mathematics

MDPI

Article

Efficient Rollout Algorithms for Resource-Constrained Project Scheduling with a Flexible Project Structure and Uncertain Activity Durations

Chunlai Yu [1,2], Xiaoming Wang [1,2,*] and Qingxin Chen [1,2]

[1] State Key Laboratory of Precision Electronic Manufacturing Technology and Equipment, Guangdong University of Technology, Guangzhou 510006, China; 1111801012@mail2.gdut.edu.cn (C.Y.); qxchen@gdut.edu.cn (Q.C.)
[2] Provincial Key Laboratory of Computer Integrated Manufacturing, Guangdong University of Technology, Guangzhou 510006, China
* Correspondence: simonwang@gdut.edu.cn

Abstract: This study addresses the resource-constrained project scheduling problem with flexible structures and uncertain activity durations. The problem is formulated as a Markov decision process, with the optimal policy determined through stochastic dynamic programming. To mitigate the curse of dimensionality in large-scale problems, several approximate methods are proposed to derive suboptimal policies. In addition to traditional methods based on priority rules and metaheuristic algorithms, we focus on the application of rollout algorithms. To improve the computational efficiency of the rollout algorithms, only the best-performing priority rules are employed for action evaluation, and the common random numbers technique is also incorporated. Experimental results demonstrate that rollout algorithms significantly outperform priority rules and metaheuristics. The common random numbers technique not only enhances computational efficiency but also improves the accuracy of action selection. The post-rollout algorithm reduces computation time by 44.37% compared to the one-step rollout, with only a 0.02% performance gap. In addition, rollout algorithms perform more stably than other methods under different problem characteristics.

Keywords: RCPSP; flexible project structure; uncertain duration; rollout algorithm; suboptimal policy

MSC: 90C27

check for updates

Academic Editor: Shih-Wei Lin

Received: 11 March 2025
Revised: 20 April 2025
Accepted: 22 April 2025
Published: 24 April 2025

Citation: Yu, C.; Wang, X.; Chen, Q. Efficient Rollout Algorithms for Resource-Constrained Project Scheduling with a Flexible Project Structure and Uncertain Activity Durations. *Mathematics* **2025**, *13*, 1395. https://doi.org/10.3390/math13091395

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The resource-constrained project scheduling problem (RCPSP) involves scheduling activities while satisfying precedence and resource constraints to optimize a specific objective function, such as minimizing project makespan or maximizing net present value [1,2]. The RCPSP is a class of problems commonly found in many important engineering fields such as construction, manufacturing, research and development programs, etc. Although the RCPSP has been proven to be NP-hard in a strong sense [3], many scholars have conducted research on it in the past few decades due to the importance of the RCPSP in practical applications. The traditional RCPSP usually assumes a deterministic environment and a fixed project network. However, real projects often face many uncertainties during the execution of activities [4,5], and project networks may be flexible [6].

In practice, many projects involve multiple decision points where managers must select among activities. For instance, a customized equipment manufacturing project in-

volves multiple selections at different stages. At the design stage, one design solution must be selected from several options to meet the client's requirements. Different options mean different manufacturing or purchasing activities. At the manufacturing stage, either outsourcing or in-house production must also be selected as the final execution activity. Ref. [6] points out that in aircraft engine maintenance, specific issues can be addressed through various repair options. The common characteristic is that scheduling includes deciding whether to implement specific optional activities and imposing the related precedence constraints. Considering this characteristic, the RCPSP with a flexible project structure (RCPSP-FPS) was first defined by [6]. In recent years, this problem has garnered increasing attention [2,7]. These studies mainly focus on variants of the RCPSP-FPS problem under a deterministic environment.

Although there may be many types of uncertainties during project execution, they ultimately affect the activity duration [4,5]. Therefore, existing research on the stochastic RCPSP mainly considers uncertain activity duration. In different studies, scholars have tried to use different probability distributions to describe the uncertainty of activity duration, such as exponential distribution [8–15], beta distribution [9–11,16], uniform distribution [8–11,16], triangular distribution [17], normal distributions [18], etc. Among them, triangular distribution has gained particular prominence in practice due to its intuitive three-parameter structure (minimum, mode, maximum) that directly corresponds to the three-point estimation technique (optimistic, most likely, pessimistic) commonly employed by project managers.

In this study, we focus on the RCPSP-FPS with uncertain activity durations. Specifically, our goal is to propose approximate algorithms that have good optimization effects and are computationally efficient for large-scale problems. Since current approximate algorithms for the large-scale stochastic RCPSP are mainly based on priority rules or metaheuristic algorithms, we verify the effectiveness of our algorithm by comparing it with these methods. The main contributions of this paper are as follows:

- Exact solutions to small-scale problems. The problem considered is formulated as a discrete-time Markov decision process (MDP), and stochastic dynamic programming (SDP) is used to obtain the optimal solution. To the best of our knowledge, this is the first time of an MDP approach being applied to the stochastic RCPSP-FPS. Exact solutions are an important benchmark for evaluating the performance of approximate algorithms.
- A comprehensive comparison of the performance of priority rule-based methods for solving the stochastic RCPSP-FPS. Although [19] made a comprehensive comparison of the performance of priority rule-based methods in solving the deterministic RCPSP-FPS, these conclusions may not be applicable to the stochastic RCPSP-FPS. Therefore, our work fills the gap in relevant research and provides reference for selecting appropriate decision rules for practical engineering applications.
- The development of efficient approximate method based on rollout algorithms. In the improved rollout algorithms, we only use the best performing paired rules to simulate and evaluate feasible actions, and we integrate the common random numbers technique. The experimental results show that it not only effectively improves the efficiency of the rollout algorithms but also enhances the accuracy of optimal action selection.

The rest of this paper is organized as follows: Section 2 provides a review of the relevant literature. Section 3 defines and formally states the problem under investigation. Section 4 constructs a discrete-time Markov decision process model. Section 5 presents the approximate methods. Section 6 reports computational experiments to evaluate the effec-

tiveness of the different algorithms. Finally, Section 7 presents conclusions and suggestions for future research.

2. Literature Review

The traditional RCPSP has been proven to be NP-hard [3]. The RCPSP-FPS is inherently NP-hard as it is a variant of the traditional RCPSP. The RCPSP mainly focuses on activity sequencing subject to precedence constraints. In contrast, the RCPSP-FPS considers not only activity sequencing but also activity selection.

The existing literature on flexible project structure mainly focuses on three types of activity selection: individual activity, activity chains, and subgraph. For individual activity selection, Ref. [6] was among the first to investigate this RCPSP-FPS problem, developing a mathematical model for the RCPSP-FPS with integer durations based on the classic time-indexed (TI) formulation and proposing a modified genetic algorithm (GA) to solve it. Ref. [20] developed a general model for the RCPSP-PS by introducing the concept of selection groups. Cutting planes and a constraint propagation technique were introduced to obtain better lower bounds. For activity chain selection, Ref. [21] investigated scenarios where multiple sets of activities in a project have sequential relationships, and these activity sets can be alternative. This scenario is defined as the RCPSP with alternative activity chains (RCPSP-AC). During project execution, once the activities in a chain begin to be implemented, activities in other chains will not be executed. A simulated annealing algorithm is used to solve this problem. In [22], cost objectives were introduced, expanding the RCPSP-AC into a multi-objective problem. Ref. [23] investigated a resource-constrained project scheduling problem with hierarchical alternative plans and stochastic activity durations, establishing a model with a stochastic chance constraint. The solution involves a hybrid metaheuristic framework combining sample average approximation with the population-based evolutionary artificial algae algorithm. Ref. [24] extended the work of [23] to a multi-project environment. For subgraph selection, Ref. [25] explored a resource-constrained project scheduling problem with alternative subgraphs (RCPSP-AS) and applied a Tabu search to solve it. Ref. [26] utilized a genetic algorithm with a boolean satisfiability solver to address the RCPSP-AS. Ref. [27] explored extensions of the RCPSP-AS with more complex relationships between alternatives, aiming to assess their impact on solution quality and computational complexity. They concluded that finding feasible or optimal solutions becomes increasingly challenging as the complexity of the project structure grows. Ref. [19] conducted a comprehensive comparative study of heuristic methods for RCPSP-AS, finding that the use of different priority rules for activity selection and sequencing significantly reduces the average project makespan. They also observed that priority rules incorporating project properties (e.g., subgraphs and links) have a substantial impact on lowering the project makespan. Except for [23], all other studies on flexible project structure assumed a deterministic environment, which deviates significantly from some real applications. In order to deal with the uncertainty of activity durations, Ref. [23] used chance constrained programming to model the problem, which can only obtain a static approximation strategy and cannot obtain the optimal dynamic strategy for the problem.

In the context of the stochastic RCPSP with a fixed project network, MDP is a common method to deal with uncertainty through state transfer. It is suitable for dynamic, multi-stage decision problems. Decisions at each moment affect the future state and are suitable for projects with long-term decision impacts. Ref. [28] investigated the PERT network with activity durations following an exponential distribution. This was one of the earliest works to employ MDP as a modeling approach for the stochastic RCPSP. Afterwards, this approach was applied to solve the RCPSP with exponential activity durations for maximizing expected net present value [12,14,29,30], as well as for minimizing expected project

makespan [13,15]. In addition, the MDP approach was also applied to the RCPSP with random resource availability [31] or random rework [32,33]. Ref. [34] studied the Resource-Constrained Multi-Project Scheduling Problem (RCMPSP) with uncertain activity durations and random new project arrivals, proposing the problem as a discrete-time Markov decision process and using dynamic programming (DP) to evaluate optimal policies. Although the SDP method can theoretically be used to solve the optimal strategy of MDP models, it is limited by the curse of dimensionality and can only be applied to small-scale problems [35,36]. Therefore, most scholars have focused on studying approximate methods for the stochastic RCPSP. For the problem studied by [34,37], they proposed a linear approximation model and a simulation-based approximate dynamic programming approach to solve for the large scale. Specifically, the existing approximate methods for the RCPSP with random activity durations are mainly based on heuristic methods [8–11,16,38–40]. Similarly, approximate methods for the RCPSP with random rework are also based on priority rules [41] and genetic algorithms [42].

As for the rollout-based policy for the stochastic RCPSP, to the best of our knowledge, there is only one relevant study [43]. The authors focused on the RCPSP with random activity durations and introduced a rollout algorithm with a hybrid look-back and look-ahead approximation architecture. To improve the efficiency of their rollout algorithm, they adopted a limited simulation approach and used the constraint programming method as the basic heuristic for the simulation. However, as shown in its experimental results, the main limitation of this method is its high computational complexity. The rollout algorithm proposed in this paper differs from prior work in that it incorporates the common random numbers (CRN) technique to reduce variance, which in turn improves simulation efficiency and the accuracy of action selection.

