European Journal of Operational Research 315 (2024) 454–469

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/eor

Discrete optimisation

A simulation-based approximate dynamic programming approach to dynamic and stochastic resource-constrained multi-project scheduling problem

U. Satic [a,b,*], P. Jacko [a,c], C. Kirkbride [a]

[a] Lancaster University Management School, Bailrigg, Lancaster, LA1 4YX, United Kingdom
[b] Abdullah Gul University Faculty of Engineering, Kocasinan, Kayseri, 38080, Turkey
[c] Berry Consultants, Abingdon, Oxfordshire, OX14 5EG, United Kingdom

ARTICLE INFO

Keywords:
Project scheduling
Markov decision processes
Approximate dynamic programming
Dynamic resource allocation
Dynamic programming

ABSTRACT

We consider the dynamic and stochastic resource-constrained multi-project scheduling problem which allows for the random arrival of projects and stochastic task durations. Completing projects generates rewards, which are reduced by a tardiness cost in the case of late completion. Multiple types of resource are available, and projects consume different amounts of these resources when under processing. The problem is modelled as an infinite-horizon discrete-time Markov decision process and seeks to maximise the expected discounted long-run profit. We use an approximate dynamic programming algorithm (ADP) with a linear approximation model which can be used for online decision making. Our approximation model uses project elements that are easily accessible by a decision-maker, with the model coefficients obtained offline via a combination of Monte Carlo simulation and least squares estimation. Our numerical study shows that ADP often statistically significantly outperforms the optimal reactive baseline algorithm (ORBA). In experiments on smaller problems however, both typically perform suboptimally compared to the optimal scheduler obtained by stochastic dynamic programming. ADP has an advantage over ORBA and dynamic programming in that ADP can be applied to larger problems. We also show that ADP generally produces statistically significantly higher profits than common algorithms used in practice, such as a rule-based algorithm and a reactive genetic algorithm.

1. Introduction

Project management and project scheduling are challenging. Engineering services, software development, IT services, construction and R&D operate in dynamic environments, often processing multiple projects simultaneously. Many unplanned factors may disturb the project execution plan with new project arrivals and delays in task processing. A recent project management survey (Wellingtone PPM, 2018) showed that only 40% of projects are completed within their planned time, 46% of projects are completed within their predicted budget, and 36% of projects realise their full benefits. In this paper we consider the dynamic arrival of new projects and stochastic durations of tasks, and we propose a comprehensive model and solution approach for the dynamic and stochastic resource-constrained multi-project scheduling problem (*dynamic and stochastic RCMPSP*).

The dynamic and stochastic RCMPSP is a generalisation of the precedence-constrained scheduling problem, which was shown to be an NP-hard problem in Garey and Johnson (1979, p. 239). Thus the dynamic and stochastic RCMPSP is also an NP-hard problem. Dynamic refers to random project arrivals from different types of projects and stochastic refers to uncertain task processing times. Dynamic generalisations of RCMPSP are the dynamic RCMPSP and the dynamic and stochastic RCMPSP. A discussion of the RCMPSP and its variants can be found in Satic et al. (2022).

The non-dynamic (i.e., static) variants of RCMPSP are extensively studied (Creemers, 2015). However, the dynamic variants of the RCMPSP where new projects randomly arrive in the system are scarce in the literature. To the best of our knowledge, there are only three research papers available for the dynamic RCMPSP which are Pamay et al. (2014), Parizi et al. (2017), Satic et al. (2020), and there are only ten research papers available for the dynamic and stochastic RCMPSP which are Adler et al. (1995), Capa and Ulusoy (2015), Chen et al. (2019), Choi et al. (2007), Cohen et al. (2005), Fliedner et al. (2012), Melchiors (2015), Melchiors and Kolisch (2009), Melchiors et al. (2018), Satic et al. (2022). They adopt different solution approaches, which have their own strengths and weaknesses.

* Corresponding author at: Lancaster University Management School, Bailrigg, Lancaster, LA1 4YX, United Kingdom.
E-mail address: ugur.satic@agu.edu.tr (U. Satic).

https://doi.org/10.1016/j.ejor.2023.10.046
Received 9 August 2021; Accepted 31 October 2023
Available online 10 November 2023

Adler et al. (1995), Cohen et al. (2005), Melchiors and Kolisch (2009) took advantage of the well-developed queueing network approach where interdependent resources process project tasks. This requires consideration of projects of relatively simple structure such as tasks requiring the allocation of a single unit of a single type of resource. Capa and Ulusoy (2015), Fliedner et al. (2012), Pamay et al. (2014) considered a reactive scheduling method which generates a baseline schedule for current projects and then updates it at each time a new project arrival disrupts the schedule. This approach can be remarkably suboptimal as evidenced in our computational study in Section 5. Melchiors et al. (2018), Satic et al. (2022) modelled the problem as a Markov decision process (MDP), using dynamic programming (DP) to evaluate optimal policies. This solution approach suffers from the curse of dimensionality and thus can only be used for unrealistically small problems. Chen et al. (2019) divided the multi-project problem into states according to the project's completion conditions and then searched best priority rules for each state, but priority rules are notably prone to be suboptimal.

Our methodological approach is similar to Choi et al. (2007), Melchiors (2015), Parizi et al. (2017) in that we also formulate the problem as an MDP and design a scheduling policy via approximate dynamic programming (ADP). However, our model is notably more comprehensive and allows for solving problems that are larger and/or have a more complex structure, which are closer to those appearing in practice. Choi et al. (2007) considered applications in the agricultural and pharmaceutical industries; thus, they focused on serial project networks, stochastic task outcomes (success or failure), a single resource type, single resource usage per task and no project due dates. Melchiors (2015, chapter 7) conducted experiments on small problems with two projects with three tasks with a single unit of resource capacity for each resource type, tasks require a single unit of resource only, identical project networks for both projects, rejection, holding and processing costs, but no project due dates. Parizi et al. (2017) considered deterministic task processing times with rejection, holding and processing costs. Their numerical study had short simulation durations with heavy discounting.

ADP is a powerful tool that provides researchers with the ability to adjust the complexity of the optimisation model to trade-off the solution complexity of large (realistic) problems at the expense of a modest suboptimality. An acceptable trade-off can be achieved by careful mathematical modelling of the problem in hand; this is in contrast to general purpose methods such as genetic algorithms and other heuristics which typically rely only on tuning of algorithm parameters. Our literature summary shows that ADP has been used in dynamic variants of the RCMPSP such as Choi et al. (2007), Melchiors (2015), Parizi et al. (2017). It has also been applied in static variants of the RCMPSP (Li & Womer, 2015; Li et al., 2023). Outside of applications in project scheduling, ADP methods have been applied in areas such as clinical trials (Ahuja & Birge, 2020), vehicle scheduling (He et al., 2018), capacity allocation (Schütz & Kolisch, 2012), machine scheduling (Ronconi & Powell, 2010) and missile defence systems (Davis et al., 2017).

We consider new projects arriving at random during the execution of ongoing projects, project completions generate rewards which are decreased by tardiness costs if completed after their respective due dates, processing times of the project tasks are uncertain, multiple types of resources are available and multiple amounts of resources can be used by each project type. Thus, our model can help optimise the use of company resources, reduce project delays, and improve overall productivity. We model the problem as an infinite-horizon discrete-time MDP and seek to maximise the expected total discounted long-run profit.

In this paper we show that ADP is a very useful and advantageous method for the dynamic and stochastic RCMPSP. We use an ADP algorithm with a linear approximation model to approximate the value function of the Bellman equation. Our approximation model uses resource consumption and decision rewards as features and can be used for online decision making after estimating the coefficients of the linear value function approximation in a simulation-based training phase.

We compare the performance of our ADP algorithm with four solution approaches from Satic et al. (2022), namely a DP algorithm that computes the optimal policy; an optimal reactive baseline algorithm (ORBA) and a genetic algorithm (GA) that both generate schedules to maximise the total profit of ongoing projects; and a rule-based algorithm (RBA) that uses the longest task first rule to guide the allocation of remaining resources to tasks.

We run our benchmark tests on the problems of Satic et al. (2022). In addition, we generate new comparison problems that are larger and include non-sequential project networks and multiple resource types. The larger size problems are computationally intractable for DP and ORBA; thus, we benchmark ADP with GA and RBA on these problems.

We contribute to the literature by (i) a new comprehensive MDP model which considers the random arrival of new projects, stochastic task durations, multiple resource types, non-sequential project networks, project completion rewards, project due dates and tardiness costs, (ii) a new approximation function that uses project completion rewards, tardiness costs and spent resource amounts for decision making, and is capable of solving much larger, more complex and much more general problems than ADPs from existing literature, (iii) an extensive simulation study illustrating the strengths and weaknesses of different approaches, (iv) benchmarking with DP and ORBA whenever tractable and with two other approaches in larger problems, (v) developing an efficient implementation of the proposed ADP method in the Julia programming language to solve dynamic and stochastic RCMPSPs.

This paper is organised as follows: In Section 2, we describe the problem setting, the MDP model and our goal function. In Section 3, we describe our ADP algorithm and its coefficient training procedure. In Section 4, we describe the comparator algorithms and discuss the comparison results in Section 5. In Section 6 we conclude.

2. The dynamic and stochastic RCMPSP model

2.1. Problem setting

A project is a group of tasks which are bound to each other with predecessor–successor relationships, also called a *project network*. We consider "finish to start" precedence relations between tasks. There are K types of renewable *resources* and the available integer number of units is represented by $B_k > 0$ for resource type $k = 1, \dots, K$. These resources need to be allocated to process *tasks* of random duration from randomly arriving *projects*.

We assume that each arriving project can be categorised as one of the possible project types ($\mathcal{J} = \{1, 2, \dots, J\}$). All projects of type j share features such as: inter-arrival time distribution Λ_j, completion reward r_j, due date F_j, and tardiness cost w_j; set of tasks $\mathcal{I}_j = \{1, 2, \dots, I_j\}$, project network with sets $\mathcal{M}_{j,i}$ of predecessors of each task i, task resource usages $b^k_{j,i}$, and task duration distribution $\Gamma_{j,i}$.

The system always accepts newly arrived projects until the system capacity for type j projects is reached and rejects the remaining newly arrived projects. We only consider *non-preemptive* task processing; thus, task processing cannot be paused from after it began until the end of its random duration. The duration of a task is not revealed to the decision maker until it is actually reached.

When all tasks of a project are processed, the project is completed, and the project completion reward is earned. However, if the project due date passes before the project is completed, the tardiness cost is incurred.

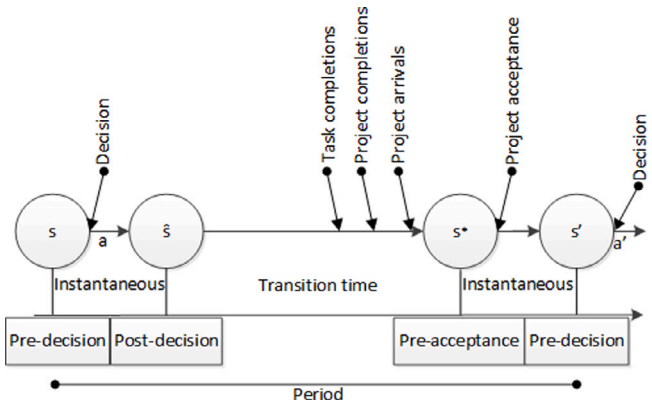

Fig. 1. Discrete-time Markov Decision Process.

2.2. Modelling framework

We consider the dynamic and stochastic RCMPSP as an infinite horizon *Discrete-Time Markov Decision Process* (DT-MDP) model. A DT-MDP is 5-tuple consisting of state space $\mathcal{S}$, set of actions $\mathcal{A}(s)$, transition function $P(s'|s,a)$, the immediate profit $R_{s,a,s'}$ and discount factor α. The DT-MDP is a discrete decision model where a decision-maker observes the system state in regular time instants, takes an action in the observed state to maximise the discounted profit, and the state randomly changes. This problem can be modelled as a DT-MDP as the transition probabilities and reward functions can be defined to depend only on the current state of the system and the action taken in that state, so the Markov property holds.

