Article

Maximizing Net Present Value for Resource Constraint Project Scheduling Problems with Payments at Event Occurrences Using Approximate Dynamic Programming

Tshewang Phuntsho [1,2,*] and Tad Gonsalves [1]

1   Department of Information and Computer Sciences, Sophia University, Chiyoda-ku, Tokyo 102-8554, Japan;
    t-gonsal@sophia.ac.jp
2   Gedu College of Business Studies, Royal University of Bhutan, Bongo, Chukha P.O. Box 21006, Bhutan
*   Correspondence: t-phuntsho-9g3@eagle.sophia.ac.jp

Abstract: Resource Constraint Project Scheduling Problems with Discounted Cash Flows (RCPSPDC) focuses on maximizing the net present value by summing the discounted cash flows of project activities. An extension of this problem is the Payment at Event Occurrences (PEO) scheme, where the client makes multiple payments to the contractor upon completion of predefined activities, with additional final settlement at project completion. Numerous approximation methods such as metaheuristics have been proposed to solve this NP-hard problem. However, these methods suffer from parameter control and/or the computational cost of correcting infeasible solutions. Alternatively, approximate dynamic programming (ADP) sequentially generates a schedule based on strategies computed via Monte Carlo (MC) simulations. This saves the computations required for solution corrections, but its performance is highly dependent on its strategy. In this study, we propose the hybridization of ADP with three different metaheuristics to take advantage of their combined strengths, resulting in six different models. The Estimation of Distribution Algorithm (EDA) and Ant Colony Optimization (ACO) were used to recommend policies for ADP. A Discrete cCuckoo Search (DCS) further improved the schedules generated by ADP. Our experimental analysis performed on the j30, j60, and j90 datasets of PSPLIB has shown that ADP–DCS is better than ADP alone. Implementing the EDA and ACO as prioritization strategies for Monte Carlo simulations greatly improved the solutions with high statistical significance. In addition, models with the EDA showed better performance than those with ACO and random priority, especially when the number of events increased.

Keywords: approximate dynamic programming; net present value; discounted cash flows; Discrete Cuckoo Search; Estimation of Distribution Algorithm; Ant Colony Optimization; Payment at Event Occurrences

1. Introduction

In project management, the precise timing for initiating project jobs within the constraints of available resources plays a crucial role in optimizing outcomes across sectors such as fleet management, supply chains, and service delivery. One such problem in this area is the Resource Constraint Project Scheduling with Discounted Cash flows (RCPSPDC), where the objective is to maximize the sum of discounted cash flows from the activities.

RCPSPDC is a crucial problem to consider when dealing with long-term projects with significant financial implications [1]. These projects usually involve two parties: the contractor, who is responsible for executing the project, and the client, who outsources the project to the contractor and provides the financing [2]. The details of the financial transactions are usually negotiated when the project's contract is signed [3]. For long-term projects, there are several payment schemes that the two parties can agree upon. Available payment options include Payment at Activities' Completion Times (PAC) [4–6], Progress

Payments (PP) [7], and Payment at Event Occurrences (PEO) [5,6]. Although PAC ensures a consistent cash flow for the contractor, it may pose logistical challenges when processing invoices for each individual activity. In PP, payments are made at regular intervals, and the final payment is disbursed upon completion of the project. PP may not be desirable for the client, as it provides little incentive for the contractor to complete the project on time. In contrast, PEO can be seen as an irregular variant of PP. Under PEO, a specific number of project tasks are treated as events, and payments are made to the contractor as soon as these events are completed. During contract negotiations, the number of events and the corresponding payment amounts can be mutually agreed upon to ensure a Pareto-optimal outcome for both the client and the contractor [8–11].

For seamless project execution, the contractor must maintain a consistent cash inflow of funds to effectively manage expenses such as payroll, equipment costs, and expenditures on raw materials and resources. This study focuses on Resource Constraint Project Scheduling with Discounted Cash flows and Payment at Event Occurrences (RCPSPDC–PEO) from the contractor's perspective. In this scenario, the contractor aims to maximize the Net Present Value (NPV) of the project under the condition that $m + 1$ payments are made throughout the project's duration. These payments are triggered as soon as m predefined activities are completed, with the final settlement upon project completion.

For instance, construction of a new residential complex is one of the practical examples of RCPSPDC–PEO. The contractor is faced with the difficult challenge of sequencing and scheduling a multitude of interdependent and dependent tasks under various resource constraints, ranging from the foundational work to the final utility installations. The allocated budget is usually not paid all at once to prevent unforeseen risks for the client; on the other hand, withholding payment at the completion of the project would also jeopardize the success of the project. Payments are therefore made according to the PEO principle. To optimize the financial outcome of this venture, the contractor aims to maximize the project's NPV, a key variable that determines the project's profitability considering the time value of money. Under the PEO scheme, the company must skillfully manage its limited resources against a backdrop of financial constraints, with cash inflows tied to the achievement of specific project milestones. This requires a judicious planning strategy that aligns task completion with the stipulated payment events to ensure a steady cash flow and, ultimately, the financial success of the project.

RCPSPDC is proven to be NP-hard [12]. Both exact (for smaller problems) and approximate solutions (metaheuristics) have been explored in research [13] that could also tackle RCPSPDC–PEO. The metaheuristics approach takes a set of feasible solutions (i.e., scheduled projects) called population and employs various exploration and exploitation techniques to find quasi-optimal solutions in a reasonable time [14]. Studies have demonstrated that hybrid approaches, intelligently combining different methods (metaheuristics or other exact methods), outperform standalone metaheuristics by leveraging their collective strengths and mitigating individual limitations [14,15]. This preference for hybridization [15,16] resonates with the No Free Lunch Theorem (NFLT) [17], which argues against the feasibility of a universal standalone metaheuristic.

Alternatively, in combinatorial problems like RCPSP and RCPSPDC, the scheduling scheme can be viewed as a Markov process [18]. That is, at any decision point in a scheduling process, a job is scheduled using a certain policy derived from the information available at the current state. In this sense, a determination of scheduling policy, also known as closed-loop policy [19], can be seen as a Markov decision process (MDP) [18,20]. However, computing the optimal closed-loop policy in this setting as Dynamic Programming (DP) suffers from the curse of dimensionality [19]. The approximation of the cost-to-go or the benefit-to-go function [21] to approximate the decision policy [22] to resolve this intractability issue results in an approximate dynamic programming (ADP). To solve a deterministic RCPSPDC–PEO, ADP can be embedded into the schedule generation scheme (SGS) [1,23,24]. SGS then decides based on a derived scheduling policy [22] when there are

multiple activities to be executed, but the quality of the final schedule is highly dependent on the policy it utilized.

Motivated by these facts, this study proposes a novel architecture: a hybridization of approximate dynamic programming (ADP) [19,22,25] with the Discrete Cuckoo Search (DCS) [26–28], Estimation of Distribution Algorithm (EDA) [29,30], and Ant Colony Optimization (ACO) [31,32] to solve the RCPSPDC–PEO. Applications of ADP to solve RCPSPDC or RCPSPDC–PEO have not been explored so far, especially in combination with metaheuristic approaches [11]. In addition, this proposal is driven by several compelling reasons. First, ADP's dynamic and Markov properties enables us to progressively explore solutions, markedly reducing computation time by eliminating the need to revisit previously executed jobs, a common occurrence in metaheuristic approaches. Second, ADP's policy approximation through Monte Carlo (MC) simulations is inherently parallelizable, leveraging the capabilities of multi-core CPU computers to our advantage. Third, the historical data derived from MC simulations can be efficiently integrated into powerful distribution-based metaheuristics such as EDA and ACO, allowing for the reuse of information. Fourth, unlike the often-challenging parameter management in metaheuristics hybrid systems, ADP offers the convenience of independent parameter control. Finally, the refined solutions produced by ADP are ideal for application in DCS, which significantly benefits from the high-quality solutions it employs. Thus, this study makes the following contributions:

1. Review and proposal of 13 different heuristics for RCPSPDC–PEO;
2. Proposal of ADP scheduling based on MC and MDP for RCPSPDC–PEO, which can be implemented on multi-core CPU computers;
3. Proposal of DCS as a solution improvement to ADP for RCPSPDC–PEO; and
4. Proposal of path estimation methods based on EDA and ACO.

This endeavor results in six models based on ADP and ADP with DCS. To compare their performance, we first customized the j30, j60, and j90 project instances of PSPLIB [33]. Our experimental analysis performed on these customized datasets designed to investigate the effectiveness of the six different models showed that ADP–DCS provided better results than ADP alone. Using EDA and ACO as priorities for MC simulations further improved the solutions. Also, the performance of EDA was relatively better than that of ACO, and it showed improvement as the number of events increased. Our customized dataset and experimental results are available online to compare with the contributions from other researchers.

This study is of great value to both finance and project managers alike. Finance managers select projects for an organization based on the NPV as a capital budgeting technique. Project managers execute projects with the goal of maximizing the NPV and thereby boosting shareholders' wealth. This study combines a series of different techniques to maximize the NPV of a project. In addition, the methods presented in this study can be adapted for use across various industries and project categories.

The remainder of the paper is organized as follows: Section 2 presents the problem statement with a brief literature review. The methods we propose are explained in detail in Section 3. Section 4 presents the experimental results and performance comparisons of the different methods. In Section 5, we present the discussion of the study followed by its conclusion, and we also outline our future research options.

2. Problem Definition and Literature Review

In a deterministic and non-preemptive project scheduling, we determine the starting time s_j for each activity from a series of activities numbered from 1 through J. Each activity j has a processing time d_j as well as precedence constraints with the immediate predecessors P_j, which must be finalized before beginning an activity j as denoted by Equation (1), and resource demand of q_j units (i.e., $q_j \in Q$) in its active duration as denoted by Equation (2). The vector Q contains the different resources required by each activity. All data points

are assumed to be fixed, known beforehand, and quantified as non-negative integers. Mathematically, this is formulated as [12,34]:

$$s_i \le s_j - d_i \, , \ \forall i \in P_j \tag{1}$$

$$q_j \cdot \le Q_{rj} \tag{2}$$

In Equation (2), the vector Q_{rj} denotes the available resources to be used by activity j and $. \le$ checks for the availability of all corresponding resources. In Equation (1), an activity j is made available for execution immediately after all its preceding jobs are completed in a zero time-lag fashion.

For RCPSPDC–PEO, a contractor receives cash whenever itcompletes the pre-specified activities and at the project completion time f_{J+1}; the objective function is expressed as:

$$Max. \ Z = I_{f_{J+1}} e^{-df_{J+1}} + \sum_{j=1}^{J} C_j e^{-df_j} + \sum_{v=1}^{m} I_v e^{-df_v} \tag{3}$$

In Equation (3), the completion of activity j is defined as $f_j = s_j + d_j$, where project completion time is $f_{J+1} = \max(\{f_j : j = 1 \dots J\})$. d, C_j and I_j denote the discount factor, cost, and income for executing job j, respectively. It is worth mentioning that the non-linearity of Equation (3) has further increased the complexity of RCPSPDC–PEO from RCPSP which is already NP-hard. Equation (3) maximizes the sum of various cash flow types discounted by the factor d.

There are two mostly used SGS [35–38] in the literature: These are the Parallel Generation Scheme (PSGS) and the Serial Generation Scheme (SSGS). SSGS operates on an activity-incrementation basis, essentially constructing a schedule without a real-time variable. The scheme proceeds through $\{1, \dots, J\}$ stages, with one activity being selected for scheduling during each stage from the pool of eligible activities. In this selection process, eligibility is determined primarily by the precedence constraints, while resource constraints are factored in later when determining the activity schedule time. The authors of [19,22] applied SSGS for stochastic scheduling with ADP in non-deterministic RCPSP.

Conversely, PSGS operates using timeline incrementation and is particularly well-suited for tackling hard problems [23]. PSGS excels at generating non-delay schedules [23]. Hence, for the novelty of this study and for its advantages over SSGS, we have employed the PSGS method to generate project schedules. The PSGS with MDP formulation is explained in Section 3.1.

RCPSP (RCPSPDC) can be divided into two categories: deterministic and non-deterministic. In deterministic RCPSP (RCPSPDC), details about resource and activity duration are known beforehand. In non-deterministic or uncertain RCPSP (RCPSPDC), this information is only partially or fully unknown. In the literature, the deterministic RCPSPDC has attracted more attention than its non-deterministic counterpart. This is primarily because non-deterministic introduces an additional layer of complexity with its inherent uncertainty, rendering the determination of cash flows associated with the project highly challenging. Consequently, our literature review focuses on the deterministic RCPSPDC, which is also the focus of our study. The simplest way to solve the deterministic RCPSPDC is using heuristics, where jobs are ranked and executed based on certain heuristics or policies [1,2]; however, this method does not yield quality solutions [39]. The recent research efforts have resulted in the proposal of various metaheuristics and their hybrids to solve RCPSPDC. Metaheuristics and their hybrids typically can produce near-optimal solutions in a reasonable time due to their exploration and exploitation capabilities [14]. Mika et al. [4] employed the renowned simulated annealing (SA) and tabu search (TS) methods to address the multi-mode RCPSPDC. Zhao et al. [40] utilized the EDA, The authors of [41] integrated a genetic algorithm (GA) with a local search strategy, and [42] implemented ACO. However, better solutions were generated using hybrids of metaheuristics. Vanhoucke [24] merged forward-backward iterations (FBI) with scatter search heuristics, while Gu et al. [43] com-

bined FBI with Lagrangian relaxation and constraint programming. Asadujjaman et al. [23] created a hybrid model combining IGA and neighborhood search with FBI. Leyman and Vanhoucke [12] used parallel GAs without information sharing to solve RCPSPDC. Though these methods were not explicitly studied for RCPSPDC–PEO, they can be adapted to solve the problem. One of the drawbacks of metaheuristics and their hybrids for RCPSP (RCPSPDC), in general, is the loss of feasible solutions in the state transition. For instance, consider GA with single point crossover [23,41]; when two parents of GA cross over to produce an offspring, the precedence conditions (i.e., Equations (1) and (2)) are usually perturbed, leading to an infeasible solution. This necessitates extra processing and computation to rectify the schedule. Additionally, in hybrid systems where multiple metaheuristics are employed either in parallel or sequentially, managing and tuning parameters pose a considerable challenge. This can sometimes diminish the system's overall effectiveness [14–16].