The relevant literature of stochastic RCPSP discussed above is summarized in Table 1. Based on the above literature review, we observe that there is a lack of research on the RCPSP-FPS in stochastic environments. Furthermore, The current approximate methods for large-scale stochastic RCPSP are mainly based on priority rules and metaheuristic algorithms, which have significant differences from the optimal solutions obtained based on MDP [13]. The only study on a rollout algorithm for the stochastic RCPSP showed that although its optimization effect was significantly better than heuristic methods, its computational cost was too high. Therefore, in the present work, we focus on the development of an efficient rollout algorithm for the large-scale stochastic RCPSP-FPS.

Table 1. Summary of research on stochastic RCPSP.

Problem Definition	Literature	Objective	Approach	Optimal
RCPSP with hierarchical alternative plans and stochastic activity durations	[23]	makespan	Chance constrained programming, sampling average approximation	N
RCPSP with phase-type distributed	[14]	net present value	Continuous-time Markov chain, SDP	Y
RCPSP with hypoexponentially distributed	[15]	makespan	Continuous-time Markov chain, SDP	Y
RCMPSP with uncertain activity durations	[31]	makespan	MDP, ruler-based approximation method	N
RCMPSP with uncertain activity durations and random new project arrivals	[34]	discounted Long-run profit	MDP, SDP	Y
	[37]	discounted Long-run profit	MDP, linear approximation model	N
RCPSP with uncertain activity durations	[16]	makespan	Ruler-based simulation	N
	[43]	makespan	MDP, rollout look-ahead policy	N
RCPSP with uncertain activity durations and transfer times	[40]	makespan	genetic programming, ruler-based simulation	N
RCPSP with uncertain durations and time-varying resource requests	[39]	makespan	RCPSP/t model, an advanced memetic algorithm	N
RCMPSP with uncertain durations and feedback	[42]	makespan	genetic algorithm	N

3. Problem Description

3.1. Discrete Triangular Distributions

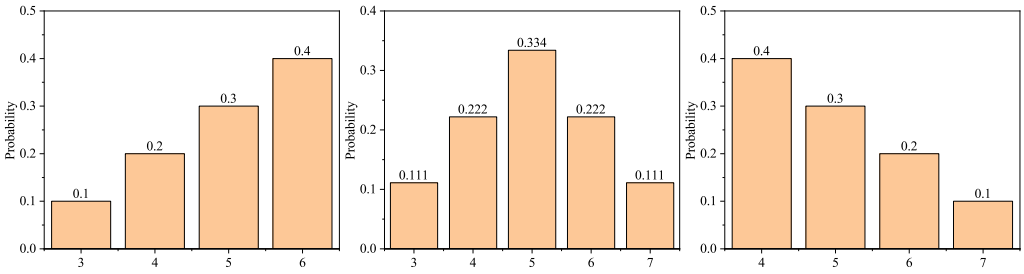
The discrete triangular distribution effectively captures the uncertainty in duration, even when historical data are scarce [44]. Furthermore, these parameters are typically obtained from project experience and expert judgment. The probability mass function of discrete triangular distribution is defined as

$$P(d_j = d) = \begin{cases} \frac{2(d-h)}{(l-h)(q-h)} & \text{if } h \le d \le l, \\ \frac{2(q-d)}{(q-l)(q-h)} & \text{if } l < d \le q, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

where h is the lower bound of d_j, q is the upper bound, and l is the most likely value. The d_j is the duration of activity j, which is assumed to be a random variable following the discrete triangular distribution. The mean duration $\bar{d}_j$ of activity j is given by

$$\bar{d}_j = \frac{h + l + q}{3} \tag{2}$$

Given the mean activity duration, we can construct three discrete triangular distributions with different distribution characteristics. For example, Figure 1 presents three discrete triangular distributions with $\bar{d}_j = 5$.



(**a**) Left-skewed. (**b**) Symmetric. (**c**) Right-skewed.

Figure 1. Three discrete triangular distributions with $\bar{d}_j = 5$.

3.2. Resource-Constrained Project Scheduling with a Flexible Project Structure

The traditional RCPSP consists of a project and a set of renewable resources R. The project is represented by an activity-on-node (AON) graph $G(N, A)$, where $N = \{0, 1, \ldots, n, n+1\}$ is the set of activities and A is the precedence relationships between activities. The activity 0 and $n+1$ are dummy activities with zero duration and resource requirements which indicate the start and end of the project, respectively. Each activity $j \in N$ requires a non-negative constant amount k_{jr} of resource $r \in R$. A resource $r \in R$ has a constant period capacity of K_r.

The main distinction between the RCPSP-FPS and the traditional RCPSP is that not all activities are necessarily implemented. The basic assumptions of the problem are outlined by [6] and are described in detail as follows:

- Let $V \subseteq N$ represent the set of mandatory activities, and the activities not in V are considered optional.
- Let $E \subseteq N$ represent the set containing multiple triggering activities. Each triggering activity $i \in E$ corresponds to an optional activity set O_i. Note that all optional sets are disjoint, meaning that each optional activity belongs to only one optional set. If the triggering activity $i \in E$ is implemented, the project manager must make a selection where exactly one activity in O_i must be implemented and the others in O_i will never

be implemented. The triggering activity may be either a mandatory or an optional activity, potentially forming a nested chain of triggers.

- Let $D \subseteq N \setminus V$ represent the set of dominant activities. A dominant activity $j \in D$ is an optional activity that has a set of dependent optional activities $\beta_j \subseteq N \setminus V \setminus \{j\}$. All dependent activities in β_j must be implemented if activity j is implemented. Conversely, if activity j is not implemented, then all the activities in β_j will not be implemented.

To avoid cycles of cause and effect, the activities are topologically ordered, ensuring that the condition $i \prec j$ holds for all $i \in E$ and $j \in O_i$, where $i \prec j$ means that activity i precedes activity j. Additionally, the dependencies between optional activities are considered. The activities are also topologically ordered so that if $j' \in \beta_j$, then $j \prec j'$.

To illustrate the flexible project structure clearly, Figure 2 presents a simplified version of a factory workstation renovation project. Based on customer demand, there are two choices: upgrading the equipment in the workstation or replacing it with new equipment. If the choice is to upgrade, parts need to be procured. If replacement is chosen, there is an additional decision regarding the production method of the parts: outsourcing or in-house production. Once the parts are ready, assembly can begin. If the upgrade option was selected earlier, the assembled components must also be integrated with the equipment. After everything is completed, the operation phase can begin. There are two optional activity sets, O_1 and O_2, corresponding to the triggering activities 0 and 2, respectively. The two activity selection decisions are nested, meaning that the optional activity corresponding to the first selection decision serves as the triggering activity for the second selection decision. If activity 2 is not implemented, neither activities 4 nor 5 will be carried out. Additionally, activities 3 and 7 depend on optional activity 1. Specifically, activities 3 and 7 will not be implemented unless activity 1 is selected in optional activity set O_1; otherwise, they will be implemented.

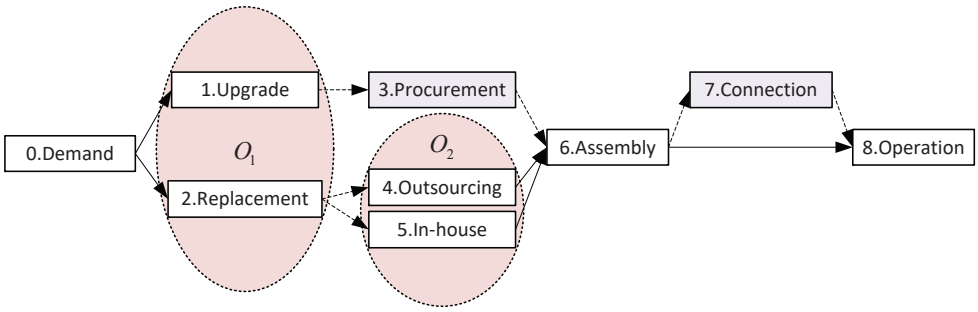


Figure 2. An example of a flexible project network.

Our goal is to find a scheduling policy that minimizes the expected project makespan. For the stochastic RCPSP-FPS, the following assumptions are made:

- The duration of activity j is assumed to be a random variable following the discrete triangular distribution.
- Preemption is not allowed.

4. MDP Model and Optimal Policy

For the problem described in the previous section, we focus on constructing the corresponding mathematical model and solution method. The overall research idea of this paper is shown in Figure 3. Different from the traditional RCPSP, the problem studied in this paper considers both flexible project structure and uncertain activity duration. Since this problem is essentially a stochastic sequential decision problem, we first model it as a discrete-time MDP (Section 4.1). For small-scale problems, the exact SDP method is em-

ployed to obtain optimal policies (Section 4.2). For large-scale problems, the state space and action space will expand rapidly as the problem size grows, making the SDP method unusable. Therefore, we focus on the approximate methods based on paired rules (Section 5.1), metaheuristic algorithms (Section 5.2), and rollout algorithms (Section 5.3). In this section, we first describe the problem as an MDP and use SDP to solve the optimal policy.

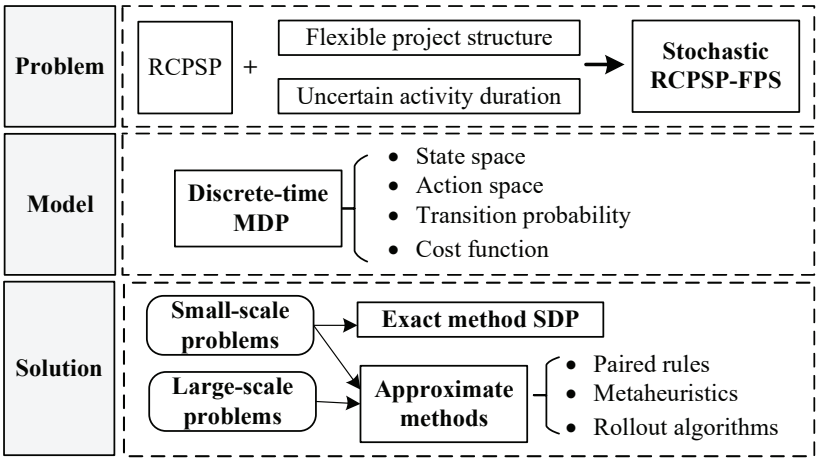


Figure 3. The overall framework of this study.

4.1. MDP Model

A sequential decision-making problem under uncertainty is typically modeled as an MDP [45]. From the perspective of project execution, the start of an activity depends solely on the completion of its predecessors and the current availability of sufficient resources. In other words, project scheduling decisions are related only to the current state and not to past states, i.e., they exhibit the Markov property. Therefore, we model the problem as a discrete-time, finite-state MDP $\{X, A, P(\cdot,\cdot), c(\cdot,\cdot)\}$, which is defined as follows.

State space X. The state must accurately and concisely express all the information required for decision-making at any time. We define the system state x as Formula (3):

$$x = [F_x, D_x, G_x, \delta_x]^T \tag{3}$$

where $F_x \subseteq N$ is the set of finished activities, $D_x \subseteq N$ is the set of never implemented activities, $G_x \subseteq N$ is the set of ongoing activities, and $\delta_x = \{\check{d}_1, \check{d}_2, \ldots, \check{d}_{|G_x|}\}$ is the set of durations for which each activity $j \in G_x$ has been implemented.