The problem is modelled using discrete time periods $t = 1, 2, \ldots$. Each period represents a unit of time (e.g., a week) which would be appropriate for a particular real-life project scheduling problem. A *decision epoch* is the beginning of each period t at which the allocation of available resources to initiate project tasks takes place. The *transition time* between the beginning and end of a period allows for completions of tasks, completions of projects, and arrivals of new projects. At the end of each period, arriving projects are accepted to the system if there is capacity to do so; otherwise, they are rejected. This process repeats for all new states over the infinite horizon. Fig. 1 illustrates this process.

We use terms pre-decision state s for a state before the action is taken, post-decision state $\hat{s}$ for a state after the action is taken, and pre-acceptance state s^* for a state immediately before the decision about accepting or rejecting projects that have arrived during the current period. We assume that the system transitions from a pre-decision to a post-decision state, and from a pre-acceptance to a pre-decision state, are instantaneous. These terms are described in detail below.

Processing task i of project j requires allocation of $b^k_{j,i}$ units of resource k in each period for the duration of the task. The unallocated resources are called *free-resources* $B^{\text{free}}_k(s)$, and allocated resources are added to free resources at the end of the period during which their assigned task is completed. Allocated resources are removed from free resources instantaneously when a task starts processing at the beginning of a period.

2.3. Modelling assumptions

A requirement of our discrete model is that project due dates and task durations are given in whole numbers of periods, and available resources and task resource usages are given in non-negative whole units. We also limit the maximum number of projects from each project type in the system to one to simplify the notation of the presented model.

To transform the inter-arrival time distribution Λ_j to discrete time, we consider *geometrically distributed inter-arrival times*. Together with the assumption that at most one project of type j can be present in the system in any given period, we can use λ_j as the probability of arrival of a single project of type j during the transition time. At the end of every period, the system only accepts a new arrival of a type j project if no type j project exists in the system, either in processing or waiting. Otherwise, the system rejects the new arrival and continues its processing as if there was no arrival.

To transform the task duration distribution $\Gamma_{j,i}$ to discrete time, we consider *stochastic task durations* such that a task may be completed in some period between minimal possible task duration $t^{\min}_{j,i}$ and maximal possible task duration $t^{\max}_{j,i}$. The probability that a task completes in the current period is $\gamma_{j,i}(\cdot)$.

See Appendix B for a more detailed discussion of these and other modelling assumptions.

2.4. Model dynamics

2.4.1. Pre-decision state

For the dynamic and stochastic RCMPSP, a pre-decision state s represents the system information available at a decision epoch. The set of all pre-decision states is called the state space $\mathcal{S}$. A pre-decision state s consists of project states $\boldsymbol{P}_j$ for all project types $j \in \mathcal{J}$:

$$s = \{\boldsymbol{P}_1, \boldsymbol{P}_2 \ldots, \boldsymbol{P}_J\}. \tag{1}$$

A project state consists of task states $x_{j,i}$ of tasks $i \in \mathcal{I}_j$ and the remaining due date state d_j:

$$\boldsymbol{P}_j = (x_{j,1}, x_{j,2}, \ldots, x_{j,I_j}, d_j), \tag{2}$$

so that the pre-decision state can be seen as[1]:

$$s = \begin{bmatrix} x_{1,1}, & x_{1,2}, & \ldots & x_{1,I_1}, & d_1 \\ x_{2,1}, & x_{2,2}, & \ldots & x_{2,I_2}, & d_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{J,1}, & x_{J,2}, & \ldots & x_{J,I_J}, & d_J \end{bmatrix}. \tag{3}$$

A task state $x_{j,i}$ represents the status of the ith task of project type j:

$$x_{j,i} \in \{-1, 0, 1, 2 \ldots, t^{\max}_{j,i} - 1\}. \tag{4}$$

If task i is "pending for processing", its state is -1. If task i is "in processing" (i.e., has been in processing for at least 1 period), its state shows the remaining task processing time to its maximal possible duration $t^{\max}_{j,i}$. If task i is "completed", its state becomes 0.

A due date state d_j is the remaining due date of a type j project and its value shows the number of remaining periods from the current decision epoch to the project due date:

$$d_j \in \{0, 1, 2, \ldots, F_j\}. \tag{5}$$

In a decision epoch, if a due date state value is zero ($d_j = 0$) while the project still has some uncompleted tasks (i.e., $x_{j,i} = -1$ or $x_{j,i} > 0$), a tardiness cost w_j is deducted from the project reward r_j at the project's completion.

The absence of a project type j is shown by a project state $\boldsymbol{P}_j$ where all task states are 0 ($\forall i : x_{j,i} = 0$). For these cases the due date state of type j project is taken as 0 ($d_j = 0$):

$$\boldsymbol{P}_j = (0, 0, \ldots, 0, 0). \tag{6}$$

An accepted arrival of a project type j at the end of the previous period is represented by a project state $\boldsymbol{P}_j$ where all task states are -1 ($\forall i \in I_j : x_{j,i} = -1$) and the due date state's value is F_j:

$$\boldsymbol{P}_j = (-1, -1, \ldots, -1, F_j). \tag{7}$$

[1] Note that (3) is not a full matrix when project types have different numbers of tasks.

An important element of the pre-decision state is also the number of free resources, $B_k^{\text{free}}(s) \in \{0, 1, \dots, B_k\}, k = 1, \dots, K$. This is not explicitly included as part of the state because it can be calculated from the other state elements via:

$$B_k^{\text{free}}(s) = B_k - \sum_{j=1}^{J} \sum_{i=1}^{I_j} b_{j,i}^k Y\{x_{j,i} > 0\}, \tag{8}$$

where $Y\{\cdot\}$ is an indicator function.

2.4.2. Action

An action a represents the decision regarding which tasks to begin processing from those tasks whose states are "pending for processing". The action consists of action elements $a_{j,i}$ for each task $i \in \mathcal{I}_j$ of all project types ($j \in \mathcal{J}$). An action element takes the value of 1 ($a_{j,i} = 1$) to represent the decision to start processing a qualifying task and takes the value 0 ($a_{j,i} = 0$) otherwise:

$$a = \begin{bmatrix} a_{1,1}, & a_{1,2}, & \dots & a_{1,I_1} \\ a_{2,1}, & a_{2,2}, & \dots & a_{2,I_2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{J,1}, & a_{J,2}, & \dots & a_{J,I_J} \end{bmatrix}. \tag{9}$$

An action a must fulfil three requirements:

(1) The selected tasks for processing must have the task state "pending for processing":

$$\forall j \in \mathcal{J}, i \in \mathcal{I}_j : a_{j,i} = 1 \Rightarrow x_{j,i} = -1. \tag{10}$$

(2) There must be enough free resources of each type to allocate for processing the selected tasks:

$$\sum_{j=1}^{J} \sum_{i=1}^{I_j} b_{j,i}^k Y\{a_{j,i} = 1\} \le B_k^{\text{free}}(s), \ k = 1, \dots, K. \tag{11}$$

(3) All predecessor tasks of the selected tasks must be completed:

$$\forall j \in \mathcal{J}, i \in \mathcal{I}_j : a_{j,i} = 1 \Rightarrow \forall m \in \mathcal{M}_{j,i} : x_{j,m} = 0. \tag{12}$$

Here, m represents a predecessor of task i ($m \in \mathcal{I}_j \setminus i, m \in \mathcal{M}_{j,i}$). The action where all action elements are zero is also a valid action and indicates that no task was selected to begin processing in that period. More than one action may fulfil all three requirements for a pre-decision state. The set of these actions is named an action set $\mathcal{A}(s)$.

2.4.3. Post-decision state

A post-decision state $\hat{s}$ represents the system information immediately after a decision epoch and just before the transition time begins. In other words, a post-decision state is the system information from the pre-decision state s updated by an action a but before any task processing or random event occurs, i.e., $\hat{s} := f(s, a)$ where f is a deterministic function defined in (14) for each element of s and a. A post-decision state consists of the post-decision project states $\hat{\boldsymbol{P}}_j$. A post-decision project state consists of post-decision task states $\hat{x}_{j,i}$ of each task $i \in \mathcal{I}_j$ and the same due date states d_j of the pre-decision project state $\boldsymbol{P}_j$:

$$\hat{s} = \begin{bmatrix} \hat{x}_{1,1}, & \hat{x}_{1,2}, & \dots & \hat{x}_{1,I_1}, & d_1 \\ \hat{x}_{2,1}, & \hat{x}_{2,2}, & \dots & \hat{x}_{2,I_2}, & d_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \hat{x}_{J,1}, & \hat{x}_{J,2}, & \dots & \hat{x}_{J,I_J}, & d_J \end{bmatrix}. \tag{13}$$

A post-decision task state $\hat{x}_{j,i}$ is the updated state of a task from the preceding pre-decision task state $x_{j,i}$. It is only the tasks that have been selected to start their processing ($a_{j,i} = 1$) that change from the pre-decision state -1 to the post-decision state $t_{j,i}^{\max}$:

$$\hat{x}_{j,i} = \begin{cases} t_{j,i}^{\max}, & \text{if } x_{j,i} = -1 \text{ and } a_{j,i} = 1, \\ x_{j,i}, & \text{otherwise.} \end{cases} \tag{14}$$

2.4.4. Pre-acceptance state

A pre-acceptance state s^* represents the system information at the end of the transition time but immediately before the pre-decision state s' at the following decision epoch. A pre-acceptance state consists of its project states $\boldsymbol{P}_j^*$ which consist of pre-acceptance task states $x_{j,i}^*$ for each task $i \in \mathcal{I}_j$ and pre-acceptance due date states d_j^*. A pre-acceptance state shows the task processing progress after a post-decision state during the transitional time without new project arrivals. Eq. (17) shows possible task state transitions from a $\hat{x}_{j,i}$ to $x_{j,i}^*$ with their probabilities. As (15) presents, a pre-acceptance due date state is zero ($d_j^* = 0$) if project type j has just been completed or if the post-decision due date state is zero. For the other possibilities, a pre-acceptance due date state is equal to the post-decision due date minus one:

$$d_j^* = \begin{cases} 0, & \text{if } d_j = 0, \\ 0, & \text{if } \exists i \in \mathcal{I}_j : \hat{x}_{j,i} \ge 0 \text{ and } \forall i \in \mathcal{I}_j : x_{j,i}^* = 0, \\ d_j - 1, & \text{otherwise.} \end{cases} \tag{15}$$

If a type j project arrived during the transition time and state $\boldsymbol{P}_j^*$ is such that a type j project is completed or not present ($\forall i \in \mathcal{I}_j : x_{j,i}^* = 0$) the system accepts the new type j project. Otherwise, the system rejects the new arrival. From a pre-acceptance state s^* to the following pre-decision state s', the new task state becomes -1 and the due date state becomes F_j.

2.4.5. Transition function

The *transition function*, $s' = s^M(s, a, c)$, represents the transformation of a system from a pre-decision state s under action a to a pre-decision state s' at the next decision epoch by random events c during the transition time. Random events include new project arrivals, task completions and project completions. All task completions and project arrivals are stochastically independent.

A project of each type may arrive in the system during a transition time according to its type's arrival probability λ_j and is accepted if no type j project exists in the system.

A task may complete processing according to a conditional probability $\gamma_{j,i}(\hat{x}_{j,i})$, if the task's processed time following this transition ($t_{j,i}^{\max} - \hat{x}_{j,i} + 1$) is equal to or greater than its minimal possible duration $t_{j,i}^{\min}$. Formally, we define $\gamma_{j,i}(\hat{x}_{j,i}) = 0$ for $\forall \hat{x}_{j,i} > t_{j,i}^{\max} - t_{j,i}^{\min} + 1$, and require $\gamma_{j,i}(\hat{x}_{j,i}) > 0$ for $\hat{x}_{j,i} = t_{j,i}^{\max} - t_{j,i}^{\min} + 1$, which guarantees that $t_{j,i}^{\min}$ is the minimal possible duration. We also require $\gamma_{j,i}(1) = 1$, which guarantees that $t_{j,i}^{\max}$ is the maximal possible duration.