An alternative approach is borrowing the techniques from stochastic scheduling [44]. The stochastic RCPSP (SRCPSP) is a category of non-deterministic RCPSP characterized by uncertainty in project parameters like job execution times and resource availabilities and are represented through distributions. One of the methods used in SRCPSP is sequential generation of a schedule based on an open- or closed-loop policy [19,45]. The open-loop policy aims to devise a complete project schedule before the project's initiation, while the closed-loop policy optimizes the schedule dynamically and adaptively as the project unfolds [46]. Greedy randomized adaptive search procedure (GRASP) [47], genetic algorithm (GA) [45] pre-processing procedure with a two-phase GA [44] and 17 priority-rule heuristics with the justification technique [48] are some of the proposed open-loop policy based techniques for SRCPSP. As the open-loop policy is static in nature, a more flexible policy based on closed-loop policy through dynamic programming (DP) [46] garnered attention for SRCPSP. A closed-loop policy [49] aims to find an optimal decision rule (i.e., policy) to execute a job at each decision point, given knowledge on the current state. If the decision or policy in the closed-loop policy is random, this is similar to any of the SGS methods [35–38] with random scheduling for the classical RCPSP (RCPSPDC). In SGS for the classical RCPSP (RCPSPDC), an activity is scheduled as soon as precedence and resource conditions (i.e., Equations (1) and (2)) are met. If there are multiple eligible activities, SGS schedules an activity based on a certain predefined priority or heuristic value [26,30,32,50]. The main distinction between SGS methods and closed-loop is that the latter uses an optimal policy.

Despite its effectiveness, the closed-loop method is limited by the curse of dimensionality [19] caused by Bellman's recursive approach in the MDP [18]. Thus, this method is applicable to only small problem sets [51]. For instance, the authors [51], used stochastic DP method to solve projects that involved only 17 activities. To address this issue in an uncertain or stochastic RCPSP, the authors of [19,22] utilized rollout algorithms [49] to approximate the policy functions for DP, thus converting it to ADP. Their conversion of DP to an ADP algorithm was based on three core techniques. First, in each decision stage, the sub-problem is constructed by approximating the recursive cost-to-go function in DP. Second, sample paths from MC simulations were used for forward iteration, bypassing the exhaustive state enumeration in traditional DP. Finally, deterministic scheduling techniques were applied within each iteration of ADP. It is should be noted that the efficiency of ADP is highly dependent on the policy (or rollout policy). An analysis of the method combining rollout and lookup table algorithms, including look-back and look-ahead using the idea of parallel rollout [49], was explored in [19]. Constraint programming [52] to solve sub-problems was also explored as the quality of a sub-problem influenced the policy estimation. For combinatorial problems like RCPSP (RCPSPDC), it is recommended to use some heuristic based policy [32] in place of exact cost/benefit-to-go function [53] as promising results have been reported in [54,55].

The dynamic and Markov nature of ADP enables several advantages for RCPSP (RCPSPDC–PEO). First, it always maintains a feasible schedule as the scheduled jobs

are not disturbed. Second, as it sequentially schedules jobs in the forward iteration, the computation complexity reduces as the project progresses. Finally, MC simulation of it can be parallelized on multi-core machines. Although the lookup table as explored in [19] serves basic requirements, it is our belief that using distribution-based algorithms would improve the solution further. However, the policies generated in ADP are approximate, leading to sub-optimal solutions.

In summary, various metaheuristics and their hybrids were proposed in the literature to solve RCPSPDC, which can be customized to solve RCPDPDC–POE. In addition, ADP was successfully used in non-deterministic RCPSP, which can also be customized to solve our problem. However, there are numerous drawbacks to both sets of methods. The main reason for our proposed hybrid architecture is to overcome these challenges. A detailed explanation of our hybrid algorithm is presented in the following section.

3. Proposed Methods

Figure 1 presents a high-level overview of our proposed method. In contrast to the literature for non-deterministic RCPSP, this study embeds ADP inside PSGS for RCPSPDC–PEO. ACO and EDA are alternatively used in addition to random in MC simulation to build priority rules for ADP. The rollout policies are selected by combining two phases using MC simulations. In the first phase, a policy for a state is approximated in advance by steps lookahead. The second phase further refines the policy using a one-step look-ahead. At any given state in an ADP, an optimal policy is derived from 13 different heuristics mentioned in Section 3.3. In addition, we have applied three priority rules in MC simulations to improve the policy optimality. We have further improved our solutions using the DCS technique, focusing on local search.
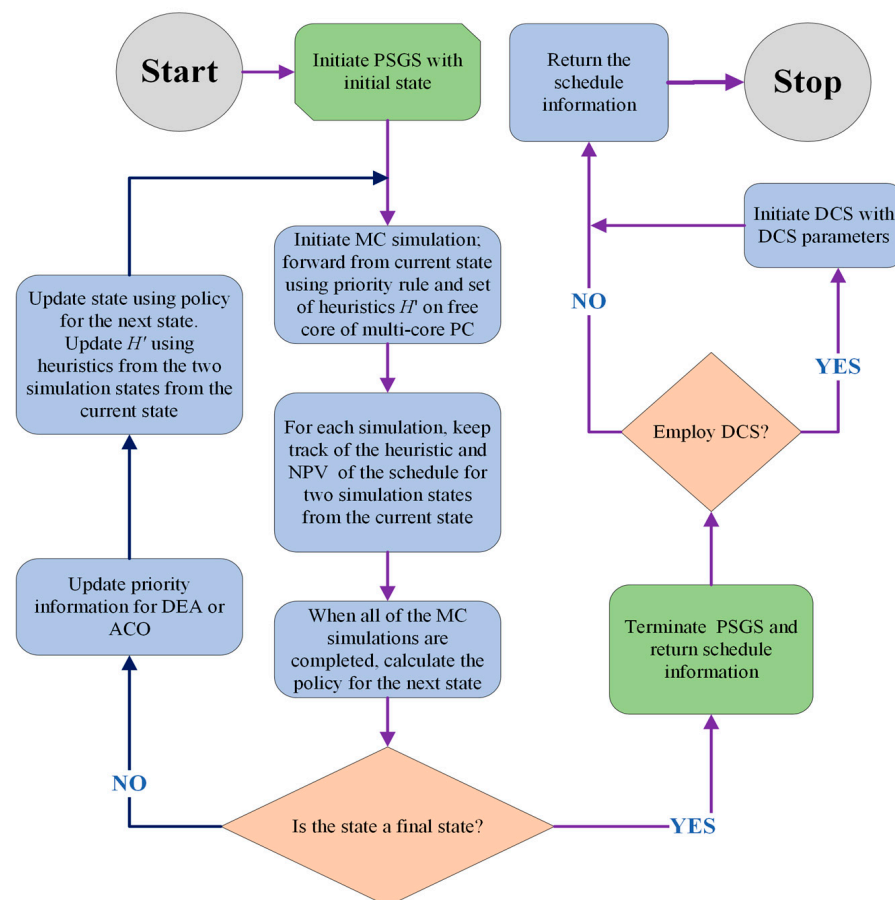

Figure 1. Flowchart of ADP using MC with ACO or EDA followed by DCS.

We have chosen ACO and EDA due to their capabilities for preserving historical information in the form of distribution. ACO and EDA have also shown effectiveness for RCPSP and RCPSPDC [29–32]. DCS was employed to improve the ADP solution. Our choice of DCS was due to its capability to search for solution space in the levy flight scheme. The levy flight characteristics allowed us to employ various search techniques. DCS was also found to produce quality solutions for RCPSPDC [26]. A thorough treatment of each part of our method is made in the following subsections.

3.1. PSGS and MDP Formulation

For deterministic scheduling, the MDP of RCPSPDC–PEO must be embedded into the SGS; in our case, we employ PSGS. Our MDP has the following five components:

1. Stages: We denote the stages by $v \in [0, \dots, \mathrm{F}]$; v gets incremented whenever a job is scheduled. In other words, whenever the number of unscheduled jobs decreases, v gets incremented. $v = 0$ corresponds to no job being scheduled and $v = F$ corresponds to the stage where all the jobs are being scheduled. The number of stages therefore corresponds to the size of the project.

2. States: The state of stage v is described as $S_v = \{A_v, C_v, Q_{rv}, A_{v_t}, t_v\}$. In S_v, the elements A_v, C_v, Q_{rv} are the vectors containing the active jobs at time v, the completed jobs at v, and the resources available at v, respectively. The vector A_{v_t} contains the completion time of the active jobs, and t_v denotes the completion time of the jobs that are completed immediately before the state S_v. The initial and final states are given by $S_0 = \{\varnothing, \varnothing, Q_{r0}, \varnothing, 0\}$ and $S_F = \{\varnothing, \{1 \dots J\}, Q_{rF}, \varnothing, f_{J+1}\}$, where $Q_{r0} \cdot = Q_{rF} \cdot = Q$; Q is the vector containing the total resources available when no job is being executed and $. =$ denotes the element wise comparison in the vectors.

3. Decision sets: Suppose $E_v := \chi(S_v)$ consists of eligible activities, that is, jobs that fulfill Equations (1) and (2), then the decision space $x_v \subseteq \chi(S_v)$ contains jobs that are feasible for execution.

4. Transition process: The transition from stage v to $v + 1$ takes place with $S_{v+1} = S^M(S_v, x_v)$ using the transition function $S^M(\cdot)$. The transitions of different components are presented in Equations (4a) to (4k):

$$\text{if } q_{j\cdot} \geq Q_{rj}, \forall j \in E_v \text{ or } E_v = \varnothing \tag{4a}$$

$$t_{v+1} = \min\left(\left\{f_j : j \in A_v, f_j \in A_{v_t}\right\}\right) \tag{4b}$$

$$C_{v+1} = C_v \cup \left\{j : f_j \leq t_{v+1}, \ \forall j \in A_v, f_j \in A_{v_t}\right\} \tag{4c}$$

$$Q_{rv+1} = Q \cdot - \sum_{k=1}^{|q|} \left[q_1 \dots q_j \dots q_l\right]^{-1}, \ j = [1, \dots, l] \in A_v \tag{4d}$$

$$A_{v+1} = \left\{j : f_j > t_{v+1}, \ \forall j \in A_v, f_j \in A_{v_t} \cup x_v,\right\} \tag{4e}$$

$$\text{if } q_j \cdot \leq Q_{rj}, \ \exists j \in E_v \text{ or } A_v = \varnothing \tag{4f}$$

$$t_{v+1} = t_v \tag{4g}$$

$$C_{v+1} = C_v \tag{4h}$$

$$A_{v+1} = A_v \cup x_v \tag{4i}$$

$$Q_{rv+1} = Q \cdot - \sum_{k=1}^{|q|} \left[q_1 \dots q_j \dots q_l\right]^{-1}, \ j = [1, \dots, l] \in A_v \tag{4j}$$

$$A_{(v+1)_t} = \left\{t_{v+1} + d_j : j \in A_{v+1}\right\} \tag{4k}$$

In this transition, the process must follow a logical control flow. That is, if Equation (4a) is true, Equations (4b)–(4e) are executed, followed by Equation (4k). Equation (4a)

checks whether E_v, an eligible set at state S_v, is empty or whether the resources required by each activity of E_v are less than the balance resources if E_v is not empty. Equation (4a) assesses whether the set of eligible activities at state S_v, denoted as E_v, is empty. If E_v is not empty, the equation further evaluates whether the resources required for each activity within E_v exceed the available balance of resources. Equation (4b) determines the times associated with state S_{v+1} by taking the minimum of the finish times of the active jobs from the state S_v. In Equation (4c), the completion set C_{v+1} of S_{v+1} is set to the union of C_v and the set containing jobs whose finish times are less than equal to t_{v+1}. The balance resources for state S_{v+1} are calculated using Equation (4d) or (4j). For each resource, the remaining resource is the difference of the total resource and the sum of the resources currently being used by active jobs at state A_v. Finally, an active set A_{v+1} is updated by adding the decision set to active set A_v whose finish times are greater than t_{v+1}. In Equations (4d) and (4j), the value of $|q| = \left|q_j\right|$, $\forall j \in \{1, .., J\}$, as this indicates the different types of resources required by an activity j. If $E_v \neq \varnothing$ in Equation (4a), then Equation (4e) is executed before Equation (4d) by setting $x_{v\prime} = x_v$; if $E_v = \varnothing$, a new E_{v+1} is determined directly after Equation (4d), and we set $x_{v\prime} = x_{v+1}$. The set x_v is determined using Equation (6). If a statement given by Equation (4f) is true, then Equations (4g)–(4k) are executed. That is, if there is no active job being processed or there exists at least one job in the eligible set E_v in which the resources required are less than the balance resources checked by Equation (4f), then the time t_{v+1} and completion set C_{v+1} of S_{v+1} are the copies from the previous state. Equation (4i) determines the active set A_{v+1} by including the decision set. Finally, Equation (4k) determines the completion times of the activities in the active set A_v. We see that the state S_{v+1} depends only on state S_v, which indicates that the process is a Markov process [56].

