ARTICLE TEMPLATE

Performance evaluation of scheduling policies for the Dynamic and Stochastic Resource-Constrained Multi-Project Scheduling Problem

Ugur Satic, Peter Jacko and Christopher Kirkbride

Lancaster University, Lancaster LA1 4YW, UK

ARTICLE HISTORY
Compiled November 26, 2020

ABSTRACT
In this study, we consider the dynamic and stochastic resource-constrained multi-project scheduling problem where projects generate rewards at their completion, completions later than a due date cause tardiness costs, task duration is uncertain, and new projects arrive randomly during the ongoing project execution both of which disturb the existing project scheduling plan. We model this problem as a discrete-time Markov decision process and explore the performance and computational limitations of solving the problem by dynamic programming. We run and compare five different solution approaches, which are: a dynamic programming algorithm to determine a policy that maximises the time-average profit, a genetic algorithm and an optimal reactive baseline algorithm, both generate a schedule to maximise the total profit of ongoing projects, a rule-based algorithm which prioritises processing of tasks with the highest processing durations, and a worst decision algorithm to seek a non-idling policy that minimises the time-average profit. The performance of the optimal reactive baseline algorithm is the closest to the optimal policies of the dynamic programming algorithm, but its results are suboptimal, up to 37.6%. Alternative scheduling algorithms are close to optimal with low project arrival probability but quickly deteriorate their performance as the probability increases.

KEYWORDS
dynamic; stochastic; resource constrained project scheduling problem; dynamic programming; reactive scheduling; genetic algorithm; scheduling policies; DSRCMPSP

1. Introduction

Many factors may bring uncertainty to the project execution plan, such as new projects arriving after the plan has begun requiring a re-evaluation of the execution order. Project management is a very challenging enterprise in that only 40% of projects are completed within their planned time, 46% of projects are completed within their predicted budget and only 36% of projects realise their full benefit APM (2018). Many sectors suffer from the uncertainty ignored by traditional project management such as engineering services, software development, IT services, construction and R&D. In this paper, we propose a comprehensive model to project scheduling under uncertainty for the dynamic and stochastic resource-constrained multi-project scheduling problem

CONTACT Ugur Satic. Email: u.satic@lancaster.ac.uk

(RCMPSP) which includes random project arrival and stochastic task duration uncertainties.

We model the problem as an infinite-horizon discrete-time *Markov decision process* (MDP) with the objective to maximise the expected time-average profit. We extend the research of Satic, Jacko, and Kirkbride (2019) by considering stochastic task durations and use their problems in our comparisons by adding early, normal and late task completion probabilities.

The resource-constrained project scheduling problem (RCPSP) and its multi-project equivalent RCMPSP generally consider a static environment. The RCMPSP is a generalisation of the RCPSP, which is an NP-hard optimisation problem; thus, RCMPSP and the other generalisation of RCPSPs are also categorised as NP-hard. (Gonçalves, Mendes, and Resende 2008). In the literature, the generalisations of RCPSP and RMCPSP with uncertainty in task durations are called the *stochastic* RCPSP (SRCPSP) and the *stochastic* RCMPSP (SRCMPSP) respectively. The common goal of deterministic problems is minimising the total completion time (Browning and Yassine 2010), while the common goal of stochastic problems is to minimise the *expected* completion time (Rostami, Creemers, and Leus 2018). These static environment problems are extensively studied in the literature. The literature review of Ortiz-Pimiento and Diaz-Serna (2018) shows that Meta-heuristic methods such as GA, particle swarm optimisation, Tabu search, Bee Colony, Ant Colony, Greedy Algorithms, Simulated annealing and Distribution Estimation Algorithm; Exact Methods such as Branch and Bound, Dynamic Programming and Stochastic programming; Special Procedures such as Priority Rules-based, Simulation Process-based or Stage-by-Stage Analysis based; Critical Chain Methods are some of the applied solution methods in the literature. The Critical Chain Method is used for stochastic RCPSP with buffer sizing method as in Zarghami et al. (2019). Also, an approximate dynamic programming method is used for stochastic RCPSP by Li et al. (2020). The literature review of (Karam and Lazarova-Molnar 2013) on recent approaches shows that the majority of the new approaches are hybrid forms of previously mentioned methods.

All the models we have described until this point were static, where the data of project arrival times and their type are known before the scheduling begins. Despite the vast number of studies in the static field, many companies accept new projects during the processing of ongoing projects (Herbots, Herroelen, and Leus 2007). That deviates from the project plan and leads to missed due dates and associated tardiness costs (Capa and Kilic 2015). So, instead of focusing only on completion times, projects are more generally modelled with completion *due dates*, completion *rewards* released as the outcome of project completion, and penalties or loss of prestige and goodwill which are collectively called the *tardiness costs* incurred if projects are completed after their due dates. The aim then becomes to find optimal schedules or scheduling policies that maximise some function of *profit*, which is the difference between the completion rewards and tardiness costs. In general, there are three standard objective functions for non-static problems: (i) the expected total profit over a finite horizon (to the best of our knowledge, this has not been used in projects scheduling literature for non-static problems), (ii) the expected total discounted profit over a finite or infinite horizon (e.g., Parizi, Gocgun, and Ghate 2017), or (iii) the expected time-average profit over an infinite horizon (e.g., Wang et al. 2015). The RCMPSP with uncertain project arrivals and deterministic task durations is called *dynamic* RCMPSP (DRCMPSP). Two main approaches are available for the DRCMPSP; (1) reactive baseline scheduling (e.g. Pamay, Bülbül, and Ulusoy (2014)), an approach which generates a baseline schedule and updates it at each project arrival which allows usage of the static RCMPSP

methods such as the GA for the DRCMPSP and (2) computation of optimal policies using approximate dynamic programming (ADP) (e.g. Parizi, Gocgun, and Ghate (2017)). Satic, Jacko, and Kirkbride (2019) applied both methods to DRCMPSP and evaluated their performances.

The RCMPSP with both random project arrivals and uncertain task durations is called the *dynamic and stochastic* RCMPSP. Only a limited number of research considered both the dynamic project arrivals and stochastic task durations together. The main approaches for this problem are processing networks (e.g. Adler et al. (1995); Cohen, Golany, and Shtub (2005)), computation of optimal policies using approximate dynamic programming (ADP) (e.g. Melchiors (2015); Choi, Realff, and Lee (2007)) and reactive baseline scheduling (e.g. Fliedner et al. (2012); Capa and Kilic (2015)).

We contribute to the literature by (i) developing a dynamic and stochastic RCMPSP model considering multi-task project types, extending the work of Melchiors et al. (2018) who only considered single-task projects, (ii) developing an efficient implementation of the value iteration algorithm in Julia programming language to solve our model with up to 4 project types, (iii) comparing the (exactly) optimal policy of value iteration with the policies of GA, rule based algorithm and optimal reactive baseline algorithm to evaluate the performance gap between solution approaches, and (iv) illustrating that even in simple problems with 2 or 3 project types, the suboptimality gap of benchmark policies commonly used in practice (genetic algorithm and longest-task-first rule) which ignore possibility of new project arrivals is remarkable.

