A Genetic Algorithm for the Proactive Resource-Constrained Project Scheduling Problem With Activity Splitting

Zhiqiang Ma, Zhengwen He, Nengmin Wang, Zhen Yang, and Erik Demeulemeester

***Abstract*—Proactive scheduling aims at the generation of robust baseline schedules, which has been studied for many years with the assumption that activity splitting is not allowed. In this paper, we focus on the proactive resource-constrained project scheduling problem in which each activity can be split at discrete time instants under the constraints of a maximum number of splitting and a minimum period of continuous execution. In this problem, setup times are also considered. A mathematical model is established and analyzed, of which two properties and one lemma are proposed. As the problem is proved to be NP-hard in the strong sense, for solving the model, we develop a genetic algorithm (GA) in which the two proposed properties and the lemma are applied as local search operators. After linearizing the proposed model, we use a commercial mathematical programming solver as a benchmark to solve the problem. From the computational results, we find that the developed GA is effective and efficient in solving the defined problem, and activity splitting improves robustness. With the growth of the maximum number of splitting, the decline in the minimum execution time, the decrease in the setup times, and the extension of the project due date, robustness increases.**

***Index Terms*—Activity splitting, genetic algorithm (GA), proactive project scheduling, setup time, solution robustness.**

I. Introduction

IT IS a well-known fact that project activities are subject to considerable uncertainties, such as accidents, resource breakdowns, and bad weather conditions, which may lead to numerous schedule disruptions during project execution and therefore incur some costs when project managers adjust the starting times of the activities to deal with them. Accordingly, proactive scheduling has been the subject of many research efforts that aim to generate robust baseline schedules that are protected against schedule disruptions. The more robust the baseline schedules are, the lower the adjustment costs will be during project execution. These research efforts have led to many models and algorithms, which are summarized in [1]–[4].

Two robustness approaches are considered in this field, i.e., quality robustness and solution robustness [5]. For quality robustness, the robust multimode discrete time/cost tradeoff problem is introduced and solved by exact and heuristic algorithms [6], [7]. Regarding solution robustness, various approaches are developed to cope with multiple disruptions, including activity duration disruptions [8], stochastic activity durations [9], [10], and stochastic resource availabilities [11], [12]. In contrast to the literature that addresses quality robustness or solution robustness separately, several studies have concentrated on the potential tradeoff between these two types of robustness. Al-Fawzan and Haouari develop a bi-objective model with an aggregation function in the absence of available information regarding the nature or size of the uncertain events [13]. With the composite objective of maximizing both schedule stability and timely project completion probability, Van de Vonder *et al*. develop a heuristic algorithm for minimizing a stability cost function [14] and they discuss the results obtained by a large experimental design that is established to evaluate several predictive-reactive resource-constrained project scheduling procedures [15]. Furthermore, Chtourou and Haouari present a two-stage algorithm in which the first stage is designed to minimize the project makespan, while the second one aims to maximize schedule robustness [16]. Deblaere *et al*. propose an objective to minimize a cost function that consists of the weighted expected activity starting time deviations and the penalties or bonuses that are associated with late or early project completion [17]. One recent study defines a new robustness measure that is completely independent of the applied reactive policy and then introduces a branch-and-cut algorithm to solve a sample average approximation of the original problem [18]. More details about the literature on proactive scheduling can be found in Appendix A.

It is noteworthy that most of the literature on proactive scheduling does not consider activity splitting, which means that these activities are not divisible. However, if activity splitting is allowed, it may be more flexible for project managers to

Manuscript received August 11, 2016; revised April 14, 2017, July 24, 2017, and December 19, 2017; accepted March 11, 2018. This work was supported in part by the China Scholarship Council under Grant 201606280246, and in part by the National Natural Science Foundation of China under Grant 71371150, Grant 71572138, Grant 71732006, Grant 71390331, Grant 71572148, and Grant 71742005. Review of this manuscript was arranged by Department Editor B. Jiang. *(Corresponding author: Nengmin Wang.)*

Z. Ma, Z. He, N. Wang, and Z. Yang are with the School of Management, Xi'an Jiaotong University, Xi'an 710049, China, and also with the Engineering Research Center (ERC) for Process Mining of Manufacturing Services in Shaanxi Province, Xi'an 710049, China (e-mail: tsyj2013@stu.xjtu.edu.cn; zhengwenhe@mail.xjtu.edu.cn; wangnm@mail.xjtu.edu.cn; zhen.yang@mail.xjtu.edu.cn).

E. Demeulemeester is with the Research Center for Operations Management, Department of Decision Sciences and Information Management, Faculty of Economics and Business, KU Leuven, Leuven 3000, Belgium (e-mail: erik.demeulemeester@kuleuven.be).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TEM.2018.2819689

schedule activities and make good use of resources and slacks to generate much more robust baseline schedules. In other words, if activity splitting contributes to higher robustness, then activities will be split into certain parts for execution; otherwise, activities will be scheduled without splitting. Note that we consider to split activities actively during the stage of baseline schedule generation, which is different from interrupting activities passively to deal with disruptions during the stage of project execution.

In previous literature, some researchers have already considered the project scheduling problem with activity splitting. The difference from what we discussed before is that their objective function is mainly focused on makespan minimization and the term they use of activity splitting is activity preemption or activity interruption. For example, Demeulemeester and Herroelen describe a branch-and-bound procedure to solve the preemptive resource-constrained project scheduling problem (PRCPSP) with the objective of minimizing project duration [19]. Following the work that reveals the potential benefits of allowing one interruption in the scheduling of activities in a resource-constrained project [20], Buddhakulsomsiri and Kim present a priority rule-based heuristic for the multimode scheduling problem with the splitting of activities around unavailable resources allowed [21]. Based on an analysis of the characterizations of the solution set for the preemptive and nonpreemptive RCPSP, Damay *et al*. present a linear programming based algorithm to solve the two problems [22]. Ballestín *et al*. mainly focus on problem 1_PRCPSP in which a maximum of one interruption per activity is allowed, and they propose a new model that covers most practical applications of discrete activity preemption [23], [24]. A genetic algorithm (GA) for the nonpreemptive multimode scheduling problem is developed and extended to the preemptive case of this problem [25]. Recently, Haouari *et al*. use a linear programming model that is based on the PRCPSP to compute a lower bound for the RCPSP [26], and Moukrim *et al*. propose an effective branch-and-price algorithm based on minimal interval order enumeration that involves column generation as well as constraint propagation [27]. For more research efforts on project scheduling problems with activity splitting, refer to [28]–[30]. In Appendix B, more details can be found about the literature on the RCPSP with activity splitting.

In practice, activities may be different with respect to activity splitting. First, activities may need to be executed continuously for certain periods before the next splitting and the duration of the continuous execution time is different. Second, some activities, such as chemical reactions, may not be split at all due to technical reasons, but some activities, such as the transportation of materials, are technically feasible to be split into certain parts. Even though the activities are all feasible for splitting, the maximum number of splitting allowed may be still different. Third, some activities, such as managerial operations, do not need setup times before execution while some activities, such as a bridge construction, may need certain periods for preparation. From the previous literature, we know that the first two differences have been considered and measured by two factors, i.e., the maximum number of activity splitting and the minimum continuous execution [24], [30], but the third one has not been considered in the scheduling research with activity splitting.

Typically speaking, activities that are split into certain parts cause additional setup times (and thus additional costs) when returning to their execution. In other words, if one activity technically needs a setup time before execution, then there will be additional setup times for the second and the subsequent parts of this activity. This implies that there will be a tradeoff between the benefits of activity splitting and the drawbacks of the increasing setup times under the objective of solution robustness maximization.

Based on the facts above, this paper presents a proactive RCPSP with activity splitting. In this problem, each activity can be split at discrete time instants under the constraints of a maximum number of splitting and a minimum period of continuous execution. Besides, additional setup times are considered when the activities return to execution from splitting. Different from the existing proactive scheduling that aims to improve schedule robustness without activity splitting, this paper aims to take activity splitting into account to seek opportunities to further improve schedule robustness. Therefore, it can be regarded as a two-stage problem: the first one is to decide how to split activities and the second stage is proactive scheduling, i.e., how to schedule activities to construct an optimal baseline schedule with the objective of solution robustness maximization. The solution robustness is obtained by inserting time buffers into the baseline schedule with the consideration of precedence, renewable resources, and project deadline constraints, and it is measured by a free slack based function, an adjusted surrogate solution robustness measure that is proposed by Lambrechts *et al*. [31]. This problem can be defined as an extension of the proactive RCPSP because activity splitting becomes allowed. As activities are handled in different ways in terms of activity splitting, this problem is also a generalization of m_PRCPSP, where all activities can be split m times. We believe that the proposed problem, which to the best of our knowledge has not thus far been investigated, may be more practical because it takes activity splitting into account and considers multiple cases of divisible activities.

Note that in previous literature on proactive scheduling activities are indivisible and treated as the basic project units. However, based on the theory of the work breakdown structure, activities are broken down by different levels. Therefore, in this paper activities are much more similar to work packages, which are not divided to the lowest level so that project managers can have the freedom to decide whether to further split the activities. Conversely, if activities have already been divided to the lowest level, we can regard them as subactivities, and then we can decide how to merge and schedule them to decrease the setup times and improve schedule robustness, which is just equivalent to scheduling activities without activity splitting in this paper.

The rest of this paper is organized as follows. In Section II, we present the notations and the problem formulation. Section III is devoted to the development of a GA that is based on the analysis of the proposed scheduling model. Section IV conducts an extensive computational experiment. Finally, in Section V, general conclusions and directions for further research are presented.

II. Problem Formulation

A. Optimization Model

Consider a project represented in an activity-on-the-node (AoN) format by means of a digraph $G = (N, A)$, where the set of nodes N represents the activities and the set of arcs A the finish-start, zero-lag precedence relations. The activities are numbered from the dummy start activity 1 to the dummy end activity n, and each activity i has a duration d_i and requires renewable resources to ensure that it is carried out. The project deadline is denoted as D. There are K different renewable resource types with an availability in each period $[t, t+1)$, $(t = 0, 1, \ldots, D)$, of R_k^ρ units, $k = 1, 2, \ldots, K$. Each activity i requires $r_{i,k}^\rho$ units of resource type k during each period in which it is processed. Dummy activities have zero duration and resource usage. We use subactivity (i, v) to denote the vth part of activity i, which has the same resource usage as activity i. The only difference between the activity and its subactivities is the duration.

For practical reasons that activities are different with respect to activity splitting, we make the following three assumptions. First, for each activity i, a required minimum execution time ε_i is predefined during which the activity must be in progress without any splitting. This forces the duration $\text{dur}_{i,v}$ of subactivity (i, v) to be at least ε_i. Second, each activity i can be split a maximum of $\eta_i (\eta_i < [\frac{d_i}{\varepsilon_i}])$ times at any discrete time instant, which results in V_i $(V_i \leq \eta_i + 1)$ precedence-connected subactivities, each of which has a resource requirement $r_{i,k}^\rho$. The first two assumptions are responses to the fact that activities cannot be split too frequently. Obviously, the case $\eta_i = 0$ or $d_i < 2\varepsilon_i$ means that activity i must be processed without splitting. In addition, as a response to the fact that in projects activities may need setup times for preparation, we assume each activity technically needs setup time θ_i before execution. Note that the setup time is not included in the activity duration, which means the actual duration of one indivisible activity is its duration plus its setup time, and there will be additional setup times for activities that are split into certain parts. Obviously, the case $\theta_i = 0$ means that activity i technically does not need setup time.