The probability of reaching a pre-decision state s' from a pre-decision state s under action a with the transition function $P(s'|s, a)$ is the joint probability of task completions $P(x'_{j,i}|\hat{x}_{j,i})$ and project arrivals $P(\boldsymbol{P}'_j|\hat{\boldsymbol{P}}_j)$:

$$P(s'|s, a) = \prod_{j=1}^{J} \left(\prod_{i=1}^{I_j} P(x_{j,i}^*|\hat{x}_{j,i}) \right) P(\boldsymbol{P}'_j|\boldsymbol{P}_j^*), \tag{16}$$

$$P(x_{j,i}^*|\hat{x}_{j,i}) = \begin{cases} \gamma_{j,i}(\hat{x}_{j,i}), & \text{if } \hat{x}_{j,i} \ge 1 \text{ and } x_{j,i}^* = 0, \\ 1 - \gamma_{j,i}(\hat{x}_{j,i}), & \text{if } \hat{x}_{j,i} \ge 2 \text{ and } x_{j,i}^* = \hat{x}_{j,i} - 1, \\ 1, & \text{if } x_{j,i}^* = \hat{x}_{j,i} \le 0, \\ 0, & \text{otherwise,} \end{cases} \tag{17}$$

$$P(\boldsymbol{P}'_j|\boldsymbol{P}_j^*) = \begin{cases} \lambda_j, & \text{if } \forall i \in \mathcal{I}_j : x'_{j,i} = -1, x_{j,i}^* = 0, \\ 1 - \lambda_j, & \text{if } \forall i \in \mathcal{I}_j : x'_{j,i} = x_{j,i}^* = 0, \\ 1, & \text{if } \forall i \in \mathcal{I}_j : x'_{j,i} = x_{j,i}^* \ne 0, \\ 0, & \text{otherwise.} \end{cases} \tag{18}$$

Here in (17), the first line represents that the post-decision task state of task i from project type j allows for task completion and, with probability $\gamma_{j,i}(\hat{x}_{j,i})$, the task will be completed by the pre-acceptance state. The second line represents that, with probability $1 - \gamma_{j,i}(\hat{x}_{j,i})$, the task will not be completed by the pre-acceptance state. The third line represents that the post-decision task state of task i from project type

j is completed or pending for processing and, with 100% probability, the task will retain its status in the pre-acceptance state.

In (18), the first line represents that, with λ_j probability, there will be an arrival of project type j during the transition time and the new type j project will take the place of the previously completed or non-existing type j project. The second line represents that, with $1-\lambda_j$ probability, there will be no new arrival of project type j during the transition time. The third line represents that, with 100% probability, the arrival of projects will not affect the status of ongoing or waiting projects of the same type as the new project will be rejected.

2.5. Objective function

The immediate profit represents the accrued profit during the state transition from s under action a to s', considering project completion rewards and tardiness costs. The immediate profit $R_{s,a,s'}$ is the sum of completed project rewards minus the tardiness cost of late completions ($d_j = 0$):

$$R_{s,a,s'} = \sum_{j=1}^{J}\left\{ \left(r_j - w_j Y\{d_j = 0\}\right) Y\left\{\exists i \in \mathcal{I}_j : \hat{x}_{j,i} > 0 \text{ and } \forall i \in \mathcal{I}_j : x'_{j,i} \le 0\right\}\right\}. \tag{19}$$

Here, the first indicator is for late project completion that takes the value 1 if a project completes later than its due date (i.e., the project's due date state $d_j = 0$) and takes the value 0 otherwise. The second indicator is for project completion that takes the value 1 if a project completes (at least one task is in progress in the post-decision state and all project tasks are complete at the end of the period) and takes the value 0 otherwise.

Our objective function seeks to find the policy that maximises the expected total discounted long-run profit:

$$V^*(s_1) = \max_{\pi \in \Pi} \mathbb{E}^{\pi}\left[\sum_{t=1}^{\infty} \alpha^{t-1} R_{s_t,a,s_{t+1}}\right]. \tag{20}$$

Here, s_1 is a given initial state; $R_{s_t,a,s_{t+1}}$ is the immediate profit of state transition from pre-decision states s_t to s_{t+1} under the action a at the time t; α is a discount factor in the interval (0,1); π is a policy from the set of all stationary deterministic policies Π that prescribe in every state s an action from the action set $\mathcal{A}(s)$.

3. Approximate dynamic programming (ADP)

In theory, the problem (20) can be solved using the Bellman equation:

$$V^*(s) = \max_{a \in \mathcal{A}(s)} \sum_{s' \in \mathcal{S}} P(s'|s,a)[R_{s,a,s'} + \alpha V^*(s')] \quad \forall s \in \mathcal{S}, \tag{21}$$

but in practice, it suffers from "the curse of dimensionality". "The curse of dimensionality" means that the number of states and computational requirements expands exponentially with the number of state variables (Sutton & Barto, 2018). Satic et al. (2022) investigated the limitations of DP and stated that a state space larger than their five projects with two tasks problem is computationally intractable for their hardware.

ADP is a modelling strategy to overcome "the curse of dimensionality" problem of DP due to the use of the Bellman equation (Powell, 2009). In our ADP algorithm, we estimate the value function of the Bellman Eq. (21) using a linear approximation model (27). A linear approximation model is a regression model that fits the value function of the Bellman equation by estimating a parameter vector θ (Powell, 2011, p 304). The linear approximation model only requires the current state and action information, and future state information and storing the states becomes unnecessary. With the linear approximation model, the decision making is done in an online fashion; thus, ADP can be used for larger size problems.

Table 1
Considered linear approximation models.

Model :	1	2	3	4	5	6	7	8	9	10	11
Feature 1	PT	CRU	TRU	DR	PT	PT	PT	CRU	TRU	PT	PT
Feature 2	–	–	–	–	CRU	TRU	DR	DR	DR	CRU	TRU
Feature 3	–	–	–	–	–	–	–	–	–	DR	DR

3.1. Linear model selection

We utilise four readily available project features and consider eleven linear approximation models comprised of different feature combinations. See Table 1 for details of each model's features. The first feature is a measure of the processing time to date (PT) of the project:

$$\sum_{i=1}^{I_j}\left\{ t_{j,i}^{\max} - \hat{x}_{j,i} + 1\right\} Y\left\{\hat{x}_{j,i} \ge 0\right\}. \tag{22}$$

Here, $t_{j,i}^{\max} - \hat{x}_{j,i}$ represents task i's processed time. Since the task's processed time is zero when the action is taken ($\hat{x}_{j,i} = t_{j,i}^{\max}$), we use the task's processed time after the transition time ends ($t_{j,i}^{\max} - \hat{x}_{j,i} + 1$) to differentiate the effect of actions.

The second feature is the current resource usage (CRU) of project j:

$$\sum_{k=1}^{K}\sum_{i=1}^{I_j} b_{j,i}^{k} Y\left\{\hat{x}_{j,i} > 0\right\}. \tag{23}$$

Here, we consider the post-decision state's resource allocation because it gives the best information about resources used by an action. For example, for single-period tasks, the resource allocation information of action may disappear after one transition time. Thus, the post-decision state's resource allocation gives the most precise information about resource usage after the action is taken.

The third feature is the total resource used (TRU) by project j to date:

$$B_j(s) + \sum_{k=1}^{K}\sum_{i=1}^{I_j} b_{j,i}^{k} Y\{\hat{x}_{j,i} > 0\}. \tag{24}$$

Here, $B_j(s)$ is the TRU of the previous state with the selected action $B_j(s_t) = TRU_j(\hat{s}_{t-1})$ and $\sum_{i=1}^{I_j} b_{j,i}^{k} Y\{\hat{x}_{j,i} > 0\}$ represents the amount of type k resource that will be used to process project type j under the latest taken action. This feature reflects the resource usage of a project and, indirectly, the time that the project has been processed in the system.

The fourth feature is decision reward (DR), which is the reward per period of processing, under the assumption the project will be processed continuously until completion:

$$\bar{R}_j = \begin{cases} \frac{r_j}{h_j} & \text{if } d_j > h_j \\ \frac{r_j - w_j}{h_j}, & \text{otherwise.} \end{cases} \tag{25}$$

In (25), the remaining late project horizon h_j is given by:

$$h_j = \sum_{i=1}^{I_j}\left\{ \hat{x}_{j,i} Y\{\hat{x}_{j,i} \ne -1\} + t_{j,i}^{\max} Y\{\hat{x}_{j,i} = -1\}\right\}, \tag{26}$$

i.e. the sum of remaining task durations assuming all remaining tasks are completed on the maximum task duration. This is compared to the remaining duration to the due date, d_j. When the remaining duration to the due date exceeds the remaining late project horizon, $\bar{R}_j$ takes the value of the project completion reward divided by the remaining late project horizon. Otherwise, $\bar{R}_j$ is reward minus tardiness cost divided by the remaining late project horizon. Using this feature, the duration to the project's due date, a worst-case estimate of the remaining processing duration, reward, and tardiness cost become decision elements.

Table 2
Counts of when the model is not statistically significantly different from the highest profit.

Model :	1	2	3	4	5	6	7	8	9	10	11
2p2t	9	10	8	10	10	0	10	10	10	9	8
2p3t	10	2	10	6	2	9	10	2	9	2	9
3p2t	3	1	2	4	2	3	6	3	3	0	5
2p10t	0	3	5	1	3	0	1	5	7	4	0
5p5t	1	0	7	1	1	3	1	0	4	1	0
5p10t	1	1	5	1	1	0	1	1	6	1	1
6p5t	0	3	0	1	2	4	0	1	0	0	3
10p10t2r	0	0	0	0	5	4	0	1	3	1	2
5p30t4r	7	1	7	7	1	0	1	1	1	2	0
2p30t2r	0	1	10	0	1	0	0	1	1	1	0
SUM	31	22	54	31	28	23	30	25	44	21	28

Table 3
Counts of when the model is not statistically significantly different from the lowest profit.

Model :	1	2	3	4	5	6	7	8	9	10	11
2p2t	0	0	0	0	0	10	0	0	0	0	0
2p3t	1	9	1	1	9	1	1	9	2	9	2
3p2t	2	4	0	0	1	2	2	3	0	3	2
2p10t	0	0	0	0	0	1	7	0	0	0	3
5p5t	0	0	0	0	0	1	0	2	0	0	8
5p10t	5	0	0	0	0	1	4	0	1	0	0
6p5t	1	2	0	1	0	2	1	2	1	1	0
10p10t2r	9	0	0	0	0	1	2	0	0	1	0
5p30t4r	0	0	0	0	0	5	2	1	2	0	2
2p30t2r	1	0	0	1	0	1	8	0	0	0	1
SUM	19	15	1	3	10	25	27	17	6	14	18

All models are evaluated on the one hundred problem scenarios that make up the numerical study of Section 5. Note that, for each of the 10 problem settings #p#t(#r) 10 problems that differ by the choice of arrival probability are considered. In the problem setting identifier, p and t are the number of projects and tasks and r, if present, is the number of types of resource (otherwise there is a single resource type). Each model has been trained using an appropriate version of Algorithm 1 in Section 3.2 with 100 iterations each having 100 simulations with 1000 periods and a discount factor $\alpha = 0.999$. In all scenarios, we used common seeds to generate project arrivals and task completions to create a fair comparison between models, e.g., the probability that the nth completion of project j's ith task is consistent across scenarios.

For our performance comparison in Table 2 and Table 3, we identify the model(s) with the highest (respectively, lowest) profit in each problem and compare this with the profits from each model via a Welch t-test at the 0.1% level of significance. We aggregate the results by recording the number of times each model was not statistically significantly different from the model(s) with the highest or lowest profit. Models 3 and 9 generated the highest profits or were not statistically significantly different from the highest profits in 54 and 44 of the 100 scenarios respectively. Other models generated the highest profits or were not statistically significantly different from the highest profits 33 or fewer times. In addition, Models 3 and 9 generated the lowest profits or were not statistically significantly different from the lowest profits in 1 and 6 of the 100 scenarios respectively.

Although Model 3 generated the highest profits more times than Model 9, we cannot claim that Model 3 is better than Model 9. In additional comparative Welch t-tests of each model to Model 3 in Tables 4 and 5, Model 3 generated higher*[2] results than Model 9 in 37 of 100 scenarios but lower* results than Model 9 in 36 of 100 scenarios.