This paper is organized as follows: In section 2, we describe the problem setting, the MDP model. In section 3, we describe the compared algorithms and discuss comparison results in subsection 4.2. In section 5, a conclusion is presented.

2. Methodology

2.1. *The problem setting*

The dynamic and stochastic RCMPSP comprises J project types, and the system capacity for each project type is limited to one. All projects of type j share the same characteristics such as arrival probability (λ_j), number of tasks (I_j), project network, resource usages ($b_{j,i}$), task duration distribution ($\gamma_{j,i}$), minimal possible completion time ($t_{j,i}^{min}$), normal possible completion time ($t_{j,i}$), maximal possible completion time ($t_{j,i}^{max}$), project due date (F_j), reward (r_j) and tardiness cost (w_j).

A project may arrive to the system at any point during the time unit, which is the duration between two decision epochs. Only one project for each type may arrive per unit time with probability λ_j for a project of type j. Projects are stored in the system until the end of unit time. Then in the next decision epoch, if the system capacity for newly arrived project type is not full, it will get accepted to the system. Otherwise, it will get rejected.

A type j project consists of I_j tasks. In this problem, tasks are connected sequentially with a successor-predecessor relationship, which defines the project network. Processing task i of project type j requires completion of its predecessor tasks ($\mathcal{M}_{j,i}$) which have an earlier place in the project network. An example project network is shown in Figure 1.

Processing task i from project type j also requires allocation of $b_{j,i}$ amount of resources during its processing. Only one type of resource is defined in this problem and the amount available is represented by B. The total number of allocated resources

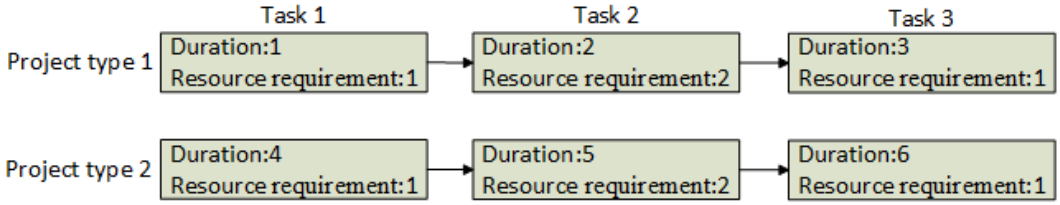

Figure 1. A project network

cannot be higher than B. The resources are assumed renewable which means they become reusable after completion of a task to which they were assigned. The number of resources which are not allocated for task processing is called free-resources (B_s^{free}). After the completion of a task, its allocated resources return to the free resources.

Task processing is considered as a stochastic process in which tasks can be completed early, normal or late according to its completion distribution ($\gamma_{j,i}$). The distribution $\gamma_{j,i}$ is assumed to be discrete, with the longest (respectively, shortest) processing time with a non-zero probability of completion denoted by $t_{j,i}^{max}$ (respectively, $t_{j,i}^{min}$).

Task processing is also assumed to be non-preemptive; thus, it cannot be paused or cancelled, i.e., once a task has begun processing, it does not leave processing until completed.

Projects are completed when all of their tasks are processed, and a project reward r_j is earned. Projects have a due date (F_j) which represents the maximum unit of time which can be spent for project completion to obtain its full reward r_j. If the due date is exceeded, the tardiness cost w_j is applied only once, after the project is completed.

2.2. *Modelling Framework*

We model the problem as an infinite horizon *Discrete Time Markov Decision Process* (DT-MDP) which is defined by five elements: time horizon, pre-decision state space, action set, transition function and profit function. Figure 2 illustrates this process.

In a DT-MDP, a decision epoch is the time where a decision is taken for a pre-decision state. Decision epochs occur as fixed intervals and the period between two consecutive decision epochs is the time unit. During a time unit, projects are processed according to the decisions made at the previous decision epoch, and the new events occur such as project arrivals and tasks completions may occur. Then the system enters a new decision epoch.

The pre-decision state (s) represents the system information relevant to the decision-making process at each decision epoch. In this research, the pre-decision states consists of the information regarding the remaining task processing times ($x_{j,i}$) to the late completions and the remaining due dates (d_j) for all projects. More details about the pre-decision state are provided in subsection 2.3.

The decisions available in a given pre-decision state s is called action (a). At a decision epoch, the decision maker selects an action a which starts the processing of the selected pending tasks. All actions must fulfil these two conditions; the available free-resources can not be exceeded, and predecessor tasks should be completed. The action is described in detail in subsection 2.4.

After the selected action a is applied in the pre-decision state s, system information is represented by the post-decision state $\hat{s} := (s, a)$. The post-decision state is a virtual state before the stochastic processes begin, and it is assumed there is not any time lag

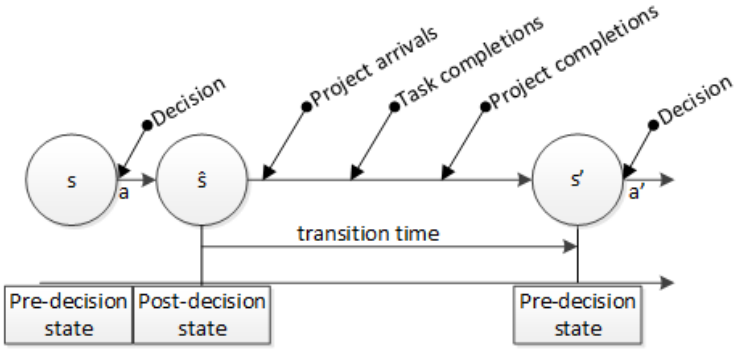

Figure 2. Discrete-time Markov Decision Process

between pre-decision state and post-decision state. The post-decision state is explained in subsection 2.5.

The transition function describes how the system evolves from one state to another as a result of decisions and information (Powell 2011). The transition function is illustrated in Figure 3 and described in detail in subsection 2.6. During the transition period; the ongoing tasks are processed for one time unit, some tasks are completed according to their completion distribution $\gamma_{j,i}$ and new projects may arrive according to arrival probabilities λ_j. Conversely, the probability of no arrival from a project of type j is $1 - \lambda_j$. If a project arrival occurs it will be accepted into the system if there is no project of the same type in the system or, if there is, that active project will complete processing in this time period. Otherwise the new project arrival will be rejected. Hence, between 0 and J project arrivals may occur in a single time unit. We assume that the arrival of projects are independent and identically distributed random variables. The above arrival process is a Bernoulli arrival process, with geometrically distributed inter-arrival times T_j with mean λ_j^{-1} and probability mass function $P(T_j = k) = (1 - \lambda_j)^{k-1}\lambda_j; k = 1, 2, \ldots$.

In this discrete-time process, due to the memoryless property of the geometric distribution, the probability of an arrival of a type j project within the current time period $P(T = 1) = \lambda_j$ regardless of the number of time periods since the last arrival. This is the discrete-time analogue of exponentially-distributed inter-arrival times in continuous time models.

