed in each activity in a resource-constrained environment. More recently, a formal

Manuscript received September 29, 2015.
Manuscript publicized December 4, 2015.

[†]The author is with Eindhoven University of Technology, Eindhoven 5611, Netherlands.
[††]The author is with Tongji University, Shanghai 200092, China.
[†††]The author is with Donghua University, Shanghai 201260, China.
[††††]The author is with University of Toyama, Toyama-shi, 930–8555 Japan.
a) E-mail: chengjj@tongji.edu.cn
b) E-mail: gaosc@eng.u-toyama.ac.jp

DOI: 10.1587/transinf.2015EDP7397

model is designed to show project managers the impacts of resource assignment strategies on project schedules by Chen et al. [13]. This model is applied with details to model a ship repair project in [14]. Although these works have addressed resource sharing problems and resolution of resource conflicts, none of them incorporates the resource conflict resolution strategies with the project time performance optimization, i.e. choose effective resolution strategies to ensure the entire project finish in a shorter execution time.

In this paper, by taking into account the resource and time factors of multiple project management scenario, we first introduce a Petri net-based model for a kind of project constrained by resources and time, called PRT-Net for short. Then, we discuss the detailed modeling and analysis approaches. Next, resource conflict detection method with corresponding algorithm is proposed. Finally, key-activity priority strategy is presented to resolve the detected resource conflicts and optimize the time performance of the entire project. By applying the strategy, corresponding resource conflict resolution controllers are designed and implemented to build a conflict-free PRT-Net. Based on this model, project manager can judge which strategy to be used by considering the real time performance requirements.

The remainder of this paper is organized as follows. Section 2 discusses the related work. Section 3 introduces a formal specification for a kind of project constrained by resource and time. Then, a simple scenario of a multi-project management is introduced. In Sect. 4, formal definition of PRT-Net is proposed and detailed modeling steps are introduced. Section 5 first defines the key activities and then, resource conflict detection approaches with corresponding algorithms are given. Section 6 addresses the real execution time evaluation of the whole project. A priority criterion including key-activity priority strategy and waiting-short priority strategy is proposed to resolve resource conflicts as well as optimizing time performance. Section 7 presents resource conflicts resolution controller design methods, Finally, Sect. 8 draws concluding remarks.

2. Related Work

This section mainly gives an overview of a previous research related to project management with resource sharing. During the last three decades, different planning and scheduling techniques such as branch and bound algorithm, zero-one programming, genetic algorithms, X-pass, Critical Path and Petri nets, have been proposed to resolve resource constraint issues in project management. Table 1 shows some related researches on solving the resource sharing and scheduling in the project management context.

These techniques have been successfully applied in different projects for many years. However, they severely suffer from several unrealistic assumptions like the infinite availability of resources for each activity of the project, i.e. they focus on project scheduling without considering if the resources are available. Moreover, these tools are incapable to resolve conflicts arising from scarcity of resources, activity execution priorities, and resource interdependencies. As Petri nets are known for their capability to describe concurrent activities and simulate the evolvement of processes, it is adopted in our work. To the best of our knowledge, some major works in [10]–[13] and [14] have been proposed to model project management procedure with resource sharing using Petri nets. In the following, we briefly summary these works and compare the differences between them accordingly.

Table 1 Research on project management with resource sharing.

Method	Year	Reference
Zero-one Programming	1973, 1976	[2], [3]
Branch and Bound	1987	[4]
Critical Path	1995, 2008	[18], [7]
Tabu Search	1998, 2002	[5], [6]
X-pass	2000	[19]
Simulated Annealing	2004	[20], [21]
Genetic Algorithms	2008, 2009	[8], [9]
Petri Nets	1989, 1995, 1998, 2008, 2013	[10], [11], [12], [13], [14]

DesignNet, which uses three types of constructs, named, places, structure operators, and execution transitions, to describe and monitor the software development process, is first presented in [10]. The places in DesginNet can be classified into four types, namely, activities, resources, products, and status reports. Structure operators, including AND/OR operators, are used to connect places of the same type to denote work breakdown structures. Execution transitions mark the start and end of the executions of activities. Moreover, the definitions for basic properties of a successful project are formulated. However, this work provides only syntactic documentation of resources needed. The firing conditions and rules of transitions subject to resource availability are not mentioned.

An interactive project management approach based on the Petri net is introduced in [11]. In this Petri net based approach, the project manager does fine tuning of the planning and scheduling with the resource constraints embedded in the plan representation. The Petri net controller maintains the reference, monitor and predictor Petri nets, which are configured with the resource database to form a feedback loop such that the controller can monitor and control the project on-line. The model also explicitly describes reusable and consumable resources shared by more than one activity. The model, however, does not provide any mechanism in deciding the priorities between the two activities.

Kumar and Ganesh [12] model temporal relationships of activities with Petri nets and draw resources from the resource-constrained environment, which is composed of a set of tables to record resources needed for each activity, the upper limit of each resource that can be accommodated, and the resource stocked in the environment during the evolution of Petri nets. When more than one transition is enabled, priorities can be assigned to transitions in an *ad hoc* manner. However, no priorities regarding resource competition among activities are designed.

Recently, Chen et al. in [13] proposed an extended Petri net model that can describe how resources are shared and assigned among concurrent activities of multiple projects. This model is named as resource assignment Petri net (RAPN), which extends an object composition Petri net with new places, transitions, attributes, and firing rules to model resource-sharing and resource assignment strategies. Moreover, they prove that RAPN can correctly model the resource consumption behaviors of projects and can correctly compute the total elapsed time of projects. In [14], RAPN is applied to model a ship repair project. This approach enables project managers to represent and evaluate different resource assignment strategies and interactions among activities and resources. Similarity, no work on conflict resolution strategies is involved. Even though the random and deterministic resource assignment strategies are involved, they cannot be well used in real-life project. More specifically, the approach on how to deciding the priority between two conflicting tasks is not mentioned. Also, as much net elements are introduced, some to the existing analysis techniques for traditional Petri nets are not applicable.

Generally speaking, these works [10]–[14] are analyzed by their capability of modeling project management with resource sharing by extending Petri nets. However, all existing work suffers at least the following limitations: (1) they extended classical Petri net with some extra elements, which makes existing analysis techniques not applicable; (2) these approaches lack detailed automatic construction steps, which makes the modeling process extremely hard and error-prone; (3) effective strategies to decide the priorities between the two conflicting activities are not discussed; and (4) project management is a real-time service where timeliness is critical to project success, i.e. project always has a deadline, thus, time optimization is needed to ensure its quality accomplishment.

3. Formal Definition for Project Constrained by Resource and Time

In this section, we first introduce the formal definition for a kind of project constrained by resources and time, and then give a simple example.

3.1 Formal Definition of Project Constrained by Resource and Time

Let $Z = \{0, 1, 2, \ldots\}$, $Z_n = \{1, 2, \ldots n\}$ where n is a positive integer and R^+ be the set of non-negative real number.

