

# An Efficient Approximate Dynamic Programming Approach for Resource-Constrained Project Scheduling with Uncertain Task Duration

Alireza Etmianiesfahani<sup>1</sup><sup>a</sup>, Hanyu Gu<sup>1</sup><sup>b</sup>, Leila Moslemi Naeni<sup>2</sup><sup>c</sup> and Amir Salehipour<sup>3</sup><sup>d</sup>

<sup>1</sup>*School of Mathematical and Physical Sciences, University of Technology Sydney, Sydney, Australia*

<sup>2</sup>*School of the Built Environment, University of Technology Sydney, Sydney, Australia*

<sup>3</sup>*The University of Sydney Business School, The University of Sydney, Sydney, Australia*

**Keywords:** Approximate Dynamic Programming, RCPSP, Priority Rule, Uncertainty.

**Abstract:** The resource-constrained project scheduling problems (RCPSP) with uncertainties have been widely studied. The existing approaches focus on open-loop task scheduling, and only a few research studies develop a dynamic and adaptive closed-loop policy as it is regarded as computationally time-consuming. In this paper, an approximate dynamic programming (ADP) approach is developed to solve the RCPSPs with stochastic task duration (SRCPSP). The solution from a deterministic average project is utilised to reduce the computational burden associated with the roll-out policy, and a parameter is introduced in the roll-out policy to control the search strength. We test the proposed approach on 960 benchmark instances from the well-known library PSPLIB with 30 and 60 tasks and compare the results with the state-of-the-art algorithms for solving the SRCPSPs. The results show that our average-project-based ADP (A-ADP) approach provides competitive solutions in a short computational time. The investigation of the characteristics of the instances also discloses that when resources are tight, it is more important to intensify the search in the roll-out policy.


## 1 INTRODUCTION


In past decades, significant attention has been devoted to scheduling projects under resource constraints. In 1969, Pritsker et al. introduced the Resource-Constrained Project Scheduling Problem (RCPSP) (Pritsker et al., 1969), which involves determining the optimal scheduling of tasks subject to precedence constraints and resource limitations over time.


While the RCPSP has many practical applications, it has limitations in addressing uncertain real-world problems, such as inaccurate estimations, new or dropped tasks, and unforeseen conditions. This has led to the development of a new research area focusing on the stochastic resource-constrained project scheduling problem (SRCPSP), which extends the RCPSP to include projects with stochastic task duration and aims to minimise the expected makespan (Schwindt and Zimmermann, 2015; Bruni et al., 2009).


Most of the literature on the SRCPSP focuses its attention on the development of heuristics and meta-heuristics (Cai et al., 2024; Guo et al., 2021). Since the task duration is random in the SRCPSP, the solution to the problem is not a fixed schedule but a scheduling policy, which determines which task to start next at each decision point. A set of policy classes is defined in (Chen et al., 2018). The policy serves a similar function to the Schedule Generation Scheme (SGS) in the RCPSP (Kolisch and Hartmann, 1999) and is used as a scheduling rule in a multi-stage decision process for the project's execution.

Most of the available approaches for solving SRCPSP are considered open-loop policies, which provide an order of all tasks at the start of the time horizon and have limited capabilities to adapt to new information in the decision process. The alternative approach is dynamic programming (DP), which allows decision-makers to utilise new information that arises between decision points (Li and Womer, 2015). These methods are called the closed-loop policy (Bertsekas, 2007). The closed-loop policy is computationally more challenging than the open-loop policy since an optimisation problem needs to be solved at each decision point.

<sup>a</sup>  <https://orcid.org/0000-0002-9780-8262>

<sup>b</sup>  <https://orcid.org/0000-0003-2035-2583>

<sup>c</sup>  <https://orcid.org/0000-0002-2577-3611>

<sup>d</sup>  <https://orcid.org/0000-0003-4866-1396>

It has been reported in (Stork, 2000) that a simple policy based on the solution of the so-called average project works well in most cases. The average project is a deterministic RCPSP model in which each task has the mean value as task duration. Since the deterministic RCPSP is solved only once, the approach is very efficient compared to the closed-loop approach. However, the performance of this approach deteriorates when uncertainties are significant. This study aims to develop an efficient average-project-based approximate dynamic programming (A-ADP) approach to solve SRCPSP.

Our contributions in this study include:

- proposing an efficient A-ADP approach for solving SRCPSPs.
- reducing the computational burden of closed-loop policy using the solution obtained from the average project.
- investigating a wide range of problems with 30 tasks and identifying the project characteristic that has the most significant impact on the performance of the A-ADP approach.
- investigating the impact of minimum slack (MSLK) priority rule on the proposed A-ADP approach in solving instances with 30 and 60 tasks.
- providing competitive results to the state-of-the-art algorithms for instances with 30 and 60 tasks in a short computational time.