The weight w_i, which is allocated to each activity i, denotes the marginal cost of deviating the completion time of activity i during project execution from its planned completion time in the baseline schedule. The cost can be regarded as the impact of such a delay on all its immediate and transitive successors. Because the successors of the subactivities are the same as those of their original activity, we assume that the weights of the subactivities are equivalent to those of their original activities. The free slack $\text{FS}_{i,v}$, which represents the time buffers after the duration of subactivity (i, v), is defined as the total amount of time this subactivity can be delayed without causing any precedence or resource constraint violations. Note that the free slack here is defined in the context of limited resources, which is an extension of the one in the framework of critical path method (CPM). Referring to Lambrechts *et al*. [31], the utility of the free slacks may decrease marginally in exponent with the increase of their amounts. For example, if one activity has a free slack of 6, then the first slack will be much more beneficial than the sixth one to absorb the disruptions because it is less likely for the activity to delay six periods. Thus, the robustness that is generated by $\text{FS}_{i,v}$ can be calculated as $w_i \sum_{b=1}^{\text{FS}_{i,v}} e^{-b}$. Then, counting the utilities of all subactivities of all activities, the robustness of a schedule (hereafter denoted as Robu) can be defined as $\sum_{i=1}^{n} [w_i (\sum_{v=1}^{V_i} \sum_{b=1}^{\text{FS}_{i,v}} e^{-b})]$.

There are three groups of decision variables in this problem, i.e., V_i, $\text{dur}_{i,v}$, and $s_{i,v}$, which, respectively, represent the number of subactivities of activity i, the duration of subactivity (i, v), and the starting time of this subactivity. Then, the goal is to decide the optimal values for V_i, $\text{dur}_{i,v}$, and $s_{i,v}$ to obtain a baseline schedule with the maximum schedule robustness Robu. The optimization model for the proactive RCPSP with activity splitting is constructed as follows. It is important to note that in our model setup times are not included in d_i but are included in $\text{dur}_{i,v}$.

$$\text{Maximize Robu} = \sum_{i=1}^{n} \left[w_i \left(\sum_{v=1}^{V_i} \sum_{b=1}^{\text{FS}_{i,v}} e^{-b} \right) \right] \tag{1}$$

Subject to

$$s_{1,1} = 0 \tag{2}$$

$$s_{i,V_i} + \text{dur}_{i,V_i} \leq s_{j,1} \quad (i, j) \in A \tag{3}$$

$$s_{i,v} + \text{dur}_{i,v} \leq s_{i,v+1} \; i = 1, \ldots, n; \; v = 1, \ldots, V_i - 1 \tag{4}$$

$$s_{n,1} \leq D \tag{5}$$

$$\sum_{i \in S(t)} r_{i,k}^\rho \leq R_k^\rho \; k = 1, 2, \ldots, K; \; t = 0, 1, \ldots, D \tag{6}$$

$$\sum_{v=1}^{V_i} \text{dur}_{i,v} = d_i + V_i \times \theta_i \;\; i = 1, 2, \ldots, n \tag{7}$$

$$V_i \leq \eta_i + 1 \;\; i = 1, 2, \ldots, n \tag{8}$$

$$\text{dur}_{i,v} - \theta_i \geq \varepsilon_i \; i = 1, 2, \ldots, n; \; v = 1, 2, \ldots, V_i \tag{9}$$

$$V_i, \text{dur}_{i,v}, \text{ and } s_{i,v} \text{ are nonnegative integers } \forall i, \forall v \tag{10}$$

In the formulation, the objective function (1) is to maximize solution robustness. Equation (2) forces the project to start at time 0. The precedence constraints given by (3) indicate that the start of activity j must wait for the end of the last subactivity of all its preceding activities, and in constraints (4) one subactivity of an activity does not start before the end of the previous subactivity of the same activity. Constraint (5) imposes a deadline on the project. As $S(t)$ is the set of activities that are in progress during time interval $[t, t+1)$, constraints (6) force the total units of utilized resources to be no greater than the available resource capacity for every period. The conditions for activity splitting are reflected in (7)–(9). Equation (7) ensures that the duration of all the subactivities of activity i must be equal to the sum of the processing time of activity i and its total setup times. The constraints (8) guarantee that the times of splitting for a given divisible activity is no more than a predefined level called η_i, while in (9) for each subactivity the duration without setup

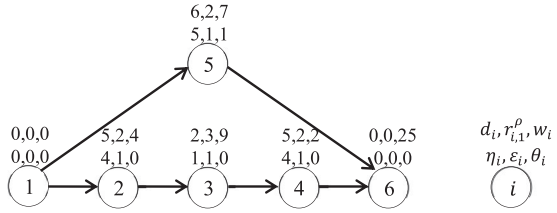


Fig. 1. Example.

time must be at least its minimum execution time. The range of values for V_i, $\text{dur}_{i,v}$, and $s_{i,v}$ are given in the constraints (10).

In this nonlinear model, we need to take constraints (7)–(9) into account to decide how to split activities and decide how to schedule those subactivities based on the constraints (2)–(6). In the first decision, there will be a tradeoff between the benefits of activity splitting and the drawbacks of the increasing setup times. In the second decision, there will be a tradeoff between inserting time buffers and the deadline constraint. For the objective function, $\text{FS}_{i,v}$ will be calculated by an algorithm that is developed in the next section. Note that $\text{FS}_{i,v}$ may not be equal to the values of time buffers. Time buffers are inserted based on the rule of marginally decreasing slack utility, activity weights, and the changes of the schedule after inserting time buffers, which together influence the improvement of the objective function value. The bigger the improvement, the bigger the possibility to insert time buffers to this activity.

B. Example

We use an example to illustrate the problem that is identified above. The AoN network of the example is depicted in Fig. 1, where activities 1 and 6 are the dummy start and end activities, respectively. The activities in the project require one renewable resource and their durations as well as resource requirements, activity weights, the maximum numbers of splitting, the minimum periods of continuous execution, and the setup times are labeled with the nodes. Other data of the project are as follows: $K = 1, R_1^{\rho} = 4, D = 14$. To demonstrate that activity splitting is beneficial to schedule robustness, we give the most robust baseline schedules without and with activity splitting, which are depicted as schedules (a) and (b), respectively, in Fig. 2, and compare the results obtained ahead.

1) Case Without Activity Splitting: In this case, we suppose that activities are indivisible during execution. Therefore, we have $\eta_i = 0$ for each activity i. Under this circumstance, schedule (a) is the optimal baseline schedule in terms of solution robustness where each activity has only one subactivity and the rounded rectangle represents the setup time of activity 5. Obviously, only activity 2 has a free slack of 2. The corresponding objective function value is equal to 2.00 and was calculated as shown in Table I.

2) Case With Activity Splitting: In this case, it is assumed that activity splitting is allowed. Based on the data shown in Fig. 1, schedule (b) is the most robust baseline schedule where activity 5 is split into two subactivities. Because of activity splitting, another setup time is needed before the execution of the second subactivity of activity 5. The corresponding objective function value is 10.90, the computation of which can be found in Table I as well.

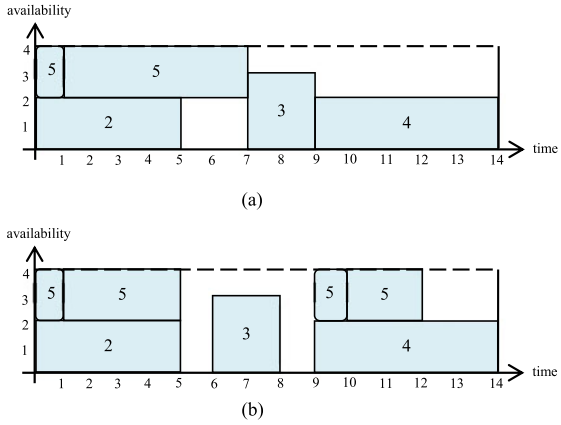


Fig. 2. Two feasible schedules for the project. (a) The most robust schedule without activity splitting. (b) The most robust schedule with activity splitting.

Comparing the results discussed above, we can find that an improvement of 445% is obtained for the free slack based objective function value, which verifies the potential benefits of making good use of activity splitting in proactive scheduling to some extent. The reason is that activity splitting enhances the flexibility of scheduling activities, which is beneficial to making good use of resources to shorten the project duration and thus spare more space to insert time buffers. Next, we will make an analysis about the cost saving when taking activity splitting into account. In this example, compared with schedule (a), schedule (b) is likely to have lower adjustment costs. For example, if the activity duration increases by 1 both for activity 3 and activity 5, we need to adjust the starting times of activities 3, 4, and 6 in schedule (a), but do nothing in schedule (b). Schedule (b) will have a much lower cost compared with (a).

III. Developed GA

In the first part, we propose two properties of the scheduling model and one lemma, which can be used for the development of the algorithm. In the second part, we first explain why we choose a GA to solve the problem and then present the framework of the developed algorithm. Afterwards, technical details are given to describe the developed GA in seven parts where the proposed properties and the lemma are used for the local search procedure.

A. Properties and the Lemma

To explain the properties and the lemma more clearly, we provide three definitions in advance.

Definition 1: In a given schedule, time period T is feasible for a minimum part of activity i, whose duration equals the minimum execution time of activity i, to be executed if the successors of activity i do not start before the end of time period T, the remaining resources in time period T can still satisfy the resource requirements of activity i, the length of time period T is no less than $\varepsilon_i + \theta_i$, and the starting time of the time period is after the completion time of the last subactivity of activity i.

TABLE I
CALCULATION OF THE OBJECTIVE FUNCTION

i	w_i	Schedule (a)			Schedule (b)		
		$FS_{i,1}$	$\sum_{b=1}^{FS_{i,1}} e^{-b}$	$w_i \sum_{b=1}^{FS_{i,1}} e^{-b}$	$FS_{i,v}$	$\sum_{v=1}^{V_i}\sum_{b=1}^{FS_{i,v}} e^{-b}$	$w_i \sum_{v=1}^{V_i}\sum_{b=1}^{FS_{i,v}} e^{-b}$
1	0	0	0.00	0.00	0	0.00	0.00
2	4	2	0.50	2.00	1	0.37	1.48
3	9	0	0.00	0.00	1	0.37	3.33
4	2	0	0.00	0.00	0	0.00	0.00
5	7	0	0.00	0.00	1+2	0.87	6.09
6	25	0	0.00	0.00	0	0.00	0.00
Robu				2.00			10.90

Definition 2: Subactivity (i, v) is divisible if $V_i \leq \eta_i$ and $\text{dur}_{i,v} - \theta_i \geq 2\varepsilon_i$.

Definition 3: Subactivity (i, v) is abundant in free slacks if $\text{FS}_{i,v} \geq 2 + \theta_i$.

Based on the above definitions, we develop two properties of the model, which are named as Pioneering and Balancing, respectively, based on the mechanism of each operation. After that, one lemma is proposed for improving the schedule robustness.