Let $W_x \subseteq N$ denote the set of waiting activities. A waiting activity refers to an activity whose predecessors have all been completed but which has not yet started. The set of waiting activities is crucial because in each state, the decision is to choose a set of activities to start while also deciding which optional activities need to be implemented. According to our state definition, we have $W_x = \{i \in N \setminus (F_x \cup D_x \cup G_x) \mid P_i^* \subseteq F_x\}$, where P_i^* is the set of (not necessarily immediate) predecessors of activity i. Additionally, the available quantity of each type of resource in state x can be easily determined based on F_x and G_x.

Since no activities are scheduled at the initial time, the initial state is $x_0 = [\emptyset, \emptyset, \emptyset, \emptyset]^T$. Once all activities are completed, the system will remain in a final absorbing state $x^* = [F^*, D^*, \emptyset, \emptyset]^T$, where $F^* \cup D^* = N$.

Action space A. Two types of decisions need to be made in each state. The first type of decision is to start a feasible set of activities subject to resource constraints. The second type of action is to select an activity to be executed from the corresponding optional activity set once a triggering activity is started. Hence, we define the action $a_x \in A_x$ as Formula (4):

$$a_x = \{\omega_x, \varphi_x\} \tag{4}$$

where $\varpi_x \subseteq W_x$ is the set of activities to be started in state x, and φ_x is the set of optional activities determined not to be implemented in state x. Note that there may be two types of optional activities in φ_x. The first type consists of activities from the set of optional activities corresponding to the triggering activities in ϖ_x, while the second type consists of dependent activities corresponding to the first type. These dependent activities will be immediately completed upon taking action a_x. Clearly, $\varphi_x = \emptyset$ if there is no triggering activity to be started in state x.

Transition probability $p(x'|x,a)$. The system state transition is driven by the initiation and completion of activities. After taking an action $a \in A_x$ in state x, the system state first transitions immediately to a post-decision state and then transitions to the next state x' after a period of time when the next ongoing activity finishes. In the post-decision state, the set of finished activities is $D(x,a) = D_x \cup \varphi_x$ and the set of ongoing activities is $G(x,a) = G_x \cup \varpi_x$, where $\check{d}_j$ for all $j \in G_x$ remain unchanged, and $\check{d}_j$ for all $j \in \varpi_x$ are initially set to zero. To simplify the state transition process, intermediate states are typically omitted. In other words, the system will transition from the current state x to the next state x' with a certain probability after taking action $a \in A_x$. Since the activity duration is the only random factor in the problem, the next state and corresponding transition probability depend on the combination of completion statuses of the ongoing activities in $G(x,a)$. Since only discrete durations are considered in this work, ongoing activities can only be completed at discrete time points. Specifically, we define the state transition step length $L_{xx'} = \min_{j \in G(x,a)} \hat{d}_j - \check{d}_j$, where $\hat{d}_j > \check{d}_j$ is the next possible value of d_j. Thus, the time span between x and x' is $L_{xx'}$. Furthermore, let the binary variable $\alpha_j = 1$ if the ongoing activity $j \in G(x,a)$ finishes in the next state x', and $\alpha_j = 0$ otherwise. The probability $p(\alpha_j = 1)$ is given by the following Formula (5), and $p(\alpha_j = 0) = 1 - p(\alpha_j = 1)$:

$$p(\alpha_j = 1) = p(d_j = L_{xx'} + \check{d}_j \mid d_j > \check{d}_j) = \frac{p(L_{xx'} + \check{d}_j)}{p(d_j > \check{d}_j)}, \forall j \in G(x,a) \tag{5}$$

We focus on those completion statuses with a probability greater than zero and obtain all possible next states by enumerating all combinations of them. The probability of each combination of completion statuses, and hence the state transition probability, can be calculated based on the independence between activities. Specifically, we have

$$p(x'|x,a) = \prod_{j \in G(x,a)} p(\alpha_j) \tag{6}$$

Cost function $c(x'|x,a)$. The expected project completion time can be regarded as the time required to transition from the initial state to the final state. This suggests that the cost function $c(x'|x,a)$ can be defined as the sojourn time in state x after taking action a, which is equal to $L_{xx'}$.

Based on the above MDP model definition, we can clearly observe the dynamic decision-making process of this problem, as shown in Figure 4. For a given state x, we first obtain the feasible actions, and the system will immediately transition to a post-decision state after an action is taken. After a certain time interval, the system will further transition to the next state when some ongoing activities have finished. Note that there are usually multiple possible next states because the activity durations are uncertain, and each possible state transition corresponds to a transition probability and cost.

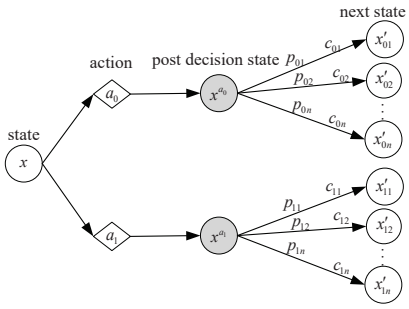


Figure 4. Schematic diagram of decision-making process and state transition.

4.2. Optimal Policy

An optimal policy of the above MDP model can be obtained using SDP based on the following Bellman equation.

$$v^{l+1}(x) = \min_{a \in A_x} E\{c(x'|x,a) + v^l(x'|x,a)\} \tag{7}$$

where l indicates the l-th iteration.

After the convergence of the value function, we can obtain the following optimal actions a^* in state $x \in X$.

$$a^* = \arg\min_{a \in A_x} E\{c(x'|x,a) + v^*(x'|x,a)\} \tag{8}$$

The optimal function value $v^*(x_0)$ is the global optimal objective function value, namely, the minimum expected project completion time.

4.3. An Illustrative Example

Figure 5 presents an example of a simple project with a flexible structure and uncertain activity durations. The project contains a selection where the triggering activity is activity 1 and the optional activity set is $O_1 = \{2, 3\}$. Activity 5 is dependent on activity 2. The duration of activity 2 has a 0.4 probability of being 1 and a 0.6 probability of being 3. The duration of activity 3 has a 0.3 probability of being 2, a 0.3 probability of being 3, and a 0.4 probability of being 4. The maximum resource is assumed to be 2.

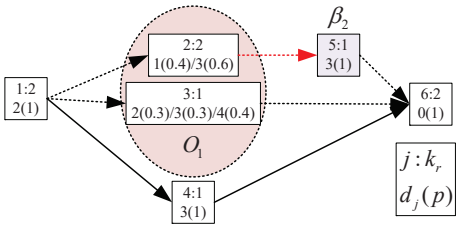


Figure 5. An example of a flexible project network with uncertain activity durations.

This problem is modeled as an MDP, and its complete decision process is shown in Figure 6. A total of 14 distinct states are identified, ranging from the initial state to the project completion state. In state x_0, activity 1 is waiting to be started. Since it is a triggered activity, a selection must be made at this stage. Selecting activity 3 will result in the action $a = \{[1], [2, 5]\}$, while selecting activity 2 will result in the action $a = \{[1], [3]\}$. The system will transition from state x_0 to state x_1 with a probability of 1, incurring a cost of 2. This occurs when the action $a = \{[1], [2, 5]\}$ is selected as the starting action. At state x_1, both

activity 3 and activity 4 are started simultaneously. The duration $\check{d}_j$ remains unchanged and is equal to 0. Given the uncertain duration of activity 3, the minimum possible duration of activity 3 in the next state is 2, with a probability of 0.3. Therefore, the minimum $L_{xx'} = 2$. At this point, activity 3 has a 0.3 probability of finishing and a 0.7 probability of not finishing. These two scenarios correspond to transitioning to the next state x_3 and x_2, respectively. For state x_2, $\check{d}_j = 2$. It is clear that the possible durations of activity 3 can only be 3 or 4. Thus, the minimum $L_{xx'} = 3 - 2 = 1$, and the probability $p(d_j > \check{d}_j) = 0.3 + 0.4 = 0.7$. At this point, activity 3 has a finishing probability of $\frac{0.3}{0.7} = 0.43$. Therefore, the transition probability from state x_2 to the next state x_5 is 0.43. Using the SDP approach, the expected project completion time is computed as 5.399. The optimal starting strategy is to begin with activity 1, then select activity 3, followed by the simultaneous start of activities 3 and 4. After waiting for activities 3 and 4 to complete, activity 6 is started.

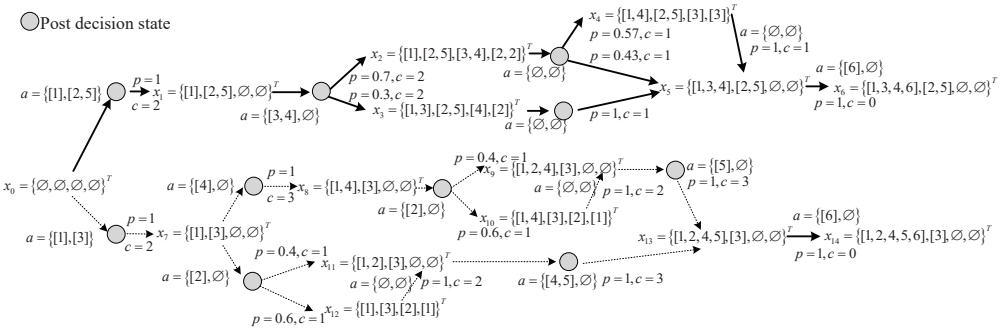


Figure 6. The complete decision process of the example.

5. Suboptimal Policies

As previously mentioned, the SDP method can only solve small-scale problems and becomes intractable for large-scale problems due to the curse of dimensionality [31,33]. To address the challenges posed by practical-scale problems, this section focuses on effective algorithms for obtaining suboptimal policies. The basic idea is to enhance solvability by compressing the action space and state space based only on the current state information. For example, in Figure 6, at state x_0, there are two possible actions to start the process. If action $a = \{[1],[2,5]\}$ is selected to start, then state x_2 and all subsequent actions and states can be ignored.

5.1. Paired Rules

In any given state, the decision problem involves activity selection and sequencing. Theoretically, the optimal activity selection should result in the shortest duration for the remaining project network. However, the presence of uncertain durations and resource constraints complicates the assessment of which decision leads to the optimal result. The paired-rule-based algorithm treats the remaining unstarted and in-progress activities at a given state x as a deterministic network. First, the activity priority is calculated using a selection rule, which results in a unique project structure. Based on this structure, the priority of activities in the waiting set W_x is calculated according to the activity sequencing rule, and the set of start activities, ϖ_x, is identified while the set of activities, φ_x, that will never be implemented is also determined. The paired rules is crucial for obtaining a high-quality starting action $a_x = \{\varpi_x, \varphi_x\}$.