## 2 MDP FORMULATION OF SRCPSP

This section introduces the SRCPSP's assumptions and problem scenario. Then its MDP formulation is presented, which serves as the basis for developing ADP algorithms in the upcoming sections.

### 2.1 Assumptions

Consider a project network represented by an activity-on-node (AON) diagram denoted as  $G(V, E)$ , where the set of tasks in the project is  $V = \{0, 1, \dots, n, n+1\}$ . Here, tasks 0 and  $n+1$  represent the beginning and end of the project, respectively. The AON network is assumed to be acyclic; once a task is started, it cannot be interrupted. The set of edges  $E$  represents the precedence relationships among the tasks, indicating that a task  $j$  cannot begin before task  $i$  has been completed if  $(i, j) \in E$ . The project requires utilising a set of resources  $K = \{1, 2, \dots, m\}$  for its execution. Each resource  $k \in K$  has a limited capacity of  $R_k$  units

available during each period. The execution of a task  $j$  requires  $r_{jk}$  units of resource  $k$ . The duration of a task  $j$ , denoted by a random variable  $D_j$ , follows a probability distribution (PD) known to the decision-maker. The duration of a task can only be observed once the task has been completed. The objective is to determine a feasible policy regarding the precedence and resource constraints that minimises the project's expected completion time (makespan).

### 2.2 SRCPSP Formulation in MDP Model

We considered the MDP model of the SRCPSP as suggested in (Li and Womer, 2015). This model includes stages, states, decisions, transition processes and cost functions.

A decision stage is defined as a point in time when a task is finished. We use  $t_i$  to represent the time associated with the  $i$ th decision stage, where  $i$  ranges from 1 to the number of decision stages, denoted by  $L$ . The state  $S_i$  contains all the necessary information to decide at each stage  $i$ . It includes the completed tasks, active tasks, the duration of the tasks that have been executed, the observed duration of active tasks up to the current stage, and the start times of all completed and active tasks. The decision  $x_i$  is a set of tasks that can start at stage  $i$  and satisfy both precedence and resource constraints.

The other component of the SRCPSP model is the transition function from the current stage  $i$  to the next stage  $i+1$ , which is denoted as  $S^M(\cdot)$ .

$$S_{i+1} = S^M(S_i, x_i, D_{j:j \in A_i}) \quad (1)$$

$A_i$  is the set of active tasks at stage  $i$ . Given the decision taken at stage  $i$   $x_i$ , and the set of observed duration of active tasks  $D_{j:j \in A_i}$ , the transition function  $S^M(\cdot)$  maps the current state  $S_i$  to the next state  $S_{i+1}$ . Equation (1) shows that the next state depends only on the current state, the decision made, and the realised duration of active tasks, not on the decision history. This is known as the Markov property.

At stage  $i \in \{1, \dots, L-1\}$ , for the time point  $t_i$ , the cost of transition from  $S_i$  to  $S_{i+1}$  is denoted by  $g(S_i, x_i, S_{i+1})$ , so the cost function from stage  $i$  and state  $S_i$ , can be written as:

$$J_i(S_i) = \mathbb{E} \left\{ \sum_{j=i}^{L-1} g(S_j, x_j, S_{j+1}) \right\} \quad (2)$$

The goal is to minimise the expected project makespan by selecting the optimal policy. Let the cost-to-go function, or the objective function from stage  $i$  and state  $S_i$  when using the optimal policy  $\pi$  is:

$$V_i(S_i) = \mathbb{E} \left\{ \sum_{j=i}^{L-1} g(S_j, x_j^\pi, S_{j+1}) \right\} \quad (3)$$

The well-known recursion Bellman function (Bellman, 2010) computes the optimal policy which is the optimal set of tasks started at each stage  $i$  as below:

$$x_i^\pi = \arg \min_{x \in \mathcal{X}(S_i)} \mathbb{E} \{ g(S_i, x_i, S_{i+1}) + V_{i+1}(S_{i+1}) \} \quad (4)$$

where  $\mathcal{X}(S_i)$  is the set of eligible tasks.

### 3 PROPOSED A-ADP APPROACH FOR SRCPSP

A successful approximation paradigm for solving MDPs is the roll-out policy for combinatorial optimisation (Bertsekas et al., 1997). At each stage, a roll-out policy replaces the exact cost-to-go function  $V_i(S_i)$  with a heuristic  $H$  to approximate it, denoted as  $V_i^H(S_i)$ , which is then used to decide in the current state. In this policy at each state  $S_i$ , a decision  $x_i^{PH}$  is obtained using the heuristic  $H$  as

$$x_i^{PH} = \arg \min_{x \in \mathcal{X}(S_i)} \mathbb{E} \{ g(S_i, x_j, S_{i+1}) + V_{i+1}^H(S_{i+1}) \} \quad (5)$$

What motivated us to use the roll-out policy is that it has been shown to be effectively applied in a wide range of NP-hard combinatorial optimisation problems, including SRCPSPs (Li and Womer, 2015; Xie et al., 2021).