5. *Benefit function*: Let $\pi = \left\{\omega_0^\pi(S_0), \ldots, \omega_F^\pi(S_F)\right\}$ be the policy containing a sequence of decisions, where $\omega_v^\pi(S_v) : S_i \to x_v^\pi$. Let $\psi(S_v, x_v^\pi, S_{v+1})$ be the change in the NPV of the project as a result of decision x_v^π at S_v that leads to state S_{v+1}. We define the benefit-to-go function with the policy π starting from stage v with state S_v as follows:

$$B_v(S_v) = \mathbb{E}\left[\sum_{j=v}^{F} \psi(S_v, x_v^\pi, S_{v+1})\right]$$

$$= \mathbb{E}[\psi(S_v, x_v^\pi, S_{v+1}) + B_{v+1}(S_{v+1})] \tag{5}$$

The benefit-to-go function $B_v(S_v)$ is the expectation of the sum of $\psi(\cdot)$ from states v to F. $B_v(S_v)$ can then be re-written as the expectation of $\psi(S_v, x_v^\pi, S_{v+1})$ and $B_{v+1}(S_{v+1})$.

To maximize the objective function given by Equation (3), we employ the Bellman equation [57,58] to determine an optimal policy:

$$x_v^{\pi^*} = \underset{x_v \subset \chi(S_v)}{\arg\max} \mathbb{E}[\psi(S_v, x_v, S_{v+1}) + B_{v+1}(S_{v+1})] \tag{6}$$

Solving Equation (6) is computationally difficult due to the well-known curse of dimensionality. Given the permutation nature of our problem, we use a rollout-policy-based [22,59] ADP to approximate $x_v^{\pi^*}$. In this scheme, we approximate $B_{v+1}(S_{v+1})$ using a certain heuristic h_{v+1} from a set of heuristics $\mathcal{H} = \{h_1, \ldots, h_k\}$. Because there are a large number of states and trajectories, we employ forward iteration with MC [45] to select the h_{v+1}. In this study, we use two-stage look-ahead to guide the selection of the one-stage look-ahead heuristic and approximate the benefit-to-go value. That is, to select h_{v+1} at state S_{v+1}, we run multiple trajectories using $h_i \in \mathcal{H}' \subset \mathcal{H}$ and keep track of heuristics $h_j \in \mathcal{H}$ at S'_{v+2}. The state S'_{v+2} is the state in each trajectory of the MC simulation (we call it the simulation stage) and is different from the actual S_{v+2}. Let Θ be the set of simulated trajectories starting at S_v; we set $h_{v+1} \in \mathcal{H}'$, which has the highest average value of the NPV of Θ. We then update $\mathcal{H}'$ using the heuristics from $\mathcal{H}$ that yielded the highest mean NPV of

Θ. This process is further explained in PSGS algorithm, which uses an MDP as shown in Algorithm 1.

Algorithm 1 PSGS algorithm with ADP

Input: S_v, A, C, t, $|\Theta|$, $\mathcal{H}'$, $\mathcal{H}$, Q, ADP (Boolean value), h, priority rule

1.   **If** ADP
2.   Extract values of A, C, t and A_t from S_v
3.   **End If**
4.   Calculate Q_r using Equation (4d)
5.   **While** $|C| < |\{1, \ldots, J\}|$ or $A \neq \varnothing$
6. $E = \left\{ j : \forall P_j \in C, \ q_{j.} \leq Q_r \right\}$
7.   **While** $E \neq \varnothing$
8.   **If** ADP
9.   **For** $\forall h \in \mathcal{H}'$
10.   **For** _ in $1 : |\Theta|$
11.   Execute PSGS algorithm given in Algorithm 1 with "ADP = false".
12.   **End For**
13.   **End For**
14.   Update state using Equations (4a)–(4k)
15.   Update $\mathcal{H}'$ using Equation (7)
16.   **Else**
17.   **If** $|C| = |C_v| + 1$
18.   Select $e \in E$ based on h
19.   **Else if** $|C| = |C_v| + 2$
20.   Select $e \in E$ based on $h_i \in \mathcal{H}$
21.   **Else**
22.   Select $e \in E$ based on priority rule.
23.   **End If**
24.   Update $A := A \cup \{e\}$
25.   Update $A_t := \left\{ j \to t + d_j : j \in A \right\}$
26.   Update Q_r using step (4)
27.   Update E using Equation (6)
28.   **End While**
29.   **If** ! ADP and $A \neq \varnothing$
30. $t := \min(f \in \{f_t : j \to f_t \in A_t\})$
31.   Update $C := C \cup \{j : f \leq t : f \in \{f_t : j \to f_t \in A_t\}\}$
32. $A := A \backslash C$
33.   Update A_t using step (26)
34.   Update Q_r using step (4)
35.   **End If**
36.   **End While**
37.   **Output**: Schedule, NPV, h_i

For a project, we execute the algorithm in Figure 1 with the inputs: S_0, $\varnothing$, $\varnothing$, 0, $|\Theta|$, $\mathcal{H}$, $\mathcal{H}$, Q, ADP (= true), h (= random($\mathcal{H}$)), priority rule. If we initialize with (ADP = false), the algorithm results in a schedule with the usual generic PSGS. The priority rule can be random, or it can be devised based on certain heuristic or meta-heuristic algorithms. In this study, we use three different priority rules as explained in Section 3.4. Because the MC simulations are independent, we take advantage of multi-core computers for parallel execution. That is, given $|\Theta|$ the size of the MC and the number of cores (nc), we distribute the simulations $|\Theta|$ to the nc cores as $\frac{|\Theta|}{nc}$.

3.2. Discrete Cuckoo Search (DCS)

After a schedule has been generated by the ADP, we further try to improve it using various local search techniques. Although there are many local search techniques [60–62], in this study we use DCS [26,50], which systematically switches between different types

of search steps based on certain predefined paths. DCS mimics a cuckoo bird searching for a host nest to lay eggs. The Lévy flight [63,64] is one of the search paths the cuckoo birds follow to explore a viable nest. In the Lévy phenomenon, there are many short hops and only rarely long hops. In this study, we set the stability parameter of the stable Lévy distribution ($\alpha = 1$), which corresponds to the Cauchy distribution [65], and consider the following search techniques corresponding to different step lengths as indicated in the algorithm shown in Algorithm 2:

1. Swap mutation [66]: In this swap mutation, two activities are picked randomly and their order is swapped if possible. Swapping positions is not always possible due to precedence constraints. Nevertheless, we try to mutate as much as possible. For instance, we select activities a and b such that $f_a < f_b$, then the schedule up to a being executed remains unchanged. The partial schedule is completed using SGS by delaying a as far as possible and prioritizing b as soon as possible.

2. $k-$ swap mutation: In this search, we perform the swap mutation k times in a row. In this study, we set $k = \lfloor 0.15 \times J \rfloor$, where J denotes the project size.

3. Scramble mutation [66]: In this mutation, n jobs are randomly selected, and their execution times are scrambled based on cost. For instance, we randomly select $a_1, a_2, \ldots, a_n$ such that $s_{a_1} < s_{a_2} < \cdots < s_{a_n}$, then we take the partial schedule just before the execution of a_1 and complete the schedule using SGS by prioritizing the activities which have the minimum costs. In our study, we set $n = \lfloor 0.15 \times J \rfloor$.

4. Inverse mutation [66]: In this mutation, n jobs are randomly selected, and their execution times are inverted. For example, we randomly select $b_1, b_2, \ldots, b_n$ such that $s_{b_1} < s_{b_2} < \cdots < s_{b_n}$, followed by completing the partial schedule containing the original schedule just before b_1 is executed. We set n the same as scramble mutation.

Algorithm 2 DCS algorithm applying various mutations

Input: schedule, τ (maximum number of steps)
1. Set $w = (\tau + 1)^{-1}$, $\lambda \in [2, \tau]$.
2. Generate τ number of step lengths from Lévy distribution with $\alpha = 1$ and set it to L.
3. Normalize L such that value of L are between 0 and 1.
4. **For each** i in L
5.   **If** $i \in [0, w]$
6.   Perform swap mutation search.
7.   Update the schedule with better NPV.
8.   **Else If** $i \in [(\lambda - 1) \times w, \lambda \times w]$
9.   Perform $k -$ swapmutation.
10.   Update the schedule with better NPV.
11.   **Else**
12.   **If** random (0,1) < 0.5
13.   Perform inverse mutation.
14.   Update the schedule with better NPV.
15.   **Else**
16.   Perform scramble mutation.
17.   Update the schedule with better NPV.
18.   **End If**
19.   **End If**
20. **End For**
21. **Output**: Updated schedule with NPV

3.3. Heuristics

One of the inputs to ADP is $\mathcal{H}$, a set containing a list of heuristics. These heuristics are defined as follows:

1. $h_{ij} = \underset{i \in P_j}{\text{argmin}} \frac{d_j}{\sum_{l \in S_i} d_l}$, which favors the shortest processing time;

2. $h_{ij} = \underset{i \in P_j}{\operatorname{argmin}} \frac{\sum q_{jk}}{\sum_{l \in S_i} \sum q_{lk}}$, which choses the minimum amount resources required first;

3. $h_{ij} = \underset{i \in P_j}{\operatorname{argmax}} \frac{\sum q_{jk}}{\sum_{l \in S_i} \sum q_{lk}}$, which favors the maximum amount of resources required;

4. $h_{ij} = \underset{i \in P_j}{\operatorname{argmin}} \frac{\sum v_{jk}}{\sum_{l \in S_i} \sum v_{lk}}$, such that $v = ([q_i > 0])_{i=1}^{|q|}$ where $q_i \in q$, which considers the activity with the minimum number of resources required;

5. $h_{ij} = \underset{i \in P_j}{\operatorname{argmax}} \frac{\sum v_{jk}}{\sum_{l \in S_i} \sum v_{lk}}$, where v is as defined in heuristics number 4 that considers the maximum number of resources required first;

6. $h_{ij} = \underset{i \in P_j}{\operatorname{argmin}} \frac{|S_j|}{\sum_{l \in S_i} |S_l|}$, which prioritizes an activity with the minimum number of successors;

7. $h_{ij} = \underset{i \in P_j}{\operatorname{argmax}} \frac{|S_j|}{\sum_{l \in S_i} |S_l|}$, which favors an activity with the maximum number successors;

8. $h_{ij} = \underset{i \in P_j}{\operatorname{argmin}} \frac{|P_j|}{\sum_{l \in S_i} |P_l|}$, which prioritizes an activity with the minimum number of predecessors;

9. $h_{ij} = \underset{i \in P_j}{\operatorname{argmax}} \frac{|P_j|}{\sum_{l \in S_i} |P_l|}$, which favors an activity with the maximum number of predecessors;

10. $h_{ij} = \underset{i \in P_j}{\operatorname{argmax}} C_i$, which favors an activity with the maximum cashflow;

11. $h_{ij} = \underset{i \in P_j}{\operatorname{argmax}} [C_i e^{-df_i} + I_i e^{-df_i}]$, which favors an activity with the maximum discounted cashflow;

12. $h_{ij} = \underset{i \in P_j}{\operatorname{argmax}} ff_i$, which prioritizes the maximum free floats; and

13. $h_{ij} =$ random combinations of any of the two heuristics (i.e., pick any two from 1 through 12).