In a deterministic RCPSP-AS, Ref. [19] demonstrated that the paired rules outperforms the single rule. Although existing classical priority rules can be applied to activity selection, they generally perform less effectively than activity-specific priority rules because they do not account for the flexible characteristics of the project network. For RCPSP-AS,

Ref. [25] proposed two activity selection rules, namely, the sum of durations (SOD) and minimum total work content (TWC), to generate the initial solution for their forbidden search algorithm. Specifically, the SOD rule selects the alternative activity with the smallest sum of durations among the current subgraph and all alternative activities in that branch. If nested alternatives exist within the remaining project network, the average duration of these alternative subgraphs is considered. The TWC rule follows the same logic as the SOD rule, with the duration replaced by the work content (i.e., the activity's resource requirement multiplied by its duration). The authors in [19] proposed seven new activity selection rules. Among them, the total time of the subgraph with its links (TTSL) and the total time of the current and following subgraphs and their links (TTXSL) rules are similar to the SOD rule. The main difference lies in how they handle optional activities that have been triggered but are uncertain, along with their dependencies. For the RCPSP-FPS, we selected four effective selection rules, as shown in Table 2, for this study.

Table 2. Four priority rules for activity selection.

Priority Rules	Rank	Remark
SOD	Min	The total duration of the activity with its dependents and the average duration of activities for its nested selections
TWC	Min	The total work content of the activity with its dependents and the average work content of activities for its nested selections
TTSL	Min	The total duration of the activity with its dependents
TTXSL	Min	The total duration of the activity with its dependents and the activities for its nested selections

For the activity sequencing, Ref. [19] tested a total of 37 priority rules from the existing literature and found that the least rank positional weight (LRPW) and the smallest cumulative resource requirement (SCRR) rules perform best for solving the RCPSP-AS with a single rule, while the rank positional weight (RPW) paired with TTSL yields the best result. Considering that the stochastic characteristics of the problem may impact the rule performance, the Latest Finish Time (LFT) rule, which performs well in the stochastic RCPSP, is also selected based on the findings of [16], and the Latest Start Time (LST) rule, which performs well in the stochastic RCMPSP, is also selected based on the study of [46]. All the activity sequencing rules are summarized in Table 3.

Table 3. Five priority rules for activity sequencing.

Priority Rules	Reference	Rank	Formula
(G)RPW	[47]	Max	$d_j + \sum_{i \in IS_j} d_i$
LRPW	[47]	Min	$d_j + \sum_{i \in IS_j} d_i$
SCRR	[48]	Min	$\sum_{k \in R} d_j \cdot r_k + \sum_{k \in R} \sum_{i \in TS_j} d_i \cdot r_k$
LFT	[49]	Min	LF_j
LST	[50]	Min	$LF_j - d_j$

Where:

- LF_j is the latest finish time of activity j calculated by CPM method.
- IS_j is set of immediate successors of activity j.
- TS_j is set of total successors of activity j.

Note that although there are many other rules that can be used for activity selection and sequencing, we tested the selected rules ourselves and they are indeed the best-performing rules. Due to space limitations, we only consider these rules in this study.

5.2. Metaheuristic Algorithms

In any given state, a deterministic RCPSP-FPS can be derived by replacing the uncertain durations of activities with their expected durations. This implies that the decision problem in each state can be solved by existing methods for deterministic RCPSP-FPS. After the best solution of the deterministic RCPSP-FPS is obtained, the start times of each activity can be obtained through a decoding process. The activities that start at the initial time are designated as ϖ_x, and φ_x can be derived based on the values of θ_l. Finally, the starting action for state x is obtained. A solution representation approach is illustrated in Formula (9).

$$l_x = (o_1, o_2, \ldots, o_{|E_x|}, j_1, j_2, \ldots, j_{|U_x|}, \theta_1, \theta_2, \ldots, \theta_{|U_x|}) \tag{9}$$

where o_l represents the selected activity from the optional set corresponding to the l-th unscheduled triggering activity in E_x, and $j_1, j_2, \ldots, j_{|U_x|}$ is the unscheduled activity list, ordered from highest to lowest priority. Note that the structure of the remaining project is determined by the activity sequence $o_1, o_2, \ldots, o_{|E_x|}$, called the selection sequence. The execution order of activities is determined by the sequence $j_1, j_2, \ldots, j_{|U_x|}$, called the activity sequence. The symbol θ_l is used to indicate whether the activity j_l will be implemented ($\theta_l = 1$) or not ($\theta_l = 0$) based on the decisions made and the triggering and dependency logic between activities as outlined in the problem description.

For deterministic RCPSP-FPS, Ref. [6] proposed a GA algorithm.The GA algorithm consists of two main operations: crossover and mutation. The crossover operation is a single-point crossover of the selection sequence and activity sequence of two individuals, respectively. The mutation operation consists of two components: An activity in the selection sequence can be reselected from the corresponding optional activity set with a certain probability m^{α}. Additionally, within the activity sequence, two adjacent activities (where $\theta_l = 1$) can be swapped with a certain probability m^{λ}. For a detailed description of the GA, refer to [6].

The Tabu search (TS) proposed by [25] also achieves good performance. The TS algorithm optimizes the selection sequence and the activity sequence alternately and separately. Specifically, the selection sequence is fixed first, and then a Tabu search is performed on the activity sequence. After two iterations of the activity sequence Tabu search, the current queue and the queue Tabu table are updated according to the greedy criterion. Immediately after fixing the latest activity sequence, a Tabu search optimization is performed on the selection sequence. For the selection sequence, the neighborhood action is to take the activity with a better priority and replace it. The activity priority is calculated according to the selection rule, randomly ignoring the dependent and nested activities. For the activity sequence, the neighborhood action is to randomly select an activity from the critical path and swap it with an adjacent activity. For a detailed description of the TS, refer to [25].

5.3. Rollout Algorithms

From the Bellman Equation (7), it can be seen that SDP obtains the optimal decision by enumerating all states and actions. Obviously, the key to obtaining the best decision is to accurately obtain $v^l(x'|x, a)$. To address this issue, Refs. [51,52] proposed a one-step rollout algorithm (Rollout), which quickly obtains a suboptimal action $\tilde{a}$ by converting the traditional backward dynamic optimization to a one-step forward dynamic programming. Specifically, it is used to estimate the function value $\tilde{v}(x'|x, a)$ of the new state $x' \in X$ after

a one-step state transition from state x. The one-step Rollout algorithm typically uses a base heuristic simulation to estimate the value of the state function. Let $\{w_1, w_2, ..., w_m\}$ represent the sample duration of the remaining activities in state x'. With the sample duration w_l, the simulation of the project is started from the specified state using the paired rules to obtain the function value $\tilde{v}(x', w_l)$. Suboptimal action $\tilde{a}$ can be obtained using Equation (10).

$$\tilde{a} = \arg\min_{a \in A_x} \frac{1}{m} \sum_{l=1}^{m} (c(c'|x, a) + \tilde{v}(x', w_l)) \tag{10}$$

However, as Refs. [53,54] pointed out, the Rollout algorithm is time-consuming for large-scale problems due to the extensive simulation required. They proposed a more efficient rollout algorithm called post-rollout (PRollout). The PRollout algorithm starts the simulation from the post-decision state (x^a) after a decision has been made, showed in Equation (11). The authors demonstrated that this algorithm significantly reduced the computational time compared with the one-step rollout.

$$\tilde{a} = \arg\min_{a \in A_x} \frac{1}{m} \sum_{l=1}^{m} \tilde{v}(x^a, w_l) \tag{11}$$

Due to the fact that the PRollout algorithm also considers all feasible actions in each state, it may still face the problem of excessive computational complexity in large-scale problems. To improve the efficiency of action evaluation, we integrate the common random numbers (CRN) technique into the action simulation evaluation. This is a classic variance reduction method that has been widely used in simulation experiments [55]. The core of CRN is to ensure that actions are assessed under the same conditions by using a consistent random number sequence. When two random variables (such as the results of two different actions) are generated by the same random number sequence, there will be a positive correlation between them. This correlation is expressed as a covariance term in the variance calculation, as shown in Equation (12). By exploiting this correlation, the interference of random noise can be reduced, thereby improving the accuracy and efficiency of the evaluation.

$$\mathrm{Var}(y_1 - y_2) = \mathrm{Var}(y_1) + \mathrm{Var}(y_2) - 2\mathrm{Cov}(y_1, y_2) \tag{12}$$

where y_1 and y_2 denote the outcomes of two alternative actions.

Note that the drawback of CRN lies in the necessity to ensure the consistency of the consumption of the random number sequence across actions. In this work, we use the inverse transform method to generate samples from a discrete triangular distribution, which ensures that each simulation consumes random numbers synchronously. However, some complex distributions cannot be sampled using the inverse transformation method and can only be sampled using the rejection method, such as the normal distribution. In this case, the synchronous consumption of random numbers cannot be guaranteed, thus reducing the effectiveness of the CRN method.

6. Computational Study

To assess the performance of the constructed models and algorithms, we conducted a number of computational experiments based on manually generated instances. The program is coded in C# and runs on a workstation that has an Intel Core CPU 2.1 GHz (Lenovo, Beijing, China) and 64 GB of RAM. The sequencing rules and GA must be combined with a schedule generation scheme (SGS, Ref. [56]) to construct a specific schedule. Combined with a priority rule or a given activity list, SGS build a feasible schedule by stepwise extension of a partial schedule. A partial schedule is composed of the activities that have been scheduled. There are two basic SGSs for the RCPSP, namely, the serial SGS

(SSGS) and the parallel SGS (PSGS). The PSGS is a non-delay schedule [56,57]. In an active schedule, none of the activities can be started earlier without delaying some other activities. Consistent with the existing literature, we combine the heuristics with the parallel SGS, as it has been validated to perform well in the stochastic RCPSP [16,46].The key parameters of the algorithms covered in this paper are summarized in Table 4.

Table 4. Parameter settings for the metaheuristic algorithms.

Algorithm	Parameter	Value	Remark
GA	G^I	125	Generations
	N^I	80	Population size
	m^α	3%	Selection sequencing mutation probability
	m^λ	10%	Activity sequencing mutation probability
TS	L^S	$0.25 \times n^o$	Selecting sequencing tabu length, where n^o is the total number of optional activities
	L^A	$0.5 \times n$	Activity sequencing Tabu length, where n is the total number of activities
	#schedules	10,000	The maximum number of schedules that can be generated
	#iterations	2	Perform activity sequence optimization #iterations times after each selection sequence optimization

6.1. Experimental Data

Ref. [6] generated a total of 4325 instances for RCPSP-PS based on the instance generator ProGen [58]. However, on the one hand, the size of these instances contains at least 30 non-dummy activities, which cannot be solved by the exact SDP method. On the other hand, the triggering and dependent activities do not strictly follow the precedence relations as mentioned in the problem description. Therefore, instead of directly using these instances, we refer to their parameter settings to generate a set of new small-scale ($n = \{15, 20\}$) and large-scale instances ($n = \{30, 60\}$) based on the instance generator RanGen2 [59]. The project scale is categorized based on whether the instance can be solved using SDP.