Property 1 (Pioneering): If subactivity (i, v) is divisible, and there is a feasible period T, whose length is denoted as ξ_T, for activity i to be executed, then schedule robustness can be improved in three steps: First, keep other activities unchanged. Second, divide this subactivity into two parts, which are denoted as (i, v_1) and (i, v_2), whose durations are $\text{dur}_{i,v} - \text{dd}$ $(\varepsilon_i \leq \text{dd} \leq \min\{\text{dur}_{i,v} - \varepsilon_i - \theta_i,\ \xi_T - \theta_i\})$ and $\text{dd} + \theta_i$, respectively. Third, schedule the two parts of this subactivity in the original and the new periods. In this way, the objective function value of the schedule can be improved.

Proof of Property 1: After the Pioneering operation, the free slack of subactivity (i, v_1) will be $\text{FS}_{i,v} + \text{dd}$, while the free slack of subactivity (i, v_2) will be $\xi_T - \text{dd} - \theta_i$. The utility of the free slacks before the operation is $U_1 = \sum_{b=1}^{\text{FS}_{i,v}} e^{-b}$, which is smaller than that after the operation $U_2 = \sum_{b=1}^{\text{FS}_{i,v}+\text{dd}} e^{-b} + \sum_{b=1}^{\xi_T - \text{dd} - \theta_i} e^{-b}$. Hence, Property 1 can be used as a rule to maximize schedule robustness.

Property 2 (Balancing): If subactivity (i, v) is divisible and has no more than one free slack, while the reverse is true for subactivity (i, p), then schedule robustness may be improved by transferring one unit of time from the duration of subactivity (i, v) to that of subactivity (i, p).

Proof of Property 2: From the prerequisites of Property 2, we can obtain the four following constraints: $\text{dur}_{i,v} - \theta_i \geq 2\varepsilon_i$, $0 \leq \text{FS}_{i,v} \leq 1, \varepsilon_i \leq \text{dur}_{i,p} - \theta_i < 2\varepsilon_i$, and $\text{FS}_{i,p} \geq 2$. After the Balancing operation, the free slack of subactivity (i, v) will be $\text{FS}_{i,v} + 1$, while the free slack of subactivity (i, p) will be $\text{FS}_{i,p} - 1$. Then, the utility of the free slacks after the operation can be calculated as

$$U_2 = \sum_{b=1}^{\text{FS}_{i,v}+1} e^{-b} + \sum_{b=1}^{\text{FS}_{i,p}-1} e^{-b} = \sum_{b=1}^{\text{FS}_{i,v}} e^{-b} + \sum_{b=1}^{\text{FS}_{i,p}} e^{-b} + (e^{-\text{FS}_{i,v}-1} - e^{-\text{FS}_{i,p}}).$$

Because of the two following constraints, i.e., $-\text{FS}_{i,v} \geq -1$ and $-\text{FS}_{i,p} \leq -2$, U_2 will be no less than the utility before the operation $U_1 = \sum_{b=1}^{\text{FS}_{i,v}} e^{-b} + \sum_{b=1}^{\text{FS}_{i,p}} e^{-b}$. Hence, Property 2 can be used to improve the objective function value.

To summarize, Pioneering facilitates the discovery of new periods for activities to be executed, and Balancing is used to balance the length of the durations between two subactivities of one activity. As the two properties can help to transform subactivities into divisible ones with abundant free slacks, they pave the way for the following lemma, which is used to divide one subactivity into subactivities that specifically share the buffer of the original subactivity as equally as possible such that the schedule robustness can be improved.

Lemma 1: For any subactivity (i, v) that is divisible and abundant in free slacks, we can first divide this subactivity into $\text{num}_{i,v} = \min([\frac{\text{dur}_{i,v} - \theta_i}{\varepsilon_i}], \frac{\text{FS}_{i,v} + \theta_i}{1 + \theta_i},\ \eta_i - V_i + 2)$ parts whose durations are no less than ε_i, and then schedule them continuously and make sure their free slacks are as equal as possible, i.e., the difference between the maximum and the minimum free slack value of the newly generated subactivities is no more than one. In this way, schedule robustness will be improved.

Proof of Lemma 1: As $\sum_{b=1}^{\infty} e^{-x} = \frac{1}{e-1} < 2e^{-1}$ is true, it would be always beneficial for improving schedule robustness by splitting a divisible subactivity with abundant free slacks into certain parts. To maximize robustness, $\text{FS}_{i,v} - (\text{num}_{i,v} - 1)\theta_i$ should be no less than $\text{num}_{i,v}$, so $\text{num}_{i,v} \leq \frac{\text{FS}_{i,v} + \theta_i}{1 + \theta_i}$. Furthermore, if the difference between the maximum and the minimum free slack value of the newly generated subactivities is more than one in the optimal improvement, for example, $\text{fs}_1 > \text{fs}_2 + 1$, then $\sum_{b=1}^{\text{fs}_1} e^{-b} + \sum_{b=1}^{\text{fs}_2} e^{-b} = (\sum_{b=1}^{\text{fs}_1 - 1} e^{-b} + e^{-\text{fs}_1}) + \sum_{b=1}^{\text{fs}_2} e^{-b} < \sum_{b=1}^{\text{fs}_1 - 1} e^{-b} + (\sum_{b=1}^{\text{fs}_2} e^{-b} + e^{-fs_2 - 1}) = \sum_{b=1}^{\text{fs}_1 - 1} e^{-b} + \sum_{b=1}^{\text{fs}_2 + 1} e^{-b}$. As there will be a contradiction, to obtain an optimal improvement based on the proposed lemma, the original subactivity should be divided into $\text{num}_{i,v}$ parts whose free slacks are as equal as possible.

Note that it is possible that none of the subactivities in a schedule is divisible and abundant in free slacks, and under this circumstance we cannot apply Lemma 1 to improve solution robustness of this schedule.

B. Developed GA

As shown in Appendix C, the proposed problem can be simplified into the RCPSP with the objective of makespan minimization. As the latter is known to be NP-hard in the strong sense [32], [33], the proposed proactive scheduling problem with activity splitting is NP-hard in the strong sense as well,

which makes the achievement of optimal solutions a computationally difficult proposition, especially for large projects. For this reason, we use a well-known metaheuristic, i.e., GA as introduced by Holland [34], to solve the problem. We choose the genetic search methodology for two reasons. First, this technique has been successfully applied to many project scheduling problems [24], [25], [29], [30], [35], [36], and second, it is easy to generate activity splitting at each iteration by using the crossover operator.

GAs work with a "population" of individuals. In our algorithm, we set the size of the population as μ, the individual of which can be initially generated by a procedure called individual generation procedure (IGP). At each iteration, which is denoted as iter, the best φ individuals of the population in terms of fitness (objective function value) are chosen to be included in the population of the next iteration, while $(\mu - \varphi)$ individuals of the population are selected following the roulette wheel sampling method to generate children with the aid of a crossover operator called crossover procedure (CRP). Then, a mutation operator called mutation procedure (MTP) is used to apply a certain change to the generated children. Each child will be decoded into a solution using the decoding procedure (DCP). If it is feasible, the solution will be buffered with the buffering procedure (BFP) and improved with a local search procedure that includes three operators called LSP_1, LSP_2, and LSP_3, respectively. As far as the termination criterion of the developed GA is concerned, we define δ as the required number of iterations and stop the algorithm once δ is reached. It is noteworthy that we work with the notion of life span to solve the problem of superindividuals. Superindividuals far exceed, in fitness, other solutions of the population, and their existence might result in premature convergence to a local optimum. We set the life span of an individual at "birth" at 0. At each iteration, the life span of each surviving individual is increased by 1. When the life span reaches a certain number, maxlife, the individual dies and is replaced by a newly generated individual with the aid of the procedure IGP.

1) Solution Representation: Referring to [24], [31], we use three lists below to codify the solutions, the length of which is denoted as nsub.

1) Subactivity list (L): This list is the sequence of subactivities. The jth element in L represents the subactivity $L_j = (i, v)_j$.
2) Duration list (DL): This list stores the duration $\text{dur}_{(i,v)_j}$ of the corresponding subactivity $(i, v)_j$ in L.
3) Buffer list (BL): This list indicates which subactivities should be buffered and by how much their finish times can be delayed beyond their earliest finish times as dictated by the serial schedule generation scheme (SSGS). For convenience, let $\text{buf}_{(i,v)_j}$ denote the buffer of the corresponding subactivity $(i, v)_j$ in L. Note that $\text{buf}_{(i,v)_j}$ represents the inserted buffer, which is different from $\text{FS}_{(i,v)_j}$.

Given the above lists, a solution can be obtained using a decoding approach, which is an extension of SSGS and is described in Algorithm 1 in Appendix D.

2) Objective Function: For a solution that is represented by the combination of the three lists, the key to calculating its objective function value is to compute the free slack $\text{FS}_{i,v}$ of each subactivity. Once they are obtained, the objective function value can be easily computed based on the formula (1). We develop a procedure called free slack calculation procedure (FSP) to compute the free slack of every subactivity, which is an extension of the procedure developed by Lambrechts *et al*. [31] and is indicated in Algorithm 2 in Appendix D.

As the decoded schedule may cause a project deadline violation, we transform the deadline constraint into a soft constraint that is based on a deadline feasibility test (DFT) function, which is defined as $\text{DFT} = \max\{0, s_{n,1} - D\}$. During the searching process, if the DFT of a solution is greater than 0, the objective function value of the solution will be penalized based on the following formula:

$$\text{Robu} = \sum_{i=1}^{n} \left[w_i \left(\sum_{v=1}^{V_i} \sum_{b=1}^{\text{FS}_{i,v}} e^{-b} \right) \right] - \text{np} \times \text{nc} \times \text{DFT}.$$

Here, np is the penalty factor, and nc denotes the number of iterations that are used by the GA since the last major improvement was found.

3) Buffering: For a feasible solution, we use a procedure called BFP, which is described in Algorithm 3 in Appendix D, to insert enough buffers into the schedule to improve its robustness, which serves as a local search of the BL. We first select a subactivity randomly and add one unit of time buffer to that. Then, we calculate the objective function value Robu$'$ of the improved solution. If the deadline constraint is violated or the objective function value has not been improved, the number of failure times ζ that is initialized at zero will increase by one. If ζ reaches a predefined maximum allowed number Z of failures, the procedure ends. Otherwise, another subactivity is chosen and the procedure continues.

4) Initial Population Generation: The individual g of the initial population can be generated through the procedure IGP, which is described in Algorithm 4 in Appendix D. To decide whether to split the activities in the initial solution, we take the constraints of the maximum number of splitting and the minimum execution time into consideration. If the two constraints are satisfied, then we generate random numbers and compare them with a predefined parameter itrpt to make the decision of activity splitting. Note that this procedure builds an individual in nsub iterations, where nsub is unknown until the end of the procedure.

5) Crossover: Children can be generated by operating on the selected individuals with the aid of a crossover procedure called CRP. This procedure is described in Algorithm 5 in Appendix D, which is similar to the one that is developed by Ballestín *et al*. [24] except that we now have a third list called BL. In our procedure, we copy the same proportion of time buffer of the parent to the child as that of the duration. Note that the selected individuals are randomly paired as parents, and each of them can be a father or a mother.

6) Mutation: We make a change on the children with the procedure MTP, which is described in Algorithm 6 in Appendix D. We must emphasize that it is a deliberate choice that the mutation operation only considers the operators of changing

Algorithm 1: Pioneering: Robu′= LSP_1 $(L,\ \mathrm{DL}, \mathrm{BL})$.