---

Algorithm 1: The Average project-based roll-out policy for SRCPSP.

---

**Input:** current time  $t$ , state  $S$ , eligible tasks  $\mathcal{X}(S)$

**Output:** Selected task;

**if**  $t=0$  **then**

    | Obtain a priority list  $\pi_L$  using  $PR_{T_{det}-MSLK}$ ;

**end**

Use MC to generate  $|\Omega|$  scenarios;

Generate the shortlisted eligible tasks  $\hat{\mathcal{X}}$

**for**  $j \in \hat{\mathcal{X}}$  **do**

**for**  $\omega=1:|\Omega|$  **do**

        |  $C(\Omega(\omega)) = \text{SSGS}(\pi_L, \Omega(\omega), S, j, t)$ ;

**end**

$Cost(j) = \frac{\sum_{\omega=1}^{|\Omega|} C(\Omega(\omega))}{|\Omega|}$ ;

**end**

Use  $EP_{T-SLK}$  to select a task  $k \in \hat{\mathcal{X}}$ ;

**return** Selected task  $k$ ;

---

Algorithm 1 represents our proposed policy for the A-ADP approach. In this policy, the priority list of the tasks ( $\pi_L$ ) is generated before starting the project

(at time  $t = 0$ ). All the tasks are ranked according to their starting time in the average project and the value of their slack. We solve the average project using the Cplex CP Optimizer (CPLEX, 2017) to find the starting time of each task. The slack value of each task  $i$  is the difference between the earliest start time and the latest start time of the task  $i$  in the average project. The tasks scheduled earlier in the average project, with lower slack value, have higher priorities to be selected in the policy. We call this priority rule as  $PR_{T_{det}-MSLK}$ .

A shortlist of eligible tasks is generated to reduce the computational burden associated with decision-making in the roll-out policy. This list comprises only a few eligible tasks, with the highest priorities according to  $\pi_L$ .

To approximately evaluate the expected cost-to-go function starting from  $S_i$ , a set of  $|\Omega|$  scenarios is generated using the Monte Carlo (MC) simulation which has been effectively employed in various algorithms for the SRCPSP (Ballestín, 2007; Xie et al., 2021; Li and Womer, 2015). The approximated value for the cost-to-go is obtained by calculating the average makespan across all  $|\Omega|$  scenarios. In the following, we explain how the makespan for each scenario is obtained by a schedule generation scheme (SGS).

SGS generates a feasible schedule from scratch by incrementally extending a partial schedule. A partial schedule refers to a schedule where only a subset of tasks have been scheduled. The serial schedule generation scheme (SSGS) is an SGS that schedules tasks as soon as possible from a given list one by one. This scheme considers both the precedence relationships and resource constraints of the project (Kolisch and Hartmann, 1999). In this research, we use a different version of SSGS to generate a schedule. In this version, the already scheduled tasks at the current time  $t_i$  are not overtaken, and just schedules the remaining tasks considering  $\pi_L$  are obtained by solving the average project. In this scheme, when a task  $i$  is taken at time  $t_i$ , the task  $j$  selected afterwards cannot be started before  $t_i$ .

An evaluation process that we call  $EP_{T-SLK}$  is conducted to identify the task associated with the minimum cost among the shortlisted eligible tasks. In  $EP_{T-SLK}$ , we rank the tasks in the shortlisted eligible tasks based on their slack value from the average project and the approximated expected cost-to-go function. The task's priority for execution is determined by its average rank.

## 4 COMPUTATIONAL EXPERIMENTS

The computational experiments provided in this paper are to investigate the efficiency of our proposed A-ADP approach. We investigate the project characteristic that has the most significant impact on the algorithm's performance and identify when it is important for the algorithm to work with the stochastic RCPSP. We also investigate the efficiency of the proposed A-ADP when using different priority rules obtained from the average project. The algorithm is tested on J30 (instances with 30 tasks) and J60 (instances with 60 tasks) from PSPLIB. The PSPLIB dataset, available at <http://www.om-db.wi.tum.de/psplib/>, is created using the ProGen project instance generator which are characterised by network complexity  $NC$ , resource factor  $RF$ , and resource strength  $RS$  (Kolisch and Sprecher, 1997).

We followed the assumptions in (Ballestín, 2007) for the stochastic task duration. Table 1 shows the employed PDs in this research. The deterministic duration for task  $j$  is denoted by  $d_j$  and the PDs include Uniform distributions, Beta distributions and exponential distribution (EXP).