Hence, a stochastic arrival process is employed to model the arrival of new projects. However, for the tractability of solutions via dynamic programming we utilise a finite buffer such that a maximum of one project of each type may be present in the system at any point in time.

The profit function of a post-decision state calculates the reward of the completed project minus any tardiness cost paid to its latest possible completion at the end of the transition. The profit calculation is expressed in subsection 2.7.

2.3. *Pre-Decision State*

The pre-decision state (s) is the system information available at a decision epoch. Pre-decision states where the resource limitations are not exceeded and predecessor tasks were completed before their successor tasks are called feasible and the set of all feasible pre-decision states is called as the state space $\mathcal{S}$. Elements of a pre-decision

Table 1. State Matrix

	Tasks			Remaining due date
Project type 1 :	$x_{1,1}$	$x_{1,2}$	$x_{1,3}$	d_1
Project type 2 :	$x_{2,1}$	$x_{2,2}$	$x_{2,3}$	d_2

state are project states ($\boldsymbol{P}_j$) for all project types:

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

A project state consists of task states ($x_{j,i}$) and the remaining due date state (d_j):

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

A task state represents the status of a task. If a task is pending for processing, its value is taken as -1. If a task is finished, its value is represented by 0. If a task is in processing, its value is the remaining processing time from its late completion duration ($t_{j,i}^{max}$). In a pre-decision state, $x_{j,i} = t_{j,i}^{max} - 1$ represents the task processing began at the previous decision epoch.

$$x_{j,i} \in \{-1, 0, 1, 2, \ldots, t_{j,i}^{max} - 1\} \tag{3}$$

The remaining due date state d_j represents the number of remaining time units from the current time epoch to complete the project j without paying any tardiness cost. When a due date is exceeded, its value becomes 0 and it expresses that the tardiness cost will be incurred at the project's completion. A newly accepted project has the highest remaining due date state value which is its due date F_j.

$$d_j \in \{0, 1, 2, 3, \ldots, F_j\} \tag{4}$$

When a type j project is completed or there is no type j project in the system, all task states ($x_{j,i} = 0, \forall i$) and remaining due date state ($d_j = 0$) of project type j are represented by 0:

$$\boldsymbol{P}_j = (0, 0, \ldots, 0, 0) \tag{5}$$

When a new type j project arrives, all its task states are set to -1 ($x_{j,i} = -1, \forall i$) and its remaining due date state is represented by its due date F_j ($d_j = F_j$):

$$\boldsymbol{P}_j = (-1, -1, \ldots, -1, F_j) \tag{6}$$

An example state matrix with two project types and three tasks is shown in Table 1. Here, rows of the matrix represent each project type j. The columns represent the task numbers but the last column of the matrix represents the due date state (d_j).

A pre-decision state determines its free resources (B_s^{free}) which is used as a constraint for the available decisions. Free resources are the remaining resources available after the resource allocation to ongoing tasks has been accounted for:

$$B_s^{\text{free}} = B - \sum_{j=1}^{J} \sum_{i=1}^{I_j} b_{i,j} \mathcal{I}\{x_{i,j} > 0\} \tag{7}$$

Table 2. Action Matrix

	Actions		
Project type 1 :	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
Project type 2 :	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$

Here, B is the total amount of resource, $b_{i,j}$ is the resource amount allocated for processing of task i from type j project, $\mathcal{I}\{.\}$ is an indicator function that takes the value 1 if the condition in parentheses is true and takes the value 0 otherwise.

2.4. *Action Representation*

An action a is a function of a pre-decision state s and it holds the processing decisions of pending tasks. An example action matrix with two project types and three tasks is shown in Table 2. If the decision includes processing a pending task i of a type j project ($x_{i,j} = -1$), the corresponding action element $a_{j,i}$ will take the value of 1 in the action matrix. Otherwise, $a_{j,i}$ will be 0. The task processing decision can be only taken if there are enough free resources to allocate ($\sum_{j=1}^{J}\sum_{i=1}^{I_j} b_{i,j}\mathcal{I}\{a_{i,j}=1\} \leq B_s^{\text{free}}$) and any predecessor tasks ($\mathcal{M}_{j,i}$) of task i are completed ($\sum_{m\in\mathcal{M}_{j,i}} x_{j,m} = 0$). Thus, an action must satisfy both of these conditions.

All the actions which meet both the resource and predecessor limitations, are called feasible and set of all feasible actions for a pre-decision state s creates the action set $\boldsymbol{A}(s)$:

$$\boldsymbol{A}(s) = \{\mathbf{0}, \boldsymbol{a}^{'}, \boldsymbol{a}^{''}, \dots\} \tag{8}$$

The action, where all action elements are zero is called "do not initiate any task" ($\mathbf{0} = (0, 0, \dots, 0)$) and it is always a member of the action set $\mathbf{0} \in \boldsymbol{A}(s)$. $\boldsymbol{a}^{'}$ and $\boldsymbol{a}^{''}$ represent alternative feasible actions. The number of alternative actions in an action set depends on the number of free resources (B_s^{free}), the unprocessed tasks (with $x_{i,j} = -1$) and the tasks with completed predecessor tasks (with $\sum_{m\in\mathcal{M}_{j,i}} x_{j,m} = 0$).

2.5. *Post-decision state*

In our model, the post-decision state ($\hat{s}$) is used to represent the task state after a decision is implemented but before any transition time passed. The transition from a pre-decision state to post-decision is a deterministic process. The post-decision state is used in this study to reduce computational effort and to store the stochastic transition outcomes of a pre-decision state and action pair with their occurrence probabilities.

The same due date state is used for a pre-decision state and its following post-decision state. Since there is not any time lag between the post-decision state and the previous pre-decision state, the remaining duration of the processing tasks and the state of completed tasks remains the same while the state of the pending tasks selected by the processing decision changes from -1 to $t_{j,i}^{max}$. This represents that these tasks begin processing. Thus the maximum remaining duration of a task in a post-decision state is $t_{j,i}^{max}$ while it is $t_{j,i}^{max} - 1$ in a pre-decision state.