Table 4
Counts of when model has higher* profit from than Model 3.

Model :	1	2	4	5	6	7	8	9	10	11
2p2t	1	2	2	2	0	2	2	2	2	2
2p3t	0	0	0	0	0	0	0	0	0	0
3p2t	6	2	9	1	7	7	4	6	3	5
2p10t	0	1	0	1	0	0	2	5	1	0
5p5t	0	0	0	0	2	0	0	2	0	0
5p10t	0	0	0	0	0	0	0	5	0	0
6p5t	3	4	3	8	7	4	5	8	6	7
10p10t2r	0	7	1	9	9	0	8	8	8	8
5p30t4r	2	0	2	0	0	1	2	0	0	0
2p30t2r	0	0	0	0	0	0	0	0	0	0
SUM	12	16	17	21	25	14	23	36	20	22

Table 5
Counts of when model has lower* profit from than Model 3.

Model :	1	2	4	5	6	7	8	9	10	11
2p2t	0	0	3	0	10	0	0	0	1	4
2p3t	0	9	6	9	1	0	9	1	9	1
3p2t	9	3	9	5	10	8	2	3	3	10
2p10t	2	7	1	5	2	2	5	0	6	5
5p5t	8	10	9	9	8	9	10	6	9	9
5p10t	10	9	9	9	10	10	9	5	9	9
6p5t	6	6	5	2	3	5	5	2	4	3
10p10t2r	10	2	9	1	1	10	2	2	2	2
5p30t4r	3	7	4	9	10	7	7	9	7	10
2p30t2r	10	9	10	9	10	10	9	9	9	10
SUM	58	62	65	58	65	61	58	37	59	63

The remaining 27 scenarios were NS differences. Since, Model 3 is not superior to Model 9, and Model 3 is a simple model with a single feature, Model 9 will be used in the remainder of the paper.

Model 9 considers relevant features of the dynamic and stochastic RCMPSP. The approximate value function $\bar{V}(\hat{s})$ for Model 9 is:

$$\bar{V}(\hat{s}) = \sum_{j=1}^{J} \left\{ \theta_j^1 \left\{ B_j(s) + \sum_{k=1}^{K} \sum_{i=1}^{I_j} b_{j,i}^k Y\{\hat{x}_{j,i} > 0\} \right\} + \theta_j^2 \bar{R}_j \right\}. \tag{27}$$

Coefficients θ_j^1 and θ_j^2 are generated using simulation training as described in Section 3.2. As an outcome of the combination of features and coefficients, the suggested approximation function considers a project's due date, reward, tardiness cost, remaining processing time, processing time to date and resource usage. All state information is used for decision making directly or indirectly. That creates a balanced approximation function.

We note that more complex models consider more features of the problem; thus, they have the potential to make better decisions using more information. However, their performance suffers from multicollinearity during the training phase. For example, Model 11, which has three features, typically generated the lowest profits in our tests. Experimenting on a single scenario where Model 11 produces the lowest profits, we heuristically combined the coefficients and features from Models 1, 2 and 4 to make a new approximating model that has the same features as Model 11. The resulting model generated higher profits than other linear models in the scenario. To sum up, the multicollinearity issue affected the training of the ADP models, so simpler linear models generated higher profits in our comparison.

[2] * represents a statistical significance level of 0.05. ** represents a statistical significance level of 0.01. *** represents a statistical significance level of 0.001. NS represents the absence of a statistically significant difference at a level of 0.05.

Algorithm 1: : ADP.

procedure ADP ALGORITHM
  $\forall j \in \mathcal{J} : \theta_j^1 = \theta_j^2 = 0$; initial state $s_1 = \mathbf{0}$. ▷ initial values
  for $itr = 1$ to $Iteration$ **do** ▷ for each iteration
    $\tilde{V}_{sim} = 0$ ▷ $\tilde{V}_{sim}$ is cumulative simulation profit
    for $sim = 1$ to $Simulation$ **do** ▷ for each simulation
      $\forall j \in \mathcal{J} : D_j^1(sim) = \left\{ B_j(s_1) + \sum_{k=1}^{K} \sum_{i=1}^{I_j} b_{j,i}^k Y\{\hat{x}_{j,i} > 0\} \right\}$
      $\forall j \in \mathcal{J} : D_j^2(sim) = \bar{R}_j$
      for $t = 1$ to $Period$ **do** ▷ for each simulation period
        find $\mathcal{A}(s_t)$ for s_t ▷ $\mathcal{A}(s_t)$ is the action set for s_t
        choose $a \in \underset{a' \in \mathcal{A}(s_t)}{\operatorname{argmax}} \sum_{j=1}^{J} \left\{ \theta_j^1 \left\{ B_j(s) + \sum_{k=1}^{K} \sum_{i=1}^{I_j} b_{j,i}^k Y\{\hat{x}_{j,i} > 0\} \right\} + \theta_j^2 \bar{R}_j \right\}$
        compute $s_{t+1} = s^M(s_t, a, c_t)$ ▷ state iteration via simulation
        $\tilde{V}_{sim} = \tilde{V}_{sim} + \alpha^{t-1} R_{s_t,a,s_{t+1}}$ ▷ $R_{s_t,a,s_{t+1}}$ explained at Section 2.5
      end for
      $s_1 = s_{Period+1}$ ▷ initial state for the next simulation
    end for
    choose $(\theta\_new_j^1, \theta\_new_j^2) \in \underset{\hat{\theta}\_new_j^1, \hat{\theta}\_new_j^2}{\operatorname{argmin}} \sum_{sim=1}^{Simulation} \left\{ \tilde{V}_{sim} - \sum_{j=1}^{J} \left(\hat{\theta}\_new_j^1 D_j^1(sim) + \hat{\theta}\_new_j^2 D_j^2(sim) \right) \right\}^2$
    $\tau = \tau_{harmonic}/(\tau_{harmonic} + itr - 1)$ ▷ harmonic step size
    $\forall j \in \mathcal{J} : \theta\_old_j^1 = \theta_j^1$ and $\theta\_old_j^2 = \theta_j^2$
    $\forall j \in \mathcal{J} : \theta_j^1 = (1-\tau)\theta\_old_j^1 + \tau\theta\_new_j^1$, $\theta_j^2 = (1-\tau)\theta\_old_j^2 + \tau\theta\_new_j^2$
  end for
  return $\forall j \in \mathcal{J} : \theta_j^1$ and θ_j^2
end procedure

3.2. Generation of approximation function coefficients

The ADP algorithm generates coefficients for features in the linear approximation model (27) using simulation and the least-squares fitting method. A least-squares fitting method minimises the sum of the squared residuals. The coefficient generation process is summarised in Algorithm 1.

Here, we train our approximation function with a set amount of iterations. In the first iteration, we assume the initial pre-decision state is an empty state with no existing project, and the coefficients are zero. In each iteration, we run a set amount of simulations, and from each simulation, we collect features D_j^1 and D_j^2 of the initial pre-decision states and cumulative simulated profit. After the simulations are completed, we estimate coefficients ($\forall j \in \mathcal{J} : \theta_j^1, \theta_j^2$) by minimising the sum of the squared deviations between the cumulative discounted profits and the linear approximation model (27) using a linear least-squares regression method. We denote the estimated coefficients by θ_new and existing coefficients by θ_old, and we use them in a dynamic step-size function (28) to generate coefficients of the new iteration:

$$\forall j \in \mathcal{J}, \forall g \in \{1,2\} : \theta_j^g = (1-\tau)\theta\_old_j^g + \tau\theta\_new_j^g. \tag{28}$$

We generate the dynamic step-size value τ with the harmonic step-size method:

$$\tau = \tau_{harmonic}/(\tau_{harmonic} + itr - 1). \tag{29}$$

We set the harmonic step-size value as $\tau_{harmonic} = 1$.[3]

We use the terminal pre-decision state of the iteration as the initial pre-decision state in the new iteration. After a specific amount of iterations, the final coefficients are used in the linear approximation model for online decision making.

During a simulation, we find the action set $\mathcal{A}(s_t)$ of the current pre-decision state s_t and select the most profitable action using the objective function:

$$\text{choose } a \in \underset{a' \in \mathcal{A}(s_t)}{\arg\max} \sum_{j=1}^{J} \left\{ \theta_j^1 \{ B_j(s) + \sum_{k=1}^{K} \sum_{i=1}^{I_j} b_{j,i}^k Y\{\hat{x}_{j,i} > 0\} \} + \theta_j^2 \bar{R}_j \right\}. \tag{30}$$

In the case of multiple actions bringing the highest profit, we select the action that processes most tasks. If the tie continues, the action that processes tasks of the higher index is selected, e.g., project type 5 is prioritised over project type 1.

The post-decision state $\hat{s}$ begins with the implementation of the best action. Then random events c_t are simulated during the transition time according to the transition function and the new pre-decision state s'_{t+1} is achieved. If any project completes during the transition time, their profit is added to the cumulative profit with discounting using the discounting function α^{t-1}. This simulation process repeats for a specific amount of periods, and the final pre-decision state is used as an initial pre-decision state in the next simulation.

Algorithm 2: : Online Decision Making.

procedure ONLINE DECISION MAKING
  find $\mathcal{A}(s_t)$ for s_t ▷ $\mathcal{A}(s_t)$ is the action set for s_t
  put set Π^* = $\underset{a' \in \mathcal{A}(s_t)}{\operatorname{argmax}} \left\{ \sum_{s'_t \in S} P(s'_t | s_t, a') R_{s_t,a',s'_t} + \alpha^{t-1} \sum_{j=1}^{J} \left\{ \theta_j^1 \{ B_j(s) + \sum_{k=1}^{K} \sum_{i=1}^{I_j} b_{j,i}^k Y\{\hat{x}_{j,i} > 0\} \} + \theta_j^2 \bar{R}_j \right\} \right\}$
  select an action $a \in \Pi^*$ ▷ Π^* is the set of all best action for s_t
  return a
end procedure

3.3. Online decision making

The approximation function evaluated by Algorithm 1 can be used for online decision making for any pre-decision state. The online decision-making process is summarised in Algorithm 2. First, all actions for the pre-decision state are generated, then the expected profits are calculated using the approximation function for each action. The action with the highest profit is used for the state, with tie-breaking applied as described in Section 3.2.

4. Compared algorithms

We used DP, ORBA, GA and RBA for benchmarking with our ADP algorithm. ORBA and GA are applied to the dynamic problem using a reactive scheduling method which reschedules the plan when a new project arrival occurs by rerunning the algorithm. ORBA and GA generate a task processing order to create an action for a given pre-decision state. DP generates the optimal policy which we denote π^*. RBA directly creates an action for a pre-decision state according to some predefined rules or criteria.

4.1. Dynamic programming (DP)

DP calculates optimal policies from an MDP model of the problem by solving the Bellman equation (Sutton & Barto, 2018). We used the value iteration method. We used DP only for problems from Satic et al. (2022) for benchmarking.

4.2. Optimal reactive baseline algorithm (ORBA)

ORBA is an exact brute force algorithm that sequentially solves a static RCMPSP from state s_t to optimality. The static problem results by ignoring the possibility of future project arrivals. New project arrivals

[3] We also considered $\tau_{harmonic} = 10$ from Powell (2011, p 451) and KESTEN's stepsize rule (Powell, 2011, p 436), but we received better results with the stated settings.

Algorithm 3: : Value Iteration.

procedure STATE VALUE ITERATION PROCEDURE
  $\beta = 0.001$ ▷ β is the stopping parameter
  $\forall s \in \mathcal{S} : V^{old}(s) = 0$ ▷ initial state values
  repeat
    for $\forall s \in \mathcal{S}$: **do**
      $V(s) = \max_{a \in A(s)} \sum_{s' \in \mathcal{S}} P(s'|s,a)(R_{s_t,a,s_{t+1}} + \alpha V^{old}(s'))$ ▷ value function
    end for
    $W_{max} = \max_{s \in \mathcal{S}} |V(s) - V^{old}(s)|$ ▷ maximum value change
    Update $\forall s \in \mathcal{S} : V^{old}(s) = V(s)$
  until $W_{max} \leq \beta(1-\alpha)/(2\alpha)$
end procedure

disrupt the current schedule and a new schedule incorporating these is required. In such a manner, ORBA represents an optimal reactive scheduling algorithm. Due to the computational requirements of brute force algorithms, ORBA runs in factorial time and only small-size dynamic and stochastic RCMPSP problems can be solved within a reasonable time. Thus, we limit our test problems with ORBA to a maximum of 10 tasks. The ORBA used here extends that in Satic et al. (2022) by allowing for multiple resource types.