$\mathrm{FS}_{(i,v)'_j} = \mathrm{FSP}\,(L, \mathrm{DL}, \mathrm{BL})$, obtain $V_i\ (i \in N)$
FOR $i = 2$ TO $n-1$ DO
  IF $V_i \le \eta_i$ AND $d_i \ge 2\varepsilon_i$ THEN
    Obtain the sets C_1 and C_2
    WHILE $C_1 \ne \emptyset$ AND $C_2 \ne \emptyset$ DO
      Choose the period T_1 from C_1 and one subactivity (i,v) from C_2
      Generate dd from $[\varepsilon_i, \min\{\mathrm{dur}_{i,v} - \varepsilon_i - \theta_i, \xi_T - \theta_i\}]$
      $\mathrm{dur}_{i,v} = \mathrm{dur}_{i,v} - \mathrm{dd}, \mathrm{dur}_{i,V_i+1} = \mathrm{dd} + \theta_i$, $\mathrm{FS}_{i,v} = \mathrm{FS}_{i,v} + \mathrm{dd}, \mathrm{FS}_{i,V_i+1} = \xi_{T_1} - \mathrm{dd} - \theta_i$
      Update the sets C_1 and C_2, $V_i = V_i + 1$
    END WHILE
  END IF
END FOR
Calculate the objective function value Robu′ of the improved solution

Algorithm 2: Balancing: Robu′= LSP_2 $(L,\ \mathrm{DL}, \mathrm{BL})$.

$\mathrm{FS}_{(i,v)'_j} = \mathrm{FSP}\,(L, \mathrm{DL}, \mathrm{BL})$
FOR $i = 2$ TO $n-1$ DO
  IF $\eta_i > 0$ AND $d_i \ge 2\varepsilon_i$ THEN
    Obtain the sets C_3 and C_4, $\mathrm{NS} = \min\{|C_3|, |C_4|\}$
    FOR $q = 1$ TO NS DO
      Choose one subactivity (i,v) from C_3 and another one (i,p) from C_4
      $\mathrm{dur}_{i,v} = \mathrm{dur}_{i,v} - 1$, $\mathrm{dur}_{i,p} = \mathrm{dur}_{i,p} + 1$
      $\mathrm{FS}_{i,v} = \mathrm{FS}_{i,v} + 1$, $\mathrm{FS}_{i,p} = \mathrm{FS}_{i,p} - 1$
      Update C_3 and C_4
    END FOR
  END IF
END FOR
Calculate the objective function value Robu′ of the improved solution

Algorithm 3: Robu′ = LSP_3 $(L,\ \mathrm{DL}, \mathrm{BL})$.

$\mathrm{FS}_{(i,v)'_j} = \mathrm{FSP}\,(L, \mathrm{DL}, \mathrm{BL})$, obtain $V_i\ (i \in N)$
FOR $i = 2$ TO $n-1$ DO
  IF $V_i \le \eta_i$ AND $d_i \ge 2\varepsilon_i$ THEN
    Obtain the set C_5
    WHILE $C_5 \ne \emptyset$ DO
      Choose one subactivity (i,v) from C_5, $\mathrm{num}_{i,v} = \min\{\lfloor\frac{\mathrm{dur}_{i,v}-\theta_i}{\varepsilon_i}\rfloor, \frac{\mathrm{FS}_{i,v}+\theta_i}{1+\theta_i},\ \eta_i - V_i + 2\}$
      Divide the subactivity (i,v) into $\mathrm{num}_{i,v}$ parts whose free slacks are as equal as possible and durations are $[\mathrm{dur}_{i,v} - (\mathrm{num}_{i,v}-1)\cdot\varepsilon_i]$, $\varepsilon_i + \theta_i, \ldots, \varepsilon_i + \theta_i$, respectively
      Update V_i and the set C_5
    END WHILE
  END IF
END FOR
Calculate the objective function value Robu′ of the improved solution

the sequences and time buffers of the subactivities and does not introduce more operators. We considered many operators, such as introducing more activity splitting, merging some subactivities, and changing the duration of subactivities. However, the preliminary tests with such operators did not lead to improved results. A reason could be that the local search procedure that is developed in the next section plays the same roles as those of these operators. For example, the procedures LSP_1 and LSP_3 can be regarded as operations that introduce more splitting of activities, and the procedure LSP_2 is structured to change the duration of the subactivities.

7) Local Search: For each feasible child, we adopt a local search procedure that includes three operators called LSP_1, LSP_2, and LSP_3, respectively, to improve its schedule robustness. The operator LSP_1, which is based on Property 1 and described in Algorithm 1, facilitates the discovery of new periods for activities to be executed. Let $C_1 = \{T_1, T_2, \ldots, T_c\}$ denote the set of feasible periods, as defined in Definition 1, and let $C_2 = \{(i,v)|\mathrm{dur}_{i,v} - \theta_i \ge 2\varepsilon_i \text{ and } V_i \le \eta_i\}$ represent the set of divisible subactivities, as defined in Definition 2.

The operator LSP_2, which is based on Property 2 and described in Algorithm 2, is used to balance the length of durations between two subactivities of one activity. Let $C_3 = \{(i,v)|\mathrm{dur}_{i,v} - \theta_i \ge 2\varepsilon_i \text{ and } \mathrm{FS}_{i,v} \le 1\}$ denote the set of subactivities that are divisible but has no more than one free slack and let $C_4 = \{(i,v)|\mathrm{dur}_{i,v} - \theta_i < 2\varepsilon_i \text{ and } \mathrm{FS}_{i,v} \ge 2\}$ represent the ones that are just the reverse.

The operator LSP_3, which is based on Lemma 1 and described in Algorithm 3, is used to divide one subactivity into subactivities that specifically share the buffer of the original subactivity as equally as possible. For the sake of description, let $C_5 = \{(i,v)|\mathrm{dur}_{i,v} - \theta_i \ge 2\varepsilon_i \text{ and } \mathrm{FS}_{i,v} \ge 2 + \theta_i\}$ denote the set of subactivities that are divisible and abundant in free slacks.

IV. Computational Results

A. Experimental Design

Based on the three developed local search operators, four different versions of the GA are presented. For the sake of description, we represent the GA without any local search operator as GA, the GA with the operator LSP_1 as GA-LSP1, the GA with operators LSP_1 and LSP_2 as GA-LSP12, and the GA with all the three operators as GA-LSP123, respectively, in the remainder of the paper. To evaluate the effectiveness of the proposed GAs, we propose the use of CPLEX as a benchmark to optimally solve the established model. Referring to the methods that are proposed to reduce zero-one polynomial formulations to zero-one linear formulations [37], the proposed nonlinear model can be linearized, just as shown in Appendix E. As many variables and constraints are introduced into the model, it may take much time to solve the problem. However, there is no loss of the quality of the solutions for the problem and therefore it

TABLE II
PARAMETER SETTINGS THAT ARE USED TO GENERATE THE DATASET

Parameter	Setting
Number of non-dummy activities	6, 8, 10, 30, 60
Network complexity, NC	1.2, 1.5, 1.8 for the sets J6, J8, and J10 1.5, 1.8, 2.1 for the sets J30 and J60
Resource factor, RF	0.25, 0.50, 0.75, 1.00
Resource strength, RS	0.2, 0.5, 0.7, 1.0
Number of instances for each combination of parameters under a given number of non-dummy activities	2
Number of initial or terminal activities	3
Maximal number of successors or predecessors	3
Activity duration, d_i	Randomly selected from interval [1, 10]
Number of resource types, K	4
Resource amounts required by activities, $r_{i,k}^{\rho}$	Randomly selected from interval [1, 10]
Weights of non-dummy activities, w_i	Randomly selected from interval [1, 10]

TABLE III
LEVELS OF THE KEY PARAMETERS

Parameter	Level	Value	
Project due date factor α	1-3	20%, 30%, 40%	
Setup times of non-dummy activities θ_i	1	$\theta_i = 0.8 * c * d_i$	
	2	$\theta_i = 1.0 * c * d_i$	
	3	$\theta_i = 1.2 * c * d_i$	
Combination of η_i and ε_i	1	$\eta_i = \min\{d_i - 1, a\}$	$\varepsilon_i = 1$
	2	$\eta_i = \min\{d_i - 1, 1.5a\}$	$\varepsilon_i = 1$
	3	$\eta_i = \min\{d_i - 1, 2a\}$	$\varepsilon_i = 1$
	4	$\eta_i = [d_i/\varepsilon_i] - 1$	$\varepsilon_i = \max\{1, b\}$
	5	$\eta_i = [d_i/\varepsilon_i] - 1$	$\varepsilon_i = \max\{1, 0.7b\}$
	6	$\eta_i = [d_i/\varepsilon_i] - 1$	$\varepsilon_i = \max\{1, 0.4b\}$
	7	$\eta_i = d_i - 1$	$\varepsilon_i = 1$

is enough for the sake of comparison of effectiveness. Note that we can use CPLEX to directly represent the algorithm that is conducted by the software. The aim of our experiment is not only to test the effectiveness of the three local search operators by comparing the performance of different versions of the GA, but also to validate the performance of the GA developed in this paper against CPLEX. Besides, it is expected to draw conclusions based on an analysis of the results.

The five algorithms are tested on five instance sets that are constructed by the ProGen project generator [38], [39], which is classified by three parameters, i.e., network complexity (NC), resource factor (RF), and resource strength (RS). Specifically, the instances with 6 or 8 or 10 nondummy activities, denoted as J6, J8, and J10, are generated by ourselves using the ProGen generator, while the instances with 30 or 60 nondummy activities, denoted as J30 and J60, are randomly (the first and the sixth instances out of the ten provided instances) chosen from the Project Scheduling Problem Library (PSPLIB), which is also generated by the ProGen generator [39]. The five sets consist of $48 \times 2 \times 5 = 480$ instances, and the parameter setting that is used to generate instances is described in Table II. Note that in consideration of the feasibility of the instance generation, we choose a different level setting of NC for J6, J8, and J10 from the one for J30 and J60. The generation of activity weight w_i is also shown in Table II.

In our experiment, the project due date D of each instance is set at $C_{\max}^{\mathrm{RCPSP}}(1+\alpha)$, where $C_{\max}^{\mathrm{RCPSP}}$ represents the minimum makespan that is optimally solved by CPLEX under a deterministic, indivisible, and nonsetup-time environment, and the due date factor α is a parameter that is chosen by the project manager and constitutes the tradeoff between project stability and project duration [14]. The value of the four key parameters, i.e., α, θ_i, η_i, and ε_i, is set at certain levels, as shown in Table III, where parameter c denotes a decimal that is randomly selected from [1/10, 1/8] and parameters a and b, respectively, denote random numbers that are selected from [0, $d_i - 1$] and [1, d_i]. Consequently, a full factorial experiment of the four parameters results in $3 \times 3 \times 7 = 63$ replicates for each instance and $480 \times 63 = 30\,240$ ones overall.

TABLE IV
VALUES OF PARAMETERS FOR INSTANCE SETS

Set	μ	φ	Z	$maxlife$	np	$itrpt$	$pmut$	δ
J6, J8, J10	64	15	2	7	25	0.4	5%	80
J30	64	11	2	7	30	0.4	5%	350
J60	64	15	2	9	50	0.4	4%	450

The following ten indices are defined to evaluate the performance of the algorithms. Specifically, the first seven indices are used to compare the performance of the four different versions of the GA, and the last three indices are additionally designed to make a comparison of the performance between the GA and CPLEX.