Definition 1 *A project constrained by resources and time* is a seven-tuple $PRT =< Activity, Resource, Time, Relation, f_{AR}, f_l, f_u >$, where
(1) $Activity = \{activity_i | i \in Z_n\}$ is the activity set;
(2) $Resource = \{resource_i | i \in Z_m\}$ is the resource set;
(3) $Time = \{time_i | i \in Z_l\}$ is the time duration set, where $time_i \geq 0$;
(4) $Relation \subseteq Activity \times Activity$ is the relation set, representing the connection relations between activities;

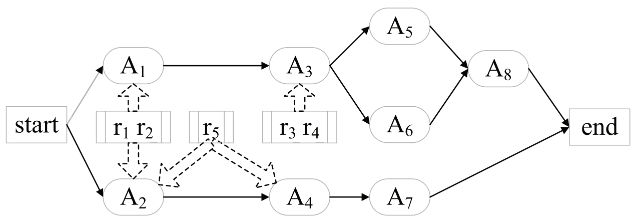


Fig. 1 A simple example of multi-project management process.

(5) f_{AR}: $Activity \rightarrow f_{AR}(Resource)$ is the resource function of a project; and
(6) Given $activity \in Activity$, $f_l(activity) \in R^+$ is the minimum time required to execute an activity while $f_u(x) \in R^+$ is the maximum one, satisfying $f_l(x) \leq f_u(x)$.

Definition 1 presents the formal specification of a kind of project constrained by resources and time, where (1) The set *Activity* defines all the activities involved in the project; (2) The set *Resource* defines the required resources of all activities in the project; (3) For $activity_1 \in Activity$ and $resource_1 \in Resource$, if $f_{AR}(activity_1) = \{resource_1\}$, it means that the execution of $activity_1$ requires $resource_1$. $\forall activity_1 \in Activity$, the execution of $activity_1$ does not require any resource if $f_{AR}(activity_1) = \emptyset$. In the project, the resource is occupied by activity exclusively, if the execution of $activity_1 \in Activity$ requires $resource_1 \in Resource$, $resource_1$ is locked while $activity_1$ is executing; (4) For $activity_1$, $activity_2 \in Activity$, if $f_{AR}(activity_1) \cap f_{AR}(activity_2) \neq \emptyset$, we say that $activity_1$ and $activity_2$ share same resources. If $resource_1 \in Resource$ is shared by $activity_1$ and $activity_2$, $resource_1$ will be locked while is $activity_1$ executing and $activity_2$ has to wait until $activity_1$ is finished and $resource_1$ is released; (5) *Relation* defines the connection relations among activities. $\forall activity_i, activity_j \in Activity$, if $(activity_i, activity_j) \in Relation$, it means $activity_j$ cannot start before $activity_i$. We call $activity_i$ be a pre-activity of $activity_j$, and $activity_j$ as a post-activity of $activity_i$; and (6) The set *Time* defines time constraints of all activities. For each activity, there are two timing functions f_l and f_u, representing the minimum and maximum execution time respectively. If its actual execution time of $activity_i$ is *Atime*, then we have $f_l(activity_i) \leq Atime \leq f_u(activity_i)$.

3.2 A Simple Example

A simple example of a multi-project management process is presented in Fig. 1.

Table 2 presents the detailed information of the project, including the time constraints, the connection relations, and the resources required by each activity.

According to the formal definition of project constrained by resource and time, we have the following explanations for Table 2: (1) The project is composed of eight activities, denoted by $Activity = \{A_i | 1 \leq i \leq 8, i \in Z\}$. More precisely, it involves two simultaneously executed projects which start at the same time, i.e. *project_1* con-

Table 2 Information of a multiple project management process.

Activity	Minimum Execution Time	Maximum Execution Time	Pre-Activities	Resources
A_1	10	12	$\varnothing$	$\{r_1, r_2\}$
A_2	3	7	$\varnothing$	$\{r_1, r_2, r_5\}$
A_3	15	20	$\{A_1\}$	$\{r_3, r_4\}$
A_4	10	15	$\{A_2\}$	$\varnothing$
A_5	5	15	$\{A_3\}$	$\varnothing$
A_6	10	20	$\{A_3\}$	$\varnothing$
A_7	10	18	$\{A_4\}$	$\{r_5\}$
A_8	1	2	$\{A_5,A_6\}$	$\varnothing$

sists of activities $\{A_1, A_3, A_5, A_6, A_8\}$ and $project\_2$ contains activities $\{A_2, A_4, A_7\}$. In $project\_1$, A_1 is a pre-activity of A_3 (or A_3 is the post-activity of A_1), which means A_3 cannot start before A_1; (2) The resources required are denoted by $Resource = \{r_j | 1 \le j \le 5, j \in Z\}$. From Table 1, we have $f_{AR}(A_1) \cap f_{AR}(A_2) = \{r_1, r_2\}$, it means that both A_1 and A_2 require resources $r1$ and $r2$, i.e., $r1$ and $r2$ are shared by A_1 and A_2; and (3) The time set is $Time = \{1, 2, 3, 5, 7, 10, 12, 15, 18, 20\}$. For example, $f_l(A_1) = 10$ and $f_u(A_1) = 12$ represent that the minimum and maximum execution time of A_1 are 10 and 12 time units (for example, days) respectively. This means that to finish A_1 needs at least 10 time units and at most 12 time units.

4. Modeling Project Constrained by Resource and Time with PRT-Net Model

In this section, we propose the formal definition of a Petri net-based model for this kind of project, named PRT-Net.

4.1 Basic Concepts of Petri Nets

It is assumed that readers are familiar with the basic concepts of Petri nets [22]–[28]. Some of the essential terminologies and notations are listed as follows.

A tuple $N = (P, T; F)$ is named a net if the following conditions are satisfied: (1) $P \cap T = \varnothing$ and $P \cup T \neq \varnothing$; (2) $F \subseteq (P \times T) \cup (T \times P)$; (3) $Dom(F) \cup Cod(F) = P \cup T$; where $Dom(F) = \{x \in P \cup T | \exists y \in P \cup T : (y, x) \in F\}$ and $Cod(F) = \{x \in P \cup T | \exists y \in P \cup T : (x, y) \in F\}$. For all $x \in P \cup T$, the set ${}^\bullet x = \{y | y \in P \cup T \wedge (y, x) \in F\}$ is the pre-set of x, and $x^\bullet = \{y | y \in P \cup T \wedge (x, y) \in F\}$ is the post-set of x.

Definition 2 A *Petri net* is a 4-tuple $\Sigma = (P, T; F, M_0)$, where $N = (P, T; F)$ is a net, and $M_0 : P \to Z$ is the initial marking of Σ. A marking M is reachable from M_0 if there is a transition firing sequence δ such that $M_0[\delta > M$. We use $R(M_0)$ to represent the set of all reachable states from M_0.

We usually use a rectangle to represent a transition, a circle to represent a place, and a dot to represent a token. An initial marking is denoted by M_0. p is marked by M iff $M(p) > 0$. A transition $t \in T$ is enabled under M, if and only if $\forall p \in {}^\bullet t : M(p) > 0$, denoted as $M[t >$. If $M[t >$ holds, t may fire, resulting in a new marking M', denoted as $M[t > M'$, such that $M'(p) = M(p) - 1$ if $\forall p \in {}^\bullet t \backslash t^\bullet, M'(p) = M(p) + 1$ if $\forall p \in t^\bullet \backslash {}^\bullet t$, and otherwise $M'(p) = M(p)$.