Uniform distributions have a constant probability, with slight variance in U1 and intermediate variances in U2. The exponential distribution (EXP) also maintains a constant failure rate but exhibits a larger standard deviation compared to the uniform distribution.

Beta distributions, characterised by slight variance in B1 and intermediate variances in B2, are well-known probability distributions in scheduling under uncertainty. These distributions are represented using two shape parameters ( $\alpha$  and  $\beta$ ) to generate random values.

To evaluate the policy in each experiment, a set  $S$  of 1000 scenarios are generated for an instance  $r$ ,  $r \in \{1, \dots, R\}$  using MC and a known PD provided in Table 1. We solve the problem by Algorithm 1 for each scenario  $i \in S$  with the makespan  $Makespan(i)$ . The expected makespan for the instance  $r$  is calculated as  $E_r = \frac{\sum_{i=1}^{|S|} Makespan(i)}{|S|}$ , where  $|S| = 1000$ .

The parameter  $Gap$  is defined to compare the results in different experiments and is the percentage of the average deviation of the expected makespan from the deterministic critical path length(CPL) of instances, i.e.,  $Gap = \frac{1}{R} \sum_{r=1}^R \frac{E_r - CPL_r}{CPL_r}$ .

Our algorithm involves two main computational components. We allocate a time limit of 2 seconds for solving the average projects with 30 tasks and 10 seconds for average projects with 60 tasks which is enough to obtain a feasible solution using Cplex cp

optimizer(Etminaniesfahani et al., 2022; Etminaniesfahani. et al., 2022). The maximum number of schedules considered for the remaining stages of the algorithm is determined as follows. The maximum number of schedules considered for a problem with  $n$  tasks when the number of tasks in the shortlisted eligible tasks is  $L_{sle}$ , and employing  $|\Omega|$  scenarios for evaluating the tasks in the policy, is  $n \times |\Omega| \times L_{sle}$ .

### 4.1 The Importance of Working with the Stochastic RCPSP

This section investigates when it is important to work with the SRCPSP. To this aim, we solved all 480 instances of J30.

Table 1: Probability distributions of task duration.

| PD  | Limits             |                    | Mean  | Variance          |
|-----|--------------------|--------------------|-------|-------------------|
|     | lb                 | ub                 | $\mu$ | $\sigma$          |
| U1  | $d_j - \sqrt{d_j}$ | $d_j + \sqrt{d_j}$ | -     | $\frac{d_j}{3}$   |
| U2  | 0                  | $2d_j$             | -     | $\frac{d_j^2}{3}$ |
| Exp | -                  | -                  | $d_j$ | $d_j^2$           |
| B1  | $\frac{d_j}{2}$    | $2d_j$             | -     | $\frac{d_j}{3}$   |
| B2  | $\frac{d_j}{2}$    | $2d_j$             | -     | $\frac{d_j^2}{3}$ |

To investigate the impact of relying on the results of deterministic RCPSP on the performance of the A-ADP approach in Algorithm 1, to obtain the  $\pi_L$ , we ignored the impact of the slack value of tasks, by considering only the starting time of each task as its priority. The tasks started earlier have the higher priority in the policy, and we call that rule  $PR_{T_{det}}$ . We also replaced  $EP_{T-SLK}$  by  $EP_{T=\arg \min_j (Cost(j))}$  for evaluating each task  $j$  in the policy based on only the expected makespan when selecting the task  $j$ .

In our experiments, We set  $L_{sle}$ , and  $(|\Omega|)$  equal to three. We call the  $Gap$  achieved when  $L_{sle}=3$  and  $PR_{T_{det}}$ , as  $Gap_{3T_{det}}$ . In another experiment, we set the length of the shortlisted eligible tasks to one and then obtain the  $Gap$ , called  $Gap_{1T_{det}}$ . When the shortlisted eligible tasks consist of only one task, the policy selects the eligible task with the highest priority without approximating the cost-to-go function. Consequently, the decision-making process relies solely on the deterministic solution derived from the average project.

The availability of resources in a problem is determined by  $RS \in \{0.2, 0.5, 0.7, 1\}$ . Figure 1 for each PD, associated with each value of  $RS$  is the difference between the  $Gap_{3T_{det}}$  and the  $Gap_{1T_{det}}$ .