1) AOV: Average objective function value.
2) APB: The percentage of instances for which the algorithm finds a solution that is equal to the best solution known, i.e., the best one among the solutions that are found by the four developed versions of the GA – GA, GA-LSP1, GA-LSP12, and GA-LSP123.

TABLE V
PERFORMANCE OF THE FOUR VERSIONS OF THE GA

Set	Version	AOV	APB (%)	API (%)	ARI (%)	ACT (s)	ACT' (s)	AOG' (%)
J6	GA	30.21	73.31	0.00	0.00	0.12	0.24	2.79
	GA-LSP1	30.23	74.11	0.23	15.94	0.12	0.24	2.76
	GA-LSP12	30.28	76.57	3.04	4.89	0.12	0.24	2.70
	GA-LSP123	30.49	96.36	10.07	9.40	0.12	0.20	1.89
J8	GA	40.00	44.71	0.00	0.00	0.16	0.39	2.98
	GA-LSP1	40.07	46.15	0.48	17.31	0.16	0.39	2.92
	GA-LSP12	40.21	50.15	3.80	5.44	0.17	0.38	2.88
	GA-LSP123	40.92	91.34	11.28	11.47	0.17	0.29	1.70
J10	GA	47.62	26.74	0.00	0.00	0.19	0.49	3.39
	GA-LSP1	47.77	28.04	1.02	9.41	0.20	0.49	3.01
	GA-LSP12	48.05	31.40	4.69	5.16	0.19	0.48	3.13
	GA-LSP123	49.38	87.12	12.68	8.45	0.20	0.34	2.00
J30	GA	158.83	3.90	0.00	0.00	3.12	9.44	6.88
	GA-LSP1	162.52	6.37	3.44	5.67	3.21	9.13	5.54
	GA-LSP12	162.83	6.66	8.55	3.60	3.13	9.10	5.69
	GA-LSP123	175.05	83.50	17.98	6.33	3.32	5.22	2.41
J60	GA	300.98	1.82	0.00	0.00	9.35	28.80	9.57
	GA-LSP1	318.10	6.30	10.85	4.23	9.73	26.32	6.40
	GA-LSP12	318.58	6.07	17.12	3.60	9.47	25.51	6.63
	GA-LSP123	349.83	85.81	30.16	7.01	10.04	13.69	4.24
Avg	GA	*115.53*	*30.10*	*0.00*	*0.00*	*2.59*	*7.87*	*5.12*
	GA-LSP1	*119.74*	*32.19*	*3.20*	*10.51*	*2.68*	*7.31*	*4.13*
	GA-LSP12	*119.99*	*34.17*	*7.44*	*4.54*	*2.62*	*7.14*	*4.21*
	GA-LSP123	*129.13*	*88.83*	*16.43*	*8.53*	*2.77*	*3.95*	*2.45*

3) API: The percentage of solutions that are improved after using the local search procedure.
4) ARI: Average rate of improvement in terms of the objective function value after using the local search procedure.
5) ACT: Average computing time.
6) ACT': Average computing time to solve the problems to obtain the best solutions known.
7) AOG': Average gap in terms of the objective function values of the worse solutions that are obtained by a specific version of the GA compared with those of the best solutions known.
8) AOG: Average gap in terms of the objective function values of the worse solutions that are obtained by the GA compared with those of the corresponding solutions that are obtained by CPLEX.
9) APN: The percentage of instances that cannot be solved by CPLEX within a predefined time limit.
10) AWS: The percentage of instances in which worse solutions are obtained by the developed GA than by CPLEX.

In our experiment, the developed algorithms are programmed in the C++ language, implemented in Microsoft Visual Studio 2013 and executed on a DELL OptiPlex 3040MT with 3.20 GHz clock-pulse and 8G RAM.

B. Parameter Selection

Our developed GA allows for different choices of eight parameters. With a focus on the value of AOV, we performed a preliminary experiment to choose the best combination of parameters. This experiment tests the instances whose project due date factor α is set at 30%, setup time is set at level 1, and the combination of η_i and ε_i is set at level 4. According to the results of the preliminary test, the parameters are set at different values to solve different instance sets, as shown in Table IV where parameter pmut represents the probability of mutation.

C. Performance of the Developed GA

1) Comparison of the Four Different Versions of the GA: The results of the performance of the four developed GAs on the five instance sets are presented in Table V, where the italic numbers in the four bottom rows represent the average values of the five instance sets. It is noteworthy that the five left indices are used to measure the performance of the GAs that stop after a predefined number of iterations while the two right ones are used to measure the performance of the GAs that stop once obtaining the best solutions known. From the table, we observe that for different instance sets the conclusion is almost the same in terms of the performance of the four versions of the GA. The indices AOV and APB of GA-LSP1 are higher than those of GA, which verifies a better performance of GA-LSP1 compared with GA. This is not surprising because the operator LSP_1 is added in GA-LSP1, which on average improves the objective function values of 3.20% of the solutions by 10.51%. Similarly, the effectiveness of the operator LSP_2 can be analyzed by comparing the versions GA-LSP12 and GA-LSP1. On average, GA-LSP12 performs better than GA-LSP1 in terms of AOV, APB, and ACT. Furthermore, we find that GA-LSP123, followed by GA-LSP12, GA-LSP1, and GA, performs the best with the highest AOV and the highest average percentage of the best solutions (APB). Corresponding with the highest value of APB, GA-LSP123 takes the least time to solve the problems again, reaching a smallest average gap of the objective function values compared with those of the best solutions known. Most of

TABLE VI
PERFORMANCE OF GA-LSP123 AND CPLEX

Set	ACT (s)		APN	AWS	AOG
	GA-LSP123	CPLEX	(%)	(%)	(%)
J6	0.12	1592.04	42.21	2.63	2.30
J8	0.17	1766.92	46.93	8.09	1.37
J10	0.20	2022.72	54.17	16.77	1.33
Avg	*0.16*	*1793.89*	*47.77*	*9.16*	*1.67*

the success is due to the application of the three operators which on average improve the objective function values of 16.43% of the solutions by 8.53%. Compared with the operators LSP_1 and LSP_2, LSP_3 is much more effective as there is a sharp increase of AOV and APB once it is included in the GA. In summary, the three developed local search operators improve the solution robustness of the baseline schedules, although it takes a somewhat longer computing time to solve the problems. Thus, GA-LSP123 is the most promising version for the problem among the four presented GAs, which can be used to compete with a commercial mathematical programming solver next.

2) Comparison of the Performance Between the GA and Commercial Software: To test the effectiveness of the algorithm that is developed in this paper, we conduct an experiment to compare the performance between GA-LSP123 and a commercial mathematical programming solver (CPLEX). In this experiment, we predefine a maximum period of 1 h for CPLEX to solve each instance. This means that even though one instance is not solved optimally by that time, we end the algorithm and save the outcome that has been obtained thus far, which includes the best solution, the objective function value, and the computing time. Because it is difficult for CPLEX to solve the problems with a lot of nondummy activities, we only choose to test the three instance sets J6, J8, and J10.

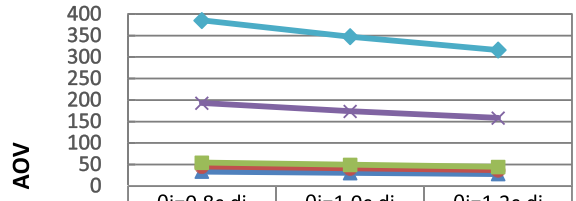
The results of the experiment can be found in Table VI. From the table, we can see that the number of instances that cannot be solved by CPLEX in the predefined period is very high and increases quickly with an increasing number of nondummy activities. Simply put, CPLEX requires a great deal of time to solve the problem. This is not surprising because many variables and constraints are introduced during the linearization process of the proposed scheduling model, which results in the difficulty of computing problems for CPLEX. By contrast, GA-LSP123 is much more efficient, with a very small computing time. Although GA-LSP123 cannot solve some instances as optimally as CPLEX, the percentage of these instances is very small, and it is acceptable of the average gap between the objective function values of the solutions for these instances that are solved by GA-LSP123 and the corresponding ones that are solved by CPLEX.

D. Sensitivity Analysis of the Key Parameters

First, we investigate the effect of different levels of the combination of η_i and ε_i on solution robustness for the five instance sets. In addition to the seven levels of the divisible case, we take level 8, which represents the indivisible case, into account. The results are described in Table VII, where for each instance set the italic numbers in the second row from the bottom represent the average values of the divisible case. From the table, two main phenomena can be observed. The first one is that the AOV under the divisible case is significantly higher than the corresponding values under the indivisible case. This indicates that activity splitting is beneficial for generating more robust baseline schedules that are likely to have lower adjustment costs during project execution. Compared with the classic proactive scheduling models where activity splitting is not allowed, this paper offers a new method to improve schedule robustness when activity splitting is allowed and generates better solutions to project management. This phenomenon can be explained as follows. When activities can be split, it will be more flexible for project managers to schedule activities at the design stage of

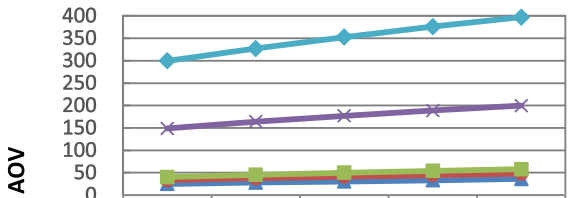
TABLE VII
EFFECT OF DIFFERENT LEVELS OF THE COMBINATION OF η_i AND ε_i

Set	Case	Level	η_i	ε_i	AOV	ACT
J6	Divisible	1	2.38	1.00	28.91	0.12
		2	2.92	1.00	29.86	0.12
		3	3.38	1.00	31.26	0.13
		4	1.17	3.26	24.49	0.11
		5	2.17	2.06	29.45	0.12
		6	3.43	1.33	33.94	0.13
		7	4.66	1.00	35.54	0.14
		Avg	*2.87*	*1.52*	*30.49*	*0.12*
	Indivisible	8	0.00	5.66	16.43	0.09
J8	Divisible	1	2.31	1.00	38.83	0.17
		2	2.84	1.00	40.81	0.17
		3	3.26	1.00	42.62	0.17
		4	1.03	3.26	31.93	0.15
		5	2.03	2.06	38.73	0.16
		6	3.25	1.34	45.17	0.18
		7	4.49	1.00	48.33	0.19
		Avg	*2.74*	*1.52*	*40.92*	*0.17*
	Indivisible	8	0.00	5.49	21.56	0.12
J10	Divisible	1	2.15	1.00	47.01	0.20
		2	2.65	1.00	48.94	0.20
		3	3.03	1.00	51.12	0.21
		4	1.07	3.04	38.93	0.18
		5	2.02	1.93	47.05	0.20
		6	3.22	1.28	54.50	0.22
		7	4.25	1.00	58.08	0.22
		Avg	*2.63*	*1.46*	*49.38*	*0.20*
	Indivisible	8	0.00	5.25	26.92	0.14
J30	Divisible	1	2.17	1.00	167.22	3.21
		2	2.63	1.00	176.05	3.30
		3	3.04	1.00	184.71	3.40
		4	1.02	3.22	132.16	2.82
		5	1.97	2.05	163.02	3.18
		6	3.20	1.34	192.29	3.54
		7	4.41	1.00	209.88	3.77
		Avg	*2.63*	*1.52*	*175.05*	*3.32*
	Indivisible	8	0.00	5.41	84.88	2.45
J60	Divisible	1	2.27	1.00	334.81	9.37
		2	2.76	1.00	353.98	9.91
		3	3.19	1.00	370.93	10.39
		4	1.07	3.25	267.95	8.24
		5	2.03	2.06	328.29	9.37
		6	3.29	1.34	379.33	10.73
		7	4.52	1.00	413.52	12.23
		Avg	*2.73*	*1.52*	*349.83*	*10.03*
	Indivisible	8	0.00	5.52	169.18	6.95



	θi=0.8c·di	θi=1.0c·di	θi=1.2c·di
J6	33.58	30.25	27.65
J8	45.15	40.77	36.83
J10	54.28	49.35	44.50
J30	192.97	173.94	158.24
J60	385.52	347.48	316.50

Fig. 3. Influence of the key parameter θ_i on AOV.