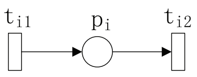


Fig. 2 PRT-net for one activity without resources.

4.2 PRT-Net

Definition 3 $\Sigma_{\alpha\beta} = (P, T; F, M_0, \alpha, \beta)$ is a *PRT-Net* if the following conditions are satisfied:
(1) $(P, T; F, M_0)$ is a Petri net;
(2) $P = P_A \cup P_R \cup P_L, P_A \cap P_R = \varnothing, P_A \cap P_L = \varnothing$ and $P_R \cap P_L = \varnothing$ where P_A is an activity place set, P_R is a resource place set and P_L is a logic place set;
(3) $\forall p_a \in P_A, p_r \in P_R$, the execution of p_a requires p_r if $p_r \subseteq (p_a^\bullet)^\bullet$ and $p_r \subseteq^\bullet ({}^\bullet p_a)$;
(4) $\alpha : P_A \to R^+$. $\forall p_a \in P_A$, $\alpha(p_a)$ is the minimum time to execute activity p_a;
(5) $\beta : P_A \to R^+$. $\forall p_a \in P_A, \beta(p_a)$ is the maximum time to execute activity p_a, where $\alpha(p_a) \le \beta(p_a)$; and
(6) $\forall p \in P, M_0(p) = 1$ if ${}^\bullet p = \varnothing \vee p \in P_R$, otherwise $M_0(p) = 0$.

The firing rule of the PRT-Net is same as that of traditional Petri nets. $\forall t \in T$ and $\forall M \in R(M_0)$, t is enabled under M if: $\forall p \in {}^\bullet t$, $M(p) \ge 1$. All properties, such as reachability, boundedness, etc., can be defined similarly to those in a traditional Petri net. The only difference between PRT-Net and Petri nets lies in that: two timing functions α and β, which represent the minimum and maximum execution time respectively, are labeled on each activity place $p_a \in P_A$.

4.3 Modeling Project Constrained by Resource and Time with PRT-Net

Modeling approaches for a project process based on the PRT-Net contain two phases: (1) modeling project activities with the PRT-Net; and (2) modeling activity dependencies with the PRT-Net. Detailed discussions for these two phases are given in the following.

4.3.1 PRT-Net Model for One Activity

(1) *PRT − Net Model for one single activity without resources*

A single activity without resources is represented by one place and two transitions in the PRT-Net, as shown in Fig. 2. Place p_i represents activity A_i, and transitions t_{i1} and t_{i2} represent the start and end of activity A_i respectively. If p_i contains a token, it means that the corresponding activity is on-going. Functions, such as $\alpha(p_i)$ and $\beta(p_i)$ are labeled on p_i. Single activity A_i can be denoted by $[t_{i1}, p_i, t_{i2}]$ in an PRT-Net. In the following, activity p_i means activity A_i.
(2) *PRT − Net Model for one single activity with resources*

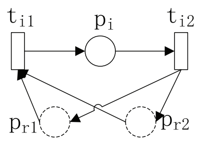


Fig. 3 PRT-net for one activity constrained by resources.

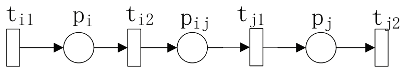


Fig. 4 PRT-net for two sequential activities.

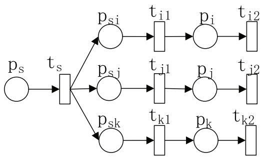


Fig. 5 PRT-net for three activities without pre-activities.

If one activity is constrained by resources, such activity is represented by one place, two transitions and another set of places in the PRT-Net. Each kind of resource is represented by a special place and no time constraint is labeled on resource places. The start of activity needs resources as input, and the activity releases resources when it finishes. The PRT-Net for one activity constrained by resources is shown in Fig. 3, where resources (p_{r1} and p_{r2}) are represented by places drawn with broken lines.

4.3.2 PRT-Net Model for the Whole Project

PRT-Net of a project process can be obtained by the following constructs. In our approach, we only consider the preorder relations between activities to obtain the entire PRT-Net. Basic structures, such as sequential and parallel ones, can be expressed with the following steps.

(1) If activity $[t_{i1}, p_i, t_{i2}]$ is one of the pre-activities of $[t_{j1}, p_j, t_{j2}]$, i.e., $(p_i, p_j) \subseteq Relation$, then place p_{ij} is added between t_{i2} and t_{j1} to connect them, as shown in Fig. 4. The new place p_{ij} satisfies $\alpha(p_{ij}) = 0$ and $\beta(p_{ij}) = 0$. Places to keep the connections, such as p_{ij}, are logic places where $p_{ij} \subseteq P_L$.

(2) For activities $[t_{i1}, p_i, t_{i2}]$ without pre-activities (i.e., ${}^\bullet t_{i1} = \varnothing$), we add start place p_s, start transitions t_s, and p_{si} to connect them. PRT-Net for three activities without pre-activities is shown in Fig. 5.The start transition t_s satisfies ${}^\bullet t_s = \{p_s\}$, $t_s^\bullet = \{p_{si}, p_{si}, p_{si}\}$ and the start place p_s satisfies ${}^\bullet p_s = \{\varnothing\}$ and $p_s^\bullet = \{t_s\}$.

(3) For activities $[t_{i1}, p_i, t_{i2}]$ without post-activities (i.e., $t_{i1}^\bullet = \varnothing$), we add end place p_e, end transitions t_e, and p_{ei} to connect them. PRT-Net for three activities without post-activities is shown in Fig. 6. The end transition t_e satisfies ${}^\bullet t_e = \{p_{ei}, p_{ei}, p_{ei}\}$ and $t_e^\bullet = \{p_e\}$ and the end place p_e satisfies ${}^\bullet p_e = \{t_e\}$ and $p_e^\bullet = \{\varnothing\}$.

(4) The initial marking M_0 of an PRT-Net satisfies: $M_0(p) = 1$ if $p = p_s \vee p \in P_R$, otherwise $M_0(p) = 0$.

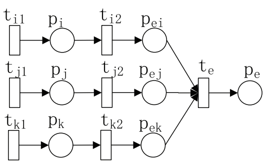


Fig. 6 PRT-net for three activities without post-activities.

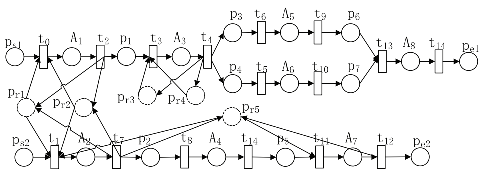


Fig. 7 PRT-net for activities with preorder relations.

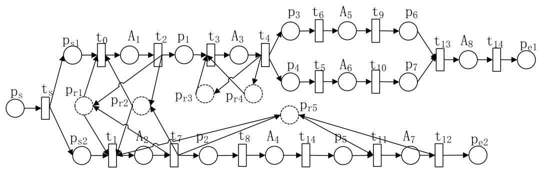


Fig. 8 Add start place to the PRT-net.