The results obtained from this experiment show that in the problems with  $RS = 0.2$ , the difference between the two policies is the maximum for all PDs. By increasing  $RS$ , which means increasing the availability of the resources, the difference between the results obtained by the policies decreases. For U1



Table 2: Comparison of results obtained by A-ADP and state-of-the-art algorithms for J30 (Zaman et al., 2021).

| Algorithm                   | Gap         |             |              |             |             |
|-----------------------------|-------------|-------------|--------------|-------------|-------------|
|                             | U1          | U2          | EXP          | B1          | B2          |
| PPGA(Ashtiani et al., 2011) | 19.87       | 30.67       | 45.56        | 19.93       | 30.76       |
| A-HBA(Li and Womer, 2015)   | 16.63       | 42.37       | 45.13        | 12.60       | 16.63       |
| LFT (Chen et al., 2018)     | 21.60       | 30.89       | 46.47        | 21.59       | 30.87       |
| SLFT (Chen et al., 2018)    | 21.60       | 30.83       | 46.32        | 21.60       | 30.76       |
| DH (Chen et al., 2018)      | 21.36       | 31.18       | 46.86        | 21.36       | 31.21       |
| S-COA (Zaman et al., 2021)  | <b>1.56</b> | <b>8.67</b> | <b>16.66</b> | <b>1.29</b> | <b>7.72</b> |
| A-ADP                       | 7.72        | 12.94       | 29.26        | 4.15        | 11.37       |

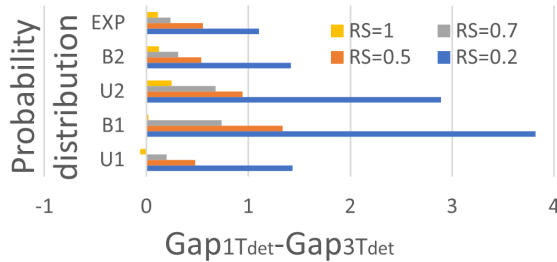


Figure 1: The impact of  $RS$  on the performance of the policy when considering uncertainty.

where the variance of the probability distribution is relatively low, when  $RS = 1$ , the policy based on the average project performs better than A-ADP when there are three tasks in the shortlisted eligible tasks. We did not find a meaningful relation between the performance of the A-ADP with characteristics  $NC$  and  $RF$ .

Table 2 compares the results of the state-of-the-art algorithms in solving SRCPSPs, with  $Gap_{3T_{det}}$  obtained by our A-ADP (Ashtiani et al., 2011; Chen et al., 2018; Li and Womer, 2015; Zaman et al., 2021). The stopping criterion for schedule generation is commonly defined as 5000 in the literature.

We set the parameter  $|\Omega|$  to 3, indicating the number of scenarios in the policy. Given that the length of the shortlisted eligible tasks is limited to 3, the policy’s maximum number of generated schedules at each stage is 9. Consequently, for a problem comprising 30 tasks, the maximum number of schedules considered is 270.

Table 2 illustrates that more variance in task duration results in a higher  $Gap$  for the same PD. Comparing other algorithms, except for the S-COA, our policy outperforms the other algorithms, including (Li and Womer, 2015), the other ADP algorithm (Li and Womer, 2015).

## 4.2 The Impact of Priority Rules and Evaluation Process on the Algorithm

This section investigates the impact of the proposed priority rule and the evaluation approach in this algorithm.

For this purpose, we employed subsets of datasets J30 and J60 from PSPLIB denoted as  $J_{48}30$  and  $J_{48}60$ . J30 and J60 comprise 48 different problem characteristic combinations, with each problem having ten distinct instances. To comprehensively assess the algorithm across all these combinations, we chose the first instance from each problem, i.e.,  $J_{48}30 = \{j301 - 1, j302 - 1, \dots, j3048 - 1\}$  and  $J_{48}60 = \{j601 - 1, j602 - 1, \dots, j6048 - 1\}$ .

We conducted a comparative analysis to assess the influence of the priority rule introduced in section 3. We compared the results obtained with the  $PR_{T_{det}-MSLK}$  to those with  $PR_{T_{det}}$ . The results of solving  $J_{48}30$  and  $J_{48}60$  are provided in Table 3, and Table 4. The  $Gap_{1T_{det}-MSLK}$ , is the  $Gap$  in which the shortlisted eligible tasks in the policy have just one task and the priority rule is  $PR_{T_{det}-MSLK}$ .

Table 3: Results of  $J_{48}30$  when  $L_{sle}=1$ .

| PD  | $Gap_{1T_{det}}$ | $Gap_{1T_{det}-MSLK}$ |
|-----|------------------|-----------------------|
| B1  | 3.91             | <b>2.33</b>           |
| B2  | 10.82            | <b>9.62</b>           |
| EXP | 28.70            | <b>27.79</b>          |
| U1  | 7.40             | <b>5.53</b>           |
| U2  | 12.63            | <b>10.98</b>          |

Table 4: Results of  $J_{48}60$  when  $L_{sle}=1$ .