Specifically, for a given pre-decision state s_t, ORBA calculates the profits and makespans of all *precedence-feasible task scheduling orders* (TSOs) then selects a TSO of maximal profit. A precedence-feasible TSO is a permutation σ of tasks such that, for any $m < n$ with $\sigma(m) = (j,i)$ and $\sigma(n) = (j,k)$, $k \notin \mathcal{M}_{j,i}$. In the case of ties, the candidate schedule with minimal makespan is selected, and the schedule of the smallest project and task indices are selected from any remaining candidate schedules.

The generated TSO is converted to non-idling action a for given pre-decision state s_t using a serial schedule-generation scheme (SGS). A non-idling action is one that will always allocate resource to tasks when it is possible to do so. In an SGS, if there are enough free resources to process the next task in the TSO, its action becomes one ($a_{j,i} = 1$), and its resource usage is subtracted from the free resources. The process then repeats for the remaining tasks in order of the TSO until either no tasks can begin processing or there is insufficient free resources to process any of the remaining tasks. The TSO is followed in future periods as long as no new project arrives. If a new project arrival disturbs the system, the current TSO becomes invalid, and ORBA generates a new TSO.

4.3. Genetic algorithm (GA)

GA is a heuristic algorithm which searches the solution space using a set of solutions (population). GA then improves the population many times (generation) using bio-inspired operators such as crossover and mutation to find better solutions. We used the genetic algorithm to benchmark with our ADP algorithm since GA is the most popular method for RCPSP family. GA is applied to dynamic problems using a reactive scheduling method. We used GA from Satic et al. (2022) and, in this paper, we extended it to multiple resource types.

For a given pre-decision state s_t, GA generates the desired population size amount of precedence-feasible TSOs, which are random permutations of the pending for processing tasks ($x_{j,i} = -1$). The algorithm evaluates profits and makespans of the TSOs and then ranks them using these values. These were evaluated via simulation under an assumption that there will be no new project arrivals and all tasks complete at $t^{\max}$. TSOs with higher profits get a higher rank. In the case of ties, TSOs with smaller makespans get a higher rank. If the tie continues, TSOs ranked according to their creation time (earliest to latest). This first set of TSOs is called the first generation, and the highest ranked TSO is called the best TSO. After the first generation is generated, GA begins iterations.

At each iteration, GA creates an empty set of TSOs and fills this new set with TSOs to the desired population size amount using elitist selection, crossover and mutation operators. The elitist selection operator copies the desired amount of highest ranked TSOs from the previous generation of TSOs to the new empty set of TSOs (new generation). Crossover and mutation operators fill the rest of the new generation.

The crossover operator randomly selects two TSOs from the previous generation and randomly selects a task inside of the first TSO. The crossover operator copies tasks, from the earliest task to be processed up to the randomly selected task of the first (selected) TSO, to (make) a new TSO without changing the order of these tasks. Then, the crossover operator re-orders the remaining tasks of the first (selected) TSO according to order of these tasks in the second (selected) TSO, then adds them to the new TSO (to after the randomly selected task). The new TSO is always a precedence-feasible TSO since it is created according to the order of tasks in both selected precedence-feasible TSOs.

The new TSO may be adjusted by the mutation operator with a desired probability. Under the mutation operation, a task is selected at random and the location of this task in the TSO is randomly re-assigned. The new location cannot be later than the task's previous order and cannot be sooner than its latest to be processed predecessor task. Thus the mutation operator also ensures that the newly generated TSO is a precedence-feasible TSOs. Then the new TSO is added to the new generation. When the size of the new generation reaches the population size amount, the TSOs are then ranked as in the first generation. GA iterates the generations until the desired number of generations is reached. The best TSO of the final generation is used for decision making.

The TSO is converted to an action in the same way as for ORBA. Similar to ORBA, GA's TSO can be used for future pre-decision states as long as a new project arrival does not disturb the system.

Since the reactive scheduling method reruns GA for each time an arrival disturbs the processing plan, the computational time requirement increases with the problem size.

4.4. Rule-based algorithm (RBA)

Rule-Based Algorithm (RBA) is a priority-based heuristic algorithm which uses the longest task first priority rule. We considered RBA in benchmarking to show the performance of a simple heuristic algorithm. Due to the simplicity of the algorithm, it runs very fast for all problem sizes.

For a given pre-decision state s_t, RBA creates a precedence-feasible TSO where the tasks with the longest task processing durations have priority over other tasks. Then the TSO is converted to an action as same as in ORBA.

5. Computational results

We simulate the dynamic project scheduling environment with random new project arrivals and stochastic task durations, and we compare the expected total discounted long-run profit performance of DP, ADP, ORBA, GA and RBA. Algorithm 4 shows the simulation procedure we used in our comparisons. The statistical significance of ADP (Model 9) against other methods are shown in the tables at three levels $(0.001, 0.01, 0.05)$. The experiments are coded in JuliaPro 1.3.1.2 on a desktop computer with Intel i5-11400F CPU with 2.60 GHz clock speed and 32 GB of RAM.

We used 100 problem scenarios in our comparison which are a combination of 10 project settings and 10 project arrival probabilities. These arrival probabilities λ_j are 0.01 and from 0.1 to 0.9 with increments of 0.1. Since we consider a dynamic environment $\lambda_j = 0$ is not used in this comparison instead $\lambda_j = 0.01$ is used to represent the

Algorithm 4: : Simulation.

procedure PROFIT SIMULATION
  for $sim = 1$ to $Simulation$ **do** ▷ for each simulation
    $\tilde{V}_{sim} = 0$, $s_1 = \mathbf{0}$ ▷ $\tilde{V}_{sim}$ is cumulative simulation profit, s_1 the initial empty pre-decision state
    for $t = 1$ to $Period$ **do** ▷ for each simulation period
      $s_{t+1} = s^M(s_t, a, c_t)$ ▷ $a \in \pi(s_t), \pi(s_t)$ is the policy of selected solution method
      $\tilde{V}_{sim} = \tilde{V}_{sim} + \alpha^{t-1} R_{s_t,a,s_{t+1}}$
    end for
  end for
  $\tilde{V} = \frac{1}{Simulation} \sum_{sim=1}^{Simulation} \tilde{V}_{sim}$
  return $\tilde{V}$
end procedure

nearly-static case. Also, $\lambda_j = 1$ is not used because it causes a non-ergodic MDP where some feasible states cannot be reachable from any states (for example a state where all projects are completed but no new project has arrived). Of the 100 scenarios, 30 represent smaller sized problems on which we are able to compare all solution algorithms. For the 70 larger sized problems it is only possible to evaluate ADP, GA and RBA.

In Algorithm 4, we run 100 simulations for 1000 simulation periods and a discount rate of $\alpha = 0.999$. (For the small instances we use 10 000 simulations for DP, ADP, RBA and ORBA.) If a period represents a day, this period would represent approximately three years of processing time. Simulations start from the empty pre-decision state ($s_1 = \mathbf{0}$). In each simulation period an action is generated with the policy being investigated (π). Then the following pre-decision state is generated given the action taken and the transition function. The profit is generated, recorded and included within $\tilde{V}_{sim}$. The discounted profits of completed projects during the transition time are added to $\tilde{V}_{sim}$. After the end of the simulations, the average discounted long-run profit $\tilde{V}$ of the investigated solution method is calculated.

ADP (Algorithm 1) is trained for 100 iterations each having 100 simulations with 1000 periods. GA is trained for 100 generations, each with 100 solutions. The elitest selection operator transfers the best 10% of solutions to the next generation. A new TSO created by the crossover operator is handled by the mutation operator with a 50% chance. We note that, while ADP requires training once prior to the simulations, GA requires multiple training occurrences during the simulations. GA is required to generate a new schedule each time a new project arrives.

In this study, we assume that the number of tasks of different project types is equal. Project tasks have a completion duration range $t^{\max} - t^{\min} + 1 = 3$ and have uniformly distributed completion probabilities. The shortest maximum task duration in our study is $t^{\max} = 2$, for such tasks the completion duration range is 2 and have an arbitrary completion probability distribution. The completion probabilities used in this computational study are:

$$\gamma_{j,i}(\hat{x}_{j,i}) = \begin{cases} \frac{1}{\hat{x}_{j,i}}, & \text{if } t_{j,i}^{\max} \geq 3,\ 1 \leq \hat{x}_{j,i} \leq 3 \\ \frac{1}{3}, & \text{if } t_{j,i}^{\max} = 2,\ \hat{x}_{j,i} = 2 \\ 1, & \text{if } t_{j,i}^{\max} = 2,\ \hat{x}_{j,i} = 1 \\ 0, & \text{otherwise.} \end{cases} \tag{31}$$

5.1. Optimality gaps for small instances

We used project settings from Satic et al. (2022). The problems' data is available in their paper and at https://github.com/ugursatic/DSRCMPSP. These problems are arbitrarily created to be small and solvable by DP. In these problems, order strength, resource factor and the number of resources are set to 1.00. In other words, project networks are serial, and all tasks use the same amount of resources. The resource strength of the first and third problems is 0.00, which means some tasks require the allocation of all resources. The resource strength of the second problem is 0.50. Here, the initial task of each project can be processed in parallel with any task of the other projects. Using these small-size problems we are able to compare ADP's performance with optimal policies of DP, scheduling orders of ORBA, GA and RBA.

Tables 6–8 illustrate the expected total discounted long-run profits (which are averages of simulations) of policy generation methods (vertical) at different arrival probabilities (horizontal). The colour of cells shows the statistical significance of compared algorithms against ADP.

The simulation results of two project types, two tasks and one resource type problems are shown in Table 6. ADP produces higher*** profits than ORBA, GA and RBA, except for $\lambda_j = 0.01$. Recall that GA is evaluated on fewer simulations. Hence, at $\lambda_j = 0.01$, RBA's profit is lower*** than ADP's while GA's profit is NS with a similar profit achieved.

The simulation results of two project types, three tasks and one resource type problems are shown in Table 7. Algorithms' profits are NS different from each other at $\lambda_j = 0.01$. From $\lambda_j = 0.1$ to $\lambda_j = 0.8$, ADP has higher** profits than ORBA, GA and RBA.

The simulation results of three project types, two tasks and one resource type problems are shown at Table 8. ADP has higher*** profits than RBA except for $\lambda_j = 0.01$. At $\lambda_j = 0.1, 0.2, 0.8, 0.9$, ADP has higher* profits than ORBA, GA and RBA.

We see the same result as in Satic et al. (2022) in that the reactive scheduling methods ORBA and GA have close to optimal profits at $\lambda_j = 0.01$ where the results are NS different from each other. However, their results usually diverge from optimum as λ_j increases.

In summary, our comparison of project settings from Satic et al. (2022) shows that ADP cannot match the optimal policy of DP. DP's advantage over the other policies is that it is able to better identify opportunities to defer the use of resources to start processing a project task to more profitable projects in later periods. ADP generated higher* profits than ORBA, GA and RBA respectively in 22, 21 and 27 of 30 problem scenarios. ADP generated NS different profits than ORBA, GA and RBA respectively in 2, 3 and 2 of 30 problem scenarios. ADP generated lower* profits than ORBA, GA and RBA respectively in 6, 6 and 1 of 30 problem scenarios. Thus, we showed that our linear ADP model performs better than ORBA, GA and RBA in up to 90% of the scenarios from Satic et al. (2022).

5.2. Test problem generation

The problems of Satic et al. (2022) were the only dynamic and stochastic RCMPSPs in the literature that have a reward after completion, a tardiness cost after a given due date, arrival probability of new projects during a transition time, randomly early, normal and late task completions. However, Satic et al. (2022) only considered small-size problems where the project network is sequential (serial, $OS_j = 1$). Thus we generate larger size test problems using ProGen/Max and MPSPLIB problems.