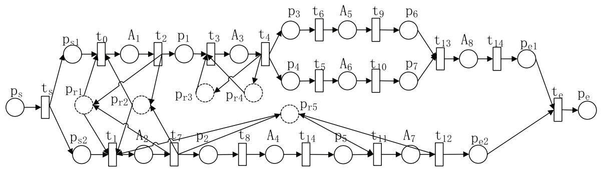


Fig. 9 Add end place to the PRT-net.

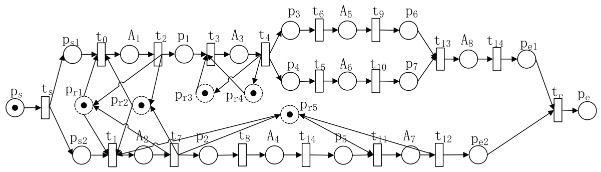


Fig. 10 Add the initial marking to the PRT-net.

With the above mentioned four steps, we can obtain the PRT-Net of a project step by step. Take the multi-project management scenario in Sect. 3.2 as an example. The detailed construction steps are given in the following:

Step 1: Connect each two activities that have precedence relations as shown in Fig. 7.

Step 2: Add p_s and t_s for activities without pre-activities, as shown in Fig. 8.

Step 3: Add p_e and t_e for activities without post-activities, as shown in Fig. 9.

Step 4: Add the initial marking of a PRT-Net, as shown in Fig. 10.

4.3.3 Reduction Rule for PRT-Net Model

According to Fig. 10, it is obviously that the PRT-Net is so

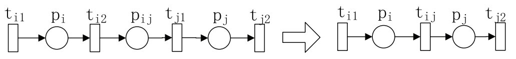


Fig. 11 Reduction example.

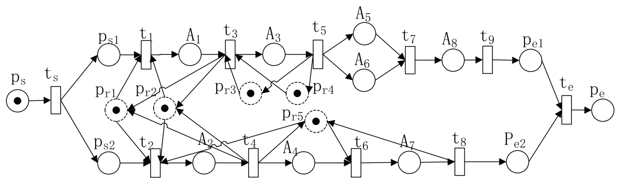


Fig. 12 Reduced PRT-net model.

complicated and contains many excessive or unnecessary places and transitions. Therefore, it is not convenient to analyze its properties. Next, a rule is introduced to reduce its scale by deleting some logic places.

Reduction Rule A logic place $p_i \in P_L$ ($p_i \neq p_s$ and $p_i \neq p_e$) can be deleted from a PRT-Net if the deletion of p_i keeps the connection relation between activities invariant. If a logic place p_i such that $(t_{i1}, p_i) \in F$ and $(p_i, t_{i2}) \in F$ is deleted, its input transition t_{i1} and output transition t_{i2} will be merged into one. This reduction process is shown in Fig. 11. After reduction, all places corresponding to activities and their connection relations stay invariant.

Take the multi-project PRT-Net model in Fig. 10 as an example, the reduced model is shown in Fig. 12. Logic places, p_{s1}, p_{s2}, p_{e1} and p_{e2}, are reserved to keep the correct connection relation between activities.

5. Resource Conflict Detection Approaches

Next, the resource conflict detection methods and a key-activity priority resolution strategy for project management are discussed. Before rendering them, the approach on how to decide key activities are addressed.

5.1 Key Activities

Without considering resource conflicts factor, if each activity is finished in its minimum time, the earliest time to start activity p, denoted by $T_{e1}(p)$, is as follows:

$$T_{e1}(p) = \begin{cases} 0 & p = p_s \\ \max\{T_{e1}(p') + \alpha(p') | p' \in {}^\bullet({}^\bullet p)\} & otherwise \end{cases} \tag{1}$$

Without considering resource conflicts factor, if each activity is finished in its maximum time, the earliest time to start activity p, denoted by $T_{e2}(p)$, is as follows:

$$T_{e2}(p) = \begin{cases} 0 & p = p_s \\ \max\{T_{e2}(p') + \beta(p') | p' \in {}^\bullet({}^\bullet p)\} & otherwise \end{cases} \tag{2}$$

Let $T_{E1} = T_{e1}(p_e)$ and $T_{E2} = T_{e2}(p_e)$, where p_e is the termination activity. T_{E1} and T_{E2} are the ideal minimum and maximum execution time respectively. To ensure the process be finished in T_{E1}, the latest time to start activity p, denoted by $T_{l1}(p)$, is as follows:

$$T_{l1}(p) = \begin{cases} T_{E1}(p) & p = p_e \\ \min\{T_{l1}(p') - \alpha(p) | p' \in (p^\bullet)^\bullet\} & otherwise \end{cases} \tag{3}$$

Similarly, to ensure the process be finished in T_{E2}, the latest time to start activity p, denoted by $T_{l2}(p)$, is as follows:

$$T_{l2}(p) = \begin{cases} T_{E2}(p) & p = p_e \\ \min\{T_{l2}(p') - \beta(p) | p' \in (p^\bullet)^\bullet\} & otherwise \end{cases} \tag{4}$$

According to the aforementioned-mentioned four computational formulas, $T_{e1}(p), T_{e2}(p), T_{l1}(p)$ and $T_{l2}(p)$ of activities in Fig. 12 are obtained and shown in Table 3.

Table 3 $T_{e1}(p), T_{e2}(p), T_{l1}(p)$ and $T_{l2}(p)$ of each activity in Fig. 12.

Activity	p_s	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	p_e
$T_{e1}(p)$	0	0	0	10	3	25	25	13	35	36
$T_{e2}(p)$	0	0	0	12	7	32	32	22	52	54
$T_{l1}(p)$	0	0	13	10	16	30	25	26	35	36
$T_{l2}(p)$	0	0	14	12	21	37	32	36	52	54

Definition 4 Let $\Sigma_{\alpha\beta} = (P, T; F, M_0, \alpha, \beta)$ be a PRT-Net, $\forall p \in P_A$ is a *key activity* if $T_{e1}(p) = T_{l1}(p)$ or $T_{e2}(p) = T_{l2}(p)$.

The key activities have their real semantics. It is obviously that if the execution time of any of key activities is extended, the execution time of the entire project will definitely be prolonged. As a result, in order to ensure the project enjoy a relatively high time performance, execution time of key activities should be maintained. According to Definition 4, key activities in our multi-project case are A_1, A_3, A_6 and A_8. For example, A_3 is a key activity and its delay will definitely postpone the finish time of the entire project, while A_4 is not a key activity, the delay of A_4 may not influence the execution time of the project. Assume that resource conflict occurs during the execution of a project, whether they are properly tackled or not is closely related with the project execution time. Therefore, methods to detect and resolve resource conflicts are highly desired to improve the time performance of the entire project.

5.2 Resource Conflict Detection

Let $\Sigma_{\alpha\beta} = (P, T; F, M_0, \alpha, \beta)$ be a PRT-Net, we assume that $T_{start}(p)$ and $T_{end}(p)$ represent the real start and finish time of activity p in the following discussion.