We first use the instance generator RanGen2 to generate standard RCPSP instances with $n = \{15, 20, 30, 60\}$. The parameter settings are similar to those of RG30 Set 1 [59]. Specifically, we set the network size $I_1 = \{15, 20, 30, 60\}$, serial or parallel indicator $I_2 = \{0.15, 0.25, \ldots, 0.75\}$, resource constraint coefficient $RC = 0.4$, and resource usage $RU = 3$. For each parameter combination, we generate 25 instances. Additionally, we consider two levels of network flexibility (high and low) for different instance scales, as shown in Table 5, and three types of activity duration distribution skewness (i.e., left-skewed, symmetrical, right-skewed), as shown in Table 6. In Table 5, $|E|$ represents the number of choices, $|O|$ denotes the number of alternative activities per choice, $|D|$ indicates the number of dependency sets, and $|\beta|$ specifies the number of dependent activities. In Table 6, $\bar{d}$ represents the expected duration, which is equal to the original activity duration. To reasonably set the parameters h, l, and q, we assume that the duration follows a discrete triangular distribution only when the original duration is greater than or equal to 3. The three parameters of the duration distribution are manually set based on the mean and skewness of the duration. Specifically, we first set $h \approx \lfloor \bar{d}/2 \rfloor$, where $\lfloor \cdot \rfloor$ represents the floor function. Then, under the condition that $h + l + q = 3\bar{d}$, let the symmetric distribution satisfy $l - h = q - l$, let the left-skewed distribution satisfy $l - h \geq q - l$, and let the right-skewed distribution satisfy $l - h \leq q - l$. Finally, for each scale, a total of 1050 ($25 \times 7 \times 2 \times 3$) instances are generated.

Table 5. Parameter settings of project network flexibility.

Parameter	n = 15		n = 20		n = 30		n = 60	
	Low	High	Low	High	Low	High	Low	High
($\|E\|$, $\|O\|$)	(1, 2)	(2, 3)	(1, 2)	(3, 3)	(1, 2)	(3, 3)	(2, 2)	(6, 3)
($\|D\|$, $\|\beta\|$)	(1, 1)	(2, 1)	(1, 1)	(3, 1)	(1, 1)	(3, 1)	(2, 1)	(6, 1)

Table 6. Parameter settings for the skewness of discrete triangular distribution durations.

$\bar{d}$	Left			Symmetric			Right		
	h	l	q	h	l	q	h	l	q
3	1	3	5	1	3	5	1	2	6
4	2	4	6	2	4	6	2	3	7
5	2	6	7	2	5	8	3	4	8
6	3	7	8	3	6	9	4	5	9
7	3	8	10	3	7	11	5	6	10
8	4	9	11	4	8	12	6	7	11
9	4	11	12	4	9	14	6	7	14
10	5	12	13	5	10	15	7	8	15

To conduct a more comprehensive analysis of the impact of flexibility on algorithm performance, in addition to the flexibility parameter, the influence of the degree of nested (denoted as *nested*) and dependent (denoted as *dependent*) activities is also considered. The parameter *nested* is defined as the ratio of all nested activities to the total number of optional activities, while the parameter *dependent* is defined as the ratio of all dependent activities to the total number of optional activities.

For small-scale instances, each method solves the problem based on states generated by the MDP. However, for large-scale instances, the enormous state space makes direct solving infeasible. Instead, states are generated through simulation. Each large-scale instance is simulated 1000 times, and the average result is used for analysis. Although more simulation times can improve the accuracy of the results, it will also bring greater computational costs. Ref. [9] tested the RCPSP with uncertain activity durations and verified that the average deviation of results between 25,000 simulations and 1000 simulations is less than 1% for the J120 dataset. Therefore, we believe that 1000 simulations is enough. The performance of each algorithm is evaluated using the indicator Dev.best(%), as shown in Formula (13). Dev.best(%) denotes the deviation of the result *Cur* with respect to the best result *Best* for the same instance.

$$\text{Dev.best}(\%) = \frac{Cur - Best}{Best} \times 100\% \tag{13}$$

6.2. Parameter Turning for Rollout Algorithms

The Rollout and PRollout algorithms require multiple simulations to evaluate the function values of states, and their optimization performance is influenced by the number of simulations. Given the relatively long computation time of the rollout algorithms, a part of the instances are selected for efficient experimentation. The test set includes all the small-scale instances and 1% randomly selected large-scale instances. When analyzing the impact of the number of simulations, both Rollout and PRollout employ the best-performing paired rules as the base heuristic (i.e., "TWC-LFT", as shown in the following section). Considering the inherent randomness in the simulation process, each algorithm is tested on the test set using 10 different random seeds to generate multiple results. The average result is taken as the final result. We refer to Rollout' and PRollout' as to the algorithms combined with the CRN technique. The results are shown in Figure 7.

It can be seen that with the increase in the number of simulations, the optimization performance of Rollout and PRollout improves progressively. The CRN technique not only enhances the optimization effect of the rollout algorithms but also accelerates its

convergence. Specifically, once the number of simulations reaches a certain threshold, the optimization effect stabilizes. In addition, after applying CRN, the optimization performance of PRollout is very close to that of Rollout. Meanwhile, it can be observed that the optimization effect of 100 simulations after integrating CRN technique has exceeded that of 250 independent simulations. This observation suggests that CRN improves the accuracy of action selection by reducing the variance in the response difference between alternative actions, thereby reducing the number of simulations required. Therefore, in subsequent experiments, all Rollout and PRollout algorithms will employ CRN, with the number of simulations set to 200. When the simulation is set to 200, the experimental results show that CRN provides an improvement of 0.18% for Rollout and 0.24% for PRollout.

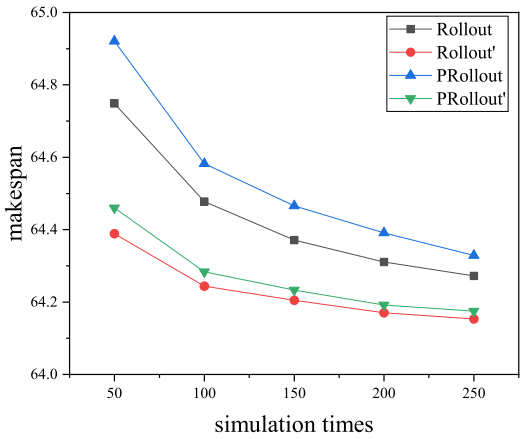


Figure 7. The impact of simulation times on rollout algorithms.

6.3. Result Analysis

Tables 7 and 8 present the results of solving small-scale and large-scale instances using various paired rules, respectively. Note that SPR* means combining with all sequencing rules from Table 3, while EPR* means combining with all selection rules from Table 2. The results demonstrate that paired rules outperform single sequencing rules. TWC is the best selection rule in terms of overall performance. LST and LFT, as activity sequencing rules, perform best for small-scale and large-scale instances, respectively. The performance of LFT is consistent with the findings of [16] in the stochastic RCPSP environment. The overall performance of TWC-SPR* exceeds that of EPR*-LFT, suggesting that incorporating multiple sequencing rules introduces greater diversity, which contributes to better results. PR** refers to the combined paired rules formed by different selection and sequencing rules, resulting in 5 × 6 = 30 distinct rule combinations. Clearly, PR** achieves the best performance because it selects the best result from all paired rule combinations. For large-scale instances, TWC-LFT is the best-performing paired rule, with an average result gap of only 1.61% compared to the best PR**.

Table 7. Comparison of priority rules based on average Dev.best(%) in small-scale instances.

Rule	$n = 15$						$n = 20$					
	—[a]	SOD	TTSL	TTXSL	TWC	EPR*	—	SOD	TTSL	TTXSL	TWC	EPR*
LST	18.14	3.59	3.67	3.69	**3.33**	**2.29**	24.50	3.28	3.64	3.42	**2.71**	**1.72**
LFT	17.79	3.77	3.89	3.86	3.48	2.51	23.73	3.75	4.06	3.87	3.34	2.32
RPW	21.23	4.34	4.37	4.42	4.32	2.98	25.70	5.45	5.84	5.59	4.79	3.89
LRPW	15.53	9.21	9.44	9.22	8.98	7.96	16.26	11.35	11.49	11.45	10.94	9.49
SCRR	17.53	10.61	10.80	10.60	10.56	9.45	18.00	12.84	12.97	12.95	12.57	11.09
SPR*		1.56	1.79	1.67	**1.46**	0.00		1.75	2.08	1.91	**1.22**	0.00

[a] the selection rule is the same as the sequencing rule.

Table 8. Comparison of priority rules based on average Dev.best(%) in large-scale instances.

Rule	$n = 30$						$n = 60$					
	–	SOD	TTSL	TTXSL	TWC	EPR*	–	SOD	TTSL	TTXSL	TWC	EPR*
LFT	10.77	2.20	2.32	2.27	**2.08**	1.36	9.56	1.47	1.62	1.49	**1.14**	**0.65**
LST	11.05	2.22	2.41	2.34	2.13	**1.25**	10.00	1.58	1.75	1.64	1.31	0.75
RPW	15.13	5.75	5.66	5.71	5.43	4.38	16.23	6.62	6.76	6.65	6.30	5.68
LRPW	13.62	10.42	10.52	10.72	10.67	9.40	15.50	12.37	12.32	12.35	11.98	11.18
SCRR	17.10	14.07	14.01	14.18	14.12	12.98	19.89	15.89	16.08	15.99	15.66	14.92
SPR*		0.98	1.11	1.05	**0.86**	0.00		0.80	0.96	0.83	**0.54**	0.00

Table 9 illustrates the impact of paired rules for rollout-based algorithms. These rules perform well in rule-based algorithms. The computation time for PR** is theoretically 30 times that of the TWC-LFT rule. After preliminary testing, the time required to solve a single instance with the combined rules exceeds 1 h for instances ($N = 60$). To improve experimental efficiency, for large-scale instances, the instances ($N = 60$) are not considered, and 1% of the newly generated instances ($N = 30$) are randomly selected as test instances. From the results, it can be observed that for Rollout' and PRollout', the average gap between TWC-LFT and PR** is 0.54% and 0.55%, respectively. This suggests that the combined paired rules do not result in significant performance improvements but instead lead to an exponential increase in computation time. Therefore, we recommend using the TWC-LFT paired rule for simulations.The results also show that the performance of the Rollout' and PRollout' algorithms remains generally consistent across different combinations of paired rules.

Table 9. Comparison of rollout algorithms using different paired rules to evaluate actions.