$$\hat{x}_{j,i} \in \{-1, 0, 1, 2, \dots, t_{j,i}^{max}\} \tag{9}$$

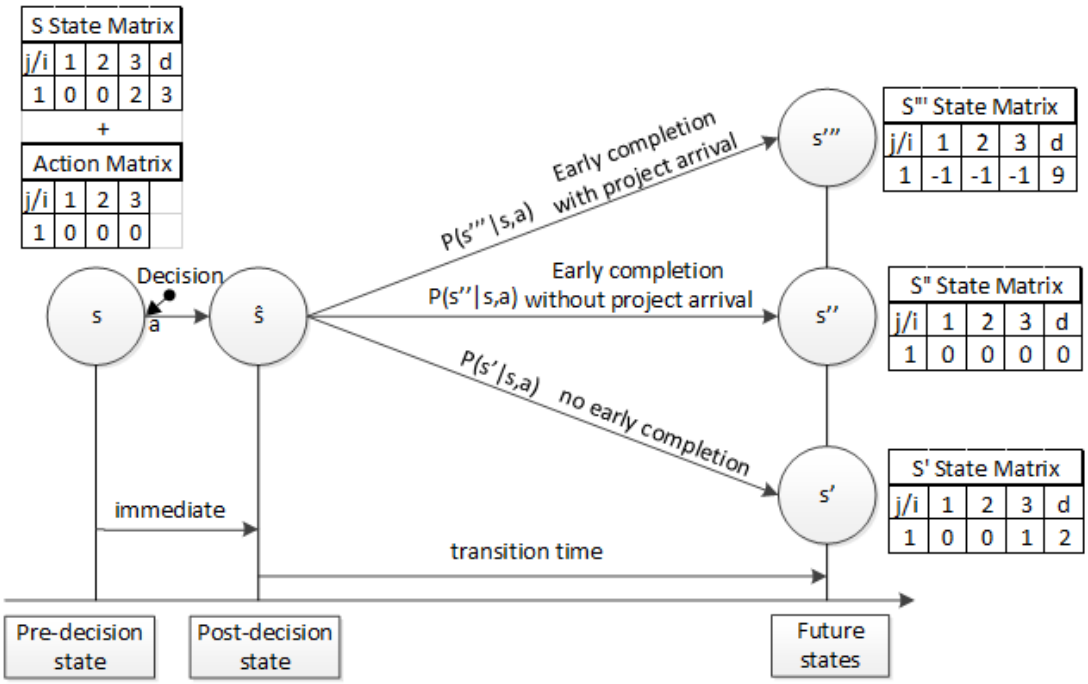

Figure 3. A state transition diagram (for a $j = 1$ type project with 3 tasks ($i = 1, 2, 3$) whose due date is $F_j = 9$ and the selected action means do not initialise any task.)

2.6. *Transition function*

The transition function determines the following pre-decision state. The task completion probability $\gamma_{j,i}(\hat{x}_{j,i})$ and the project arrival probability λ_j affect the transition probability of the next pre-decision state. In our model a task may complete within $t_{j,i}^{min}$ and $t_{j,i}^{max}$ periods once it has begun processing. With a maximum of $\hat{x}_{j,i}$ periods remaining until task i of project type j's completion, the task will complete with probability $\gamma_{j,i}(\hat{x}_{j,i})$, where $\gamma_{j,i}(\hat{x}_{j,i}) = 0$ for $\hat{x}_{j,i} > 1 + t_{j,i}^{max} - t_{j,i}^{min}$ and $\sum_{x=1}^{1+t_{j,i}^{max}-t_{j,i}^{min}} \gamma_{j,i}(x) = 1$. The project arrival probability is considered when it is possible for task I_j to be completed before the next decision epoch, at which point capacity for a newly arriving project of type j will become available. The system transition probability is given by

$$P(s'|s,a) = \prod_{j=1}^{J} \prod_{i=1}^{I_j} P(x'_{j,i}|\hat{x}_{j,i}) \tag{10}$$

$$P(x'_{j,i}|\hat{x}_{j,i}) = \begin{cases} \lambda_j \gamma_{j,i}(\hat{x}_{j,i}), & \text{for } 1 \leq \hat{x}_{j,i} \leq 1 + t^{max}_{j,i} - t^{min}_{j,i}, x'_{j,i} = -1, i = I_j \\ (1-\lambda_j)\gamma_{j,i}(\hat{x}_{j,i}), & \text{for } 1 \leq \hat{x}_{j,i} \leq 1 + t^{max}_{j,i} - t^{min}_{j,i}, x'_{j,i} = 0, i = I_j \\ \gamma_{j,i}(\hat{x}_{j,i}), & \text{for } 1 \leq \hat{x}_{j,i} \leq 1 + t^{max}_{j,i} - t^{min}_{j,i}, x'_{j,i} = 0, i < I \\ 1 - \gamma_{j,i}(\hat{x}_{j,i}), & \text{for } 1 \leq \hat{x}_{j,i} \leq 1 + t^{max}_{j,i} - t^{min}_{j,i}, x'_{j,i} = \hat{x}_{j,i} - 1 \\ \lambda_j, & \text{for } \hat{x}_{j,i} = 0, x'_{j,i} = -1, i = I_j \\ 1 - \lambda_j, & \text{for } \hat{x}_{j,i} = 0, x'_{j,i} = 0, i = I_j \\ 1, & \text{for } \hat{x}_{j,i} = 0, x'_{j,i} = 0, i < I_j \\ 1, & \text{for } \hat{x}_{j,i} > 1 + t^{max}_{j,i} - t^{min}_{j,i}, x'_{j,i} = \hat{x}_{j,i} - 1 \\ 1, & \text{for } \hat{x}_{j,i} = -1, x'_{j,i} = -1 \end{cases}$$

In Figure 3 an example transition process has been shown. The transition probability of the first alternative future pre-decision state, where the last task of type j project is finished and a new type j project arrived, is $P(s'''|s,a) = \lambda_j \cdot \gamma_{j,i}(\hat{x}_{j,i})$. The transition probability of the second alternative future pre-decision state, where the last task type j project is finished and no project arrived, is $P(s''|s,a) = (1-\lambda_j) \cdot \gamma_{j,i}(\hat{x}_{j,i})$. The transition probability of the third possible alternative future pre-decision state, where the last task type j project is not finished thus a project arrival is not considered, is $P(s'|s,a) = 1 - \gamma_{j,i}(\hat{x}_{j,i})$.

2.7. *Profit Representation*

The profit ($R_{\hat{s}}$) of the post-decision state $\hat{s}$ is the sum of rewards (r_j) of completed projects in the period between current and next decision epoch minus the tardiness cost of late completions which depend on the remaining due dates.

$$R_{\hat{s}} = \sum_{j=1}^{J} r_j \mathbb{E}\Big[\mathcal{I}\big\{\hat{x}_{j,I} > 0 \wedge x'_{j,I} \leq 0\big\}\Big] - \sum_{j=1}^{J} w_j \mathbb{E}\Big[\mathcal{I}\big\{\hat{x}_{j,I} > 0 \wedge x'_{j,I} \leq 0 \wedge d_j = 0\big\}\Big] \quad (11)$$

Here, the first indicator is for project completion and takes the value 1 if a project completes and is 0 otherwise. The second indicator is for late project completion. It takes the value 1 if a project's due date has already passed (i.e., the projects remaining dues date $d_j = 0$) and is 0 otherwise. Recall that, in post-decision state $\hat{s}$, $\hat{x}_{j,I}$ represents the remaining maximum processing time of the final task of a type j project. $x'_{j,I}$ is the remaining maximum processing time of the final task of project j at the future pre-decision state (s').

2.8. *Goal function*

The goal of the dynamic and stochastic RCMPSP is to find the policy π that maximises the long-term average profit per unit time.

$$g^* = \max_{\pi \in \Pi} \lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}^{\pi}[R_{\hat{s}(t)}] \quad (12)$$

Here, $R_{\hat{s}(t)}$ is the profit function dependent of time epoch t. π is a policy from the set of all feasible non-anticipating policies (Π) presenting the action set $\boldsymbol{A}(s)$. A feasible

policy is a sequence of action which considers both the resource limitation and project network.