ProGen/Max is an RCMPSP generation software which is developed by Schwindt (1998) which extends its predecessor ProGen (Kolisch et al., 1995) with an option to consider the minimum and maximum time lags between the start of activities.

We used ProGen/Max to generate RCPSPs with different activity-on-node networks, order strength (denoted OS_j), task durations, resource usage, and resource availability. We combined these RCPSPs problems and added stochastic task completion, project arrival probability, project completion reward, late completion cost, and due date. We add reasonable completion rewards and tardiness costs to each project. We used the generated task durations as expected task durations $t_{j,i}$ and added one minimal possible ($t_{j,i}^{\min} = t_{j,i} - 1$) and one maximal possible ($t_{j,i}^{\max} = t_{j,i} + 1$) duration options. We tested all problems with ten different λ_j options which are $0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9$. We generated the due date of the project via (32), where ρ is an arbitrary factor, which we set to 1.5. We adjusted the resource availability using

Table 6
Two project types and two tasks problem.

λ_j	0.01	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
DP	77	503	759	907	1000	1063	1109	1143	1169	1190
ADP	75	481	710	835	911	960	995	1020	1038	1051
ORBA	73	466	669	768	817	840	846	844	837	829
GA	72	452	636	708	745	752	754	735	735	724
RBA	72	413	529	551	542	525	507	491	480	473

Table 7
Two project and three tasks problem.

λ_j	0.01	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
DP	115	590	798	905	970	1013	1044	1066	1083	1097
ADP	115	584	786	889	949	988	1015	1034	1048	983
ORBA	115	575	768	862	915	947	966	979	988	995
GA	114	573	772	857	911	952	965	987	984	999
RBA	115	573	762	854	905	937	954	968	975	983

Table 8
Three project and two tasks problem.

λ_j	0.01	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
DP	199	878	1044	1122	1197	1263	1325	1378	1427	1468
ADP	181	746	857	936	1012	1089	1163	1229	1342	1373
ORBA	182	728	854	950	1042	1129	1207	1273	1325	1366
GA	186	727	848	947	1040	1128	1209	1277	1327	1367
RBA	183	736	815	862	921	986	1050	1114	1173	1228

Significantly lower results than ADP	p<0.05,	p<0.01,	p<0.001
Significantly better results than ADP	p<0.05,	p<0.01,	p<0.001

(33) because the combination of resource availabilities of multiple single project problems makes the multi-project problem resource-rich:

$$F_j \approx \left\{ (1-OS_j)\max\left\{ J\frac{\sum_{k=1}^{K}\left(\frac{\sum_{i=1}^{I_j}(t_{j,i}b_{j,i}^k)}{B_k}\right)}{K}, \max\{t_{j,1},t_{j,2},\dots,t_{j,I_j}\}\right\} + (OS_j)\max\left\{\sum_{i=1}^{I_j}t_{j,i}, J\frac{\sum_{k=1}^{K}\left(\frac{\sum_{i=1}^{I_j}(t_{j,i}b_{j,i}^k)}{B_k}\right)}{K}\right\}\right\}\rho, \tag{32}$$

$$B_k \approx \sum_{j=1}^{J}\left(B_k^j\left(\frac{50-4J}{100}\right)\right). \tag{33}$$

The MPSPLIB (http://www.mpsplib.com/) is a RCMPSP library which contains the problem set of Homberger (2007). These RCMPSP problems are made by combining single project problems of PSPLIB (http://www.om-db.wi.tum.de/psplib) which is RCPSP library (Kolisch & Sprecher, 1996). PSPLIB problems are generated with ProGen.

The MPSPLIB contains 140 instances that differ by project type number, project number, task number, global resource type number and arrival times. Global resources are shared among all projects, and local resources are only used for a single project. Compared to our ProGen/Max generated problems, MPSPLIB problems have predefined tardiness costs and due dates. However, these due dates ($Best_j$) are the shortest completion time found for single project problems, which we need to modify to use in the dynamic multi-project setting.

From the MPSPLIB, we only considered 30 tasks per project problems for algorithm benchmarking. We combined the given global and local resources for each resource type. Then we reduced the resource amount using (33). We use the predefined tardiness costs and twice the tardiness cost as completion rewards to each project. We use the stochastic task completion and arrival probabilities as same as ProGen/Max generated problems. We generated the due date of the project as:

$$F_j \approx Best_j + J\frac{\sum_{k=1}^{K}\frac{\sum_{i=1}^{I_j}(t_{j,i}b_{j,i}^k)}{B_k}}{K}. \tag{34}$$

5.3. Performance analysis for larger instances

We created five project settings with ProGen/Max and two project settings with MPSPLIB problems. The parameters of these problems are shown in Appendix A. Our ProGen/Max problems are not resource-rich, and their resource strengths are between 0.007 and 0.176. Maximum task durations of these problems range between 2 and 21 according to a uniform distribution. More detailed information (e.g., task duration, project network, resource usages) about these problems and more detailed test results are available at https://github.com/ugursatic/DSRCMPSP. The size of these problems exceeds the computational limits of DP and ORBA on our hardware, so we only compared ADP, GA and RBA.

In a number of the larger scale problems, there can be relatively small increases in profit as the arrival probability increases. We have included the full range of results for completeness and to appraise potential differences in performance at different arrival probabilities. A broader discussion on the effect of arrival probabilities is provided in Appendix C.

In five project types, five tasks and four resource types problems, shorter projects are more profitable than longer ones. Thus, RBA with longer tasks first rule is disadvantaged, and we expect that ADP outperforms RBA. Table 9 shows that ADP produces better*** results than GA and RBA in all λ_j values.

In two project types, ten tasks and two resource types problems, project type 1 is more profitable based on its reward/project horizon[4] ratio. Also, tasks of project type 1 are longer than project type 2. Thus

[4] The project horizon is the sum of its maximal task durations.

Table 9
Five project types, five tasks and four resource types problem.

λ_j	0.01	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
ADP	937	1492	1453	1495	1475	1494	1369	1473	1295	1485
GA	869	1128	1143	1141	1158	1146	1143	1148	1148	1145
RBA	698	601	576	562	558	549	527	538	517	507

Table 10
Two project types, ten tasks and two resource types problem.

λ_j	0.01	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
ADP	292	424	490	502	519	754	756	759	759	760
GA	304	448	454	455	457	457	457	458	459	455
RBA	290	420	423	427	437	445	453	454	464	467

Table 11
Five project types, ten tasks and two resource types problem.

λ_j	0.01	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
ADP	530	2463	2489	2464	2442	2508	2436	2457	2513	2509
GA	1757	2324	2330	2337	2336	2339	2346	2341	2339	2342
RBA	1769	2341	2356	2377	2376	2378	2389	2377	2392	2389

Table 12
Six project types, five tasks and two resource types problem.

λ_j	0.01	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
ADP	1113	1632	824	1408	1660	1443	1740	1538	1310	1560
GA	1226	1308	1313	1312	1307	1301	1301	1299	1318	1311
RBA	1210	1145	1107	1106	1127	1084	1100	1098	1095	1086

this problem is advantageous for the longest task first priority rule of RBA. However, Table 10 shows that ADP usually produces higher*** profits than alternatives except for low project arrival rates $\lambda_j = 0.01$ and $\lambda_j = 0.1$.

In five project types, ten tasks and two resource types problems, the tardiness costs are set to approximately 10% of the project rewards. Table 11 shows that ADP's profits are higher*** than GA and RBA from $\lambda_j = 0.1$ to $\lambda_j = 0.9$.

Six project types, five tasks and two resource types problems consist of six copies of the same project with different reward and tardiness cost combinations. In Table 12 ADP produces higher*** profits for most project arrival probabilities except for $\lambda_j = 0.01$ and $\lambda_j = 0.2$. ADP's and GA's profits are NS different at $\lambda_j = 0.8$.

The ten project types, ten tasks and two resource types problems are the largest problems we generate with ProGen/Max. In this problem, we arbitrarily assigned rewards and tardiness costs. Table 13 shows that ADP has higher*** profits than GA and RBA at all project arrival probabilities.

Two project types, thirty tasks and four resource types problem is the smallest MPSPLIB problem in our comparison. The problem name is mp_j30_a2_nr2_set in MPSPLIB, and it has one global and three local resources. The resource strength of type one resource is 0.207. But other resource strength values are 0.01, 0 and 0. These values represent that at least one large task cannot be processed in parallel with different tasks and might create a bottleneck. Table 14 shows that ADP leads to higher*** profits than GA at most arrival probabilities except for $\lambda_j = 0.1$. ADP's profits are higher*** than RBA at most arrival probabilities except for $\lambda_j = 0.1$ and $\lambda_j = 0.3$.

Five project types, thirty tasks and four resource types problem is the largest problem we have used in our comparison. The problem name is mp_j30_a5_nr4_set in MPSPLIB. The original problem has three global and one local resource type. We used the given due dates in the problem without changing them. The resource strengths of resource types one, two and three are high (0.449–0.687) and the order strengths of these projects are low. In other words, many tasks can be processed together, and free-resource availability may allow it easily. Table 15 shows that ADP produces higher*** results than RBA in $\lambda_j = 0.1$ and $\lambda_j = 0.5$.

In summary, our comparison of larger size problems shows that ADP's profit was higher* than GA in 53 of 70 problem scenarios; GA's profit was higher* for 13 problem scenarios, and there was NS difference in the remaining 4 problem scenarios. ADP generated higher* profits than RBA in 55 problem scenarios; RBA's profit was higher* than ADP only in 12 problem scenarios, and results between ADP and RBA are NS different in 3 problem scenarios. These results show that the overall performance of ADP is better* in the majority of larger-size problems than GA and RBA.

6. Conclusion

Paper summary. In this paper, we modelled the dynamic and stochastic RCMPSP as an infinite-horizon discrete-time MDP where projects have identical arrival probabilities at each transition time, and tasks have random durations. Our objective function maximises the expected total discounted long-run profit. We used a linear approximation model to design a practical scheduling policy and showed that it performs near-optimally in small problems and compares favourably to existing heuristics in large problems.

The motivation of this study is to create a more comprehensive project scheduling model by considering the uncertainties of stochastic task durations, random new project arrivals, multiple types of resource usages and bigger and complex project networks. For this purpose, we suggest a linear approximation model which generates decisions based on resource consumption and decision rewards. Our linear approximation model generated the best profits after the exact methods in our comparisons and contributed to the literature by extending the work of Satic et al. (2022) which only considered small-sized projects with sequential networks and single resource type.

This study provides an efficient ADP algorithm for dynamic and stochastic RCMPSP, which generates profits that are significantly higher than or equal to the profits of ORBA, GA and RBA in respectively 80%, 81% and 87% of our comparisons. DP produces better* results than ADP for small-size problems. However, it suffers from the curse of dimensionality and is not suitable for larger problems. ADP is a viable

Table 13
Ten project types, ten tasks and two resource types problem.

λ_j	0.01	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
ADP	650	592	576	960	928	865	889	894	932	884
GA	522	536	546	544	539	542	546	540	540	548
RBA	552	550	555	554	553	556	549	552	554	555

Table 14
Two project types, thirty tasks and four resource types problem.

λ_j	0.01	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
ADP	176	184	221	206	224	212	213	214	214	214
GA	168	197	195	192	193	192	192	190	189	193
RBA	161	200	208	209	207	209	210	209	209	210

Table 15
Five project types, thirty tasks and four resource types problem.

λ_j	0.01	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
ADP	721	1515	864	1143	904	1748	913	1104	1591	1595
GA	719	1546	1665	1706	1727	1740	1741	1748	1755	1754
RBA	717	1493	1573	1589	1609	1614	1617	1627	1623	1627

Table A.16
Two project types and two tasks problem.

Project attributes		
Number of project types	2	
Project type no	1	2
Completion reward	3	10
Due date	8	5
Tardiness cost	1	9
Order strength	1.00	1.00
Resource attributes		
Resource factor	1.00	
Number of resource types	1	
Resource type no	1	
Resource amounts	3	
Resource strength	0.00	

Table A.17
Two project types and three tasks problem.