Paired Rule		n = 15		n = 20		n = 30	
		Rollout'	PRollout'	Rollout'	PRollout'	Rollout'	PRollout'
TWC-LFT	Dev.best(%)	0.46	0.46	0.65	0.68	0.95	0.99
	RunTime (s)	0.15	0.09	1.01	0.65	45.68	27.50
TWC-SPR*	Dev.best(%)	0.14	0.16	0.16	0.16	0.18	0.33
	RunTime (s)	0.83	0.51	5.88	3.96	137.70	73.64
EPR*-LFT	Dev.best(%)	0.45	0.44	44.93	0.65	0.91	0.93
	RunTime (s)	0.59	0.36	4.25	2.75	125.08	71.97
PR**	Dev.best(%)	0.15	0.14	0.14	0.18	0.14	0.17
	RunTime (s)	3.64	2.28	24.92	16.23	958.96	592.01

Tables 10 and 11 show the deviation of each algorithm from the best result under different scales. The results for PR** are obtained by the rule-based algorithm. For small-scale instances, the SDP method provides the optimal result, so the deviation is measured relative to the optimal result. The state space of the instances grows rapidly as the number of nodes increases. Due to the memory size limitations of the machine, the maximum solvable space is restricted. Experiments show that when solving large-scale instances using SDP, the memory usage exceeds 128 GB, which prevents it from solving large-scale instances. The final experiments demonstrate that both Rollout' and PRollout' perform well, with only a 0.02% gap between PRollout' and Rollout' in large-scale instances. As shown in Figure 4, Rollout starts the simulation from the next state, which essentially means using the heuristic policy to generate a full path to the last activity. In contrast, PRollout generates a full path starting from the post-state. Although PRollout starts earlier, it ignores more random factors, which leads to a poorer optimization effect. Ref. [54] points out that the difference in the performance of the Rollout and PRollout algorithms depends on the consistency of the paths generated by the two. When there is only one next state, the results of PRollout and Rollout are equal. As the number of next states increases, this difference

becomes larger. Since the final action is chosen based on the expected rewards of all paths, the gap between the two algorithms is not large. In addition, as illustrated in Table 10, the performance gap between rollout algorithms and SDP went from 1.3 to 2.2 when the project size increased from 15 to 20. This result is intuitive, as it becomes more difficult for rollout algorithms to obtain an optimal solution for larger problems due to the rule-based action evaluation. Although we cannot obtain the specific growth trend of the rollout algorithm performance gap with the problem size, we know that it will inevitably increase with the increase in the problem size.

Table 10. Comparison of methods for small-scale instances based on average Dev.best(%).

Method	*n*		Flexibility		Skewness			Total
	15	20	Low	High	Left	Symm	Right	
PR**	2.98	4.45	4.06	3.38	3.75	3.66	3.75	3.72
TS	1.52	2.47	2.40	1.58	1.96	2.02	2.00	1.99
GA	1.43	2.16	2.28	1.32	1.79	1.80	1.80	1.80
PRollout'	1.30	2.23	1.99	1.54	1.78	1.75	1.75	1.76
Rollout'	1.30	2.21	1.98	1.53	1.79	1.73	1.73	1.75
SDP	0	0	0	0	0	0	0	0.00

Table 11. Comparison of methods for large-scale instances based on average Dev.best(%).

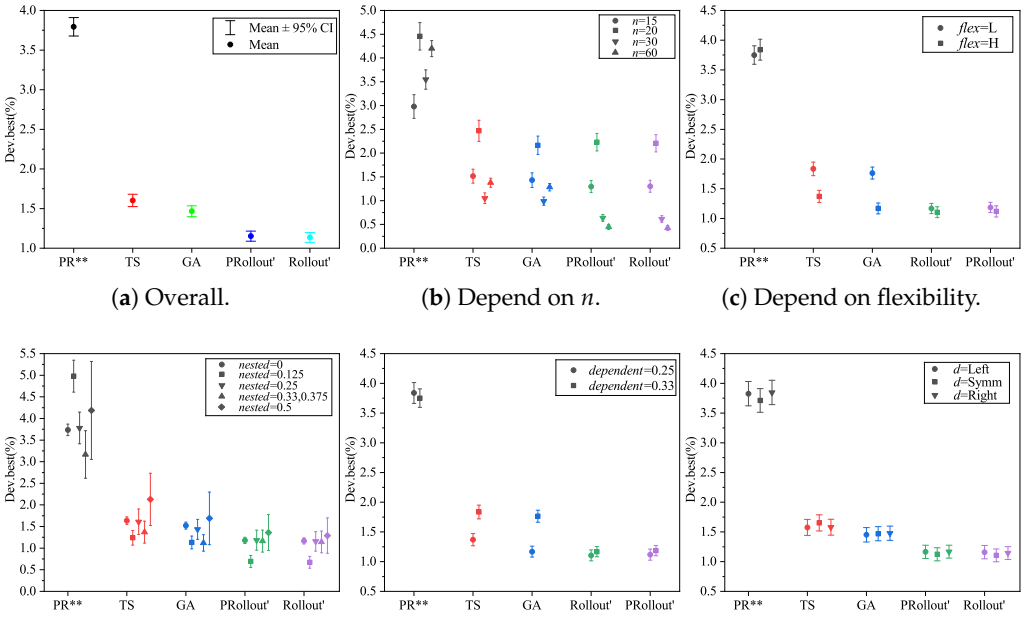
Method	*n*		Flexibility		Skewness			Total
	30	60	Low	High	Left	Symm	Right	
PR**	3.55	4.20	3.44	4.30	3.90	3.76	3.94	3.87
TS	1.05	1.38	1.27	1.16	1.19	1.28	1.16	1.21
GA	0.99	1.28	1.25	1.02	1.11	1.14	1.16	1.14
PRollout'	0.64	0.45	0.38	0.70	0.55	0.50	0.58	0.54
Rollout'	0.61	0.42	0.36	0.68	0.52	0.48	0.55	0.52

TS and GA have an average deviation gap of 0.33% and 0.47%, respectively, compared to the Rollout algorithm across different scales. Since TS and GA use expected durations for evaluation, they struggle to account for the impact of the randomness in durations on the scheduling method. Although PR** considers multiple rule combinations, its performance is the worst due to the limited solution space it covers.

In order to analyze the statistical significance of the difference between the results of different methods, two-tailed Wilcoxon signed-rank test (2-tailed Wilcoxon signed-rank test) was applied to the algorithm results and the obtained p-value results are shown in Table 12. If the p-value between two algorithms is less than 0.05, it means that there is a significant difference between the two algorithms, otherwise there is no significant difference. Note that the reason why we did not use the ANOVA method or t-test here is because the data do not follow a normal distribution.The results of the test showed significant differences between each method. Note that although the Rollout' method and PRollout' method tests are significant, the actual differences are small, as shown in Figure 8a.

Table 12. The p-values of 2-tailed Wilcoxon signed-rank test for different methods.

	PR**	TS	GA	Rollout'	PRollout'
PR**	0.00	<0.001	0.000	0.000	0.000
TS		0.000	<0.001	0.000	<0.001
GA			0.000	0.000	<0.001
Rollout'				0.000	<0.001
PRollout'					0.000



(**a**) Overall. (**b**) Depend on n. (**c**) Depend on flexibility.

(**d**) Depend on *nested*. (**e**) Depend on *dependent*. (**f**) Depend on skewness.

Figure 8. Analysis of the impact of different problem features on algorithm performance based on 95% confidence intervals (CI).

In terms of computational efficiency, Table 13 shows the average computation time (in seconds) for each method. The results for PR** are obtained by the rule-based algorithm. The computation time for small-scale instances ($N = 15, 20$) refers to the average time required to solve an instance based on the MDP framework. It is seen that the state space size of the approximate algorithms is much smaller than those of SDP. The computation time for large-scale instances ($N = 30, 60$) refers to the average time required to complete 1000 simulations. Although PR** performed the worst, its computation time is negligible. The metaheuristic algorithms and the Rollout algorithms experience a significant increase in solution time as the project size increases. In addition, although the performance of PRollout' is not as good as that of Rollout', there is only a 0.02% gap. More importantly, the PRollout' algorithm reduces the solution time by 44.37% compared to Rollout'. This shows that PRollout' can better balance the optimization effect and solution time and is therefore a better choice for practical applications.

Table 13. Comparison of the average computation time in seconds (state space size ($\times 10^3$)) of each method.

Instance Scale	PR**	TS	GA	Rollout'	PRollout'	SDP
15	0.01 (0.35)	2.55 (0.57)	2.51 (0.5)	0.15 (0.49)	0.09 (0.48)	0.44 (57.28)
20	0.02 (1.01)	14.00 (1.9)	12.40 (1.82)	1.01 (1.76)	0.65 (1.77)	6.51 (534.35)
30	0.02	40.50	35.96	44.76	26.32	-
60	0.08	256	240	240.94	132.53	-

We further analyzed the impact of the problem characteristics on the algorithm performance. Figure 8 shows the 95% confidence intervals for each algorithm at each factor level. Figure 8b also shows the impact of the instance size on the algorithm performance. For small-scale instances ($N = 15, 20$), the performance of the algorithms is compared with that of the optimal solution. As the instance size increases, the solution complexity rises, and the performance of each algorithm deteriorates. For large-scale instances ($N = 30, 60$), rollout algorithms outperform metaheuristics as the scale rises. Figure 8c–e illustrate the impact

of the flex–feature algorithm. High flexibility helps improve the performance of both metaheuristics and rollout algorithms. With the same number of activities, higher flexibility means that fewer activities need to be implemented. Fewer activities result in a smaller solution space for queue optimization. The rollout algorithms show greater stability in response to the variation of flexibility. As the number of nested and dependent activities increases, both rollout algorithms and metaheuristics perform worse. This is due to the increased complexity of flexibility. Note that the rollout algorithms and the metaheuristic algorithms perform better when the network has a small number of nested activities ($nested \leqslant 0.125$) than when there are no nested activities ($nested = 0$). This is because the moderate nesting of optional activities reduces the number of possible structural combinations and reduces the flexibility complexity. Figure 8f shows that different skewness of triangular distribution durations do not have a significant impact on the algorithm performance. In theory, a right-skewed distribution means that the mean is greater than the median, and the mean overestimates the typical value of most data points, while a left-skewed distribution means that the mean is less than the median, and the mean underestimates the typical value of most data points. There are two main reasons why skewness has no significant effect on algorithm performance. One possibility is that the skewness is not large enough. Another possible reason is that all approximate methods overestimate or underestimate the duration at the same time, thus offsetting the impact of skewness on decision making.

6.4. Analysis of the Impact of Duration Distribution and Variance on the Algorithm

To better investigate how different distributions and variance affect the algorithm's performance, we also consider cases where the project duration follows a uniform distribution or a Beta distribution for large-scale instances, as shown in Table 14. These distributions align with the probability distribution types and parameters outlined in [16]. As shown in Table 14, U1 and B1 exhibit relatively low variance, while U2 and B2 demonstrate significantly higher variance.

Table 14. Four continuous distributions for activity durations.