Definition 5 Let $\Sigma_{\alpha\beta} = (P, T; F, M_0, \alpha, \beta)$ be a PRT-Net. For any two activities $p_i, p_j \in P_A (p_i \neq p_j)$, p_i and p_j have *resource dependency*, denoted as $p_i \Theta p_j$ if ${}^\bullet({}^\bullet p_i) \cap {}^\bullet({}^\bullet p_j) \cap P_R \neq \emptyset$.

Definition 6 Let $\Sigma_{\alpha\beta} = (P, T; F, M_0, \alpha, \beta)$ be a PRT-Net. For any two activities $p_i, p_j \in P_A(p_i \neq p_j)$, p_i and p_j are in a *resource conflict*, denoted as $p_i \otimes p_j$, if (1) $p_i \Theta p_j$; and (2) $[T_{start}(p_i), T_{end}(p_i)]$ and $[T_{start}(p_j), T_{end}(p_j)]$ are overlapping.

Here, based on Definitions 5-6 we present an algorithm to detect resource conflicts in the project constrained by resource and time.

```
Algorithm 1: Detect Resource Conflicts
Input: $\Sigma_{\alpha\beta} = (P, T; F, M_0, \alpha, \beta)$
Output: $ConflictSet = \{(p_i, p_j) | p_i \otimes p_j\}$
/*Step 1: initialization*/
Step 1: $ConflictSet \leftarrow \varnothing, T_{e1}(p_s) \leftarrow 0, T_{e2}(p_s) \leftarrow 0,$
$T_{l1}(p_s) \leftarrow 0, T_{l2}(p_s) \leftarrow 0;$
/*Step 2: to detect resource dependency among
activities*/
Step 2: FOR $\forall p_i, p_j \in P_A(p_i \neq p_j)$ DO
  IF $^\bullet(^\bullet p_i) \cap ^\bullet(^\bullet p_j) \cap P_R \neq \varnothing$ THEN
  $ConflictSet \leftarrow ConflictSet \cup \{(p_i, p_j)\};$
  END IF
  END DO
/*Step 3: to detect if time interval of activities with
resource dependency are overlapping*/
Step 3: IF $ConflictSet \neq \varnothing$ THEN
  For $\forall (p_i, p_j) \in ConflictSet$ DO
  (3.1) Calculate $T_{e1}(p_i), T_{e2}(p_i), T_{e1}(p_j), T_{e2}(p_j);$
  (3.2) IF $[T_{e1}(p_i), T_{e2}(p_i) + \beta(p_i)] \cap [T_{e1}(p_j), T_{e2}(p_j)$
  $+\beta(p_j)] \neq \varnothing$ THEN
  GOTO Step3;
  (3.3) $ConflictSet \leftarrow ConflictSet - \{(p_i, p_j)\};$
  END DO
  END IF
Step 4: Output $ConflictSet$
```

In Algorithm 1, the complexity of Step 2 is $O(|P_A|^2)$. Because $O(|ConflictSet|) \leq O(|P_A|)$, that of Step 3 is $O(|P_A|)$. Hence, Algorithm 1 has its computational complexity $O(|P_A|^2)$. By executing Algorithm 1, we can obtain that A_3 and A_4 are in resource conflicts for the multi-project management scenario in Sect. 3.2. Resource conflicts may exert a negative effect (delay or even suspend) on the execution of a project. In this way, conflict resolution strategies are needed to resolve these conflicts and ensure the whole project be finished in a shorter time.

6. Time Performance Analysis

Resource conflict detection methods and its corresponding algorithm for a kind of project constrained by resources and time are discussed in last section. In this section, time performance analysis of this kind of project is conducted with details.

6.1 Real Execution Time Calculation

In a multi-project execution context, one activity starts only after the end of all its pre-activities and under condition that its required resources are available.

During the project execution, even if each activity can be finished in its minimum time, $T_{e1}(p)$ may not be the earliest time to start p because some activities prior to p may be delayed due to resource conflicts. The actual earliest time to start p is calculated as follows:

$$E_1(p) = \begin{cases} 0 & p = p_s \\ T_{e1}(p) + W_1(p, p_1) | p \otimes p_1 & otherwise \end{cases} \tag{5}$$

where $W_1(p, p_1)$ is the waiting time of p for p_1 when p and p_1 execute in its minimum time.

$$W_1(p, p_1) = \begin{cases} 0 & T_{e1}(p_1) + \alpha(p_1) \\ & \leq T_{e1}(p) \\ T_{e1}(p_1) + \alpha(p_1) - T_{e1}(p) & otherwise \end{cases} \tag{6}$$

Similarly, if each activity is finished in its maximum time, the earliest time to start p is:

$$E_2(p) = \begin{cases} 0 & p = p_s \\ T_{e2}(p) + W_2(p, p_1) | p \otimes p_1 & otherwise \end{cases} \tag{7}$$

where $W_2(p, p_1)$ is the waiting time of p for p_1 when p and p_1 execute in its maximum time.

$$W_2(p, p_1) = \begin{cases} 0 & T_{e2}(p_1) + \beta(p_1) \\ & \leq T_{e2}(p) \\ T_{e2}(p_1) + \beta(p_1) - T_{e2}(p) & otherwise \end{cases} \tag{8}$$

For $\forall p \in P_A$, $E_1(p)$ and $E_2(p)$ are the real earliest time to start p if all activities p_i before p can be finished in $\alpha(p_i)$ and $\beta(p_i)$ respectively. If we denote $TE_1 = E_1(p_e)$ and $TE_2 = E_2(p_e)$ where p_e is the end place, then TE_1 and TE_2 are the earliest time to finish the project if each activity is finished in its minimum and maximum time respectively.

6.2 Time Performance Evaluation

Next, we discuss two resolution strategies, named key-activity priority strategy (KPS) and waiting-short priority strategy (WPS) respectively, to remove resource conflicts and achieve better time performance.

6.2.1 Key-activity Priority Strategy

Definition 7 *Key − activity priority strategy* : $\forall p_i \otimes p_j$, if p_i is a key activity but p_j is not, the priority of p_i is higher than p_j, i.e., $W(p_i, p_j) = 0$ and $W(p_j, p_i) > 0$.

The key-activity priority strategy defines two priority levels, a key level and a non-key level, for all project activities. Activities with key level can be obtained by Definition 4 and for our scenario case the key level contains $\{A_1, A_3, A_6$ and $A_8\}$ and the rest activities belong to the non-key level. All key activities will influence the finish time of a project, and a key activity will have a higher priority than a non-key activity while they are in a resource conflict. The key-activity priority strategy only suits to resolve the conflicts between key activities and non-key activities, but it is not effective enough when facing the resource conflicts between two non-key activities or two key activities. Therefore, another resolution strategy will ensure that the waiting time of all activities be as short as possible so that the entire project can be finished at the earliest possible time.

Definition 8 *Waiting − short priority strategy* : $\forall p_i \otimes p_j$, if $W(p_i, p_j) \leq W(p_j, p_i)$, let $W(p_i, p_j) = 0$. Otherwise, let $W(p_j, p_i) = 0$.