Project attributes		
Number of project types	2	
Project type no	1	2
Completion reward	12	6
Due date	10	15
Tardiness cost	8	5
Order strength	1.00	1.00
Resource attributes		
Resource factor	1.00	
Number of resource types	1	
Resource type no	1	
Resource amounts	3	
Resource strength	0.50	

Table A.18
Three project types and two tasks problem.

Project attributes			
Number of project types	3		
Project type no	1	2	3
Completion reward	8	5	20
Due date	10	8	10
Tardiness cost	5	3	19
Order strength	1.00	1.00	1.00
Resource attributes			
Resource factor	1.00		
Number of resource types	1		
Resource type no	1		
Resource amounts	3		
Resource strength	0.00		

solution method for more practical, larger scale problems that cannot be tackled by DP.

Also, this study gives insights to project managers to determine more suitable methods for their environment by providing a performance comparison of ADP, DP, ORBA, GA and RBA methods in various project settings and various arrival probabilities. We suggest using DP for problems that are within the computational limitations of DP. However, we suggest using ADP methods for larger problems.

Future research direction. In real life, there are more dynamic and stochastic elements in dynamic project scheduling environments than those (stochastic task durations, uncertain new project arrivals, finish to start project networks, multiple resource types and usage) considered in this paper. Future work might consider other elements of the dynamic project scheduling environment, such as stochastic resource availability or multiple modes of task processing.

Acknowledgements

We acknowledge Mahshid Salemi Parizi for making their code available. The first author acknowledges the Ministry of National Education of The Republic of Turkey for providing a PhD scholarship. We thank the two anonymous reviewers for their careful and detailed reviews of the paper.

Appendix A. Problem attributes

See Tables A.16–A.25.

Appendix B. Modelling assumptions and model generality

Models are always simplifications of reality, but may still be useful for making decisions if they capture the key problem features and are solvable within a practical time amount.

Project due dates and task durations are given in whole numbers of periods, and available resources and task resource usages are given in non-negative whole units. If a task would in practice last for a fractional number of periods, we round it up to an integer assuming that the resource employed on that task cannot be re-allocated to a new task until the beginning of the next period (for the remainder of the period during which a task is completed, the resource may be allowed to take a vacation or be allocated to other tasks which are shorter than one period or are preemptive; these are however not included in

Table A.19
Five project types, five tasks and four resource types problem.

Project attributes					
Number of project types	5				
Project type no	1	2	3	4	5
Completion reward	63	50	19	48	46
Due date	54	105	232	96	139
Tardiness cost	21	25	15	25	25
Order strength	0.80	0.70	0.50	0.50	0.40
Resource attributes					
Resource factor	0.80				
Number of resource types	4.00				
Resource type no	1	2	3.00	4.00	
Resource amounts	18	19	15	18	
Resource strength	0.12	0.1	0.08	0.13	

Table A.20
Two project types, ten tasks and two resource types problem.

Project attributes		
Number of project types	2	
Project type no	1	2
Completion reward	10	72
Due date	125	129
Tardiness cost	5	15
Order strength	0.51	0.51
Resource attributes		
Resource factor	0.75	
Number of resource types	2	
Resource type no	1	2
Resource amounts	12	16
Resource strength	0.10	0.18

Table A.21
Five project types, ten tasks and two resource types problem.

Project attributes					
Number of project types	5				
Project type no	1	2	3	4	5
Completion reward	99	83	85	98	92
Due date	131	138	166	135	180
Tardiness cost	10	8	9	10	9
Order strength	0.36	0.36	0.36	0.69	0.67
Resource attributes					
Resource factor	0.70				
Number of resource types	2				
Resource type no	1	2			
Resource amounts	19	20			
Resource strength	0.11	0.13			

Table A.22
Six project types, five tasks and two resource types problem.

Project attributes						
Number of project types	6					
Project type no	1	2	3	4	5	6
Completion reward	99	75	50	30	10	80
Due date	177	177	177	177	177	177
Tardiness cost	25	25	25	25	5	60
Order strength	0.50	0.50	0.50	0.50	0.50	0.50
Resource attributes						
Resource factor	1.00					
Number of resource types	2					
Resource type no	1	2				
Resource amounts	24	18				
Resource strength	0.15	0.11				

the considered system). Analogously, if a task would in practice use a fractional number of resource units, we round it up to an integer assuming that each resource unit employed on that task cannot be split between several tasks (the remainder of the fractional resource unit may be allowed to take a vacation or be allocated to other tasks which require less than one resource unit; these are however not included in the considered system).

Geometrically distributed inter-arrival times are the discrete time analogue of exponentially distributed inter-arrival times in a continuous time setting that are commonly deployed to model random arrivals. The benefit of the memoryless property of this distribution means that the probability of a project arrival is constant and independent of the system state. Other discrete probability distributions can be used with some additional effort. For example, assume that the project arrival probability distribution is defined by the number of periods since the last project arrival of the same type. In that case, by expanding the state to include this information, we can evaluate the conditional probability of an arrival in the next period given the number of periods since the last arrival.

The technical assumptions of our model such as the existence of project types and the limit of 1 project of each type in the system are features that provide structure for our model which is then exploited to find a solution efficiently. The model is however still useful even in situations that do not strictly fit these technical assumptions, as we describe below. The model could be extended in a straightforward manner allowing for more than 1 project of each type, at an expense of a more complicated notation (required additional indexing by project number) and dynamics, which would potentially lead to an efficient algorithm to solve it, but this modification was omitted in this paper as we believe the current model captures the key problem features.

Our model (and algorithm) can be directly used to allow for any fixed number of projects instead of just one project of each type. For instance, if a company considers 2 real project types and is willing to accept in parallel up to three projects of the first real project type and up to five projects of the second real project type, then our model with 8 model project types (limiting each model project type to one project) can be employed, by defining each of the first three model project types identically to the first real project type and each of the remaining five model project types identically to the second real project type. So, our model will treat projects of the same type as different types. The arrival probabilities of each model project type can be approximately obtained by dividing the arrival probability of the real project type by the number of projects acceptable in parallel.

Actually, one of our examples uses this modelling trick. In our "Six project types, five tasks and two resource types" problem, all projects are copies of each other so there is actually only one type of project but its capacity in the system becomes six. Although in that problem we assigned different rewards, duration and tardiness costs, these project variables could have been identical. So, multiple projects of the same type can be accommodated in our model.

The number of projects of each type in the system is naturally limited by the amount of resources and the ratio between the due date and the minimum project duration. Suppose that a company is managing one resource and the due dates are tight enough not allowing to complete two projects one after the other by the earlier due date (i.e., the above ratio is strictly lower than 2). Then it would unlikely be a good idea to have three or more projects of the same type in the system at any given time, because every project beyond the first two (in the order of completion) would surely incur the tardiness cost. Therefore, perhaps except in some rare situations in which processing a project incurring the tardiness cost is still better than processing other projects, it would be wise for the company to limit the number of projects of each type to two, which would then allow them to use our model to optimise the scheduling of the projects to achieve the highest possible profit.

Another limitation arising in practice, in some sectors such as software development, production, construction and R&D, is where some types of resources are shared among all types of projects but some resources are specific to one project type. For example, imagine a

Table A.23
Ten project types, ten tasks and two resource types problem.

Project attributes										
Number of project types	10									
Project type no	1	2	3	4	5	6	7	8	9	10
Completion reward	36	37	93	69	38	62	12	80	52	62
Due date	429	354	654	771	447	288	448	1124	288	708
Tardiness cost	29	14	70	24	18	21	2	50	40	29
Order strength	0.36	0.58	0.89	1.00	0.51	0.53	0.51	0.53	0.69	0.67
Resource attributes										
Resource factor	0.70									
Number of resource types	2									
Resource type no	1	2								
Resource amounts	11	11								
Resource strength	0.01	0.01								

Table A.24
Two project types, thirty tasks and four resource types problem.

Project attributes				
Number of project types	2			
Project type no	1	2		
Completion reward	40	19		
Due date	137	74		
Tardiness cost	20	9		
Order strength	0.33	0.57		
Resource attributes				
Resource factor	0.50			
Number of resource types	4			
Resource type no	1	2	3	4
Resource amounts	21	11	10	10
Resource strength	0.20	0.01	0	0

Table A.25
Five project types, thirty tasks and four resource types problem.

Project attributes					
Number of project types	5				
Project type no	1	2	3	4	5
Completion reward	48	2	22	32	56
Due date	62	82	72	89	72
Tardiness cost	24	1	11	16	28
Order strength	0.45	0.44	0.39	0.50	0.44
Resource attributes					
Resource factor	0.25				
Number of resource types	4				
Resource type no	1	2	3.00	4.00	
Resource amounts	58	78	102	19	
Resource strength	0.45	0.65	0.69	0.13	

software development company that has two types of projects: mobile app development and web app development. Each type of project requires a different set of programming skills and testing environments. If these special resources are limited to only one, only one project from each type can be processed in the system at once, while different project types can be processed in parallel. In these environments, the system cannot process multiple projects of the same type at once and therefore it makes sense not to accept more than one project of each type in the system.

Classifying projects into types may often be possible in practice, even for companies that deal with bespoke projects. Our model can be employed at a chosen level of detail, but there is a trade-off. A higher level of detail would possibly lead to more different types of projects, i.e., to a larger problem which may be harder to solve although the arrival probabilities would be smaller (or zero in the most extreme case of the highest level with all projects bespoke). On the other hand, a lower level of detail would allow for grouping projects according to their key characteristics (such as the number of tasks, durations of tasks and required resources) into project types while potentially heterogeneous project rewards and tardiness costs can be replaced by their corresponding average values.

Appendix C. The effect of arrival probabilities

A common feature of the algorithms' performance in Section 5 is that accrued profit is relatively consistent for scenarios with higher arrival probabilities. This is particularly true in the larger instances. Fig. C.2 highlights this phenomenon for four of the project scenarios by considering relative measures of profit generation and resource consumption for ADP Model 3. \$/Max is the expected total discounted long-run profit achieved for the given arrival probability expressed as a percentage of the maximal profit achieved across all tested arrival probabilities. Rk_usage, $k = 1, \dots, K$ is the average usage of type k resource expressed as a percentage of total type k resource available. Profits increase in line with the increase in arrival probability, but the increase typically arrests as the resource usage plateaus. This indicates there is little additional profit gain as resources approach their consumption limits or capabilities for processing tasks in parallel.

We argue that there is value in investigating the algorithmic performance across the test range of arrival probabilities used. In order to achieve these profits, policies deployed by the algorithms should respond to the differing arrival rates. For example, take a problem with two identical project types that differ only by one project's net profits (on-time and tardy) exceeding the other project's. Under low arrival probabilities, the system will experience periods where no, or only one project, is available for processing. A sensible use of resources will be to progress both projects towards on-time completions. At high arrival probabilities, the availability of both project types for processing is more common. A sensible policy will focus on processing the higher return project and deploys resource to the other project when it is viable to do, for example, in parallel or if it is the only project in the system at that time.

All policies we consider do this to some extent, as seen in Section 5, with strongly performing policies better able to take advantage. DP will do this optimally, at the expense of evaluating the policy. ADP can learn to do this through appropriate training, by taking into account the downstream profit implications of their actions. Reactive baseline algorithms, by assuming no future arrivals, have a limited view of the future profit implications of their actions. But good policies in this class would aim to maximise the net profit from clearing the existing projects from the system and, hence, would prioritise the higher value project. Rule-based algorithm performance depends on how the rule is applied. If the two projects were in an identical state and there was sufficient resource to process a single task from one of these projects, then the longest task rule would randomly decide on which project to action. This could unnecessarily delay one or both project completions resulting in lower net profits.