Distribution Type	Name	Range	Variance
Uniform distribution	U1	$U(\bar{d}-\sqrt{\bar{d}}, \bar{d}+\sqrt{\bar{d}})$	$\bar{d}/3$
	U2	$U(0, 2\bar{d})$	$\bar{d}^2/3$
Beta distribution	B1	$B\left(\frac{\bar{d}}{2}, 2\bar{d}, \frac{\bar{d}}{2}-\frac{1}{3}, \bar{d}-\frac{2}{3}\right)$	$\bar{d}/3$
	B2	$B\left(\frac{\bar{d}}{2}, 2\bar{d}, \frac{1}{6}, \frac{1}{3}\right)$	$\bar{d}^2/3$

Note that the one-step Rollout algorithm cannot be applied to the problem instance with continuous distributed durations because it corresponds to a continuous state space problem. The PRollout' algorithm can be used to solve this type of problem because it does not require state transition. Table 15 presents the average Dev.best(%) of each algorithm, and Table 16 provides the results of statistical tests.

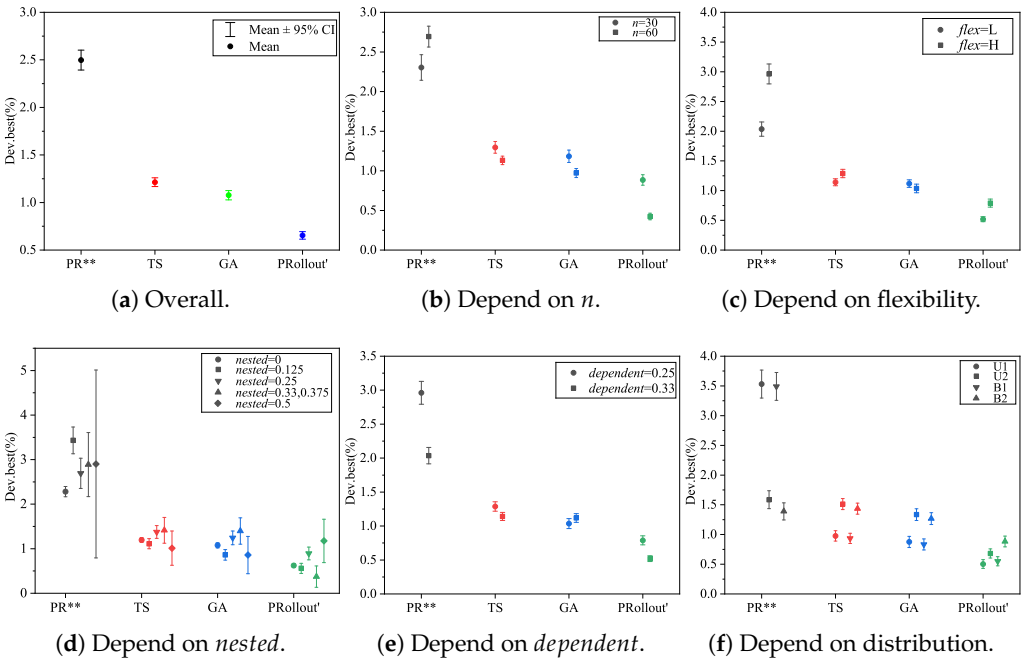
Table 15. Comparison of methods for large-scale instances with continuous duration distribution based on average Dev.best(%).

Method	*n*		Flexibility		Distribution				Total
	30	60	Low	High	U1	U2	B1	B2	
PR**	2.30	2.69	2.03	2.96	3.53	1.58	3.49	1.39	2.50
TS	1.30	1.13	1.14	1.29	0.98	1.51	0.93	1.43	1.21
GA	1.18	0.97	1.12	1.04	0.88	1.33	0.83	1.27	1.08
PRollout'	0.89	0.42	0.52	0.79	0.50	0.68	0.55	0.88	0.65

Table 16. The *p*-values of 2-tailed Wilcoxon signed-rank test for different methods in large-scale instances with continuous duration distribution.

	PR**	**TS**	**GA**	**PRollout'**
PR**	0.00	<0.001	<0.001	<0.001
TS		0.000	0.528	<0.001
GA			0.000	<0.001
PRollout'				0.000

It can be observed that the performance of each method in the continuous duration problem is generally consistent with that in the discrete triangular distribution problem. Specifically, the PRollout' algorithm still outperforms others. Among all algorithms, no significant difference exists between TS and GA, while other pairwise comparisons show statistically significant differences. This is because the variances of uniform and Beta distributions are larger than those of triangular distributions, and the increased randomness masks the differences between metaheuristics. Figure 9 displays the 95% confidence intervals for each algorithm at each factor level, and Figure 9f illustrates the impact of variance on algorithm performance when durations follow uniform or Beta distributions. It is evident that higher variance (i.e., greater randomness) degrades the performance of both the PRollout' algorithm and metaheuristics.



(**a**) Overall. (**b**) Depend on n. (**c**) Depend on flexibility.

(**d**) Depend on *nested*. (**e**) Depend on *dependent*. (**f**) Depend on distribution.

Figure 9. Analysis of the impact of different problem features on algorithm performance based on 95% CI in large-scale instances with continuous duration distribution.

7. Summary and Conclusions

This paper investigates the RCPSP with flexible structures and uncertain activity durations, modeled as an MDP with the optimal policy obtained using SDP. Due to the curse of dimensionality in large-scale problems, several approximate methods are proposed to obtain suboptimal policies.Through experiments with different combinations of priority rules, better-performing rules are identified and applied to rollout algorithms. Additionally, a CRN technique is incorporated in the rollout algorithms. The experimental results show that the CRN technique enhances not only the simulation efficiency but also the accuracy of action selection. Furthermore, experimental results reveal that the Rollout algorithm significantly outperforms both the priority rules and metaheuristic algorithms.

The PRollout' algorithm achieves a close performance (with only a 0.02% gap) to the Rollout' algorithm while reducing computation time by 44.37%. It is also observed that an increase in flexibility (larger *nested* or larger *dependent*) and variability of activity duration lead to a decrease in algorithm performance. Our research findings provide valuable guidance for practical applications. Specifically, our study shows that although the metaheuristic algorithm performs better in deterministic problem environments, the Rollout algorithm is significantly better than it in stochastic environments, and this advantage becomes more significant as the problem size increases.

The limitation of the proposed PRollout' algorithm lies in the solution quality and computational efficiency for large-scale problem instances. On the one hand, the solution quality of the proposed PRollout' algorithm depends on rule-based simulation, which will deteriorate as the problem size increases. On the other hand, the PRollout' algorithm enumerates all feasible actions in each state. When the problem size increases, the action size will also become very large, and it may even be impossible to fully enumerate. Although the CRN technique significantly improves its computational efficiency, it still requires a considerable amount of time. In summary, our approach is mainly applicable to medium-sized problems, which covers most practical problems.

In the future, this research work can be expanded in the following three directions. First, a more efficient simulation strategy (e.g., non-uniform allocation strategy) can be adopted to improve the efficiency of the simulation evaluation. In the field of simulation optimization, many scholars have conducted research on the non-uniform allocation strategies and verified their effectiveness. For instance, Ref. [60] pointed out that the efficiency and accuracy of the alternative solution selection can be improved by combining Bayesian models with adaptive stopping rules. Second, in terms of random factors, this paper only considers random activity durations. In the future, we can further consider random resource availability or random project network (e.g., random rework). Third, we can use the latest artificial intelligence techniques (such as reinforcement learning and deep learning) to obtain high-quality policies and increase the scale of solvable problems.

Author Contributions: Conceptualization, X.W. and Q.C.; methodology, C.Y. and X.W.; investigation, C.Y. and X.W.; data curation, C.Y.; formal analysis, C.Y.; visualization, C.Y.; writing—original draft, C.Y.; writing—review and editing, X.W. and Q.C.; supervision, X.W. and Q.C.; funding acquisition X.W. and Q.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China [Grants 71972053, 61973089 and 51505090] and Guangdong Basic and Applied Basic Research Foundation [Grants 2024A1515010906 and 2022A1515011165].

Data Availability Statement: The generated problem instance data are available from the corresponding author upon reasonable request.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

RCPSP	Resource-constrained project scheduling problem
RCPSP-FPS	Resource-constrained project scheduling problem with a flexible project structure
RCPSP-AC	Resource-constrained project scheduling problem with alternative activity chains
RCPSP-AS	Resource-constrained project scheduling problem with alternative subgraphs
CRN	Common random numbers
GA	Genetic algorithm
MDP	Markov decision process
LST	Latest Start Time

LFT	Latest Finish Time
SCRR	Smallest cumulative resource requirement
LRPW	Least rank positional weight
(G)RPW	Rank positional weight
SOD	Sum Of durations
TWC	Total work content
TTSL	Total processing time of the subgraph with its links
TTXSL	Total time of the current and following subgraphs and their links
CPM	Critical Path Method