Given an activity j, free float (ff_j) denotes the amount of time an activity can be delayed without hampering the project completion time. The free float is calculated as:

$$ff_j = ES_i - EF_j, \ i \in S_j \tag{7}$$

where ES_i and EF_j are called early start of activity i and early finish of activity j. To compute these values, we set $ES_0 = EF_0 = 0$ (i.e., the early start and early finish of no activity), then ES_i and EF_j are solved recursively as:

$$ES_i = \max\{EF_k : k \in P_i\}; EF_j = ES_j + d_j \tag{8}$$

3.4. Priority Rules

In each state S_v of ADP, we use MC and simulate $|\Theta|$ schedules. These schedules are usually forgotten when we try to approximate policy functions for the subsequent states. Such a technique can be called random prioritization. In contrast, we can also learn from the schedules generated by MC simulations to promote exploitation in the subsequent MC simulations. In this study, we explore two techniques based on metaheuristics: EDA and ACO. EDA [29] involves the estimation of the distribution model based on selected elite ancestral generations. The distribution model can be represented in a matrix form. An element $a_{ij} \in P_{J \times J}$ denotes the probability of activity i being scheduled at position j. We initialized this matrix as $P_{J \times J}(0)$ such that $a_{ij}(0) = \frac{1}{J}, \forall i, j \in [1, J]$ represents a uniform probability for the execution of each activity at each position. Given a schedule s in state S_v, we update the elements of P as:

$$a_{ij} := (1 - \gamma)a_{ij} + \gamma[I_i - C_i][s[i] = j] \tag{9}$$

where $\gamma \in (0, 1]$, I_i and C_i are defined as in Equation (3). $[S[i] = j]$ is in Iverson bracket notation, which returns 1 whenever the job at position i in schedule s equals j and 0 otherwise. Once the elements are updated, a column normalization is performed. Given $|\Theta|$ schedules, only the top $\lfloor 0.25 \times |\Theta| \rfloor$ solutions are selected to update P because our distribution must represent the best values of the NPVs. In this study, we set $\gamma = 0.3$ to give more importance to the historical information. To use P in ADP, we substitute the "priority" by P as an input to the "PSGS algorithm with ADP", as shown in Figure 1. The algorithm then schedules an activity i at position j, which has the highest value of a_{ij} in P in an Epsilon-Greedy Algorithm [67,68] fashion with $(1 - \epsilon = 0.90)$ when it has multiple options. Alternatively, we can use the information generated by ACO [42] as the priority rule. ACO depicts the foraging phenomenon of ants. In this study, the pheromone information of the ants is stored in a matrix $\phi(S_v) := \phi(S_v)_{J \times J}$. The elements of $\phi(S_0)$ are set to:

$$\tau_{ij} = a_{ij}(s_0) := |S_i|^{-1} \left[i \in P_j \right] \tag{10}$$

where $\left[i \in P_j \right]$ is in the Iverson bracket notation checking the existence of job i in the precedence of j. S_i is the set of successors of an activity i. Given a schedule s, the pheromone is updated as:

$$\tau_{ij} := (1 - \mu)\tau_{ij} + \mu \frac{Z_{s_i}}{(\vartheta \times Z_b - Z_s)} [s \to (i, j)] \tag{11}$$

where $\mu \in [0, 1]$, Z_{s_i} is the cumulative objective value of schedule s up to activity i calculated using Equation (3) are the final objective values of the historical best schedule and schedule s, respectively. In this study, we set $\vartheta = 2$, and $s \to (i, j)$ denotes the job i schedule immediately before activity j in schedule s. To use this priority rule in our ADP, we substitute $\phi(S_v)$ for "priority" as an input to the "PSGS algorithm with the ADP" shown in Figure 1. In the MC simulations, an activity with the minimum value of u_{ij} given by Equation (12) is selected from the eligible set with a certain probability $(1 - \epsilon = 0.90)$ [67,68].

$$u_{ij} := \frac{\tau_{ij}^{\omega} h_{ij}^{1-\omega}}{\sum_{l \in S_i} \tau_{il}^{\omega} h_{ij}^{1-\omega}} \left[i \in P_j \right] \tag{12}$$

For each simulation in Θ, we first choose the parameter control (ω) of the pheromone and then a heuristic. We randomly set ω to any value from $\{0, 0.2, 0.4, 0.6, 0.8, 1\}$, and the heuristic is randomly selected from Section 3.3. We also set μ in Equation (11) to 0.3 to give more weight to the historical information. As with EDA, we update the Equation (11) given Θ, using only elite solutions in each state S_v. The size of the elite solution is set to $e_n = \lfloor 0.25 \times |\Theta| \rfloor$. To minimize the computation, updates for EDA and ACO are made only for jobs that are not scheduled (i.e., $\{1, \ldots, J\} \backslash C \cup A$).

4. Experimental Results

In this study, we used j30 (projects with 30 jobs), j60 (projects with 60 jobs), and j90 (projects with 90 jobs), each set containing 480 instances from PSPLIB [69] for analysis. The project instances in PSPLIB have been generated by the standard project generator ProGen [69] by authors of [70]. Tables 1 and 2 contain the summary of the parameters used by the ProGen software to generate the project instances of single-mode RCPSP with renewable resources.

Table 1. Base parameter setting for ProGen.

Range	d_j	$|Q|$	U_Q	K_Q	S_1	S_j	P_J	P_j
Min	1	4	1	1	3	1	3	1
Max	10	4	10	2	3	3	3	3

Table 2. Variable parameter setting for ProGen.

Parameter	Levels			
NC	1.50	1.80	2.10	
RF	0.25	0.50	0.75	1.00
RS	0.20	0.50	0.70	1.00

The base parameters are adjusted individually for each benchmark set, and variable parameters are varied within each benchmark set. As usual, $d_j, |Q|, U_Q, K_Q, S_1, S_j, P_J$, and P_j in Table 1 denote processing time for activity j, total number of unique resources, per-period usage of particular resource, number of different categories of resources consumed, number of start activities, number of successors of an activity j, number of finish activities, and number of predecessors of an activity j, respectively. In Table 2, the abbreviations NC, RF, and RS correspond to network complexity, resource factor, and resource strength, respectively, for resources belonging to category k within K_Q. NC is quantified as the average number of unique arcs for each node, considering dummy nodes as well. The RF represents the average proportion of resources from category $k \in K_Q$ that are utilized. It can be seen as an average measure of how much resource is typically used by the activities in a project. Meanwhile, resource strength RS refers to the robustness of the resource constraints for the specified resource category $k \in K_Q$. It measures the scarcity or abundance of a resource relative to the demand for that resource in a project. If a resource has high strength, it means there is plenty of it available in relation to how much is needed for the project. For each project set, 480 instances were created using the full factorial design of variable parameters with 10 replications (i.e., $3 \times 4 \times 4 \times 10 = 480$). A file named *J9040_9.SM*, corresponds to project set j90, with variable combination 40 and replication number 9 of single-mode.

In the experimental design of [70], the authors have used four steps to generate a project in the form of an acyclic directed graph employing a number of fundamental graph theories. The first three steps generate the base project, and the fourth step is repeated until conditions imposed by the combination sought from the parameters in Table 2 are met. For instance, to generate J9001 (i.e., project with 90 jobs and combination 01) the steps are as follows:

Step 1: Generate a random integer for each of S_1 and P_J using the range from Table 1. In our case, a random integer numbers from a closed interval ($|S_1| := rand[3,3] = 3$) for S_1 and ($|P_J| := rand[3,3] = 3$) for P_J are generated. Then, $|S_1|$ number of nodes are connected by an arc to a dummy node (source) denoted by 1. These $|S_1|$ nodes are then numbered from 2 to $|S_1|+1$ and are called starting activities. Similarly, $|P_J|$ nodes are connected by an arc to a dummy node (sink) denoted by $(90 + 1)$ where $|P_J|$ are numbered $(90 - |P_J|)$ to 90. These nodes are then called finish-activities.

Step 2: A random predecessor node (activity) is assigned starting with the lowest indexed (non-start) node until all the jobs (i.e., $|S_1| + 2$ to $(90 - |P_J| - 1)$) are assigned.

Step 3: If any node has no successor, a random node is assigned as the successor to the node.

In step 2 and step 3, the S_j and P_j range values indicated in Table 1 are maintained. Also, for each job j, d_j is determined using the random integer value from the closed interval from the corresponding range values given by Table 1. We then proceed to the final step.

Step 4: After randomly determining the resource allocations made based on Table 1 (i.e., after generating random integers from the closed intervals of $|Q|, U_Q, K_Q$ in a similar process to step 1), the values of NC, RF, and RS of the network are determined. The additional arcs are further added randomly in the network to reach a desired NC, and resource allocations for each job are further tuned to achieve the needed combination value. For J9001, NC = 1.50, RF = 0.25, and RS = 0.20 must be achieved by tuning the number of arcs and resource values. Readers may refer

to [70] for simple formulas to calculate these values and a detailed description of the parameters and their realizations.

This process (i.e., steps 1–4) was computed using ProGen software as it is computationally expensive. In addition, in this study, to meet the practicality of the data, two additional values (i.e., cost and income) were determined. The monetary cost of executing each activity is determined as follows:

$$C_i := \frac{d_i \sum q_i}{\sum_{k=1}^{J} d_k \sum q_k} \times 100 \tag{13}$$

In Equation (13), d_i and q_i have the same meanings as defined in Equations (1) and (2). For each project, we randomly select a set of three, four, five, and six activities as events. For each set of events, we assign an income from performing these activities to random positive numbers such that the total income always comes to 100. (Files containing detailed information regarding each income for each event for all projects can be downloaded along with the dataset and associated results using the link provided at the end of this paper). Using just cost and incomes in this setting has no reward for a contractor as the sum of costs equals the income without taking into account the time value of money. To reward a contractor, a final settlement of C_{J+1} is computed using Equation (14):

$$C_{J+1} := \frac{30 \times J}{|E|} \tag{14}$$

where $|E|$ denotes the total number of events or jobs that generate income for the contractor, in our case, $|E| \in \{3,4,5,6\}$. The client pays the contractor without any delay whenever the contractor executes activities specified in the event set. In this study, we set $|\Theta| = 2(J - |C \cup A|)$ where C and A denote the completed and active jobs, respectively. That is, $|\Theta| = 2J$ when the state is S_0 and $|\Theta| = 0$ when the state is S_F, and $|\Theta|$ decreases linearly as a function of the state S_v. The numerical experiment is designed to investigate the effectiveness of six different models developed in this study. These models include: 1. ADP with random priority rule, 2. ADP with EDA as priority rule, 3. ADP with ACO as priority rule, 4. ADP–DCS with random priority rule, 5. ADP–DCS with EDA as priority rule, and 6. ADP–DCS with ACO as priority rule. ADP–DCS indicates the improvements made by DCS after the ADP process.

The experiments were conducted on an AMD Ryzen Threadripper 3970X 32-core processor, 128GB RAM machine in the Julia language. We note that a direct comparison between models associated with ADP and ADP–DCS cannot be made as ADP–DCS tries to make improvements on ADP solutions by generating additional schedules. Therefore, our study aims to show whether it is possible to improve the APD-generated solutions by applying DCS. However, comparisons within the same group of models can be made because they share the same parameters and have the same number of solutions. To allow fair comparisons within the models of ADP–DCS, all models in this family are terminated after DCS has generated 500 individuals. The computational results are shown in Table 1 for j30, j60, and j90.

Table 3 shows the average values of the NPV for the j30, j60, and j90 project sets corresponding to the six models for different numbers of event sets. The results show that the mean values of ADP–DCS are better than ADP for all three project sets. We also see that the mean values produced by EDA and ACO as priorities for MC are better than the mean values of the random priority for MC. In particular, EDA priority has generated the highest average values in the majority of the cases. Figures 2 and 3 show the bar charts of the mean values of the NPV plotted against different job sets by varying the size of event occurrences. We see that ADP(–DCS)–EDA consistently outperforms the other methods.

Table 3. Experimental results containing average values of j30, j60, and j90 projects for six different algorithms in terms of $|E|$.

Algorithm	Priority Rule	$|E|$	$\overline{Z}$ * for j30	$\overline{Z}$ * for j60	$\overline{Z}$ * for j90	Overall $\overline{Z}$ *
ADP	Random	3	25.18	132.47	212.65	123.43
		4	−17.05	64.72	126.22	57.96
		5	−43.29	24.95	74.68	18.78
		6	−61.31	−1.85	41.01	−7.38
	EDA	3	29.05	141.05	223.39	131.16
		4	−13.73	71.95	134.82	64.35
		5	−40.16	30.07	81.84	23.92
		6	−58.10	2.90	47.77	−2.48
	ACO	3	27.56	137.84	219.88	128.43
		4	−14.95	68.99	132.07	62.04
		5	−41.12	28.32	79.84	22.35
		6	−59.09	1.32	45.59	−4.06
ADP–DCS	Random	3	30.80	142.79	225.73	133.11
		4	−12.14	73.28	137.20	66.11
		5	−37.66	32.15	84.52	26.34
		6	−53.37	5.11	49.71	0.48
	EDA	3	31.16	143.92	226.67	133.92
		4	−11.85	73.42	137.75	66.44
		5	−37.03	32.22	84.47	26.55
		6	−52.19	5.13	50.54	1.16
	ACO	3	30.03	141.12	223.80	131.65
		4	−12.72	72.10	135.70	65.03
		5	−37.73	31.10	82.92	25.43
		6	−53.42	4.08	48.67	−0.22