	α=20%	α=25%	α=30%	α=35%	α=40%
J6	24.84	27.92	30.58	33.00	36.07
J8	33.50	37.55	41.25	44.55	48.00
J10	40.26	45.44	49.95	54.26	57.92
J30	148.85	164.22	176.79	188.93	199.51
J60	299.68	327.29	352.77	375.96	397.05

Fig. 4. Influence of the key parameter α on AOV.

the baseline schedules, which may help to obtain higher solution robustness. Essentially, the solution space of the divisible case is extended because of the constraint relaxation. The second phenomenon is that the AOV increases with the growth of η_i or the decline of ε_i. Activities can be split more frequently with a higher value of η_i or with a lower value of ε_i, which improves the scheduling feasibility, and thus this is beneficial for obtaining a higher objective function value.

Second, we investigate the influence of the key parameter θ_i on the index AOV for the five instance sets. The results are shown in Fig. 3, from which we can see that the growth of θ_i has a negative effect on the AOV. This is because there will be less space for inserting time buffers when taking more setup times into account.

In addition, we investigate the influence of the key parameter α on the index AOV for the five instance sets, and we take two more levels of α, 25% and 35%, into account. The results are shown in Fig. 4, from which we can see that the growth of α has a positive effect on the AOV. This is reasonable because there will be more inserted buffers in the schedule as the project due date constraint becomes less strict.

V. Conclusions

This paper presents a proactive RCPSP with activity splitting where each activity can be split at discrete time instants under the constraints of a maximum number of splitting and a minimum period of continuous execution. Besides, in this problem setup times are considered. Based on the analysis of the established model, two properties and one lemma are proposed and applied in our developed GA to improve the local search efficiency. In addition, after linearizing the proposed model, we use a commercial software as a benchmark to solve the problem. A computational experiment that is performed on datasets generated by the ProGen is designed and executed, from which the following conclusions are drawn.

1) The two developed properties and the proposed lemma can be used to maximize the objective function, and the GA with a combination of the three local search operators performs the best.
2) Compared with commercial software, the developed GA is much more efficient to solve the proposed scheduling problem, and the gap in terms of the objective function value is acceptable.
3) Due to the increase in flexibility of executing activities, activity splitting enhances the robustness of baseline schedules that are likely to have lower adjustment costs during project execution. Compared with the classic proactive scheduling models where activity splitting is not allowed, this paper offers a new method to improve schedule robustness when activity splitting is allowed and generates better solutions to project management.
4) With the growth of the maximum number of splitting, the decline in the minimum execution time, the decrease in the setup times, and the extension of the project due date, schedule robustness increases.

Note that the research in this paper is based on specific assumptions of activity splitting, so further research can provide support for quantitative decisions on project management under more complex and realistic conditions of activity splitting, such as cases in which activity splitting is allowed at arbitrary rational times. In addition, more effective and efficient algorithms can be developed to solve the proposed scheduling problem, and other efficient methods can be proposed to solve the zero-one polynomial formulations.

Appendix A
More Details About the Literature on Proactive Scheduling

See Table VIII top of the next page.

Appendix B
More Details About the Literature on the Resource-Constrained Project Scheduling Problem With Activity Splitting

See Table IX top of the next page.

TABLE VIII
DETAILS ABOUT THE LITERATURE ON PROACTIVE SCHEDULING

Reference	Mode		Objective		Surrogate robustness measure	Slacks		Algorithm	
	Single	Multi-	Robustness	Others		Time	Resource	Exact	Heuristic
[5]	√			Trade-off		√			CC/BM & ADFF
[6]		√		Trade-off	√	√		A two-phase approach	
[7]		√		Trade-off	√	√		Benders decomposition	Tabu search
[8]	√		√		√	√			EWD1
[9]	√		√			Resource allocation		Brand & bound	
[10]	√		√			√			Multiple procedures
[11]	√		√			√	√		Priority rule based
[12]	√		√			√			Steepest descent
[13]	√			Trade-off	√	√			Tabu search
[14]	√			Trade-off		√			RFDFF
[16]	√			Trade-off	√	√			Priority rule based
[17]	√			Cost		√			STC + D heuristic
[18]	√		√		√	Confidence level		CCP method	
[31]	√		√		√	√			Tabu search
Mine	√		√		√	√			Genetic algorithm

TABLE IX
DETAILS ABOUT THE LITERATURE ON THE RESOURCE-CONSTRAINED PROJECT SCHEDULING PROBLEM WITH ACTIVITY SPLITTING

Reference	Mode		Minimum execution time	Maximum splitting times	Setup times	Objective		Algorithm	
	Single	Multi-				Makespan	Others	Exact	Heuristic
[19]	√					√		Branch & bound	
[20]		√				√		Branch & bound	
[21]		√				√			Priority-based
[22]	√					√			Local search
[23]	√					√			Improved RCPSP
[24]	√		√	√		√			Evolutionary
[25]		√				√			Genetic algorithm
[26]	√					√		Lower bound	
[27]	√					√		Branch-and-price	
[28]		√				√		Branch & bound	
[29]	√						Cost		Genetic algorithm
[30]		√	√	√			Trade-off		Evolutionary
Mine	√		√	√	√		Robustness		Genetic algorithm

APPENDIX C
SIMPLIFICATION OF THE PROPOSED PROBLEM

In the proposed problem, activities can be split into certain parts, which is decided by two parameters, the maximum splitting times η_i and the minimum continuous execution time ε_i. If we set $\eta_i = 0$ or $\varepsilon_i = d_i$ for each activity i, then the problem will be simplified into the proactive scheduling problem without activity splitting. As the dummy end activity is assumed to start and end at the project deadline, we can add an extra dummy activity $(n\ -\ 1)$ in the project network $G' = (N, A')$ where $(n-1, n) \in A'$ and $(i, n-1) \in A'$, $\forall (i, n) \in A$. Then, we can set the weight of activity $(n\ -\ 1)$ as 1 and the weights of all other activities as 0. In this way, the objective function (maximize $\sum_{i=1}^{n} [w_i (\sum_{v=1}^{V_i} \sum_{b=1}^{FS_{i,v}} e^{-b})]$ will be simplified to maximize $\sum_{b=1}^{FS_{n-1,1}} e^{-b}$, which is equivalent to minimizing the project makespan $C_{\max}^{\mathrm{RCPSP}}$. As there are also precedence and resource constraints in our model and there would not be a deadline constraint if we set the project deadline much bigger than $C_{\max}^{\mathrm{RCPSP}}$, the proposed problem can be simplified into RCPSP with the objective of makespan minimization.

APPENDIX D
MORE DETAILS OF THE PROCEDURES IN THE GAS

Algorithm 1: Decoding procedure: $s_{(i,v)_j} = \mathrm{DCP}\,(L, \mathrm{DL}, \mathrm{BL})$.

1: $s_{(i,v)_1} = 0$
2: FOR $j = 2$ TO nsub DO
3: $\quad s_{(i,v)_j} = \max_{(i,v)_h \in P_{(i,v)_j}} (s_{(i,v)_h} + \mathrm{dur}_{(i,v)_h} + \mathrm{buf}_{(i,v)_h})$
4: $\quad$ WHILE $\exists k, t : \sum_{h \in S(t)} r_{h,k}^{\rho} > R_k^{\rho}$ $(k = 1, \ldots, K$ and $t = s_{(i,v)_j}, \ldots, s_{(i,v)_j} + \mathrm{dur}_{(i,v)_j} + \mathrm{buf}_{(i,v)_j} - 1)$ DO
5: $\quad\quad s_{(i,v)_j} = s_{(i,v)_j} + 1$
6: $\quad$ END WHILE
7: END FOR
8: $s_{(i,v)_{\mathrm{nsub}}} = \max(s_{(i,v)_{\mathrm{nsub}}}, D)$

Note: $P_{(i,v)_j}$ represents the set of predecessors of subactivity $(i, v)_j$.

Algorithm 2: Slack calculation: $\text{FS}_{(i,v)'_j} = \text{FSP}\,(L,\ \text{DL}, \text{BL})$.

1: $s_{(i,v)_j} = \text{DCP}\,(L,\ \text{DL}, \text{BL})$
2: Obtain the list L'
3: $\text{ES}_{(i,v)'_1} = \text{LS}_{(i,v)'_1} = s_{(i,v)'_1}$, $\text{FS}_{(i,v)'_1} = 0$
4: FOR $j = 2$ TO nsub DO
5: $\text{ES}_{(i,v)'_j} = s_{(i,v)'_j}$, $\text{LF}_{(i,v)'_j} = \min\{\text{ES}_h | h \in S_{(i,v)'_j}\}$, $\text{LS}_{(i,v)'_j} = \text{LF}_{(i,v)'_j} - \text{dur}_{(i,v)'_j}$
6: WHILE $\exists k, t : \sum_{h \in S(t)} r^{\rho}_{h,k} > R^{\rho}_k$ $(k = 1, \ldots, K$ and $t = \text{ES}_{(i,v)'_j}, \ldots, \text{LF}_{(i,v)'_j} - 1)$ DO
7: $\text{LF}_{(i,v)'_j} = \text{LF}_{(i,v)'_j} - 1$, $\text{LS}_{(i,v)'_j} = \text{LS}_{(i,v)'_j} - 1$
8: END WHILE
9: $\text{FS}_{(i,v)'_j} = \text{LS}_{(i,v)'_j} - \text{ES}_{(i,v)'_j}$
10: END FOR

Note: L' represents the list of subactivities that are ordered according to their nonincreasing completion times (the tiebreaker is the highest subactivity number). For convenience, $(i,v)'_j$ denotes the subactivity in position j of the ordered list L'. Additionally, $\text{dur}_{(i,v)'_j}$ represents the duration of the subactivity $(i,v)'_j$, and $S_{(i,v)'_j}$, $\text{ES}_{(i,v)'_j}$, $\text{LS}_{(i,v)'_j}$, and $\text{LF}_{(i,v)'_j}$ respectively, denote the set of immediate successors, the earliest starting time, the latest starting time, and the latest completion time of the subactivity $(i,v)'_j$.

Algorithm 3: Buffering: Robu′= BFP $(L,\ \text{DL}, \text{BL})$.