2.9. *Solution by Dynamic Programming*

Dynamic Programming (DP) is a collection of algorithms which calculates optimal policies from the MDP model of the solution environment (Sutton and Barto 2018). In this research we used Dynamic Programming Value Iteration. Value Iteration calculates a sequence of value functions (Tijms 1994). The value function approximates the cumulative reward minus the tardiness cost. The per-period change in the value function approximates the maximum long-term average profit. The process steps of the algorithm are below;

For each state $\forall s \in \mathcal{S}, V^{old}(s) = 0$
Do
  For each state $\forall s \in \mathcal{S}$
    $V(s) = \max_{a \in A}[R_{\hat{s}} + \sum_{s' \in \mathcal{S}} p(s'|s,a)V^{old}(s')]$
  End For
  $W_{max} = \max_{s \in \mathcal{S}}[V(s) - V^{old}(s)]$
  $W_{min} = \min_{s \in \mathcal{S}}[V(s) - V^{old}(s)]$
  $\Delta = W_{max} - W_{min}$
  Update for $\forall s \in \mathcal{S}, V^{old}(s) = V(s)$
While $\Delta > \beta \times W_{min}$

Here, V represents the value function of a pre-decision state s. $R_{\hat{s}}$ is profit function as explained in subsection 2.7. $p(s'|s,a)$ is the state transition probability. s' stands for the future pre-decision state of s. $V^{old}(s')$ is the value of s' from next decision epoch. β is pre-specified tolerance number (0.000001). W_{min} and W_{max} are respectively minimum and maximum value changes between two iterations. Δ is the difference between the minimum and the maximum value changes. $\mathcal{S}$ is the state space which is defined at subsection 2.3. These processes are repeated until the stopping criteria is met.

3. Results and Comparisons

We used two heuristic algorithms and one exact algorithm with reactive scheduling and one worst decision algorithm to compare their performance to optimal. A reactive scheduling method generates decisions within a deterministic approach without considering the future uncertainties (Pamay, Bülbül, and Ulusoy 2014). Then, it iteratively fixes its first schedule according to random changes and makes the schedules feasible again (Rostami, Creemers, and Leus 2018). We used a genetic algorithm, an optimal reactive baseline algorithm and a priority rule algorithm; note that all three are based on the reactive scheduling method. Both the optimal DP and the worst decision algorithm are proactive scheduling methods.

3.1. *Genetic Algorithm*

The GA is one of the search algorithms which searches for the global optimum on the solution space by improving the search samples at each iteration (Mori and Tseng

1997). The GA uses bio-inspired operators (e.g. Elitist selection, Crossover and Mutation) to develop the population, which is a solution set, in each iteration. The GA is the most-used algorithm for project scheduling problems. However, the algorithm is not suitable for dynamic problems, and a reactive scheduling method is required to apply GA to a dynamic problem. The reactive scheduling method converts each state of the dynamic problem to a static problem, and solution methods generate a baseline schedule for each state. Fliedner et al. (2012) and Capa and Kilic (2015) proposed reactive scheduling methods based on GA for the dynamic and stochastic RCMPSP. Thus we included GA to our compared algorithms.

The goal of the genetic algorithm (GA) in this research is maximising the profit. The algorithm uses the total completion time as tiebreakers between schedules with equal rewards. If the tie continues, the model prioritises processing of lower project type numbers and lower task numbers. We adapted GA from Satıç (2014). For each pre-decision state, random numbers are assigned to unprocessed tasks, and this assignment is stored as an individual of the population. Individuals are created until the population number (here, one hundred) is reached. The random numbers represent task processing priorities and this method called as the random key representation. The random keys are converted to a schedule using the serial scheduling scheme as Kolisch and Hartmann (1999) described. Then the population is ordered according to their total profit and total completion time. So, the first member of the population represents the best schedule found with highest profit and shortest completion time while the last member represents the worst schedule.

The first population is iterated one hundred times using the genetic operators. The best ten percent of the population is transferred to the next population without any change, and the rest of the next population is created with the crossover operator. The crossover operator, firstly, selects two individuals from the previous population, then, copies some random keys from the first individual, after that, copies the rest from another individual, and finally, creates a new individual. The new individual is mutated with a fifty per cent probability before joining to the next population. The mutation operator randomly selects an unprocessed task and re-assigns its random number. When the new population reaches to one hundred individuals, the random keys are converted to schedules with the serial scheduling scheme. Then the population is ordered from the shortest total completion time to longest, and it is ordered again from the maximum total profit to the minimum. After the one-hundredth generation is created; the first schedule in the population (the best schedule) is selected as the baseline schedule. Then the baseline schedule is converted to action.

3.2. *Optimal reactive baseline algorithm*

The optimal reactive baseline algorithm (ORBA) converts each pre-decision state to a static RCMPSP with the reactive scheduling method and generates all possible schedules for the static RCMPSP. The profit and the total makespan of the schedules are calculated using the serial scheduling scheme. The schedule with highest profit is selected as the best schedule and converted to action. In case of more than one schedule with the highest profit, algorithm prioritizes the shortest total makespan between these schedules. If the tie continues, algorithm randomly selects one schedule. We included the ORBA to show the best possible result of the reactive scheduling method.

3.3. *Priority rule (Longest task first)*

An alternative policy is created with a priority based heuristic algorithm. The algorithm uses a single-pass priority rule called the longest task first rule. Single-pass rules generate only one action for the given state. The rule based algorithm (RBA) prioritises the smallest numbered project type, if two tasks have the same duration, e.g., project type 1 is prioritised over type 2 or type 3. For each pre-decision state, the algorithm generates a baseline schedule using the priority rule and the serial scheduling scheme. Then, the baseline schedule is converted to an action. We included the RBA to show the performance of a simple heuristic algorithm.

3.4. *Worst decision algorithm*

A mix of value iteration and priority rule methods are used as the worst decision algorithm (WDP) which seeks a policy (π') to get the minimum profit per unit time (g'). We used this method in our comparison to show the minimum profit of the worst non-idling policy.

$$g' = \min_{\pi' \in \Pi'} \lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}^{\pi'}[R_{\hat{s}(t)}]. \tag{13}$$

Here, π' is a policy from the set of all feasible non-anticipating active policies (Π') which does not include the "do not active any task" ($\mathbf{0}$) actions unless it is the only possible action in the action set ($|\boldsymbol{A}(s)| = 1$). Since the reward and tardiness costs are modelled to be received after project completions, a minimum profit algorithm without the priority rule ($|\boldsymbol{A}(s)| \neq 1 \Rightarrow \mathbf{0} \notin \pi'$) delays project completions infinitely to halt rewards.

4. Computational results

4.1. *Experimental setup*

In this section, we will explore the limits of DP on the dynamic and stochastic RCMPSP, and compare its performance with the two heuristic reactive baseline scheduling algorithms, the optimal reactive baseline algorithm and the worst decision algorithm. The DP and the compared algorithms are coded in JuliaPro 1.0.1.1. All tests are performed on a desktop computer with Intel i5-6500T CPU with 2.50 GHZ clock speed and 32 GB of RAM.