* The objective value is given by Equation (3). $\overline{Z}$ denotes the average value of Z.
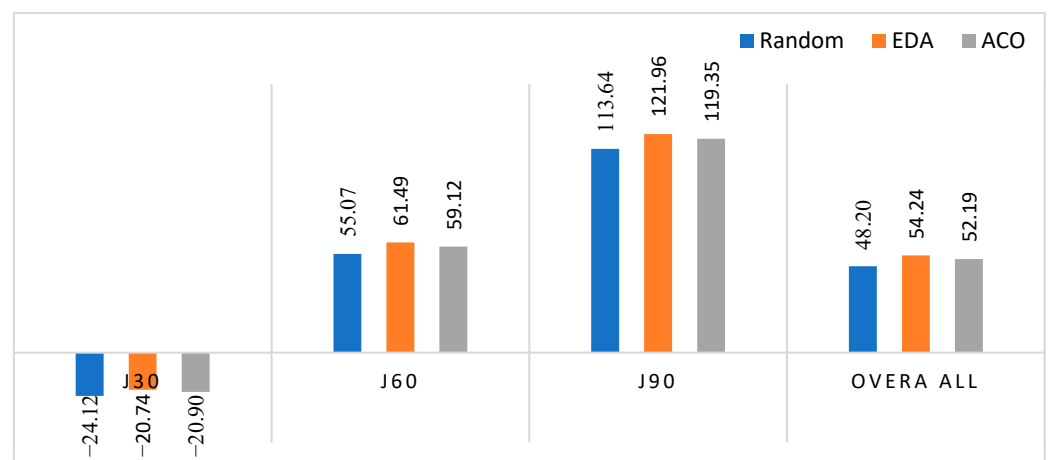
Figure 2. Bar chart of the mean values of the NPV corresponding to each job set against each priority rule for ADP in terms of $|E|$.
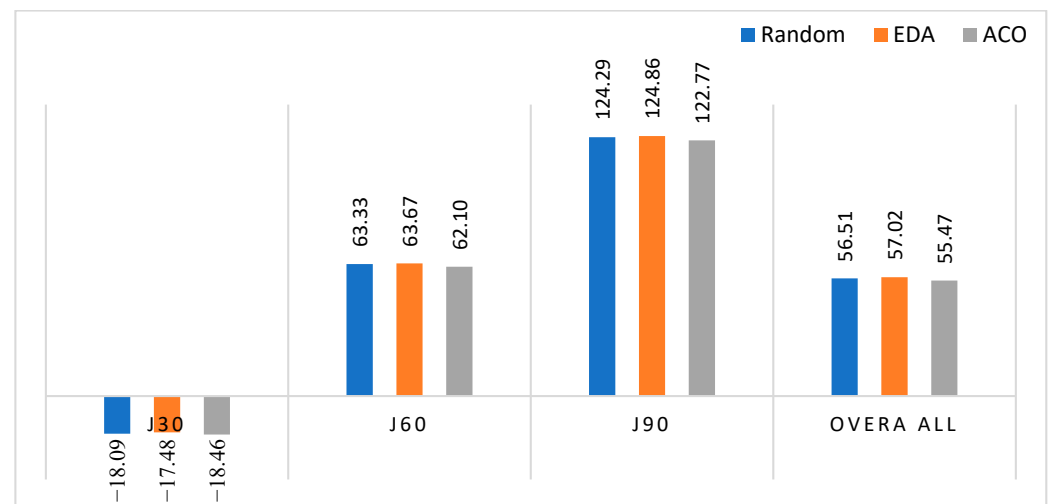
Figure 3. Bar chart of the mean values of the NPV corresponding to each job set against each priority rule for ADP–DCS in terms of $|E|$.

Table 4 presents the average values of the NPV for the j30, j60, and j90 project sets corresponding to the six models for different levels of NC. As NC increases, the average value decreases for each project size, which is logical because, as the complexity of the network increases, it becomes harder for each algorithm to find the best NPV. We see that EDA and ACO used as priority for MC have increased the NPV. We can also note that EDA variant is much resistant to changes in NC. Similarly, Figures 4 and 5 confirm our observation from Table 4.

Table 4. Average values of the NPV for j30, j60, and j90 projects for six different algorithms in terms of NC.

| Algorithm	Priority Rule	NC	$\overline{Z}$ for j30	$\overline{Z}$ for j60	$\overline{Z}$ for j90	Overall $\overline{Z}$
ADP	Random	1.5	−88.82	247.42	511.69	223.43
		1.8	−97.04	217.01	443.85	187.94
		2.1	−103.53	196.45	408.15	167.02
	EDA	1.5	−73.86	274.41	548.44	249.66
		1.8	−83.38	242.47	476.26	211.78
		2.1	−91.56	218.32	438.76	188.51
	ACO	1.5	−78.57	240.97	536.95	233.12
		1.8	−87.90	211.98	467.86	197.31
		2.1	−96.34	192.60	427.35	174.54
ADP–DCS	Random	1.5	−62.48	283.69	558.86	260.02
		1.8	−72.34	250.34	485.65	221.22
		2.1	−82.35	225.94	446.97	196.85
	EDA	1.5	−59.62	286.08	561.93	262.80
		1.8	−69.97	251.91	487.49	223.14
		2.1	−80.13	226.09	448.88	198.28
	ACO	1.5	−63.90	279.09	552.46	255.88
		1.8	−73.65	245.00	480.23	217.19
		2.1	−83.97	221.09	440.60	192.57

$\overline{Z}$ denotes the average value of Z.
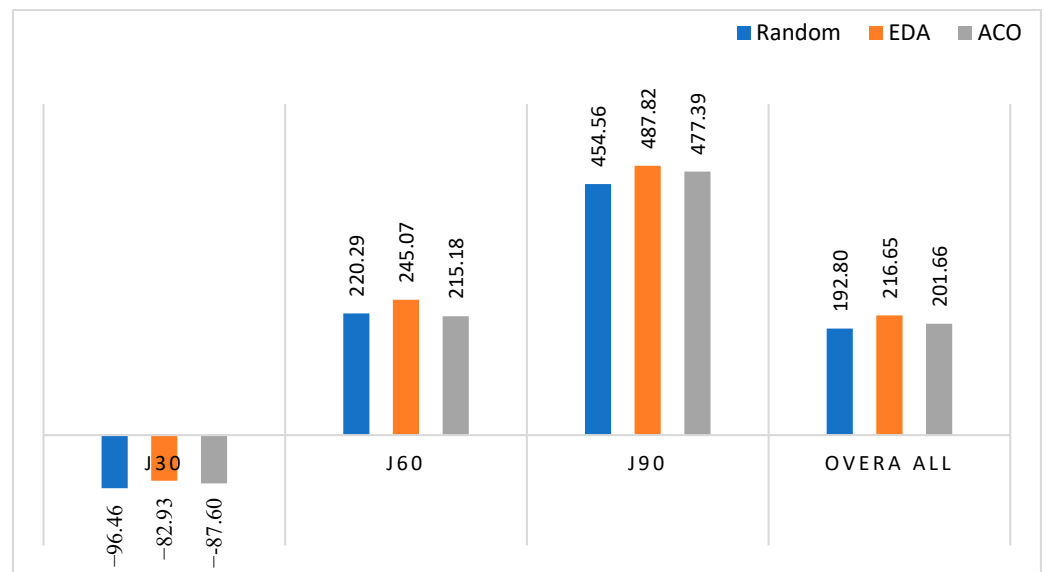
Figure 4. Bar chart of the mean values of the NPV corresponding to each job set against each priority rule for ADP in terms of NC.
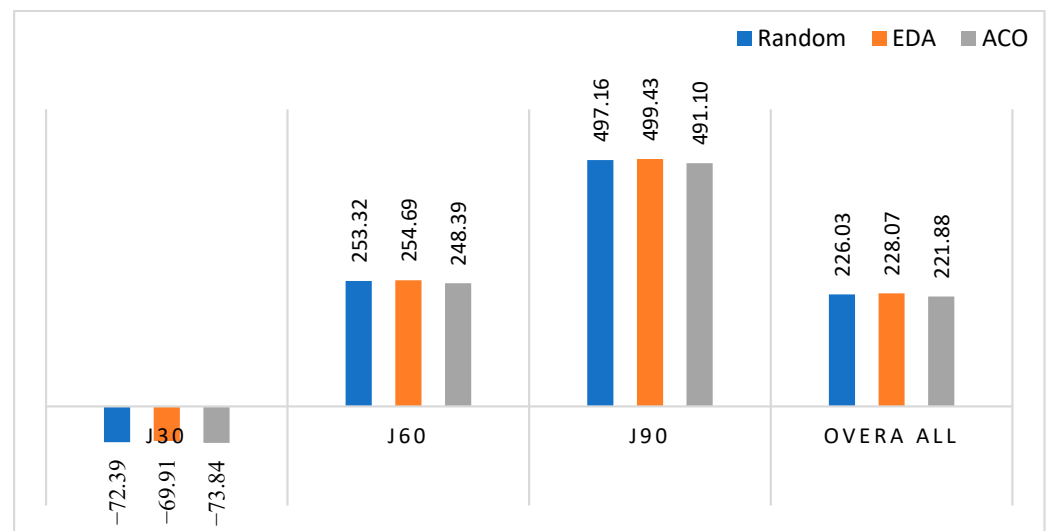

Figure 5. Bar chart of the mean values of the NPV corresponding to each job set against each priority rule for ADP–DCS in terms of NC.

Table 5 presents the average values of the NPV for the j30, j60, and j90 project sets corresponding to the six models for different levels of RF. The efficacy of all algorithms decreases as the RF increases except for EDA. Again, we see that EDA and ACO used as priority for MC have increased the NPV. Figures 6 and 7 again show that ADP(-DCS)-EDA demonstrates superior performance over other methods in terms of RF.

Table 6 shows the average values of the NPV for the j30, j60, and j90 project sets corresponding to the six models for different levels of RS. As the RS increases, the average values of all algorithms increase. In this case, we see that APD_DCS using MC with random priority generated better solutions than with ACO as priority; however, EDA maintained its superiority. The robustness of ADP(-DCS)-EDA in terms of RS is confirmed by Figures 8 and 9 as it consistently outperforms other methods in all different project sizes.

Table 5. Average values of the NPV for the j30, j60, and j90 projects for six different algorithms in terms of RF.

Algorithm	Priority Rule	RF	$\overline{Z}$ for j30	$\overline{Z}$ for j60	$\overline{Z}$ for j90	Overall $\overline{Z}$
ADP	Random	0.25	−94.42	242.17	503.54	217.10
		0.50	−94.79	223.66	459.50	196.12
		0.75	−100.79	210.90	418.82	176.31
		1.00	−95.87	204.44	436.40	181.66
	EDA	0.25	−81.94	265.69	536.65	240.13
		0.50	−81.43	250.52	494.32	221.14
		0.75	−87.34	234.15	451.11	199.31
		1.00	−81.03	299.904	469.18	229.35
	ACO	0.25	−84.32	239.36	526.28	227.11
		0.50	−86.06	221.01	484.56	206.50
		0.75	−93.46	201.69	440.58	182.94
		1.00	−86.59	198.68	458.11	190.07
ADP–DCS	Random	0.25	−69.54	277.07	550.84	252.79
		0.50	−71.90	258.76	503.66	230.17
		0.75	−77.61	241.69	458.55	207.54
		1.00	−70.51	235.77	475.59	213.62
	EDA	0.25	−67.11	277.85	551.04	253.93
		0.50	−69.67	258.54	505.59	231.49
		0.75	−74.97	243.67	461.72	210.14
		1.00	−67.89	238.69	479.39	216.73
	ACO	0.25	−69.06	273.40	544.29	249.54
		0.50	−72.89	253.55	497.70	226.12
		0.75	−80.14	235.21	452.63	202.57
		1.00	−73.29	231.42	469.76	209.30

$\overline{Z}$ denotes the average value of Z.
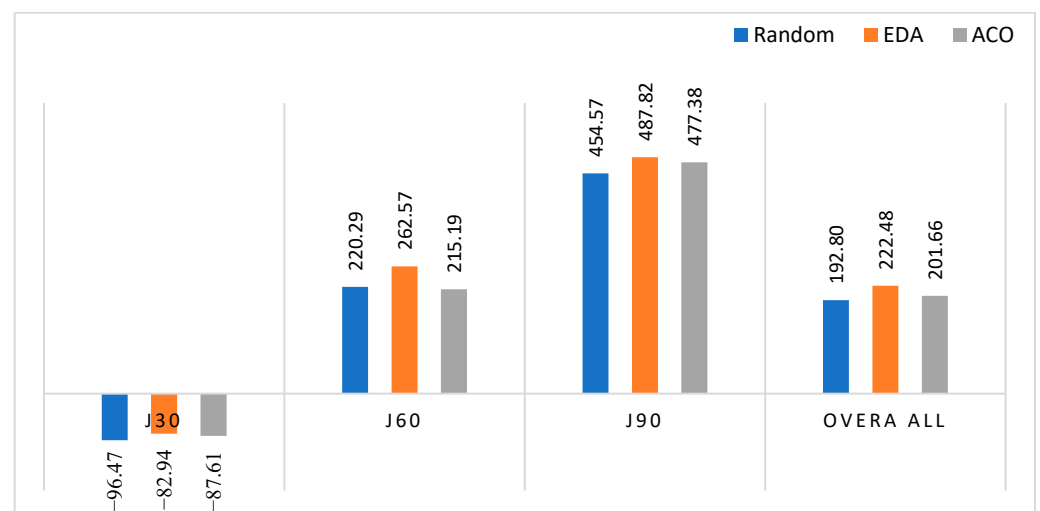
Figure 6. Bar chart of the mean values of the NPV corresponding to each job set against each priority rule for ADP in terms of RF.
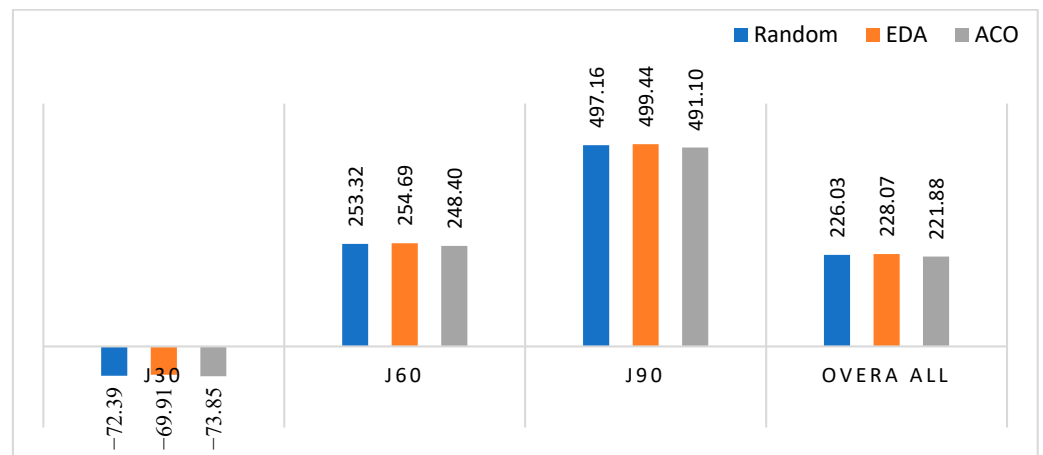
Figure 7. Bar chart of the mean values of the NPV corresponding to each job set against each priority rule for ADP–DCS in terms of RF.