References

1. Hartmann, S.; Briskorn, D. A survey of variants and extensions of the resource-constrained project scheduling problem. *Eur. J. Oper. Res.* **2010**, *207*, 1–14. . [CrossRef]
2. Hartmann, S.; Briskorn, D. An updated survey of variants and extensions of the resource-constrained project scheduling problem. *Eur. J. Oper. Res.* **2022**, *297*, 1–14. . [CrossRef]
3. Blazewicz, J.; Lenstra, J.; Kan, A. Scheduling subject to resource constraints: Classification and complexity. *Discret. Appl. Math.* **1983**, *5*, 11–24. [CrossRef]
4. Herroelen, W.; Leus, R. Project scheduling under uncertainty: Survey and research potentials. *Eur. J. Oper. Res.* **2005**, *165*, 289–306. [CrossRef]
5. Hazır, Ö.; Ulusoy, G. A classification and review of approaches and methods for modeling uncertainty in projects. *Int. J. Prod. Econ.* **2020**, *223*, 107522. [CrossRef]
6. Kellenbrink, C.; Helber, S. Scheduling resource-constrained projects with a flexible project structure. *Eur. J. Oper. Res.* **2015**, *246*, 379–391. [CrossRef]
7. Ding, H.; Zhuang, C.; Liu, J. Extensions of the resource-constrained project scheduling problem. *Autom. Constr.* **2023**, *153*, 104958. [CrossRef]
8. Ballestín, F. When it is worthwhile to work with the stochastic RCPSP? *J. Sched.* **2007**, *10*, 153–166. [CrossRef]
9. Ballestín, F.; Leus, R. Resource-Constrained Project Scheduling for Timely Project Completion with Stochastic Activity Durations. *Prod. Oper. Manag.* **2009**, *18*, 459–474. [CrossRef]
10. Ashtiani, B.; Leus, R.; Aryanezhad, M.B. New competitive results for the stochastic resource-constrained project scheduling problem: Exploring the benefits of pre-processing. *J. Sched.* **2011**, *14*, 157–171. [CrossRef]
11. Rostami, S.; Creemers, S.; Leus, R. New strategies for stochastic resource-constrained project scheduling. *J. Sched.* **2018**, *21*, 349–365. [CrossRef]
12. Creemers, S.; Leus, R.; Lambrecht, M. Scheduling Markovian PERT networks to maximize the net present value. *Oper. Res. Lett.* **2010**, *38*, 51–56. [CrossRef]
13. Creemers, S. Minimizing the expected makespan of a project with stochastic activity durations under resource constraints. *J. Sched.* **2015**, *18*, 263–273. [CrossRef]
14. Creemers, S. Maximizing the expected net present value of a project with phase-type distributed activity durations: An efficient globally optimal solution procedure. *Eur. J. Oper. Res.* **2018**, *267*, 16–22. [CrossRef]
15. Creemers, S. The preemptive stochastic resource-constrained project scheduling problem. *Eur. J. Oper. Res.* **2019**, *277*, 238–247. [CrossRef]
16. Chen, Z.; Demeulemeester, E.; Bai, S.; Guo, Y. Efficient priority rules for the stochastic resource-constrained project scheduling problem. *Eur. J. Oper. Res.* **2018**, *270*, 957–967. [CrossRef]
17. Kerkhove, L.P.; Vanhoucke, M. Optimised scheduling for weather sensitive offshore construction projects. *Omega* **2017**, *66*, 58–78. [CrossRef]
18. Trietsch, D.; Mazmanyan, L.; Gevorgyan, L.; Baker, K.R. Modeling activity times by the Parkinson distribution with a lognormal core: Theory and validation. *Eur. J. Oper. Res.* **2012**, *216*, 386–396. [CrossRef]
19. Nekoueian, R.; Servranckx, T.; Vanhoucke, M. Constructive heuristics for selecting and scheduling alternative subgraphs in resource-constrained projects. *Comput. Ind. Eng.* **2023**, *182*, 109399. [CrossRef]
20. van der Beek, T.; van Essen, J.; Pruyn, J.; Aardal, K. Exact solution methods for the Resource Constrained Project Scheduling Problem with a flexible Project Structure. *Eur. J. Oper. Res.* **2025**, *322*, 56–69. [CrossRef]
21. Tao, S.; Dong, Z.S. Scheduling resource-constrained project problem with alternative activity chains. *Comput. Ind. Eng.* **2017**, *114*, 288–296. [CrossRef]
22. Tao, S.; Dong, Z.S. Multi-mode resource-constrained project scheduling problem with alternative project structures. *Comput. Ind. Eng.* **2018**, *125*, 333–347. [CrossRef]

23. Tao, S.; Wu, C.; Sheng, Z.; Wang, X. Stochastic Project Scheduling with Hierarchical Alternatives. *Appl. Math. Model.* **2018**, *58*, 181–202. [CrossRef]
24. Hauder, V.A.; Beham, A.; Raggl, S.; Parragh, S.N.; Affenzeller, M. Resource-constrained multi-project scheduling with activity and time flexibility. *Comput. Ind. Eng.* **2020**, *150*, 106857. [CrossRef]
25. Servranckx, T.; Vanhoucke, M. A tabu search procedure for the resource-constrained project scheduling problem with alternative subgraphs. *Eur. J. Oper. Res.* **2019**, *273*, 841–860. [CrossRef]
26. Servranckx, T.; Coelho, J.; Vanhoucke, M. A genetic algorithm for the Resource-Constrained Project Scheduling Problem with Alternative Subgraphs using a boolean satisfiability solver. *Eur. J. Oper. Res.* **2024**, *316*, 815–827. [CrossRef]
27. Servranckx, T.; Coelho, J.; Vanhoucke, M. Various extensions in resource-constrained project scheduling with alternative subgraphs. *Int. J. Prod. Res.* **2022**, *60*, 3501–3520. [CrossRef]
28. Kulkarni, V.G.; Adlakha, V.G. Markov and Markov-Regenerative pert Networks. *Oper. Res.* **1986**, *34*, 769–781. [CrossRef]
29. Sobel, M.J.; Szmerekovsky, J.G.; Tilson, V. Scheduling projects with stochastic activity duration to maximize expected net present value. *Eur. J. Oper. Res.* **2009**, *198*, 697–705. [CrossRef]
30. Hermans, B.; Leus, R. Scheduling Markovian PERT networks to maximize the net present value: New results. *Oper. Res. Lett.* **2018**, *46*, 240–244. [CrossRef]
31. Wang, X.; Chen, Q.; Mao, N.; Chen, X.; Li, Z. Proactive approach for stochastic RCMPSP based on multi-priority rule combinations. *Int. J. Prod. Res.* **2015**, *53*, 1098–1110. [CrossRef]
32. Luh, P.B.; Liu, F.; Moser, B. Scheduling of design projects with uncertain number of iterations. *Eur. J. Oper. Res.* **1999**, *113*, 575–592. [CrossRef]
33. Wang, X.; Leus, R.; Creemers, S.; Chen, Q.; Mao, N. A CTMDP-Based Exact Method for RCPSP with Uncertain Activity Durations and Rework. In Proceedings of the Operations Research Proceedings, Berlin, Germany, 6–8 September 2017; Kliewer, N., Ehmke, J.F., Borndörfer, R., Eds.; Springer: Cham, Switzerland, 2018; pp. 559–565.
34. Satic, U.; Jacko, P.; Kirkbirde, C. Performance evaluation of scheduling policies for the dynamic and stochastic resource-constrained multi-project scheduling problem. *Int. J. Prod. Res.* **2022**, *60*, 1411–1423. [CrossRef]
35. Dreyfus, S.E. Richard Bellman on the Birth of Dynamic Programming. *Oper. Res.* **2002**, *50*, 48–51. [CrossRef]
36. Powell, W.B. *Approximate Dynamic Programming—Solving the Curses of Dimensionality*; Wiley Series in Probability and Statistics; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 2007.
37. Satic, U.; Jacko, P.; Kirkbride, C. A simulation-based approximate dynamic programming approach to dynamic and stochastic resource-constrained multi-project scheduling problem. *Eur. J. Oper. Res.* **2023**, *315*, 454–469. [CrossRef]
38. Igelmund, G.; Radermacher, F.J. Preselective strategies for the optimization of stochastic project networks under resource constraints. *Networks* **1983**, *13*, 1–28. . [CrossRef]
39. Rahman, H.F.; Chakrabortty, R.K.; Ryan, M.J. Scheduling project with stochastic durations and time-varying resource requests: A metaheuristic approach. *Comput. Ind. Eng.* **2021**, *157*, 107363. [CrossRef]
40. Zhang, H.; Demeulemeester, E.; Li, L.; Bai, S. Surrogate-assisted cooperative learning genetic programming for the resource-constrained project scheduling problem with stochastic activity durations and transfer times. *Comput. Oper. Res.* **2025**, *173*, 106816. [CrossRef]
41. Browning, T.R.; Yassine, A.A. Managing a Portfolio of Product Development Projects under Resource Constraints. *Decis. Sci.* **2016**, *47*, 333–372. [CrossRef]
42. Yassine, A.A.; Mostafa, O.; Browning, T.R. Scheduling multiple, resource-constrained, iterative, product development projects with genetic algorithms. *Comput. Ind. Eng.* **2017**, *107*, 39–56. [CrossRef]
43. Li, H.; Womer, N.K. Solving stochastic resource-constrained project scheduling problems by closed-loop approximate dynamic programming. *Eur. J. Oper. Res.* **2015**, *246*, 20–33. [CrossRef]
44. Clark, C.E. Letter to the Editor—The PERT Model for the Distribution of an Activity Time. *Oper. Res.* **1962**, *10*, 405–406. [CrossRef]
45. Puterman, M.L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*; John Wiley & Sons: Hoboken, NJ, USA, 2014.
46. Wang, Y.; He, Z.; Kerkhove, L.P.; Vanhoucke, M. On the performance of priority rules for the stochastic resource constrained multi-project scheduling problem. *Comput. Ind. Eng.* **2017**, *114*, 223–234. [CrossRef]
47. Olaguíbel, R.A.V.; Goerlich, J.T. Chapter 5—Heuristic Algorithms for Resource-Constrained Project Scheduling: A Review and an Empirical Analysis. In *Project Scheduling*; Springer: New York, NY, USA, 1989.
48. Yang, K.K. A comparison of dispatching rules for executing a resource-constrained project with estimated activity durations. *Omega-Int. J. Manag. Sci.* **1998**, *26*, 729–738. [CrossRef]
49. Davis, E.W.; Patterson, J.H. A Comparison of Heuristic and Optimum Solutions in Resource-Constrained Project Scheduling. *Manag. Sci.* **1975**, *21*, 944–955. [CrossRef]
50. Cooper, D.F. Heuristics for Scheduling Resource-Constrained Projects: An Experimental Investigation. *Manag. Sci.* **1976**, *22*, 1186–1194. [CrossRef]

51. Bertsekas, D.P.; Tsitsiklis, J.N.; Wu, C.C. Rollout Algorithms for Combinatorial Optimization. *J. Heuristics* **1997**, *3*, 245–262. [CrossRef]
52. Bertsekas, D.P.; Castañón, D.A. Rollout Algorithms for Stochastic Scheduling Problems. *J. Heuristics* **1998**, *5*, 89–108. [CrossRef]
53. Goodson, J.C.; Ohlmann, J.W.; Thomas, B.W. Rollout Policies for Dynamic Solutions to the Multivehicle Routing Problem with Stochastic Demand and Duration Limits. *Oper. Res.* **2013**, *61*, 138–154. [CrossRef]
54. Goodson, J.C.; Thomas, B.W.; Ohlmann, J.W. A rollout algorithm framework for heuristic solutions to finite-horizon stochastic dynamic programs. *Eur. J. Oper. Res.* **2017**, *258*, 216–229. [CrossRef]
55. Kleijnen, J.P.C. Antithetic Variates, Common Random Numbers and Optimal Computer Time Allocation in Simulation. *Manag. Sci.* **1975**, *21*, 1176–1185. [CrossRef]
56. Kolisch, R. Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. *Eur. J. Oper. Res.* **1996**, *90*, 320–333. [CrossRef]
57. Sprecher, A.; Kolisch, R.; Drexl, A. Semi-active, active, and non-delay schedules for the resource-constrained project scheduling problem. *Eur. J. Oper. Res.* **1995**, *80*, 94–102. [CrossRef]
58. Kolisch, R.; Sprecher, A.; Drexl, A. Characterization and Generation of a General Class of Resource-Constrained Project Scheduling Problems. *Manag. Sci.* **1995**, *41*, 1693–1703. [CrossRef]
59. Vanhoucke, M.; Coelho, J.; Debels, D.; Maenhout, B.; Tavares, L.V. An evaluation of the adequacy of project network generators with systematically sampled networks. *Eur. J. Oper. Res.* **2008**, *187*, 511–524. [CrossRef]
60. Branke, J.; Chick, S.E.; Schmidt, C. Selecting a Selection Procedure. *Manag. Sci.* **2007**, *53*, 1916–1932. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.