We will use DRCMPSPs with deterministic task durations from Satic, Jacko, and Kirkbride (2019) and generate their dynamic and stochastic RCMPSP equivalents with stochastic task duration by adding early and late completion options to these problems. For each project in the experiment, a project's tasks are performed in sequential numerical order, i.e., a project starts with task one which is a predecessor of task two which is a predecessor of task three. See Figure 1.

The model uses the state space as defined in subsection 2.3. The number of states grows exponentially with the number of tasks in a project, the number of project types, task durations and due dates, and the large state space becomes computationally intractable which is called "the curse of dimensionality" (Sutton and Barto 2018). In our experiment, a state space for more than five project types with two tasks each

Table 3. 2 project types and 2 tasks problem.

2 project types and 2 tasks problem						
Project type (j)	Reward (r_j)	Tardiness cost (w_j)	Due date (F_j)	Task no (i)	Normal task duration ($t_{j,i}$)	Resource usage ($b_{j,i}$)
1	3	1	8	1	2	2
				2	2	2
2	10	9	5	1	3	1
				2	1	3
						Resource capacity = 3

Table 4. 2 project types and 2 tasks problem, differences (percent lower) from optimal results of DP.

2 project and 2 task problem with deterministic task durations										
λ_j	1%	10%	20%	30%	40%	50%	60%	70%	80%	90%
ORBA	0.008%	0.5%	1.4%	2.3%	3.1%	4.0%	4.9%	6.0%	7.1%	8.3%
GA	0.7%	6.5%	11.7%	15.4%	18.1%	20.0%	21.2%	21.4%	20.4%	17.0%
RBA	2.1%	19.9%	35.2%	46.1%	53.7%	59.3%	63.7%	67.3%	70.4%	72.7%
WDP	2.8%	25.6%	43.8%	55.4%	62.7%	67.3%	70.2%	72.1%	73.5%	75.5%
2 project and 2 task problem with uniform stochastic task durations (1E 1N 1L)										
λ_j	1%	10%	20%	30%	40%	50%	60%	70%	80%	90%
ORBA	0.7%	4.9%	7.4%	9.0%	10.1%	10.9%	11.5%	11.9%	12.3%	12.5%
GA	1.2%	9.3%	15.0%	18.5%	20.6%	21.8%	22.5%	22.8%	22.9%	22.8%
RBA	2.0%	17.5%	30.1%	39.2%	45.8%	50.6%	54.3%	57.0%	59.0%	60.2%
WDP	2.4%	21.0%	35.1%	44.5%	50.8%	55.1%	58.0%	61.0%	63.6%	65.9%

becomes computationally intractable. We limited our problem sizes to four project types and two tasks.

The problems considered in our experiments vary by number of project types, number of tasks, resource usages, different reward-tardiness cost settings and lengths of due dates. We call the difference between a project's due date and the sum of its expected tasks durations as slack time. This value also varies for each project in the problems. The total resource capacity is taken $B = 3$ for all problems.

The completion times of the deterministic task duration problem are the normal task completion times ($t_{j,i}$) of the stochastic task duration problem. Further, we assume that a task can complete 1 period earlier ($t^{min}_{j,i} = t_{j,i} - 1$) or later ($t^{max}_{j,i} = t_{j,i} + 1$) than normal in the stochastic version. With completion probabilities uniformly distributed between $[t^{min}_{j,i}, t^{max}_{j,i}]$ when $t_{j,i} \geq 2$; and with $\gamma_{j,i}(2) = 1/3$, $\gamma_{j,i}(1) = 2/3$ when $t_{j,i} = 1$.

We test each problem and its versions consecutively from 1% to 90% project arrival probabilities, increment by 10%. 0% and 100% arrival probabilities are not used in this comparison, because 0% arrival probability makes the problem static and 100% arrival probability causes a non-ergodic MDP, e.g., the empty state where no project has arrived cannot be reachable again from any states. For deterministic task durations and 100% arrival probability, the system can never be empty and results in a cycle. Thus the stopping criteria ($\Delta > \beta \times W_{min}$) of the value iteration (subsection 2.9) can no be reachable, the W_{min} value remains as zero, and the W_{max} value does not change after the best policy is found.

4.1.1. 2 project types and 2 tasks problem

The two project types and two tasks problem (see Table 3) is the smallest problem in our test sample with 1424 states. Since project type two has a higher completion reward and higher tardiness cost with a smaller slack time. The project type two

Table 5. 2 project types and 3 tasks problem.

2 project types and 3 tasks problem						
Project type (j)	Reward (r_j)	Tardiness cost (w_j)	Due date (F_j)	Task no (i)	Normal task duration ($t_{j,i}$)	Resource usage ($b_{j,i}$)
1	12	8	10	1	1	1
				2	2	2
				3	5	1
2	6	5	15	1	4	1
				2	3	2
				3	4	1
						Resource capacity = 3

Table 6. 2 project types and 3 tasks problem, differences (percent lower) from optimal results of DP.

2 project and 3 task problem with deterministic task durations										
λ_j	1%	10%	20%	30%	40%	50%	60%	70%	80%	90%
ORBA	0.003%	0.2%	0.4%	0.6%	0.8%	1.0%	1.1%	1.1%	1.1%	0.8%
GA	0.003%	0.2%	0.4%	0.6%	0.8%	1.0%	1.1%	1.1%	1.1%	0.8%
RBA	0.4%	3.0%	5.0%	7.1%	9.7%	13.2%	17.6%	23.2%	30.6%	40.6%
WDP	0.9%	8.1%	13.6%	18.2%	23.8%	30.0%	36.2%	42.3%	48.2%	53.6%
2 project and 3 task problem with uniform stochastic task durations (1E 1N 1L)										
λ_j	1%	10%	20%	30%	40%	50%	60%	70%	80%	90%
ORBA	0.2%	1.0%	1.5%	1.9%	2.2%	2.6%	2.9%	3.1%	3.3%	3.5%
GA	0.2%	1.0%	1.5%	1.9%	2.2%	2.6%	2.9%	3.1%	3.3%	3.5%
RBA	0.5%	3.0%	4.6%	5.8%	6.9%	7.9%	8.8%	9.6%	10.3%	10.9%
WDP	1.6%	11.5%	17.0%	20.3%	22.6%	24.6%	26.3%	27.7%	28.9%	29.9%

contributes larger reward opportunities, however, its late completion is less rewarding compared to the late completion of the project type one. Resource usage of both types of projects allow parallel processing of any task of project type one with the first task of project type two. Thus, the processing decision of the second task of the project type two or any tasks of the project type one, is a bottleneck for this problem.

The minimum difference with optimum is seen at the 1% arrival probability for both version of the problem. The maximum difference is seen at 70% arrival probability for deterministic task durations and 80% arrival probability for stochastic task durations.