Table 6. Average values of the NPV for j30, j60, and j90 projects for six different algorithms in terms of RS.

| Algorithm	Priority Rule	*RS*	$\overline{Z}$ for j30	$\overline{Z}$ for j60	$\overline{Z}$ for j90	Overal $\overline{Z}$
ADP	Random	0.20	−115.18	113.49	261.08	86.46
		0.50	−88.25	241.39	476.89	210.01
		0.70	−95.49	258.23	524.38	229.04
		1.00	−86.94	268.06	555.91	245.68
	EDA	0.20	−100.47	144.72	302.29	115.51
		0.50	−72.93	276.19	523.70	242.32
		0.70	−80.76	281.97	558.67	253.29
		1.00	−77.59	277.39	566.61	255.47
	ACO	0.20	−108.06	107.51	293.79	97.75
		0.50	−78.70	232.58	506.16	220.01
		0.70	−85.32	252.59	544.02	237.10
		1.00	−78.35	268.06	565.58	251.76
ADP–DCS	Random	0.20	−91.48	151.22	309.74	123.16
		0.50	−64.38	280.27	528.21	248.03
		0.70	−68.95	290.07	567.03	262.72
		1.00	−64.77	291.74	583.66	270.21
	EDA	0.20	−88.38	152.33	310.85	124.93
		0.50	−62.39	282.69	532.31	250.87
		0.70	−67.16	291.13	569.31	264.43
		1.00	−61.70	291.63	585.27	271.73
	ACO	0.20	−93.05	144.51	302.47	117.98
		0.50	−66.85	272.76	518.77	241.56
		0.70	−71.36	285.45	560.22	258.10
		1.00	−64.11	290.88	582.94	269.90

$\overline{Z}$ denotes the average value of Z.
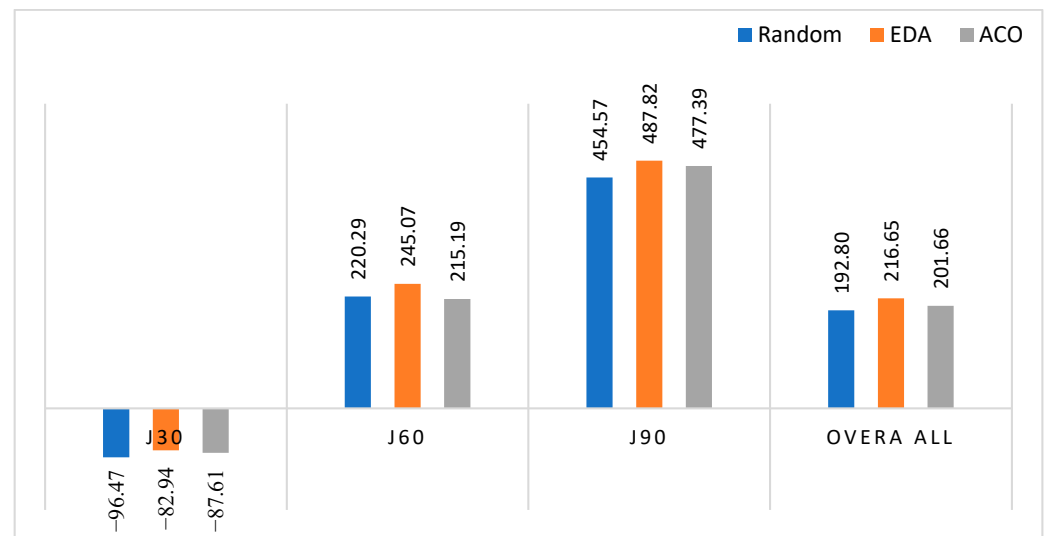
Figure 8. Mean bar chart of the NPV corresponding to each job set against each priority rule for ADP in terms of RS.
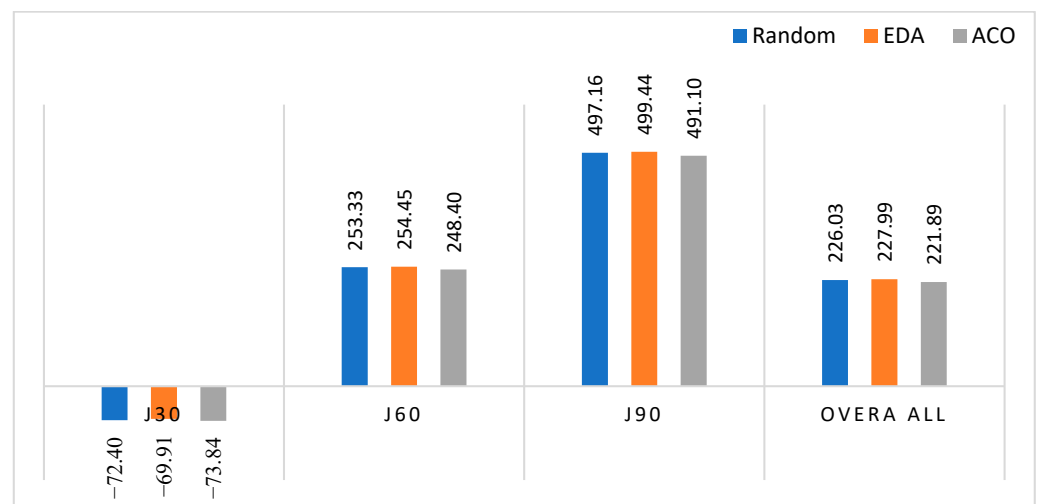

Figure 9. Mean bar chart of the NPV corresponding to each job set against each priority rule for ADP–DCS in terms of RS.

From Tables 3–6, we observe the performance of each model is corelated to the parameter under consideration with ADP–DCS using EDA as priority, producing superior results in most of the cases. As the focus of our study is on PEO, we further investigate the effect of different values of $|E|$ by performing a series of statistical tests to compare their means.

Table 7 shows Anderson's normality test for all project sets. In view of the sample size of 480, the Anderson's test is more suitable than the Shapiro–Wilk test. All the *p*-values except ADP-random with $|E| = 5$ are less than 0.05. This indicates that the objective values are not normally distributed except for ADP-random with $|E| = 5$. To compare the mean values, we therefore perform a non-parametric test with the Mann–Whitney U test for each case.

Table 7. Anderson's normality test for all project sets in terms of $|E|$.

Algorithm	Priority Rule	|E|	A, *p*-Value for j30	A, *p*-Value for j60	A, *p*-Value for j90
ADP	Random	3	1.704, 0.0002	4.545, 0.0000	5.342, 0.0000
		4	0.982, 0.0135	3.906, 0.0000	4.305, 0.0000
		5	0.541, 0.1641	1.985, 0.0000	3.292, 0.0000
		6	0.772, 0.0444	1.436, 0.0011	2.0934, 0.000
	EDA	3	1.496, 0.0007	4.903, 0.0000	6.135, 0.0000
		4	0.856, 0.0277	4.824, 0.0000	4.627, 0.0000
		5	0.751, 0.0502	2.812, 0.0000	3.786, 0.0000
		6	1.458, 0.0009	1.374, 0.0015	2.311, 0.0000
	ACO	3	2.066, 0.000	4.509, 0.0000	5.224, 0.0000
		4	1.193, 0.0041	3.719, 0.0000	3.969, 0.0000
		5	0.828, 0.0325	1.923, 0.0000	3.291, 0.0000
		6	1.337, 0.0017	1.132, 0.0058	1.801, 0.0001
ADP–DCS	Random	3	1.664, 0.0003	5.037, 0.0000	6.363, 0.0000
		4	0.904, 0.0210	4.569, 0.0000	4.770, 0.0000
		5	1.332, 0.0019	2.635, 0.0000	4.140, 0.0000
		6	1.454, 0.0009	1.633, 0.0003	2.589, 0.0000
	EDA	3	1.884, 0.0000	4.965, 0.0000	6.515, 0.0000
		4	0.923, 0.0189	5.059, 0.0000	4.849, 0.0000
		5	1.469, 0.0008	2.905, 0.0000	3.941, 0.0000
		6	1.537, 0.0006	1.294, 0.0023	2.922, 0.0000
	ACO	3	1.603, 0.0003	5.111, 0.0000	5.366, 0.0000
		4	0.663, 0.0826	5.010, 0.0000	4.239, 0.0000
		5	1.394, 0.0013	2.679, 0.0000	3.862, 0.0000
		6	1.066, 0.0083	1.4319, 0.0011	2.539, 0.0000

The *p*-value 0.0000 indicates that the value is smaller than five decimal places.

Table 8 shows the mean difference between ADP and ADP–DCS using MC with random priority. Because all the *p*-values are below 0.05, we conclude that there is a statistically significant difference between the mean values of ADP and ADP–DCS. Based on the values in Table 1, it can be concluded that ADP–DCS is better than ADP when MC with random priority is used.

Table 8. A non-parametric test using the Mann–Whitney U test between ADP and ADP–DCS for random priority.

|E|	W, *p*-Value j30	W, *p*-Value j60	W, *p*-Value j90
3	92,689, 0.0000	96,618, 0.0000	99,976, 0.0000
4	85,251, 0.0000	92,772, 0.0000	98,027, 0.0000
5	72,734, 0.0000	90,775, 0.0000	93,704, 0.0000
6	54,287, 0.0000	84,734, 0.0000	91,958, 0.0000

Table 9 shows the mean differences between ADP and ADP–DCS using MC with EDA priority. Their mean values are statistically significantly different for all j30 instances and for $|E| = 6$ for j60 and j90. For j60 and j90, the p-value improves as $|E|$ increases. In this case, ADP–DCS is better than ADP for smaller projects and when $|E|$ increases for larger projects.

Table 9. A non-parametric test using the Mann–Whitney U test between ADP and ADP–DCS for EDA priority.

$|E|$	W, p-Value j30	W, p-Value j60	W, p-Value j90
3	105,902, 0.0304	110,006, 0.1879	110,887, 0.3154
4	103,263, 0.0055	108,726, 0.1066	110,142, 0.2391
5	89,583, 0.0000	107,760, 0.06605	209,112, 0.1565
6	68,310, 0.0000	105,614, 0.01949	107,070, 0.05843

Similarly, Table 10 presents the mean difference between ADP and ADP–DCS using MC with ACO prioritization. We see that the p-values of both j30 and j60 are less than 0.05. This indicates that ADP–DCS with ACO is statistically better than ADP with ACO for j30 and j60. The superiority of ADP–DCS improves for j90 as $|E|$ increases.

Table 10. Non-parametric test using the Mann-Whitney U test between ADP and ADP–DCS for ACO priority.

$|E|$	W, p-Value j30	W, p-Value j60	W, p-Value j90
3	105,452, 0.0233	97,082, 0.0000	109,940, 0.2208
4	101,898, 0.0019	92,158, 0.0000	108,898, 0.1424
5	88,290, 0.0000	90,745, 0.0000	107,896, 0.0890
6	69,670, 0.0000	84,909, 0.0000	106,188, 0.03591

Table 11 shows the mean difference between ADP with EDA and ADP–DCS with ACO as their priorities. In many cases, the mean differences are not statistically significant. However, the p-value improves with increasing value of $|E|$ for all project groups.

Table 11. A non-parametric test using the Mann–Whitney U test between ADP with EDA and ADP–DCS with ACO is their priority.

$|E|$	W, p-Value j30	W, p-Value j60	W, p-Value j90
3	110,909, 0.3179	115,377, 0.9439	114,834, 0.9321
4	108,934, 0.1447	112,690, 0.4877	113,998, 0.7796
5	95,083, 0.0000	111,910, 0.3816	112,744, 0.5675
6	77,254, 0.0000	110,352, 0.2162	112,499, 0.5296

Table 12 shows the mean difference between ADP with ACO and ADP with EDA as their priorities. Again, it shows that most of the means of j30 and j90 are not statistically different; however, for j60, the means are different with high statistical significance. In addition, the p-values decrease as $|E|$ increases. This suggests that for ADP, using EDA is slightly better for j60 but does not make much of a difference for smaller $|E|$ for j30 and j90.

Table 12. Non-parametric test using the Mann–Whitney U test between ADP with ACO and ADP with EDA as their priorities.

|E|	W, *p*-Value j30	W, *p*-Value j60	W, *p*-Value j90
3	120,671, 0.2028	134,082, 0.0000	120,046, 0.2593
4	122,326, 0.0971	136,325, 0.0000	120,164, 0.2478
5	122,504, 0.0890	137,435, 0.0000	120,058, 0.2581
6	124,479, 0.0308	141,544, 0.0000	121,506, 0.1422

Table 13 shows the mean difference between ADP–DCS with ACO and ADP–DCS with EDA as their priorities. For ADP–DCS, the choice between EDA and ACO is not meaningful as the mean differences are not statistically significant.