1: $\text{FS}_{(i,v)'_j} = \text{FSP}\,(L, \text{DL}, \text{BL})$, $\zeta = 0$
2: Calculate the objective function value Robu
3: WHILE $\zeta \le Z$ DO
4: Choose one subactivity $(i,v)_j$ from the list L, and then $\text{buf}_{(i,v)_j} = \text{buf}_{(i,v)_j} + 1$
5: $\text{FS}_{(i,v)'_j} = \text{FSP}\,(L, \text{DL}, \text{BL})$, and calculate its new objective function value Robu′
6: IF $s_{(i,v)_{nsub}} > D$ OR Robu′ $\le$ Robu THEN
7: $\zeta = \zeta + 1$, $\text{buf}_{(i,v)_j} = \text{buf}_{(i,v)_j} - 1$
8: ELSE
9: Robu = Robu′
10: END IF
11: END WHILE

Algorithm 4: Individual generation: $(L, \text{DL}, \text{BL}) = \text{IGP}(g)$.

1: DO
2: Initialize Elig and the three lists, $j = 0, V_i = 0\,(\forall i \in N), leftd(i) = d_i\,(\forall i \in N)$
3: WHILE Elig $\neq \emptyset$ DO
4: Select an activity i from Elig, $V_i = V_i + 1$, $j = j + 1$
5: $L_j = (i, V_i)$, $\text{buf}_{(i,v)_j} = 0$, nsub $= j$
6: Generate a random number m_1 between 0 and 1
7: IF $m_1 >$ itrpt AND $V_i \le \eta_i$ AND $leftd(i) \ge 2\varepsilon_i$ THEN
8: Generate a random number m_2 from $[\varepsilon_i, leftd(i) - \varepsilon_i]$

Algorithm 4: Continued.

9: $\text{dur}_{(i,v)_j} = m_2 + \theta_i$, $leftd(i) = leftd(i) - m_2$
10: ELSE
11: $\text{dur}_{(i,v)_j} = leftd(i) + \theta_i$, update Elig
12: END IF
13: END WHILE
14: WHILE $(s_{(i,v)_{\text{nsub}}} > D)$
15: Robu′= BFP$(L,\ \text{DL}, \text{BL})$

Note: Let $leftd(i)$ represent the number of duration units of activity i that have not yet been assigned (setup times are not included in $leftd(i)$), and let Elig, defined as $\text{Elig} = \{i \,|\, leftd(i) > 0 \text{ and } leftd(j) = 0,\ (j,i) \in A\}$, be the set of eligible activities.

Algorithm 5: Crossover: $(\text{L}_C, \text{DL}_C, \text{BL}_C) = \text{CRP}(\text{L}_F, \text{DL}_F, \text{BL}_F, \text{L}_M, \text{DL}_M, \text{BL}_M)$.

1: Generate a random number m between 1 and nsub_F
2: Copy the first m elements of every list of the father to the child
3: Obtain V_i and $leftd(i)$ of the child after copy, $j = \sum_{i \in N} (\eta_i + 1)$
4: FOR $q = \text{nsub}_M$ TO 1 DO
5: $i = L^M_q$, $d = \text{dur}_{(i,v)^M_q} - \theta_i$
6: IF $leftd(i) > 0$ THEN
7: IF $V_i \ge \eta_i$ OR $leftd(i) < d$ THEN
8: $dur_{(i,v)^C_j} = leftd(i) + \theta_i$, $leftd(i) = 0$
9: ELSE
10: $\text{dur}_{(i,v)^C_j} = d + \theta_i$, $leftd(i) = leftd(i) - d$
11: END IF
12: $L^C_j = L^M_q$, $\text{buf}_{(i,v)^C_j} = \text{buf}_{(i,v)^M_q} \times [(\text{dur}_{(i,v)^C_j} - \theta_i)/d]$
13: $V_i = V_i + 1$, $j = j - 1$
14: END IF
15: END FOR
16: Erase the blank cells from L_C, DL_C, and BL_C

Note: The parameters that are labeled with F, M, and C, respectively, represent the corresponding lists of the father, the mother, and the child.

Algorithm 6: Mutation: $(\text{L}_Q, \text{DL}_Q, \text{BL}_Q) = \text{MTP}(L, \text{DL}, \text{BL})$.

1: FOR $q = 1$ TO $\text{pmut} \cdot \text{nsub}$ DO
2: Randomly generate a number m_3 from $\{0, 1\}$, a number j from $[1, \text{nsub}]$
3: IF $m_3 = 0$ THEN
4: Calculate the possible positions $[a,\ b]$ of subactivity $(i,v)_j$ in the list L without causing the precedence constraint violation
5: Generate a random number m_4 from $[a,\ b]$
6: Place subactivity $(i,v)_j$ in position m_4 and update the lists
7: ELSE

Algorithm 6: Continued.

```
    Generate a random number m_5 from {0, 1}
    IF m_5 = 0 THEN
      buf_(i,v)_j = buf_(i,v)_j + 1
    ELSE
      IF buf_(i,v)_j ≥ 1 THEN
        buf_(i,v)_j = buf_(i,v)_j − 1
      END IF
    END IF
  END IF
END FOR
```

Note: The parameters that are labeled by Q represent the corresponding lists of the mutated individual.

APPENDIX E
LINEARIZATION OF THE MODEL

To conduct the linearization, we redefine V_i as the maximum number of subactivities of activity i, which is a constant value that is known in advance instead of being a decision variable. We use M to denote a large positive number and introduce U_i to represent the maximum number of free slacks of activity i, which is calculated as the length of the time window of activity i without the resource constraints under an indivisible scheduling environment. Then, the free slack $\text{FS}_{i,v}$ ranges from 0 to U_i. Additionally, five groups of binary variables are defined as follows:

$$y_{i,v} = \begin{cases} 0 & \text{if the duration of subactivity } (i,v) \text{ is zero} \\ 1 & \text{otherwise} \end{cases}$$

$$x_{i,v,u} = \begin{cases} 0 & \text{if free slack } u \text{ of subactivity } (i,v) \text{ is zero} \\ 1 & \text{otherwise} \end{cases}$$

$$\alpha_{i,v,t} = \begin{cases} 1 & \text{if } s_{i,v} \le t \\ 0 & \text{otherwise} \end{cases}$$

$$\beta_{i,v,t} = \begin{cases} 1 & \text{if } s_{i,v} + \text{dur}_{i,v} + \text{FS}_{i,v} > t \\ 0 & \text{otherwise} \end{cases}$$

$$\gamma_{i,v,t} = \begin{cases} 1 & \text{if } \alpha_{i,v,t} = \beta_{i,v,t} = 1 \\ 0 & \text{otherwise} \end{cases}.$$

There are seven groups of decision variables in the transformed linear model, i.e., $y_{i,v}$, $\text{dur}_{i,v}$, $s_{i,v}$, $x_{i,v,u}$, $\alpha_{i,v,t}$, $\beta_{i,v,t}$, and $\gamma_{i,v,t}$. Compared with those decision variables in the nonlinear model, $y_{i,v}$ is used to replace V_i while $\text{dur}_{i,v}$ and $s_{i,v}$ stay the same. In addition, $x_{i,v,u}$ is used to take the place of the computation of the free slack, while $\alpha_{i,v,t}$, $\beta_{i,v,t}$, and $\gamma_{i,v,t}$ will decide the set of activities that are in progress at time t. Based on the above definitions, the nonlinear scheduling model can be transformed into a linear one, as follows.

$$\text{Maximize Robu} = \sum_{i=1}^{n}\left[w_i\left(\sum_{v=1}^{V_i}\sum_{u=1}^{U_i} e^{-u} x_{i,v,u}\right)\right] \tag{1}$$

$$s_{1,1} = 0 \tag{2}$$

$$s_{i,V_i} + \text{dur}_{i,V_i} + \text{FS}_{i,V_i} \le s_{j,1} \;\; (i,j) \in A \tag{3}$$

$$s_{i,v} + \text{dur}_{i,v} + \text{FS}_{i,v} \le s_{i,v+1} \;\forall i;\; v = 1, \ldots, V_i - 1 \tag{4}$$

$$s_{n,1} \le D \tag{5}$$

$$\sum_{i=1}^{n}\left(r_{i,k}^{\rho}\sum_{v=1}^{V_i}\gamma_{i,v,t}\right) \le R_k^{\rho} \;\forall k, \forall t \tag{6}$$

$$\sum_{v=1}^{V_i}\text{dur}_{i,v} = d_i + \theta_i\sum_{v=1}^{V_i} y_{i,v} \;\forall i \tag{7}$$

$$V_i = \eta_i + 1 \tag{8}$$

$$\text{dur}_{i,v} \ge (\varepsilon_i + \theta_i) \times y_{i,v} \;\forall i;\; v = 1, 2, \ldots, V_i \tag{9}$$

$$y_{i,v+1} \le y_{i,v} \;\forall i;\; v = 1, 2, \ldots, V_i - 1 \tag{10}$$

$$\text{dur}_{i,v} + FS_{i,v} \le y_{i,v} \times M \;\forall i, \forall v \tag{11}$$

$$\sum_{u=1}^{U_i} x_{i,v,u} = \text{FS}_{i,v} \;\forall i, \forall v \tag{12}$$

$$M\,(\alpha_{i,v,t} - 1) \le t - s_{i,v} < M \times \alpha_{i,v,t} \;\forall i, \forall v, \forall t \tag{13}$$

$$M\,(\beta_{i,v,t} - 1) < s_{i,v} + \text{dur}_{i,v} + \text{FS}_{i,v} - t \le M \times \beta_{i,v,t} \tag{14}$$

$$2\,\gamma_{i,v,t} \le \alpha_{i,v,t} + \beta_{i,v,t} \le \gamma_{i,v,t} + 1 \;\forall i, \forall v, \forall t \tag{15}$$

$$\text{FS}_{i,v}, \text{dur}_{i,v}, \text{ and } s_{i,v} \text{ are nonnegative integers} \tag{16}$$

$$y_{i,v},\, x_{i,v,u},\, \alpha_{i,v,t},\, \beta_{i,v,t},\, \gamma_{i,v,t} \in \{0,1\} \;\forall i, \forall v, \forall t. \tag{17}$$

In the formulation, the objective function is transformed into a new linear one, while two constraints, (2), and (5), stay the same. In addition, six constraints, (3), (4), (6), (7), (8), and (9), are adjusted into new ones, and six constraints, from (10) to (15), are added. Specifically, $\text{FS}_{i,v}$ should be included in the precedence constraints (3) and (4). As the decision variable $x_{i,v,u}$ is used to decide the value of $\text{FS}_{i,v}$ through (12), it should also replace $\text{FS}_{i,v}$ in the objective function. In constraints (7), we now use $\sum_{v=1}^{V_i} y_{i,v}$ to represent the number of nondummy subactivities. Because V_i now represents the maximum number of subactivities of activity i, which is calculated by constraints (8), there will be dummy subactivities whose durations and free slacks should be zero. Hence, constraints (9) force the duration of each subactivity to be at least its minimum execution time plus its setup time, but only if it is a nondummy one. Moreover, constraints (10) and (11) ensure that the dummy subactivities are the last ones of each activity and that their duration and free slack are zero. Further, with three added constraints, which are shown in (13)–(15), to describe the set $S(t)$ based on the definition $S(t) = \{i | s_{i,v} \le t < s_{i,v} + \text{dur}_{i,v} + \text{FS}_{i,v}\}$, the resource constraints are transformed into new ones, as stated in constraints (6).

REFERENCES

[1] W. Herroelen and R. Leus, "Robust and reactive project scheduling: A review and classification of procedures," *Int. J. Prod. Res.*, vol. 42, pp. 1599–1620, 2004.