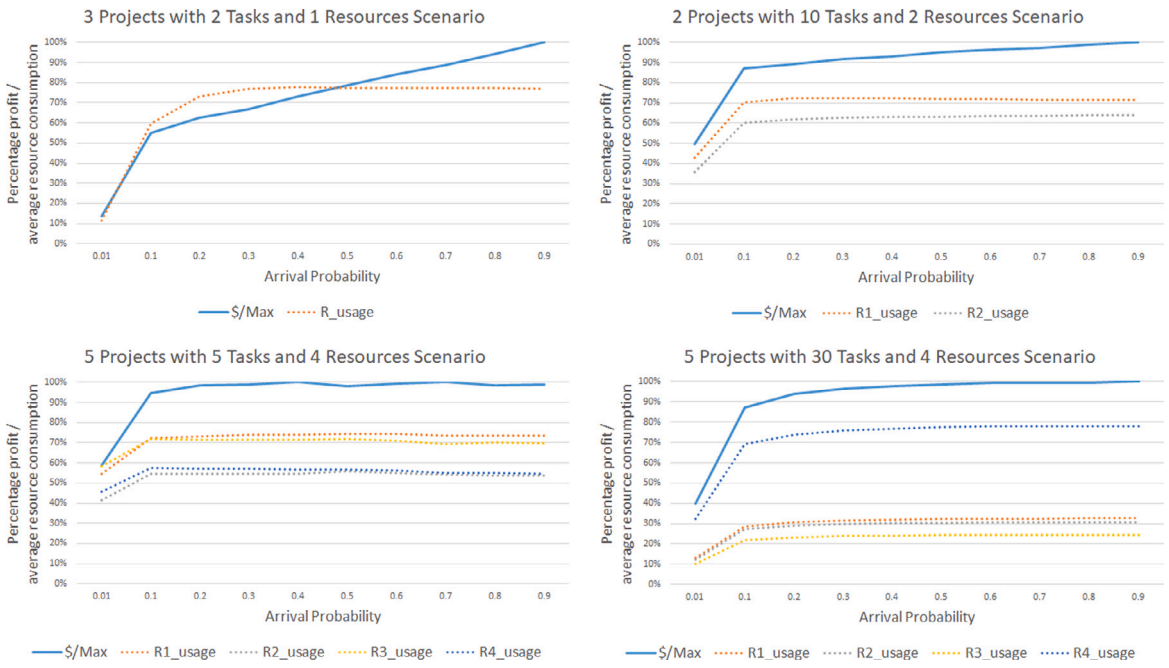

Fig. C.2. Relative profit performance and resource consumption for ADP Model 3 in four problem scenarios.

Table C.26
Performance analysis of DP policies design for problem given arrival probabilities applied in problems for different arrival probabilities.

λ_j	DP	1%	10%	20%	30%	40%	50%	60%	70%	80%	90%
1%	199.3	199.3	199.3	199.4	199.2	198.9	198.3	198.0	198.0	197.8	197.6
10%	878.3	877.4	878.3	875.8	862.4	844.2	815.6	803.9	801.6	796.0	785.3
20%	1043.9	1036.1	1039.6	1043.9	1033.7	1016.0	986.0	973.0	968.4	963.3	953.1
30%	1122.0	1085.0	1093.0	1110.7	1122.0	1117.8	1099.1	1090.3	1086.2	1081.7	1074.1
40%	1196.6	1114.1	1124.3	1155.4	1189.2	1196.6	1189.7	1186.4	1182.2	1178.6	1170.9
50%	1263.5	1141.4	1153.5	1195.6	1245.9	1260.8	1263.5	1263.1	1259.2	1255.7	1250.7
60%	1324.7	1171.5	1184.1	1234.7	1294.5	1314.5	1320.8	1324.7	1323.4	1321.9	1316.0
70%	1377.6	1205.3	1217.4	1272.3	1334.5	1360.4	1370.3	1375.5	1377.6	1376.2	1373.2
80%	1426.8	1247.8	1254.0	1309.8	1369.4	1401.2	1413.5	1419.4	1425.0	1426.8	1422.1
90%	1467.9	1302.5	1295.0	1347.7	1396.6	1440.4	1451.9	1455.9	1464.5	1468.3	1467.9

We expand the discussion by considering the robustness of policies to deviations in the arrival probabilities that they were designed for. We focus on optimal DP policies for the three projects, two tasks and one resource problem. In Table C.26 we present the expected total discounted long-run profit of the optimal policy for each $\lambda_j, j \in \mathcal{J}$, applied to all arrival probability scenarios. In each instance, we ran 10,000 simulations. The first two columns show the optimal profit (DP) for the problem scenario with arrival rate λ_j. The remaining columns show the profit performance of the DP policy optimised for the arrival probability in the column header applied to the problem with arrival probability given in the λ_j-row. Naturally, the diagonal of these columns is exactly the optimal profit in column DP. Away from the diagonal, we see the value in a policy being able to adapt to different conditions. As we move along a row, away from the optimal profit, the performance of policies designed for different arrival rates deteriorates. For low and high arrival probability there can be a close level of performance for policies designed for the adjacent arrival probabilities.

In summary, our investigations highlight the effect of project arrival probabilities on profitable project selection. Comparing policy performance across a broad range of arrival probabilities is useful to establish the overall effectiveness of the algorithms considered.

References

Adler, P. S., Mandelbaum, A., Nguyen, V., & Schwerer, E. (1995). From project to process management: An empirically-based framework for analyzing product development time. *Management Science*, *41*(3), 458–484. http://dx.doi.org/10.1287/mnsc.41.3.458.

Ahuja, V., & Birge, J. R. (2020). An approximation approach for response adaptive clinical trial design. *INFORMS Journal on Computing*, *32*(4), 877–894. http://dx.doi.org/10.1287/ijoc.2020.0969.

Capa, C., & Ulusoy, G. (2015). Proactive project scheduling in an R&D department a bi-objective genetic algorithm. In *2015 International conference on industrial engineering and operations management (IEOM), Vol. 1* (pp. 1–6). http://dx.doi.org/10.1109/IEOM.2015.7093733.

Chen, H., Ding, G., Zhang, J., & Qin, S. (2019). Research on priority rules for the stochastic resource constrained multi-project scheduling problem with new project arrival. *Computers & Industrial Engineering*, *137*, Article 106060. http://dx.doi.org/10.1016/j.cie.2019.106060.

Choi, J., Realff, M. J., & Lee, J. H. (2007). A Q-learning-based method applied to stochastic resource constrained project scheduling with new project arrivals. *International Journal of Robust and Nonlinear Control*, *17*(13), 1214–1231. http://dx.doi.org/10.1002/rnc.1164.

Cohen, I., Golany, B., & Shtub, A. (2005). Managing stochastic, finite capacity, multi-project systems through the cross-entropy methodology. *Annals of Operations Research*, *134*(1), 183–199. http://dx.doi.org/10.1007/s10479-005-5730-1.

Creemers, S. (2015). Minimizing the expected makespan of a project with stochastic activity durations under resource constraints. *Journal of Scheduling*, *18*(3), 263–273. http://dx.doi.org/10.1007/s1095.

Davis, M. T., Robbins, M. J., & Lunday, B. J. (2017). Approximate dynamic programming for missile defense interceptor fire control. *European Journal of Operational Research*, *259*(3), 873–886. http://dx.doi.org/10.1016/j.ejor.2016.11.023.

Fliedner, T., Gutjahr, W., Kolisch, R., & Melchiors, P. (2012). Solving the dynamic stochastic resource-constrained multi-project scheduling problem with SRCPSP-methods. In *Proceedings of the 13th international conference on project management and scheduling, Leuven, Belgium: KU Leuven* (pp. 148–151).

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. New York, NY: W. H. Freeman & Co..

He, F., Yang, J., & Li, M. (2018). Vehicle scheduling under stochastic trip times: An approximate dynamic programming approach. *Transportation Research Part C (Emerging Technologies)*, *96*, 144–159. http://dx.doi.org/10.1016/j.trc.2018.09.010.

Homberger, J. (2007). A multi-agent system for the decentralized resource-constrained multi-project scheduling problem. *International Transactions in Operational Research*, *14*(6), 565–589. http://dx.doi.org/10.1111/j.1475-3995.2007.00614.x.

Kolisch, R., & Sprecher, A. (1996). PSPLIB: A project scheduling problem library. *European Journal of Operational Research*, *96*(1), 205–216. http://dx.doi.org/10.1016/S0377-2217(96)00170-1.

Kolisch, R., Sprecher, A., & Drexl, A. (1995). Characterization and generation of a general class of resource-constrained project scheduling problems. *Management Science*, *41*(10), 1693–1703. http://dx.doi.org/10.1287/mnsc.41.10.1693.

Li, H., & 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. http://dx.doi.org/10.1016/j.ejor.2015.04.015.

Li, H., Zhang, X., Sun, J., & Dong, X. (2023). Dynamic resource levelling in projects under uncertainty. *International Journal of Production Research*, *61*(1), 198–218. http://dx.doi.org/10.1080/00207543.2020.1788737.

Melchiors, P. (2015). *Lecture notes in economics and mathematical systems*, *Dynamic and stochastic multi-project planning*. Cham, Switzerland: Springer, http://dx.doi.org/10.1007/978-3-319-04540-5.

Melchiors, P., & Kolisch, R. (2009). Scheduling of multiple R&D projects in a dynamic and stochastic environment. In *Operations research proceedings 2008* (pp. 135–140). Heidelberg: Springer, http://dx.doi.org/10.1007/978-3-642-00142-0_22.

Melchiors, P., Leus, R., Creemers, S., & Kolisch, R. (2018). Dynamic order acceptance and capacity planning in a stochastic multi-project environment with a bottleneck resource. *International Journal of Production Research*, *56*(1–2), 459–475. http://dx.doi.org/10.1080/00207543.2018.1431417.

Pamay, M. B., Bülbül, K., & Ulusoy, G. (2014). Dynamic resource constrained multi-project scheduling problem with weighted earliness/tardiness costs. In P. S. Pulat, S. C. Sarin, & R. Uzsoy (Eds.), *International series in operations research & management science*: *vol. 200*, *Essays in production, project planning and scheduling* (pp. 219–247). Springer US, http://dx.doi.org/10.1007/978-1-4614-9056-2_10.

Parizi, M. S., Gocgun, Y., & Ghate, A. (2017). Approximate policy iteration for dynamic resource-constrained project scheduling. *Operations Research Letters*, *45*(5), 442–447. http://dx.doi.org/10.1016/j.orl.2017.06.002.

Powell, W. B. (2009). What you should know about approximate dynamic programming. *Naval Research Logistics*, *56*(3), 239–249. http://dx.doi.org/10.1002/nav.20347.

Powell, W. B. (2011). *Approximate dynamic programming: Solving the curses of dimensionality*. Hoboken, NJ: John Wiley, http://dx.doi.org/10.1002/9781118029176.

Ronconi, D. P., & Powell, W. B. (2010). Minimizing total tardiness in a stochastic single machine scheduling problem using approximate dynamic programming. *Journal of Scheduling*, *13*, 597–607. http://dx.doi.org/10.1007/s10951-009-0160-6.

Satic, U., Jacko, P., & Kirkbride, C. (2020). Performance evaluation of scheduling policies for the DRCMPSP. In M. Gribaudo, E. Sopin, & I. Kochetkova (Eds.), *Analytical and stochastic modelling techniques and applications, Vol. 12023* (pp. 100–114). Cham: Springer International Publishing, http://dx.doi.org/10.1007/978-3-030-62885-7_8.

Satic, U., Jacko, P., & Kirkbride, C. (2022). Performance evaluation of scheduling policies for the dynamic and stochastic resource-constrained multi-project scheduling problem. *International Journal of Production Research*, *60*(4), 1411–1423. http://dx.doi.org/10.1080/00207543.2020.1857450.

Schütz, H.-J., & Kolisch, R. (2012). Approximate dynamic programming for capacity allocation in the service industry. *European Journal of Operational Research*, *218*(1), 239–250. http://dx.doi.org/10.1016/j.ejor.2011.09.007.

Schwindt, C. (1998). *Generation of resource constrained project scheduling problems subject to temporal constraints*: *Report WIOR-543*, Kaiserstrasse 12, D-76128 Karlsruhe, Germany: Universitat Karlsruhe.

Sutton, R. S., & Barto, A. G. (2018). *Adaptive computation and machine learning, Reinforcement learning: An introduction* (second ed.). Cambridge, MA, USA: MIT Press.

Wellingtone PPM (2018). The state of project management annual survey 2018. http://www.wellingtone.co.uk/wp-content/uploads/2018/05/The-State-of-Project-Management-Survey-2018-FINAL.pdf, Accessed October 31, 2023.