Table 13. Non-parametric test using the Mann–Whitney U test between ADP–DCS with ACO and ADP–DCS with EDA as their priorities.

|E|	W, *p*-Value j30	W, *p*-Value j60	W, *p*-Value j90
3	119,980, 0.2658	120,907, 0.2252	119,037, 0.3718
4	120,758, 0.1957	119,508, 0.3745	118,965, 0.3808
5	121,192, 0.1631	119,678, 0.3535	118,796, 0.4025
6	125,314, 0.0185	120,348, 0.2787	120,376, 0.2283

Finally, Table 14 shows the mean difference between ADP with ACO and ADP–DCS with EDA as their priorities. The mean differences are statistically highly significant for all job sets. Based on Table 3, we can conclude that ADP–DCS with EDA is better than ADP–ACO.

Table 14. Non-parametric test using the Mann–Whitney U test between ADP with ACO and ADP–DCS with EDA as their priorities.

|E|	W, *p*-Value j30	W, *p*-Value j60	W, *p*-Value j90
3	100,706, 0.0000	92,147, 0.0000	106,220, 0.0365
4	96,530, 0.0000	88,793, 0.0000	105,306, 0.0213
5	83,050, 0.0000	86,859, 0.0000	104,550, 0.0317
6	61,439, 0.0000	80,468, 0.0000	101,088, 0.0010

It is important to recognize that for each cardinality of |E|, a total of 480 distinct projects were analyzed for each project set (i.e., j30, j60, and j90). The average values of the NPV corresponding to four different parameters are shown in Tables 3–6. To gain further insights of the models' performances, bar charts of the mean values of the NPV in terms of four parameters for all possible models for various project sizes are shown in Figures 2–9. For different values of $|E|$, after performing normality test, we compared the mean difference using the Mann–Whiteney U test in seven different scenarios. ADP–DCS leads to better results than ADP alone. This makes sense as DCS further improves the solution of ADP. Using EDA and ACO for MC as a priority also improved the solution. Although the choice between EDA and ACO is difficult, EDA performed relatively better, and its performance improves with increasing $|E|$.

5. Discussion

The authors of [70] mention that the parameters considered for generation of projects from *ProGen* reflect real-life practical projects. In this study, having made a comprehensive analysis based on all important parameters, the observations and conclusions made above

hold for project sizes between 30 and 90 jobs. That is, for each parameter we analyzed, the performances of different models by varying values of parameters of interest resulted in a sensitivity analysis. We observe that in all instances, ADP(–DCS)–EDA consistently outperforms other models, indicating its robustness. Our statistical analysis performed on varying cardinalities of E also confirms ADP(–DCS)–EDA's superiority. We must also note that by nature of our experimental design, the hybrid methods proposed are theoretically superior to their standalone counterparts. Our analysis indicates that the approximation of policy rule using distribution estimated by EDA is better than random approximation and approximation by ACO. It can also be seen that policy approximation by ACO is relatively better in most cases than random approximations.

However, considering the NP-hard classification of the problem at hand, the results obtained may not represent the theoretical optimum, suggesting that there is room for the further refinement and improvement of these values. Furthermore, the stochastic nature of the algorithms under consideration means that repeated executions may lead to slightly varying results. It is also important to note that no comparison of computation times was made in this study, as such metrics are significantly affected by the computational capacity of the processing machines used. Moreover, because our experiment involved j30, j60, and j90, the results may not hold true for larger problem sets (projects with more than 90 jobs) or smaller projects (less than 30 activities). An analysis of these models on larger projects remains out of the scope of this study but is reserved for the future.

This study offers numerous practical implications. First, as RCPSPDC–PEO is the most logical mode of agreement a contractor and a client can enter, practitioners can employ our models off-the-shelves to maximize their profitability. As indicated in the introduction, these models have multitude of advantages, including the capacity to execute parallelly on multiple cores of computing systems. This not only enhances computational efficiency but also makes our method accessible and scalable. Our models also offer customization capabilities to meet the needs of practitioners. For instance, an user can only employ the ADP part of the model in a quicker manner to determine a reasonably good schedule. Customizable models like ours are also important tools for an agile project management, which is prevalent in innovative product development. Second, though we have studied RCPSPDC–PEO from the contractor's perspective, it can be easily implemented from the client's perspective by swapping the cash flows (i.e., the cash inflow of the contractor becomes the cash outflow of the contractor). Our models could then be used to guide the formulation of pareto-optimal contract designs which foster fair and efficient contracting practices that protect the interests of both contractors and clients. Third, as our proposed models are versatile, they can be adapted for various industries such as software development, manufacturing, and logistics, whose objectives may not necessarily be financial optimization.

As ADP and Bellman's equations underpin reinforcement learning (RL), our research lays the foundation for integrating RL into project scheduling. An RL for project scheduling will involve generating a schedule within a short time that optimizes certain pre-defined objectives, given the project characteristics. Our proposed model generate schedules for RCPSPDC–PEO consuming less computation time, but it is problem-specific as it has not learned other project characteristics. RL that solves RCPSPDC–PEO can benefit from our model in multiple ways. First, our model can help a model-free RL agent learn by using the generated schedules to update value function estimates. Second, our model and meta-heuristics can be used with policy iteration methods similar to our experimental design.

This study focused on the deterministic single-mode RCPSPDC–PEO. However, not all project characteristics can fit into this scenario. Often, due to unavoidable situations, the activity durations are disturbed or not known in advance. Moreover, some tasks can be performed in one of several modes. These situations lead to stochastic and multi-mode RCPSPDC–PEOs. Through stochastic RCPSP using ADP in the literature, a similar and robust extension can be made using our model to solve stochastic RCPSPDC–PEO. Multi-mode RCPSPDC–PEO can also benefit from our models as solving multi-mode version

involves determining the mode and the job execution times. The dynamic nature of our framework can determine both the mode and job execution times in one go using the MC simulations. Thus, we plan to extend this approach to address robust and stochastic single-mode and multi-mode RCPSPDC–PEOs.

Furthermore, we intend to explore more metaheuristics, in addition to EDA and ACO, to enhance the priority rules. We will also consider alternative local search techniques beyond DCS in the future.

Because our problem is NP-hard, potential improvements to our proposed method could be to expand the policy decision space or improve the search techniques after the base schedule is generated using PSGS–ADP. One promising approach to improve the search technique could be to use another set of metaheuristics hybrids if computational resources permit.

6. Conclusions

Studies have shown that integrating various algorithms into hybrid metaheuristics can outshine singular techniques by drawing on their mutual advantages and balancing their individual weakness. This preference for hybrid methods is in line with the *NFLT*. This study proposed a hybridization of ADP with three different metaheuristics methods (i.e., EDA, ACO, and DCS) to overcome their inherent weakness for RCPSPDC–POE. To implement ADP within this problem context, we incorporated the MDP framework with the PSGS. PSGS iteratively explores various MDP states by applying an optimal policy to schedule an activity. Given the computational intractability of MDP to determine an exact optimal policy due to the curse of dimensionality, we approximated the optimal policies using heuristics-based MC simulations. This process resulted in a dynamic and adaptive solution for RCPSPDC–PEO.

To guide and refine the MC simulations, we have introduced two priority rules derived from EDA and ACO in addition to random search. These priority rules are applied within the MC simulations using an Epsilon-Greedy algorithm to assist in the selection of policy. This has enabled effective search strategies. The solutions of ADP are further enhanced using DCS, which employs various local and global search techniques based on Lévy flight paths, showcasing the potential for significant improvements in scheduling outcomes. This process resulted in six hybrid architectures that use 13 distinct heuristics as a core contribution of our work. The hybridization of conventional operations research techniques (i.e., ADP) with metaheuristics (i.e., EDA, ACO, and DCS), a combination previously unexplored within the literature for our problem, diminishes their individual weaknesses and harnesses their collective strengths besides reducing computation burden due to the dynamic nature of ADP. These hybrid approaches have demonstrated their capabilities for yielding adequate solutions on practical benchmark problems, underscoring their relevance and applicability to real-world project scheduling challenges.

In addition, we parallelized our architectures on a multi-core computer. This approach not only enhances computational efficiency, but also makes our method accessible and scalable, catering to the demands of large-scale project management scenarios. Moreover, as ADP underpins reinforcement learning (RL), our research serves as a foundational step towards integrating RL into project scheduling, opening new avenues for future investigations and applications in adaptive and intelligent project management solutions.

The findings from the experiment on j30, j60, and j90 showed that using ADP–DCS gives better results than using ADP alone, which is not surprising as DCS is known to improve the solution of ADP. Additionally, incorporating EDA and ACO as priorities for MC simulations further improved the solutions. While choosing between EDA and ACO can be challenging, the performance of EDA was relatively better and showed improvement with the increasing value of $|E|$. EDA also showed superior performance over ACO and random priority in all three project characters. The validity of these findings for very large or small projects, although it remains untested in the current study, will be our future endeavor. This study's scope was limited to deterministic single-mode RCPSPDC–PEO; future work

will explore robust and stochastic versions, additional metaheuristics and alternative local search strategies. Given the NP-hard nature of the problem, further enhancements could include broadening the policy decision space or refining search techniques following initial schedule generation with PSGS–ADP.

Author Contributions: Conceptualization, T.P.; Methodology, T.P.; Validation, T.P. and T.G.; Formal analysis, T.P.; Investigation, T.P. and T.G.; Resources, T.G; Data curation, T.P.; Writing—original draft, T.P.; Writing—review & editing, T.G.; Visualization, T.P.; Supervision, T.G.; Project administration, T.G. All authors have read and agreed to the published version of the manuscript.

Institutional Review Board Statement: This article does not contain any studies performed by any of the authors with human participants or animals.

Data Availability Statement: The dataset used in the research is PSPLIB [69]. Additional information regarding cost and income associated with each activity in each project was made on the dataset. A folder containing this information with our computation results can be downloaded from https://drive.google.com/drive/folders/17dbCm1NyCE9l4vy6tHCZ95tcniIWaCC_?usp=sharing (accessed on 1 April 2024).

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

References

1. Phuntsho, T.; Gonsalves, T. Hybrid of Simplified Small World and Group Counseling Optimization Algorithms with Matured Random Initialization and Variable Insertion Neighborhood Search Technique to Solve Resource Constrained Project Scheduling Problems with Discounted Cash Flows. In Proceedings of the 2022 5th Artificial Intelligence and Cloud Computing Conference, Osaka, Japan, 17–19 December 2022; ACM: Piscataway, NJ, USA, 2022; pp. 66–72.
2. Klimek, M. Financial Optimization of the Resource-Constrained Project Scheduling Problem with Milestones Payments. *Appl. Sci.* **2021**, *11*, 661. [CrossRef]
3. Szmerekovsky, J.G. The Impact of Contractor Behavior on the Client's Payment-Scheduling Problem. *Manag. Sci.* **2005**, *51*, 629–640. [CrossRef]
4. Mika, M.; Waligóra, G.; Węglarz, J. Simulated Annealing and Tabu Search for Multi-Mode Resource-Constrained Project Scheduling with Positive Discounted Cash Flows and Different Payment Models. *Eur. J. Oper. Res.* **2005**, *164*, 639–668. [CrossRef]
5. Ulusoy, G.; Sivrikaya-Şerifoğlu, F.; Şahin, Ş. Four Payment Models for the Multi-Mode Resource Constrained Project Scheduling Problem with Discounted Cash Flows. *Ann. Oper. Res.* **2001**, *102*, 237–261. [CrossRef]
6. Leyman, P.; Vanhoucke, M. Payment Models and Net Present Value Optimization for Resource-Constrained Project Scheduling. *Comput. Ind. Eng.* **2016**, *91*, 139–153. [CrossRef]
7. Vanhoucke, M.; Demeulemeester, E.; Herroelen, W. Progress Payments in Project Scheduling Problems. *Eur. J. Oper. Res.* **2003**, *148*, 604–620. [CrossRef]
8. Teich, J.E.; Wallenius, H.; Wallenius, J.; Zionts, S. Identifying Pareto-Optimal Settlements for Two-Party Resource Allocation Negotiations. *Eur. J. Oper. Res.* **1996**, *93*, 536–549. [CrossRef]
9. Ehtamo, H.; Hämäläinen, R.P. Interactive Multiple-Criteria Methods for Reaching Pareto Optimal Agreements in Negotiations. *Group Decis. Negot.* **2001**, *10*, 475–491. [CrossRef]
10. Dayanand, N.; Padman, R. Project Contracts and Payment Schedules: The Client's Problem. *Manag. Sci.* **2001**, *47*, 1654–1667. [CrossRef]
11. Bahrami, F.; Moslehi, G. Study of Payment Scheduling Problem to Achieve Client–Contractor Agreement. *Int. J. Adv. Manuf. Technol.* **2013**, *64*, 497–511. [CrossRef]
12. Leyman, P.; Vanhoucke, M. A New Scheduling Technique for the Resource–Constrained Project Scheduling Problem with Discounted Cash Flows. *Int. J. Prod. Res.* **2015**, *53*, 2771–2786. [CrossRef]
13. Asadujjaman, M.D.; Rahman, H.F.; Chakrabortty, R.K.; Ryan, M.J. Multi-Operator Immune Genetic Algorithm for Project Scheduling with Discounted Cash Flows. *Expert Syst. Appl.* **2022**, *195*, 116589. [CrossRef]
14. Raidl, G.R.; Puchinger, J.; Blum, C. Metaheuristic Hybrids. In *Handbook of Metaheuristics*; Gendreau, M., Potvin, J.-Y., Eds.; International Series in Operations Research & Management Science; Springer International Publishing: Cham, Switzerland, 2019; Volume 272, pp. 385–417, ISBN 978-3-319-91085-7.
15. Ting, T.O.; Yang, X.-S.; Cheng, S.; Huang, K. Hybrid Metaheuristic Algorithms: Past, Present, and Future. In *Recent Advances in Swarm Intelligence and Evolutionary Computation*; Yang, X.-S., Ed.; Studies in Computational Intelligence; Springer International Publishing: Cham, Switzerland, 2015; Volume 585, pp. 71–83, ISBN 978-3-319-13825-1.