Similarly, the waiting-short priority strategy also defines two priority levels, denoted as a waiting-short level and a non-waiting-short level. A waiting-short activity will have a higher execution priority than a non-waiting-short activity when they are in a resource conflict. Whether an activity is a waiting-short activity or not can be decided by Definition 8. By integrating the key-activity priority strategy and the waiting-short priority strategy, the priority criteria used in our work for resource conflict resolution are as follows: (1) First priority: activities with key level; and (2) Second priority: activities with waiting-short level. More accurately, when resource conflicts occurs if one of the conflicting activities is a key activity, then the key-activity priority strategy will be applied and the waiting-short priority strategy will not be considered. In other cases, the later strategy will be chosen.

6.2.2 Project Execution Time Evaluation and Comparison

To demonstrate the effectiveness and advantages of our resolution strategies, in this section, time performance of the project is evaluated, i.e. we compare the project execution time by using our proposed conflict resolution strategies with that not using our strategies. In the following discussion, we denote the methods using our proposed strategy criteria as Method$_1$. Formally, the key-activity priority strategy and the waiting-short priority strategy are applied with our aforementioned priority criteria in Method$_1$. This means that the priority of a key activity regarding the shared resources is higher than that of a non-key activity. More, when facing the resource conflicts between two non-key activities or two key activities, this method choose the short-level activities with higher execution priority. On the contrary, neither key-activity priority strategy nor waiting-short priority strategy is applied in Method$_2$. This means that the priority of a non-key activity regarding the shared resources is higher than that of a key activity. When facing the resource conflicts between two non-key activities or two key activities, this method choose the non-short-level activities with higher execution priority.

Following the approaches in Sect. 6.1, the real execution time of each activity of the multi-project case, denoted as $E_1(p)$ and $E_2(p)$, are obtained and demonstrated in Tables 4 and 5 respectively. Table 4 shows the execution time of the whole project by applying our resolution strategies (Method$_1$) while Table 5 is the result without any strategy (Method$_2$).

Table 4 $T_{E1}(p)$ and $T_{E2}(p)$ of each activity in Fig. 12 with method$_1$.

Activity	p_s	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	p_e
$W_1(p, p_1)$	0	0	10	0	0	0	0	0	0	0
$E_1(p)$	0	0	10	10	13	25	25	23	35	36
$W_2(p, p_1)$	0	0	12	0	0	0	0	0	0	0
$E_2(p)$	0	0	12	12	19	32	32	34	52	54

Table 5 $T_{E1}(p)$ and $T_{E2}(p)$ of each activity in Fig. 12 with method$_2$.

Activity	p_s	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	p_e
$W_1(p, p_1)$	0	3	0	0	0	0	0	0	0	0
$E_1(p)$	0	3	0	13	3	28	28	13	38	39
$W_2(p, p_1)$	0	7	0	0	0	0	0	0	0	0
$E_2(p)$	0	7	0	19	7	39	39	22	59	61

As mentioned $E_1(p_e)$ and $E_2(p_e)$ are the earliest time to finish the entire project if each activity is finished in its minimum and maximum time respectively. From Table 4, we have $E_1(p_e) = 36$ and $E_2(p_e) = 54$, which are same as those in Table 3 (the ideal execution time without considering resource conflicts factor). While in Table 5, $E_1(p_e) = 39$ and $E_2(p_e) = 61$, which shows that the execution time of the project is prolonged by resource conflicts. Therefore, we can conclude that our proposed resolution strategies can effectively guarantee the resource scheduling and time performance of the project with high level.

7. Resource Conflict Resolution Controller Design

In this section, we detail how to design resolution controller to resolve the detected resource conflicts and construct the conflict-free PRT-Net using the previously introduced resolution strategies. Based on the preceding resource conflict resolution strategies, such as KPS and WPS, a conflict between two activities can be resolved by setting one activity executes first, and the other has to wait. To explicitly express the waiting time in a PRT-Net, we introduce the notion of *idle activity* to represent the waiting time. By adding idle activities to a PRT-Net, it is constructed to be conflict-free as the execution duration of the conflicting activities are separated.

Definition 9 Let $\Sigma_{\alpha\beta} = (P, T; F, M_0, \alpha, \beta)$ be a PRT-Net. An *idle activity*, denoted as t_{vi}, satisfying $\alpha(A_{vi}) = W_1(A_m, A_n)$ and $\beta(A_{vi}) = W_2(A_m, A_n)$ where $A_m \otimes A_n$ and A_{vi} is added before A_m.

As $W(A_m, A_n)$ is the waiting time of A_m for resources that are currently occupied by A_n if A_m and A_n are checked to be in a resource conflict. More specifically, we have $W_1(A_m, A_n) = T_{e1}(A_n) + \alpha(A_n) - T_{e1}(A_m)$ and $W_2(A_m, A_n) = T_{e2}(A_n) + \beta(A_n) - T_{e2}(A_m)$ to represent the minimum and maximum execution time respectively. After adding idle activities, a PRT-Net is denoted as Ψ_{PRT}. And a Ψ_{PRT} is proved to be conflict-free as the execution duration of the conflict-

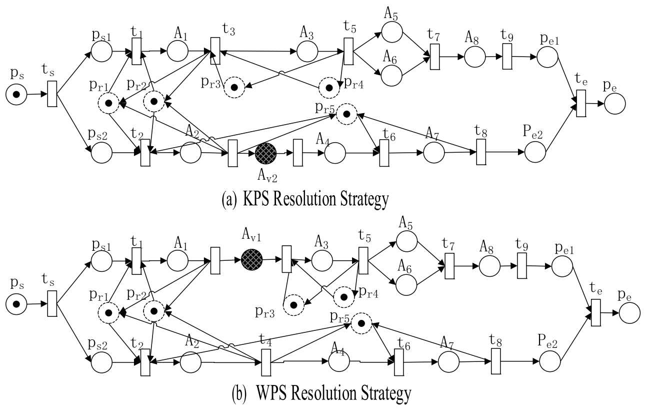


Fig. 13 Resource conflict resolution controller design.

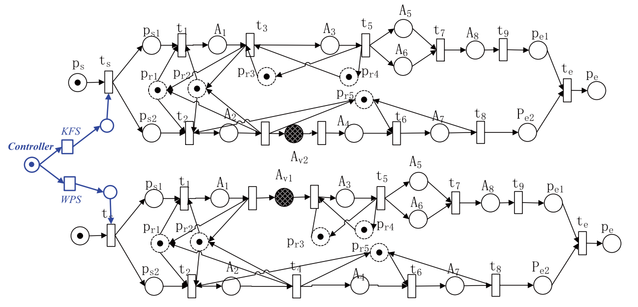


Fig. 14 Integrated resource conflict resolution controller design.

ing activities are separated. The detailed proof is given in the next Theorem.

Theorem 1 Let $\Sigma_{\alpha\beta} = (P, T; F, M_0, \alpha, \beta)$ be a PRT-Net. Ψ_{PRT} is conflict-free if it is obtained by adding idle activities.