4.1.2. 2 project types and 3 tasks problem

The two project types and three tasks problem (see Table 5) has 16612 states. Most of the task combination can be processed together, except for second tasks. The project type one is as twice as much profitable. However the slack time of project type one is shorter so its due date may easily be exceeded which leads to pay a tardiness cost.

The GA's results are equal to optimal reactive baseline algorithm's results and both algorithm very close to the optimum at this problem for all arrival probabilities and task duration variations. The RBA's results are less good as its performance deteriorates with higher arrival probabilities.

4.1.3. 3 project types and 2 tasks problem

Three project types and two tasks problem (see Table 7) has 212568 states. Three types of project can not be processed together and this leads to paying tardiness cost at least for one type of project. A maximum of two project types can be processed at the same time and only project type one can be processed with others. Project type

Table 7. 3 project types and 2 tasks problem.

3 project types and 2 tasks problem						
Project type (j)	Reward (r_j)	Tardiness cost (w_j)	Due date (F_j)	Task no (i)	Normal task duration ($t_{j,i}$)	Resource usage ($b_{j,i}$)
1	8	5	10	1	5	1
				2	2	1
2	5	3	8	1	1	2
				2	3	1
3	20	19	10	1	2	3
				2	7	2
						Resource capacity = 3

Table 8. 3 project types and 2 tasks problem, differences (percent lower) from optimal results of DP.

3 project and 2 task problem with deterministic task durations										
λ_j	1%	10%	20%	30%	40%	50%	60%	70%	80%	90%
ORBA	0.02%	1.7%	6.1%	12.6%	20.0%	26.5%	31.0%	34.1%	36.3%	37.6%
GA	0.1%	4.9%	13.0%	22.0%	31.1%	39.1%	45.6%	51.6%	58.1%	67.2%
RBA	1.5%	15.3%	25.1%	30.1%	32.3%	32.6%	31.1%	28.2%	23.5%	15.4%
WDP	4.1%	34.3%	49.9%	59.0%	66.4%	72.6%	77.1%	80.2%	82.2%	83.3%
3 project and 2 task problem with uniform stochastic task durations (1E 1N 1L)										
λ_j	1%	10%	20%	30%	40%	50%	60%	70%	80%	90%
ORBA	0.3%	4.2%	8.6%	13.2%	18.0%	21.6%	24.4%	26.5%	28.2%	29.5%
GA	0.6%	6.7%	12.8%	19.1%	25.3%	30.3%	33.9%	36.4%	38.0%	38.9%
RBA	1.3%	13.2%	20.4%	23.4%	24.8%	24.8%	24.2%	23.2%	22.1%	21.0%
WDP	3.7%	29.2%	40.6%	45.9%	50.1%	53.3%	55.7%	57.3%	58.3%	59.0%

three has the largest reward and highest tardiness cost.

As it can be seen from Table 8, the algorithms diverge from the optimum rapidly as arrival probability increases. The GA produced the closest to optimal and to optimal reactive scheduling results at 1% to 30% percent arrival probabilities. After that the RBA generates better results than the GA. The RBA processes type one and type two projects together and delays the processing of type three. Only in this problem, the RBA has produced better results than the GA.

4.1.4. 4 project types and 2 tasks problem

The four project types and two tasks problem (see Table 9) is the largest problem in our experiments with 1509132 states. All project types have the same resource usage and sum of task durations. A first task of any project type can be processed with up to two first tasks or one second task of other project types, however, a second task can only be processed with a task one of another project type. The slack times of project one and two are negative, project three's slack times is zero and project four's slack time is one. This implies that most of the projects will be completed later than their planned due date and the tardiness payment will be inevitable.

According to results which is shown in Table 10, the GA's results are close to optimal reactive scheduling results. Best results of the alternative algorithms are seen at 1% arrival probability and the worst results are seen at 90% arrival probability.

Table 9. 4 project types and 2 tasks problem.

4 project and 2 task problem						
Project type (j)	Reward (r_j)	Tardiness cost (w_j)	Due date (F_j)	Task no (i)	Normal task duration ($t_{j,i}$)	Resource usage ($b_{j,i}$)
1	18	3	4	1	5	2
				2	1	1
2	27	4	5	1	4	2
				2	2	1
3	18	5	6	1	3	2
				2	3	1
4	18	6	7	1	2	2
				2	4	1
						Resource capacity = 3

Table 10. 4 project types and 2 tasks problem, differences (percent lower) from optimal results of DP.

4 project and 2 task problem with deterministic task durations										
λ_j	1%	10%	20%	30%	40%	50%	60%	70%	80%	90%
ORBA	0.0003%	0.2%	1.0%	3.2%	4.4%	4.7%	6.4%	10.0%	13.8%	17.8%
GA	0.020%	1.2%	2.9%	5.8%	6.9%	6.8%	8.0%	11.5%	15.4%	19.0%
RBA	0.4%	6.6%	14.6%	21.4%	25.1%	26.8%	28.7%	31.4%	33.9%	36.1%
WDP	1.4%	21.3%	37.8%	46.2%	50.5%	52.8%	54.8%	57.3%	59.4%	61.5% [a]
4 project and 2 task problem with uniform stochastic task durations (1E 1N 1L)										
λ_j	1%	10%	20%	30%	40%	50%	60%	70%	80%	90%
ORBA	0.008%	0.4%	1.5%	3.4%	4.8%	5.7%	6.7%	8.1%	9.4%	10.4%
GA	0.021%	0.8%	2.0%	4.1%	5.7%	6.4%	7.3%	8.5%	9.7%	10.8%
RBA	0.3%	5.3%	11.9%	17.4%	21.0%	23.0%	24.5%	26.0%	27.4%	28.6%
WDP	1.1%	19.1%	34.7%	42.3%	46.3%	48.5%	50.1%	51.5%	52.7%	53.7%

[a] approximate

4.2. *Discussion*

The results shown in section 4 illustrate that none of alternative algorithms produces the optimum results as the DP. However, the ORBA and GA produce almost optimal solutions in 1% arrival probability and close to optimal solutions with the other low arrival probabilities. The ORBA's results were from 0.0003% to 37.6% worse than the optimum results, and they always deteriorate from optimal as the arrival probability increases.

The GA's results are generally closer to optimal reactive scheduling results compared to RBA for the majority of the considered problems and their task duration variations. The GA's results were from 0.003% to 67.2% lower than the optimum results. For the 2 project types and 3 task problem, GA'results are equal to optimal reactive scheduling results. In two of the problems; the results with the deterministic task durations were closer to optimum, while in the other two, it was vice versa.

The RBA's results are between the GA and the WDP for most of the test problem. The RBA's results were from 0.3% to 72.7% lower than the optimum results. In three project types with two tasks problem, the RBA produced better results than the GA at higher arrival probabilities. However, in most of the cases, its results were closer to the WDP than the optimum since the RBA might prioritise the less rewarding project types over others. Thus, it can be said that using a single priority rule usually does not produce good results unless the given priority rule is designed well for problem features.