| PD  | $Gap_{1T_{det}}$ | $Gap_{1T_{det}-MSLK}$ |
|-----|------------------|-----------------------|
| B1  | 15.78            | <b>13.75</b>          |
| B2  | 22.94            | <b>21.02</b>          |
| EXP | 42.65            | <b>40.84</b>          |
| U1  | 18.87            | <b>16.66</b>          |
| U2  | 24.24            | <b>22.15</b>          |

When comparing  $Gap_{1T_{det}}$  and  $Gap_{1T_{det}-MSLK}$  for all PDs in order to solve  $J_{48}30$  and  $J_{48}60$ , the tables

Table 5: Results of  $J_{4830}$  for different  $|\Omega|$ , priority rule and evaluation approaches when  $L_{sle}=3$ .

| PD  | $ \Omega =1$     |        |                       |        | $ \Omega =3$     |        |                       |        |
|-----|------------------|--------|-----------------------|--------|------------------|--------|-----------------------|--------|
|     | $Gap_{3T_{det}}$ |        | $Gap_{3T_{det}-MSLK}$ |        | $Gap_{3T_{det}}$ |        | $Gap_{3T_{det}-MSLK}$ |        |
|     | $EP_{T-SLK}$     | $EP_T$ | $EP_{T-SLK}$          | $EP_T$ | $EP_{T-SLK}$     | $EP_T$ | $EP_{T-SLK}$          | $EP_T$ |
| B1  | 2.22             | 3.48   | <b>1.87</b>           | 2.96   | 2.32             | 3.31   | 2.00                  | 3.01   |
| B2  | 9.68             | 11.01  | 9.49                  | 10.32  | 9.76             | 10.89  | <b>9.44</b>           | 10.51  |
| EXP | 27.93            | 28.89  | <b>27.42</b>          | 28.87  | 27.82            | 28.69  | <b>27.98</b>          | 28.05  |
| U1  | 5.60             | 7.15   | <b>5.45</b>           | 6.58   | 5.80             | 7.02   | 5.52                  | 6.65   |
| U2  | 10.98            | 12.62  | 10.94                 | 12.15  | 10.96            | 12.36  | <b>10.92</b>          | 11.76  |

Table 6: Results of  $J_{4860}$  for different  $|\Omega|$ , priority rule and evaluation approaches when  $L_{sle}=3$ .

| PD  | $ \Omega =1$     |        |                       |        | $ \Omega =3$     |        |                       |        |
|-----|------------------|--------|-----------------------|--------|------------------|--------|-----------------------|--------|
|     | $Gap_{3T_{det}}$ |        | $Gap_{3T_{det}-MSLK}$ |        | $Gap_{3T_{det}}$ |        | $Gap_{3T_{det}-MSLK}$ |        |
|     | $EP_{T-SLK}$     | $EP_T$ | $EP_{T-SLK}$          | $EP_T$ | $EP_{T-SLK}$     | $EP_T$ | $EP_{T-SLK}$          | $EP_T$ |
| B1  | 14.40            | 15.77  | 13.55                 | 14.65  | 14.49            | 15.62  | <b>13.50</b>          | 14.28  |
| B2  | 21.28            | 23.40  | <b>20.91</b>          | 21.68  | 21.61            | 23.41  | 20.98                 | 21.74  |
| EXP | 41.18            | 43.72  | <b>40.78</b>          | 42.41  | 41.36            | 43.16  | 40.84                 | 42.33  |
| U1  | 16.94            | 19.16  | 16.67                 | 18.07  | 19.14            | 17.46  | <b>16.47</b>          | 17.92  |
| U2  | 23.18            | 24.71  | <b>21.82</b>          | 23.55  | 23.08            | 24.13  | 22.55                 | 22.67  |

Table 3 and Table 4 clearly demonstrate that using the priority rule  $PR_{T_{det}-MSLK}$ , which takes into account the minimum slack value of tasks, yields superior results when compared to cases where slack value is not considered in the priority rule.

Additionally, we examined two approaches for evaluating each task in the eligible shortlisted tasks. One was the evaluation method proposed in section 3 denoted as  $EP_{T-SLK}$ , while the other involved assessing tasks by disregarding their slack values,  $EP_T$ . Furthermore, we explored how the number of scenarios, represented as  $|\Omega|$ , impacts the algorithm’s efficiency. Our investigation included a comparison of results when  $|\Omega|$  was set to 3, with the results obtained when  $|\Omega|$  was equal to 1, and the average project was the single scenario in the policy. This analysis shows how altering the number of scenarios impacts the algorithm’s performance.

Table 5 represents the results of  $J_{4830}$ . Comparing the results for different PDs, there is no meaningful difference when  $|\Omega|$  is one or three, and in terms of computational efficiency, employing the task duration of an average project to calculate the cost-to-go function when  $|\Omega|=1$  reduces the computations by up to a third compared to the other approach of finding the expected makespan by averaging scenarios. Table 6 displays the results for the  $J_{4860}$  experiment, which leads us to the same conclusion. Consequently, we focus on using the average project with  $|\Omega|$  set to one.