16. Raidl, G.R. A Unified View on Hybrid Metaheuristics. In *Hybrid Metaheuristics*; Almeida, F., Blesa Aguilera, M.J., Blum, C., Moreno Vega, J.M., Pérez Pérez, M., Roli, A., Sampels, M., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2006; Volume 4030, pp. 1–12, ISBN 978-3-540-46384-9.

17. Joyce, T.; Herrmann, J.M. A Review of No Free Lunch Theorems, and Their Implications for Metaheuristic Optimisation. In *Nature-Inspired Algorithms and Applied Optimization*; Yang, X.-S., Ed.; Studies in Computational Intelligence; Springer International Publishing: Cham, Switzerland, 2018; Volume 744, pp. 27–51, ISBN 978-3-319-67668-5.

18. Puterman, M.L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*; John Wiley & Sons: Hoboken, NJ, USA, 2014.

19. 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]

20. Fianu, S.; Davis, L.B. A Markov Decision Process Model for Equitable Distribution of Supplies under Uncertainty. *Eur. J. Oper. Res.* **2018**, *264*, 1101–1115. [CrossRef]

21. Novoa, C.; Storer, R. An Approximate Dynamic Programming Approach for the Vehicle Routing Problem with Stochastic Demands. *Eur. J. Oper. Res.* **2009**, *196*, 509–515. [CrossRef]

22. Xie, F.; Li, H.; Xu, Z. An Approximate Dynamic Programming Approach to Project Scheduling with Uncertain Resource Availabilities. *Appl. Math. Model.* **2021**, *97*, 226–243. [CrossRef]

23. Asadujjaman, M.D.; Rahman, H.F.; Chakrabortty, R.K.; Ryan, M.J. An Immune Genetic Algorithm for Solving NPV-Based Resource Constrained Project Scheduling Problem. *IEEE Access* **2021**, *9*, 26177–26195. [CrossRef]

24. Vanhoucke, M. A Scatter Search Heuristic for Maximising the Net Present Value of a Resource-Constrained Project with Fixed Activity Cash Flows. *Int. J. Prod. Res.* **2010**, *48*, 1983–2001. [CrossRef]

25. Powell, W.B. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*; John Wiley & Sons: Hoboken, NJ, USA, 2011.

26. Bibiks, K.; Hu, F.; Li, J.-P.; Smith, A. Discrete Cuckoo Search for Resource Constrained Project Scheduling Problem. In Proceedings of the 2015 IEEE 18th International Conference on Computational Science and Engineering, Porto, Portugal, 21–23 October 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 240–245.

27. Bibiks, K.; Hu, Y.-F.; Li, J.-P.; Pillai, P.; Smith, A. Improved Discrete Cuckoo Search for the Resource-Constrained Project Scheduling Problem. *Appl. Soft Comput.* **2018**, *69*, 493–503. [CrossRef]

28. Quoc, H.D.; Nguyen The, L.; Doan, C.N.; Phan Thanh, T. Solving Resource Constrained Project Scheduling Problem by a Discrete Version of Cuckoo Search Algorithm. In Proceedings of the 2019 6th NAFOSTED Conference on Information and Computer Science (NICS), Hanoi, Vietnam, 12–13 December 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 73–76.

29. Wang, L.; Fang, C. An Effective Estimation of Distribution Algorithm for the Multi-Mode Resource-Constrained Project Scheduling Problem. *Comput. Oper. Res.* **2012**, *39*, 449–460. [CrossRef]

30. Fang, C.; Kolisch, R.; Wang, L.; Mu, C. An Estimation of Distribution Algorithm and New Computational Results for the Stochastic Resource-Constrained Project Scheduling Problem. *Flex. Serv. Manuf. J.* **2015**, *27*, 585–605. [CrossRef]

31. Merkle, D.; Middendorf, M.; Schmeck, H. Ant Colony Optimization for Resource-Constrained Project Scheduling. *IEEE Trans. Evol. Comput.* **2002**, *6*, 333–346. [CrossRef]

32. Zhao, J.-C.; Zhang, Y.-M.; Qu, H.-Y.; Qi, H. Ant Colony Optimization for Resource-Constrained Multi-Project Scheduling. In Proceedings of the 2009 International Workshop on Intelligent Systems and Applications, Wuhan, China, 23–24 May 2009; IEEE: Piscataway, NJ, USA, 2009; pp. 1–5.

33. Smith-Daniels, D.E.; Aquilano, N.J. Using a late-start resource-constrained project schedule to improve project net present value. *Decis. Sci.* **1987**, *18*, 617–630. [CrossRef]

34. Phuntsho, T.; Gonsalves, T. Maximizing the Net Present Value of Resource-Constrained Project Scheduling Problems Using Recurrent Neural Network with Genetic Algorithm. In Proceedings of the 2023 International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT), Bengaluru, India, 5–7 January 2023; IEEE: Piscataway, NJ, USA, 2023; pp. 524–530.

35. Liu, C.; Yang, S. A Serial Insertion Schedule Generation Scheme for Resource-Constrained Project Scheduling. *J. Comput.* **2011**, *6*, 2365–2375. [CrossRef]

36. Kim, J.-L.; Ellis, R.D. Comparing Schedule Generation Schemes in Resource-Constrained Project Scheduling Using Elitist Genetic Algorithm. *J. Constr. Eng. Manag.* **2010**, *136*, 160–169. [CrossRef]

37. Kim, J.-L. *Proposed Methodology for Comparing Schedule Generation Schemes in Construction Resource Scheduling*; ACM: New York, NY, USA, 2009.

38. Pinha, D.C.; Ahluwalia, R.S.; Carvalho, A.N. Parallel Mode Schedule Generation Scheme. *IFAC-Pap.* **2015**, *48*, 794–799. [CrossRef]

39. Demeulemeester, E.; Herroelen, W.S.; Herroelen, W. *Project Scheduling: A Research Handbook*; International Series in Operations Research & Management Science; Kluwer Academic Publishers: Boston, MA, USA, 2002; ISBN 978-1-4020-7051-8.

40. Zhao, C.; Ke, H.; Chen, Z. Uncertain Resource-Constrained Project Scheduling Problem with Net Present Value Criterion. *J. Uncertain. Anal. Appl.* **2016**, *4*, 12. [CrossRef]

41. Phuntsho, T.; Gonsalves, T. Solving NPV-Based Resource Constrained Project Scheduling Problem Using Genetic Algorithm. In Proceedings of the 2022 10th International Conference on Information and Education Technology (ICIET), Matsue, Japan, 9–11 April 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 409–414.

42. Chen, W.-N.; Zhang, J.; Chung, H.S.-H.; Huang, R.-Z.; Liu, O. Optimizing Discounted Cash Flows in Project Scheduling—An Ant Colony Optimization Approach. *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.* **2010**, *40*, 64–77. [CrossRef]

43. Gu, H.; Schutt, A.; Stuckey, P.J. A Lagrangian Relaxation Based Forward-Backward Improvement Heuristic for Maximising the Net Present Value of Resource-Constrained Projects. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*; Gomes, C., Sellmann, M., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2013; Volume 7874, pp. 340–346, ISBN 978-3-642-38170-6.

44. Rostami, S.; Creemers, S.; Leus, R. New Strategies for Stochastic Resource-Constrained Project Scheduling. *J. Sched.* **2018**, *21*, 349–365. [CrossRef]

45. Ballestín, F. When It Is Worthwhile to Work with the Stochastic RCPSP? *J. Sched.* **2007**, *10*, 153–166. [CrossRef]

46. Bertsekas, D.P. *Dynamic Programming and Optimal Control*, 4th ed.; Athena Scientific Optimization and Computation Series; Athena Scientific: Nashua, NH, USA, 2012; ISBN 978-1-886529-43-4.

47. Ballestín, F.; Leus, R. Resource-Constrained Project Scheduling for Timely Project Completion with Stochastic Activity Durations. *Prod. Oper. Manag.* **2009**, *18*, 459–474. [CrossRef]

48. 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]

49. Bertsekas, D.P. Rollout Algorithms for Discrete Optimization: A Survey. In *Handbook of Combinatorial Optimization*; Pardalos, P.M., Du, D.-Z., Graham, R.L., Eds.; Springer: New York, NY, USA, 2013; pp. 2989–3013, ISBN 978-1-4419-7996-4.

50. Li, X.; Yin, M. A Discrete Artificial Bee Colony Algorithm with Composite Mutation Strategies for Permutation Flow Shop Scheduling Problem. *Sci. Iran.* **2012**, *19*, 1921–1935. [CrossRef]

51. Choi, J.; Realff, M.J.; Lee, J.H. Dynamic Programming in a Heuristically Confined State Space: A Stochastic Resource-Constrained Project Scheduling Application. *Comput. Chem. Eng.* **2004**, *28*, 1039–1058. [CrossRef]

52. Baptiste, P.; Le Pape, C.; Nuijten, W. *Constraint-Based Scheduling Applying Constraint Programming to Scheduling Problems*; Springer: Berlin/Heidelberg, Germany, 2013; ISBN 978-1-4613-5574-8.

53. Kolisch, R.; Hartmann, S. Experimental Investigation of Heuristics for Resource-Constrained Project Scheduling: An Update. *Eur. J. Oper. Res.* **2006**, *174*, 23–37. [CrossRef]

54. Xu, N.; McKee, S.A.; Nozick, L.K.; Ufomata, R. Augmenting Priority Rule Heuristics with Justification and Rollout to Solve the Resource-Constrained Project Scheduling Problem. *Comput. Oper. Res.* **2008**, *35*, 3284–3297. [CrossRef]

55. Li, H.; Womer, K. Scheduling Projects with Multi-Skilled Personnel by a Hybrid MILP/CP Benders Decomposition Algorithm. *J. Sched.* **2009**, *12*, 281–298. [CrossRef]

56. Sammut, C.; Webb, G.I. (Eds.) Markov Process. In *Encyclopedia of Machine Learning*; Springer: Boston, MA, USA, 2011; p. 646, ISBN 978-0-387-30768-8.

57. Gross, E. On the Bellman's Principle of Optimality. *Phys. Stat. Mech. Its Appl.* **2016**, *462*, 217–221. [CrossRef]

58. Krylov, N.V. On Bellman's Equations with VMO Coefficients. *Methods Appl. Anal.* **2010**, *17*, 105–122. [CrossRef]

59. Pardalos, P.M.; Du, D.; Graham, R.L. (Eds.) *Handbook of Combinatorial Optimization*, 2nd ed.; Springer Reference; Springer: New York, NY, USA, 2013; ISBN 978-1-4419-7996-4.

60. Grobler, J.; Engelbrecht, A.P.; Kendall, G.; Yadavalli, V.S.S. Investigating the Use of Local Search for Improving Meta-Hyper-Heuristic Performance. In Proceedings of the 2012 IEEE Congress on Evolutionary Computation, Brisbane, Australia, 10–15 June 2012; IEEE: Piscataway, NJ, USA, 2012; pp. 1–8.

61. Blot, A.; Kessaci, M.-É.; Jourdan, L. Survey and Unification of Local Search Techniques in Metaheuristics for Multi-Objective Combinatorial Optimisation. *J. Heuristics* **2018**, *24*, 853–877. [CrossRef]

62. Tsai, H.-K.; Yang, J.-M.; Kao, C.-Y. Solving Traveling Salesman Problems by Combining Global and Local Search Mechanisms. In Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600), Honolulu, HI, USA, 12–17 May 2002; IEEE: Piscataway, NJ, USA, 2002; Volume 2, pp. 1290–1295.

63. Pantaleo, E.; Facchi, P.; Pascazio, S. Simulations of Lévy Flights. *Phys. Scr.* **2009**, *T135*, 014036. [CrossRef]

64. Pavlyukevich, I. Lévy Flights, Non-Local Search and Simulated Annealing. *J. Comput. Phys.* **2007**, *226*, 1830–1844. [CrossRef]

65. Đorić, D. New Generalizations of Cauchy Distribution. *Commun. Stat. Theory Methods* **2011**, *40*, 3764–3776. [CrossRef]

66. Katoch, S.; Chauhan, S.S.; Kumar, V. A Review on Genetic Algorithm: Past, Present, and Future. *Multimed. Tools Appl.* **2021**, *80*, 8091–8126. [CrossRef] [PubMed]

67. Graves-Morris, P.R.; Roberts, D.E.; Salam, A. The Epsilon Algorithm and Related Topics. *J. Comput. Appl. Math.* **2000**, *122*, 51–80. [CrossRef]

68. Tuong-Bach, N.; Isabelle, S. Epsilon-Covering: A Greedy Optimal Algorithm for Simple Shapes. In Proceedings of the CCCG 2016 28th Canadian Conference on Computational Geometry, Vancouver, BC, Canada, 3–5 August 2016.

69. Kolisch, R.; Sprecher, A. PSPLIB—A Project Scheduling Problem Library. *Eur. J. Oper. Res.* **1997**, *96*, 205–216. [CrossRef]

70. 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]