[2] W. Herroelen and R. Leus, "Project scheduling under uncertainty: Survey and research potentials," *Eur. J. Oper. Res.*, vol. 165, pp. 289–306, 2005.

[3] S. Van de Vonder, E. Demeulemeester, W. Herroelen, and R. Leus, "Proactive-reactive project scheduling trade-offs and procedures," in *Perspectives in Modern Project Scheduling*. New York, NY, USA: Springer, 2006, ch. 2, pp. 25–51.

[4] E. Demeulemeester and W. Herroelen, *Robust Project Scheduling*. Boston, USA: Now Publishers, 2011.

[5] S. Van de Vonder, E. Demeulemeester, W. Herroelen, and R. Leus, "The use of buffers in project management: The trade-off between stability and makespan," *Int. J. Prod. Econ.*, vol. 97, pp. 227–240, 2005.

[6] Ö. Hazır, M. Haouari, and E. Erel, "Robust scheduling and robustness measures for the discrete time/cost trade-off problem," *Eur. J. Oper. Res.*, vol. 207, pp. 633–643, 2010.

[7] Ö. Hazır, E. Erel, and Y. Günalay, "Robust optimization models for the discrete time/cost trade-off problem," *Int. J. Prod. Econ.*, vol. 130, pp. 87–95, 2011.

[8] W. Herroelen and R. Leus, "The construction of stable project baseline schedules," *Eur. J. Oper. Res.*, vol. 156, pp. 550–565, 2004.

[9] R. Leus and W. Herroelen, "Stability and resource allocation in project planning," *IIE Trans.*, vol. 36, pp. 667–682, 2004.

[10] S. Van de Vonder, E. Demeulemeester, and W. Herroelen, "Proactive heuristic procedures for robust project scheduling: An experimental analysis," *Eur. J. Oper. Res.*, vol. 189, pp. 723–733, 2008.

[11] O. Lambrechts, E. Demeulemeester, and W. Herroelen, "Proactive and reactive strategies for resource-constrained project scheduling with uncertain resource availabilities," *J. Scheduling*, vol. 11, pp. 121–136, 2008.

[12] O. Lambrechts, E. Demeulemeester, and W. Herroelen, "Time slack-based techniques for generating robust project schedules subject to resource uncertainty," *Ann. Oper. Res.*, vol. 186, pp. 443–464, 2011.

[13] M. A. Al-Fawzan and M. Haouari, "A bi-objective model for robust resource-constrained project scheduling," *Int. J. Prod. Econ.*, vol. 96, pp. 175–187, 2005.

[14] S. Van De Vonder, E. Demeulemeester, W. Herroelen, and R. Leus, "The trade-off between stability and makespan in resource-constrained project scheduling," *Int. J. Prod. Res.*, vol. 44, pp. 215–236, 2006.

[15] S. Van de Vonder, E. Demeulemeester, and W. Herroelen, "A classification of predictive-reactive project scheduling procedures," *J. Scheduling*, vol. 10, pp. 195–207, 2007.

[16] H. Chtourou and M. Haouari, "A two-stage-priority-rule-based algorithm for robust resource-constrained project scheduling," *Comput. Ind. Eng.*, vol. 55, pp. 183–194, 2008.

[17] F. Deblaere, E. Demeulemeester, and W. Herroelen, "Proactive policies for the stochastic resource-constrained project scheduling problem," *Eur. J. Oper. Res.*, vol. 214, pp. 308–316, 2011.

[18] P. Lamas and E. Demeulemeester, "A purely proactive scheduling procedure for the resource-constrained project scheduling problem with stochastic activity durations," *J. Scheduling*, vol. 19, pp. 409–428, 2016.

[19] E. Demeulemeester and W. Herroelen, "An efficient optimal solution procedure for the preemptive resource-constrained project scheduling problem," *Eur. J. Oper. Res.*, vol. 90, pp. 334–348, 1996.

[20] J. Buddhakulsomsiri and D. S. Kim, "Properties of multi-mode resource-constrained project scheduling problems with resource vacations and activity splitting," *Eur. J. Oper. Res.*, vol. 175, pp. 279–295, 2006.

[21] J. Buddhakulsomsiri and D. S. Kim, "Priority rule-based heuristic for multi-mode resource-constrained project scheduling problems with resource vacations and activity splitting," *Eur. J. Oper. Res.*, vol. 178, pp. 374–390, 2007.

[22] J. Damay, A. Quilliot, and E. Sanlaville, "Linear programming based algorithms for preemptive and non-preemptive RCPSP," *Eur. J. Oper. Res.*, vol. 182, pp. 1012–1022, 2007.

[23] F. Ballestin, V. Valls, and S. Quintanilla, "Pre-emption in resource-constrained project scheduling," *Eur. J. Oper. Res.*, vol. 189, pp. 1136–1152, 2008.

[24] F. Ballestín, V. Valls, and S. Quintanilla, "Scheduling projects with limited number of preemptions," *Comput. Oper. Res.*, vol. 36, pp. 2913–2925, 2009.

[25] V. V. Peteghem and M. Vanhoucke, "A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem," *Eur. J. Oper. Res.*, vol. 201, pp. 409–418, 2010.

[26] M. Haouari, A. Kooli, E. Neron, and J. Carlier, "A preemptive bound for the resource constrained project scheduling problem," *J. Scheduling*, vol. 17, pp. 237–248, 2014.

[27] A. Moukrim, A. Quilliot, and H. Toussaint, "An effective branch-and-price algorithm for the preemptive resource constrained project scheduling problem based on minimal interval order enumeration," *Eur. J. Oper. Res.*, vol. 244, pp. 360–368, 2015.

[28] M. Vanhoucke and D. Debels, "The impact of various activity assumptions on the lead time and resource utilization of resource-constrained projects," *Comput. Ind. Eng.*, vol. 54, pp. 140–154, 2008.

[29] S. Quintanilla, Á. Pérez, P. Lino, and V. Valls, "Time and work generalised precedence relationships in project scheduling with pre-emption: An application to the management of Service Centres," *Eur. J. Oper. Res.*, vol. 219, pp. 59–72, 2012.

[30] M. Tavana, A. R. Abtahi, and K. Khalili-Damghani, "A new multi-objective multi-mode model for solving preemptive time-cost-quality trade-off project scheduling problems," *Expert Syst. Appl.*, vol. 41, pp. 1830–1846, 2014.

[31] O. Lambrechts, E. Demeulemeester, and W. Herroelen, "A tabu search procedure for developing robust predictive project schedules," *Int. J. Prod. Econ.*, vol. 111, pp. 493–508, 2008.

[32] J. Blazewicz, J. K. Lenstra, and A. H. G. R. Kan, "Scheduling subject to resource constraints: Classification and complexity," *Discrete Appl. Math.*, vol. 5, pp. 11–24, 1983.

[33] R. Leus, "The generation of stable project plans: Complexity and exact algorithms," PhD dissertation, Faculty Econ. Bus., Katholieke Universiteit Leuven, Belgium, 2003.

[34] J. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis With Applications to Biology, Control and Artificial Intelligence*. Michigan, USA: Univ. Michigan Press, 1975.

[35] W. Huang and L. Ding, "Project-scheduling problem with random time-dependent activity duration times," *IEEE Trans. Eng. Manage.*, vol. 58, no. 2, pp. 377–387, May 2011.

[36] C. Fang, F. Marle, M. Xie, and E. Zio, "An integrated framework for risk response planning under resource constraints in large engineering projects," *IEEE Trans. Eng. Manage.*, vol. 60, no. 3, pp. 627–639, Aug. 2013.

[37] L. J. Watters, "Reduction of integer polynomial problem to zero-one linear programming problems," *Oper. Res.*, vol. 15, pp. 1171–1174, 1967.

[38] R. Kolisch, A. Sprecher, and A. Drexl, "Characterization and generation of a general class of resource-constrained project scheduling problems," *Manage. Sci.*, vol. 41, pp. 1693–1703, 1995.

[39] R. Kolisch and A. Sprecher, "PSPLIB – A project scheduling problem library," *Eur. J. Oper. Res.*, vol. 96, pp. 205–216, 1997.

Zhiqiang Ma received the B.S. degree in marketing from the China University of Petroleum, Qingdao, China, in 2013. He is currently working toward the Ph.D. degree in management science and engineering with Xi'an Jiaotong University, Xi'an, China.

His research interests include project scheduling and robust optimization.

Zhengwen He received the B.S. degree in mechanical engineering from Xi'an Jiaotong University, Xi'an, China, in 1989, the M.S. degree in mechanical and electronic engineering from the Xi'an University of Technology, Xi'an, in 2001, and the Ph.D. degree in business administration from Xi'an Jiaotong University, in 2004.

He is currently a Professor with the Industrial Engineering Department, Xi'an Jiaotong University, and is a member of the Scheduling Committee of the Operations Research Society of China, Beijing, China. He has authored 18 articles in international journals, including the *European Journal of Operational Research*, *Annals of Operations Research*, the *Journal of the Operational Research Society*, *Computers and Operations Research*, *Computers and Industrial Engineering*, and the *International Journal of Production Research*.

Nengmin Wang received the B.S. degree in investment economy and the M.S. degree in management science and engineering from the Central South University of Technology, Changsha, China, in 1997 and 2000, respectively, and the Ph.D. degree in management science and engineering from Xi'an Jiaotong University, Xi'an, China, in 2003.

He is currently a Professor with the Industrial Engineering Department, Xi'an Jiaotong University. He has authored 30 articles in international journals, including the *European Journal of Operational Research*, *Annals of Operations Research*, the *International Journal of Production Research*, the *International Journal of Production Economics*, the *Journal of Management Information Systems*, the IEEE TRANSACTIONS ON ENGINEERING MANAGEMENT, the IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, *Computers and Operations Research*, *Computers and Industrial Engineering*, *Information Systems Frontiers*, *Expert Systems With Applications*, and the *Journal of Operational Research Society*. His research interests include supply chain management and big data.

Zhen Yang received the Ph.D. degree in system security and optimization from the Université de Technologie de Troyes, Troyes, France, in 2010.

He is currently an Assistant Professor with the Industrial Engineering Department, Xi'an Jiaotong University, Xi'an, China. He has authored or coauthored five articles in international journals, such as *Computers and Industrial Engineering*, the *European Journal of Operational Research*, *Computers and Operations Research*, and the *International Journal of Production Research*. His research interests include supply chain management and combinatorial optimization.

Erik Demeulemeester received the Business Engineering degree in management informatics, the master's degree in business administration, and the Ph.D. degree in applied economics from KU Leuven, Leuven, Belgium, in 1987, 1988, and 1992, respectively.

Since 2001, he has been a full-time Professor with the Research Center for Operations Management, KU Leuven, and currently teaches a course on project management and scheduling, a doctoral course on combinatorial optimization and local search techniques, as well as a seminar on operations management. His main research interests are situated in the field of project scheduling and health care planning, and he has published many papers on these topics.

Prof. Demeulemeester serves on the editorial board for the *European Journal of Operational Research*, the *Journal of Scheduling*, *Computers and Operations Research*, and the *European Journal of Industrial Engineering*. He was also appointed as a Member of the Program Committee for the EURO XXVIII Conference in Poznan, Poland (July 3–6, 2016). Additionally, he became a core jury member for the EURO Excellence in Practice Award (EEPA) that were and will be awarded at the EURO-k conferences in 2016, 2018, and 2019.