When we compare the different evaluation methods in both tables, regardless of the PDs and whether we’re looking at  $Gap_{3T_{det}}$  or  $Gap_{3T_{det}-MSLK}$ ,  $EP_{T-SLK}$  consistently outperforms  $EP_T$ . Moreover, if we specifically consider the  $EP_{T-SLK}$  approach, it is clear that the values of  $Gap_{3T_{det}-MSLK}$  consistently surpass

those of  $Gap_{3T_{det}}$ .

### 4.3 Comparing the Proposed A-ADP Algorithm with the State-of-the-Art Algorithms

This section compares our obtained  $Gap_{3T_{det}-MSLK}$ , when  $|\Omega|=1$ , and using  $EP_{T-SLK}$  with the state-of-the-art algorithms. We solved all J30 and J60 sets of instances. The results of our proposed algorithm are compared with other algorithms for different PDs, i.e., U1, U2, EXP, B1, and B2.

Table 7: Comparison of  $Gap_{3T_{det}-MSLK}$  and  $Gap$  in the state-of-the-art algorithms for J30 (Zaman et al., 2021).

| Algorithm | Gap         |             |              |             |             |
|-----------|-------------|-------------|--------------|-------------|-------------|
|           | U1          | U2          | EXP          | B1          | B2          |
| PPGA      | 19.87       | 30.67       | 45.56        | 19.93       | 30.76       |
| A-HBA     | 16.63       | 42.37       | 45.13        | 12.60       | 16.63       |
| LFT       | 21.60       | 30.89       | 46.47        | 21.59       | 30.87       |
| SLFT      | 21.60       | 30.83       | 46.32        | 21.60       | 30.76       |
| DH        | 21.36       | 31.18       | 46.86        | 21.36       | 31.21       |
| S-COA     | <b>1.56</b> | <b>8.67</b> | <b>16.66</b> | <b>1.29</b> | <b>7.72</b> |
| A-ADP     | 5.64        | 11.17       | 27.97        | 1.82        | 9.59        |

Table 8: Comparison of  $Gap_{3T_{det}-MSLK}$  and  $Gap$  in the state-of-the-art algorithms for J60 (Zaman et al., 2021).

| Algorithm | Gap          |              |              |              |              |
|-----------|--------------|--------------|--------------|--------------|--------------|
|           | U1           | U2           | EXP          | B1           | B2           |
| PPGA      | 18.91        | 29.08        | 45.74        | 18.98        | 29.17        |
| A-HBA     | 14.14        | 28.57        | 45.36        | 18.31        | 28.77        |
| LFT       | 19.94        | 28.49        | 44.97        | 19.95        | 28.63        |
| SLFT      | 19.89        | 28.42        | 44.94        | 19.90        | 28.55        |
| S-COA     | <b>12.60</b> | <b>19.31</b> | <b>28.70</b> | <b>12.62</b> | <b>18.54</b> |
| A-ADP     | 18.03        | 23.71        | 43.07        | 14.57        | 22.26        |

Table 9: Comparison of the computational time of A-ADP and S-COA.

| Prob. | Alg.  | CPU Time (seconds) |              |              |              |              |
|-------|-------|--------------------|--------------|--------------|--------------|--------------|
|       |       | U1                 | U2           | EXP          | B1           | B2           |
| J30   | S-COA | 58.72              | 69.09        | 70.32        | 64.42        | 67.24        |
|       | A-ADP | <b>4.68</b>        | <b>4.61</b>  | <b>4.37</b>  | <b>4.87</b>  | <b>4.63</b>  |
| J60   | S-COA | 181.21             | 185.45       | 193.87       | 183.7        | 190.88       |
|       | A-ADP | <b>36.98</b>       | <b>36.94</b> | <b>35.37</b> | <b>38.27</b> | <b>39.37</b> |

As  $L_{ste}=3$  and  $|\Omega|=1$ , the maximum number of generated schedules in solving J30 is 90, and in solving J60 is 180.

Table 7 and table 8, compare the results of our proposed A-ADP algorithm with the state-of-the-art algorithm. The stopping criterion of the other algorithms is 5000 generated schedules. As is shown in Table 7 in solving J30 instances except for A-COA, our algorithm outperforms other state-of-the-art algorithms. Table 8 presents the results for J60 instances. When solving J60 with the U1 probability, A-HBA and S-COA perform better than our A-ADP. However, S-COA is the best-performing algorithm for the remaining probability distributions, and A-ADP is the second-best compared to the state-of-the-art algorithms. To show the efficiency of A-ADP in comparison to S-COA, the CPU times of these algorithms are presented in Table 9. As is highlighted, A-ADP demonstrates significantly shorter computational times than S-COA when solving both J30 and J60 problems.