Proof. Assume that A_i and A_j are two conflict activities. If A_i and A_j are executed in their minimum time, they satisfy (1) $A_i \Theta A_j$; and (2) $[T_{e1}(A_i), T_{e1}(A_i) + \alpha(A_i)] \cap [T_{e1}(A_j), T_{e1}(A_j) + \alpha(A_j)] \neq \emptyset$. If A_j executes first, then A_i has to wait. We then add an idle activity A_{vi}, such that $\alpha(A_{vi}) = W_1(A_i, A_j)$ to represent its minimum duration in Ψ_{PRT}. In a Ψ_{PRT}, the execution interval of A_i and A_j are $[T_{e1}(A_i) + W_1(A_i, A_j), T_{e1}(A_i) + \alpha(A_i) + W_1(A_i, A_j)]$ and $[T_{e1}(A_j), T_{e1}(A_j) + \alpha(A_j)]$, where $W_1(A_i, A_j) = T_{e1}(A_j) + \alpha(A_j) - T_{e1}(A_i)$. Obviously, $[T_{e1}(A_j) + \alpha(A_j), T_{e1}(A_j) + \alpha(A_j) + \alpha(A_i)] \cap [T_{e1}(A_j), T_{e1}(A_j) + \alpha(A_j)] = \emptyset$. Similarly, if A_i and A_j are executed in their maximum time, they also satisfy (1) $A_i \Theta A_j$; and (2) $[T_{e2}(A_i), T_{e2}(A_i) + \beta(A_i)] \cap [T_{e2}(A_j), T_{e2}(A_j) + \beta(A_j)] \neq \emptyset$. If A_j executes first, then A_i has to wait. We then add an idle activity A_{vi}, such that $\beta(A_{vi}) = W_2(A_i, A_j)$ to represent its maximum duration in Ψ_{PRT}. In a Ψ_{PRT}, the execution interval of A_i and A_j are $[T_{e2}(A_i) + W_2(A_i, A_j), T_{e1}(A_i) + \beta(A_i) + W_2(A_i, A_j)]$ and $[T_{e2}(A_j), T_{e2}(A_j) + \beta(A_j)]$, where $W_2(A_i, A_j) = T_{e2}(A_j) + \beta(A_j) - T_{e2}(A_i)$. Obviously, $[T_{e2}(A_j) + \beta(A_j), T_{e2}(A_j) + \beta(A_j) + \beta(A_i)] \cap [T_{e2}(A_j), T_{e2}(A_j) + \beta(A_j)] = \emptyset$.

Other activities in resource conflict can be treated the same way. Thus, Ψ_{PRT} is conflict-free.

Consider for example the PRT-Net model in Fig. 12, where we have A_3 and A_4 are in resource conflicts. According to the resolution strategies, no matter what strategy is chosen, one of the conflict activity executes first and the other has to wait. On one hand, suppose that we apply the *KPS*, and activity A_3 executes first and A_4 has to wait. Then, we add an idle activity A_{v2} before A_4 to force the waiting time in the PRT-bet, as shown in Fig. 13 (a). The idle activity A_{v2} satisfies that $\alpha(A_{v2}) = T_{e1}(A_3) + \alpha(A_3) - T_{e1}(A_4)$ and $\beta(A_{v2}) = T_{e2}(A_3) + \beta(A_3) - T_{e2}(A_4)$. On one hand, suppose that we apply the *WPS*, and activity A_4 executes first and A_3 has to wait in this circumstance. Then, we add an idle activity A_{v1} before A_3 to force the waiting time in the PRT-Net, as shown in Fig. 13 (b). The idle activity A_{v1} is drawn with a rectangle with grid, satisfying that $\alpha(A_{v1}) = T_{e1}(A_4) + \alpha(A_4) - T_{e1}(A_3)$ and $\beta(A_{v1}) = T_{e2}(A_4) + \beta(A_4) - T_{e2}(A_3)$. The Ψ_{PRT} in Figs. 13 (a)-(b) are conflict-free because the time interval of A_3 and A_4 are separated by adding virtual activities A_{v1} and A_{v2}, which is prove in Theorem 1.

The integrated resolution controller which supports both the *WPS* and the *KPS* to resolve the resource conflict between A_3 and A_4 is shown in Fig. 14. If the *transition* (*KPS*) is fired, it means that the *KPS* is selected to resolve the conflict and it is functionally identical with the model in Fig. 13 (a). If the *transition* (*WPS*) is fired, it means that the *WPS* is selected to resolve the conflict, and it is functionally identical with the model in Fig. 13 (b). In real-life cases, it is determined by the project managers which strategies to be applied according to the project execution time demands.

Essentially, the integrated resolution controller combines all available resolution strategies for a specific resource conflict. It is determined by the users (here we refer to project managers) preference to choose which one to execute. It is worth mentioning that some resource conflicts cannot be resolved by certain strategies. For example, if two activities in conflicts start at the same time, the *SAPS* cannot work effectively. Moreover, the *KAPS* only suits to resolve the conflicts between key activities and non-key activities, but it is not effective enough when facing the resource conflicts between two non-key activities or two key activities. In this way, we need to work out which resolution strategies are applicable for each specific resource conflict.

8. Conclusion

In this paper, the formal modeling and analysis methods for a kind of project constrained by the resource and time factors based on Petri net is addressed. Resource conflict detection and resolution strategies are discussed to achieve a high time performance. Our analysis method is presented before the start of a project, which are based on the specification at build-time. During its real execution, the minimum and maximum execution time in the specification will be replaced by the actual execution time. In this case, we can analyze how executed activities and resources influence the execution of the remainder activities, with the goal to minimize the time to execute the entire project.

This work opens the door to the following future research. A complex project is often executed by geographically dispersed partners or different organizations. As a solution for dealing with the decentralized nature, a complicated project can be fragmented into small pieces and scheduled to different organizations or teams for its execution. Such fragmentation algorithms should be developed for complex and large-scale projects. Deadlock exists during the execution of the complex project with shared resources. In this way, advanced deadlock control methods should be introduced for a PRT-Net.

Acknowledgments

This work is partially supported by the National Natural Science Foundation of China (Grants Nos. 61472284, 61472004), the Shanghai Rising-Star Program (No. 14QA1400100), and Natural Science Foundation Programs of Shanghai under Grant No. 13ZR1443100.

References

[1] J.H. Payne, "Management of multiple simultaneous projects: a state-of-the-art review," International journal of project management, vol.13, no.3, pp.163–168, 1995.

[2] J.H. Patterson, "Alternate methods of project scheduling with limited resources," Naval Research Logistics Quarterly, vol.20, no.4, pp.767–784, 1973.

[3] J.H. Patterson and G.W. Roth, "Scheduling a project under multiple resource constraints: a zero-one programming approach," AIIE transactions, vol.8, no.4, pp.449–455, 1976.

[4] N. Christofides, R. Alvarez-Valdés, and J.M. Tamarit, "Project scheduling with resource constraints: A branch and bound approach," European Journal of Operational Research, vol.29, no.3, pp.262–273, 1987.