## 5 CONCLUSION AND FUTURE WORK

The results show that the A-ADP approach provides competitive results in solving SRCPSP with 30 and 60 tasks. Investigating different problem characteristics also indicates that the larger shortlisted eligible tasks are required for the problems with the lower availability of resources. In making and evaluating the shortlisted eligible tasks, our experiments show the importance of prioritising tasks with lower slack values and tasks started earlier in the average project.

Future work includes investigating higher dimensional instances and their characteristics to find a machine learning-based approach to automatically select policy for solving SRCPSPs. Extending this approach to another type of uncertainty, i.e., uncertain resources and multi-mode instances, is also interesting.

## ACKNOWLEDGMENT

Alireza Etminaniesfahani is the recipient of the UTS International Research Scholarship (IRS) and UTS President’s Scholarship (UTSP).

## REFERENCES

Ashtiani, B., Leus, R., and Aryanezhad, M.-B. (2011). New competitive results for the stochastic resource-constrained project scheduling problem: Exploring the benefits of pre-processing. *J. Scheduling*, 14:157–171.

Ballestín, F. (2007). When it is worthwhile to work with the stochastic RCPSP? *J. Scheduling*, 10:153–166.

Bellman, R. E. (2010). *Dynamic programming*. Princeton university press.

Bertsekas, D., Tsitsiklis, J., and Wu, C. (1997). Rollout algorithms for combinatorial optimization. *Journal of Heuristics*, 3:245–262.

Bertsekas, D. P. (2007). *Dynamic Programming and Optimal Control, Vol. II*. Athena Scientific, 3rd edition.

Bruni, M., Guerriero, F., and Pinto, E. (2009). Evaluating project completion time in project networks with discrete random activity durations. *Computers & Operations Research*, 36:2716–2722.

Cai, H., Bian, Y., and Liu, L. (2024). Deep reinforcement learning for solving resource constrained project scheduling problems with resource disruptions. *Robotics and Computer-Integrated Manufacturing*, 85:102628.

Chen, Z., Demeulemeester, E., Bai, S., and Guo, Y. (2018). Efficient priority rules for the stochastic resource-constrained project scheduling problem. *European Journal of Operational Research*, 270(3):957–967.

CPLEX, I. I. (2017). version 12.8.0.

Etminaniesfahani, A., Gu, H., Naeni, L., and Salehipour, A. (2022). A forward–backward relax-and-solve algorithm for the resource-constrained project scheduling problem. *SN Computer Science*, 4:104–114.

Etminaniesfahani, A., Gu, H., and Salehipour, A. (2022). An efficient relax-and-solve algorithm for the resource-constrained project scheduling problem. In *Proceedings of the 11th International Conference on Operations Research and Enterprise Systems - ICORES*, pages 271–277. INSTICC, SciTePress.

- Guo, W., Vanhoucke, M., Coelho, J., and Luo, J. (2021). Automatic detection of the best performing priority rule for the resource-constrained project scheduling problem. *Expert Systems with Applications*, 167:114116.
- Kolisch, R. and Hartmann, S. (1999). *Heuristic Algorithms for the Resource-Constrained Project Scheduling Problem: Classification and Computational Analysis*, pages 147–178. Springer US, Boston, MA.
- Kolisch, R. and Sprecher, A. (1997). PSPLIB - A project scheduling problem library: OR Software - ORSEP Operations Research Software Exchange Program. *European Journal of Operational Research*, 96(1):205–216.
- Li, H. and Womer, N. K. (2015). Solving stochastic resource-constrained project scheduling problems by closed-loop approximate dynamic programming. *European Journal of Operational Research*, 246(1):20–33.
- Pritsker, A., Waiters, L. J., and Wolfe, P. (1969). Multi-project scheduling with limited resources: A zero-one programming approach. *Management Science*, 16:93–108.
- Schwindt, C. and Zimmermann, J. (2015). *Handbook on Project Management and Scheduling Vol. 2*. Springer Cham.
- Stork, F. (2000). Branch-and-bound algorithms for stochastic resource-constrained project scheduling. *Technical rep.*, pages 702–2000.
- Xie, F., Li, H., and Xu, Z. (2021). An approximate dynamic programming approach to project scheduling with uncertain resource availabilities. *Applied Mathematical Modelling*, 97:226–243.
- Zaman, F., Elsayed, S., Sarker, R., Essam, D., and Coello Coello, C. A. (2021). An evolutionary approach for resource constrained project scheduling with uncertain changes. *Computers & Operations Research*, 125:105104.