[5] P.R. Thomas and S. Salhi, "A tabu search approach for the resource constrained project scheduling problem," J. Heuristics, vol.4, no.2, pp.123–139, 1998.

[6] K. Nonobe and T. Ibaraki, "Formulation and tabu search algorithm for the resource constrained project scheduling problem," in Essays and surveys in metaheuristics, vol.15, pp.557–588, Springer, 2002.

[7] J.H. Patterson, "A comparison of exact approaches for solving the multiple constrained resource, project scheduling problem," Management science, vol.30, no.7, pp.854–867, 1984.

[8] M. Cervantes, A. Lova, P. Tormos, and F. Barber, "A dynamic population steady-state genetic algorithm for the resource-constrained project scheduling problem," in New Frontiers in Applied Artificial Intelligence, pp.611–620, Springer, 2008.

[9] J.R. Montoya-Torres, E. Gutierrez-Franco, and C. Pirachicán-Mayorga, "Project scheduling with limited resources using a genetic algorithm," International Journal of Project Management, vol.28, no.6, pp.619–628, 2010.

[10] L.-C. Liu and E. Horowitz, "A formal model for software project management," IEEE Trans. Softw. Eng., vol.15, no.10, pp.1280–1293, 1989.

[11] J. Kim, A.A. Desrochers, and A.C. Sanderson, "Task planning and project management using petri nets," Proc. IEEE International Symposium on Assembly and Task Planning, pp.265–271, IEEE, 1995.

[12] A. Kumar and L.S. Ganesh, "Use of petri nets for resource allocation in projects," IEEE Trans. Eng. Manag., vol.45, no.1, pp.49–56, 1998.

[13] Y.-L. Chen, P.-Y. Hsu, and Y.-B. Chang, "A petri net approach to support resource assignment in project management," IEEE Trans. Syst., Man, Cybern. A, vol.38, no.3, pp.564–574, 2008.

[14] K. Salimifard, G. Jamali, and S. Behbahaninezhad, "Resolving resource conflicts in a ship repair project," International Journal of Modeling & Optimization, pp.618–621, 2012.

[15] S.M.T.F. Ghomi and B. Ashjari, "A simulation model for multi-project resource allocation," International Journal of Project Management, vol.20, no.2, pp.127–130, 2002.

[16] P. Ballesteros-Pérez, M.C. González-Cruz, and M. Fernández-Diego, "Human resource allocation management in multiple projects using sociometric techniques," International Journal of Project Management, vol.30, no.8, pp.901–913, 2012.

[17] L.C. e Silva and A.P.C.S. Costa, "Decision model for allocating human resources in information system projects," International Journal of Project Management, vol.31, no.1, pp.100–108, 2013.

[18] S.E. Elmaghraby, E.I. Baxter, and M.A. Vouk, "An approach to the modeling and analysis of software production processes," International Transactions in Operational Research, vol.2, no.1, pp.117–135, 1995.

[19] A. Schirmer, "Case-based reasoning and improved adaptive search for project scheduling," Naval Research Logistics (NRL), vol.47, no.3, pp.201–222, 2000.

[20] V. Valls, F. Ballestín, and S. Quintanilla, "Justification and rcpsp: A technique that pays," European Journal of Operational Research, vol.165, no.2, pp.375–386, 2005.

[21] V. Valls, S. Quintanilla, and F. Ballestín, "Resource-constrained project scheduling: a critical activity reordering heuristic," European Journal of Operational Research, vol.149, no.2, pp.282–301, 2003.

[22] T. Murata, "Petri nets: Properties, analysis and applications," Proceedings of the IEEE, vol.77, no.4, pp.541–580, 1989.

[23] C. Liu, Q. Zeng, J. Zou, F. Lu, and Q. Wu, "Invariant decomposition conditions for petri nets based on the index of transitions," Information Technology Journal, vol.11, no.7, pp.768–774, 2012.

[24] C. Liu, Q. Zeng, H. Duan, and F. Lu, "Petri net based behavior description of cross-organization workflow with synchronous interaction pattern," in Process-Aware Systems, vol.495, pp.1–10, Springer, 2015.

[25] C. Liu, Q. Zeng, H. Duan, M. Zhou, F. Lu, and J. Cheng, "E-net modeling and analysis of emergency response processes constrained by resources and uncertain durations," IEEE Trans. Syst. Man Cybern, Syst., vol.45, no.1, pp.84–96, 2015.

[26] Q.-T. Zeng, F.-M. Lu, C. Liu, and D.-C. Meng, "Modeling and analysis for crossorganizational emergency response systems using petri nets," Chin. J. Comput, vol.36, no.11, pp.2290–2302, 2013.

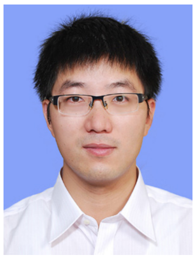
[27] Q. Zeng, F. Lu, C. Liu, H. Duan, and C. Zhou, "Modeling and verification for cross-department collaborative business processes using extended petri nets," IEEE Trans. Syst. Man Cybern, Syst., vol.45, no.2, pp.349–362, 2015.

[28] J. Cheng, C. Liu, M. Zhou, Q. Zeng, and A. Yla-Jaaski, "Automatic composition of semantic web services based on fuzzy predicate petri nets," IEEE Trans. Automat. Sci. Eng., vol.12, no.2, pp.680–689, 2015.

Cong Liu received his M.S. degree in computer science and technology from Shandong University of Science and Technology, Qingdao, China in 2015. He is currently a Ph.D. candidate within the section of Information Systems (IS) at the Department of Mathematics and Computer Science of Eindhoven University of Technology. His research interests are in the areas of Business Process Management, Process mining, Petri nets and Service composition.

Jiujun Cheng received his Ph.D. degree from Beijing University of Posts and Telecommunications in 2006. He is presently an associate professor of Tongji University, Shanghai, China. In 2009, he was a Visiting Professor at the Aalto University, Espoo, Finland. He has over 30 publications including conference and journal papers. His research interests span the area of mobile computing and social network with a focus on mobile/Internet interworking, service computing, and Internet of Vehicles.

Yirui Wang received the B.S. degree form Donghua University, Shanghai, China in 2014. He is currently working toward the M.S. degree with the College of Information Sciences and Technology, Donghua University, Shanghai, China. His research interests are computational intelligence, swarm intelligent algorithms, and combinatorial optimizations.



Shangce Gao received his Ph.D. degree in innovative life science from University of Toyama, Toyama, Japan in 2011. He is currently an Associate Professor with the Faculty of Engineering, University of Toyama, Japan. From 2011 to 2012, he was an associate research fellow with the Key Laboratory of Embedded System and Service Computing, Ministry of Education, Tongji University, Shanghai, China. From 2012 to 2014, he was an associate professor with the College of Information Sciences and Technology, Donghua University, Shanghai, China. His main research interests include mobile computing, nature-inspired technologies, and neural networks for optimization and real-world applications. He was a recipient of the Shanghai Rising-Star Scientist award, the Chen-Guang Scholar of Shanghai award, the Outstanding Academic Performance Award of IEICE, and the Outstanding Academic Achievement Award of IPSJ.