Since the GA, the ORBA and the RBA are reactive baseline scheduling algorithms,

they generate their decisions without considering the future uncertainties such as early or late task completion or new project arrivals. Thus we may accept that the result of a reactive baseline scheduling algorithm deteriorates compared to the optimum as problem deviates from the static assumption i.e. no project arrivals. However, some anomalies were observed for very high arrival probabilities. These anomalies occur since the tardiness cost is only paid once when a project is completed. In the current model, high arrival probabilities lead to postponing some project types infinitely. Thus, they stay in the system without causing a tardiness cost while the other project types continue processing without causing much tardiness cost.

5. Conclusion

Paper summary. In this paper, we studied the resource-constrained multi-project scheduling problem with uncertain project arrivals and uncertain task duration. We modelled the problem as an infinite-horizon discrete-time MDP. We inspected both the cases where the task durations are deterministic or stochastic. We used the uniform distribution for the stochastic task durations.

We used DP value iteration to maximise the long-term average profit per unit time. We tested the limits of the DP on the dynamic and stochastic RCMPSP and generated four test problems with both deterministic and stochastic task duration variations. Our approach generates the optimum policies for the dynamic and stochastic RCMPSP and contributes to literature by extending the work of Melchiors et al. (2018) which only considered single-task projects. We used two heuristic and one exact reactive baseline scheduling methods and a worst-decision DP on the same problems and compared their results with exact results of the DP. We used GA and RBA as heuristic and ORBA as exact reactive baseline scheduling methods.

According to our findings, a reactive baseline scheduling method with a GA produced closer to optimal results with and without considering arrivals than the priority rule heuristic for the most of the test problem with different arrival probabilities and deterministic or stochastic task durations options. The GA produced the optimal reactive scheduling results for one problem but not for the others even though the setting for GA were the same. The RBA generally produced results between the GA and the WDP. Since reactive baseline scheduling does not consider the random changes before they occurred, the GA's, ORBA's and the RBA's results are closer to optimal at low arrival probabilities and diverge from optimum at high arrival probabilities. The GA's and the RBA's results are closer to optimal at deterministic task durations than the stochastic task durations. However, a few exceptions have been observed.

Managerial insights. This study provides a performance comparison of the methods and give insights to project managers for determination of the solution method in highly dynamic and stochastic environments. We have seen that DP suffers from the curse of dimensionality even for the small size problems and reactive baseline scheduling methods do not produce close to optimum results at the high arrival probabilities or stochastic task durations. We suggest using DP for small problems and the reactive baseline scheduling methods such as GA for the environments with low uncertainties. We don't recommend using GA or other reactive baseline scheduling methods in highly uncertain environments since our test results showed that GA may generate up to 67.2% less average profit per unit time compared to optimal in these environments. We suggest considering other methods for larger and more complex problems with high or moderate uncertainties.

Future research direction. Our work showed that for environments which change frequently, the most popular method GA and other reactive scheduling methods perform poorly, and alternative solution methods should be considered. Future work might seek other solution methods and compare their performances in these environments. An ADP algorithm with a well-designed and tuned approximation model is a modern example of alternative solution methods. See, for example, Melchiors (2015), Choi, Realff, and Lee (2007), Parizi, Gocgun, and Ghate (2017), which are to the best of our knowledge the only attempts in this direction for similar problems. An extension would be to develop an approximate model and/or approximate solution approach which would not suffer from the curse of dimensionality while also considering both the uncertainties of new project arrivals and task durations as considered here. Other important future research topics are to consider additional uncertainties such as stochastic resource availability or multiple modes of task processing.

References

Adler, Paul Simon, Avi Mandelbaum, Viên Nguyen, and Elizabeth Schwerer. 1995. "From Project to Process Management: An Empirically-based Framework for Analyzing Product Development Time." *Management Science* 41 (3): 458–484.

APM. 2018. "The State of Project Management Annual Survey 2018." Accessed 2018-11-12. `http://www.wellingtone.co.uk/wp-content/uploads/2018/05/The-State-of-Project-Management-Survey-2018-FINAL.pdf`.

Browning, Tyson R., and Ali A. Yassine. 2010. "Resource-Constrained Multi-project Scheduling: Priority Rule Performance Revisited." *International Journal of Production Economics* 126 (2): 212–228.

Capa, Canan, and Kemal Kilic. 2015. "Proactive Project Scheduling with a Bi-Objective Genetic Algorithm in an R&D Department." In *2015 International Conference on Industrial Engineering and Operations Management (IEOM)*, Vol. 1, 1–6.

Choi, Jaein, Matthew J Realff, and Jay Hyung Lee. 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.

Cohen, Izack, Boaz Golany, and Avraham Shtub. 2005. "Managing Stochastic, Finite Capacity, Multi-Project Systems through the Cross-Entropy Methodology." *Annals of Operations Research* 134 (1): 183–199.

Fliedner, Thomas, Walter Gutjahr, Rainer Kolisch, and Philipp Melchiors. 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*, 148–151.

Gonçalves, José Fernando, Jorge JM Mendes, and Mauricio GC Resende. 2008. "A Genetic Algorithm for the Resource Constrained Multi-Project Scheduling Problem." *European Journal of Operational Research* 189 (3): 1171–1190.

Herbots, Jade, Willy Herroelen, and Roel Leus. 2007. "Dynamic Order Acceptance and Capacity Planning on a Single Bottleneck Resource." *Naval Research Logistics (NRL)* 54 (8): 874–889.

Karam, Ahmed, and Sanja Lazarova-Molnar. 2013. "Recent Trends in Solving the Deterministic Resource Constrained Project Scheduling Problem." In *9th International Conference on Innovations in Information Technology (IIT)*, 03, 124–129.

Kolisch, Rainer, and Sönke Hartmann. 1999. *Heuristic Algorithms for the Resource-Constrained Project Scheduling Problem: Classification and Computational Analysis*, 147–178. Springer US.

Li, Hongbo, Xianchao Zhang, Jinshuai Sun, and Xuebing Dong. 2020. "Dynamic resource

levelling in projects under uncertainty." *International Journal of Production Research* 0 (0): 1–21.

Melchiors, Philipp. 2015. *Dynamic and Stochastic Multi-Project Planning*. Lecture Notes in Economics and Mathematical Systems. Springer.

Melchiors, Philipp, Roel Leus, Stefan Creemers, and Rainer Kolisch. 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.

Mori, Masao, and Ching Chih Tseng. 1997. "A Genetic Algorithm for Multi-Mode Resource Constrained Project Scheduling Problem." *European Journal of Operational Research* 100 (1): 134–141.

Ortiz-Pimiento, Nestor Raul, and Francisco Javier Diaz-Serna. 2018. "The Project Scheduling Problem with Non-deterministic Activities Duration: A Literature Review." *Journal of Industrial Engineering and Management (JIEM)* 11 (1): 116–134.

Pamay, M Berke, Kerem Bülbül, and Gündüz Ulusoy. 2014. "Dynamic Resource Constrained Multi-Project Scheduling Problem with Weighted Earliness/tardiness Costs." In *Essays in Production, Project Planning and Scheduling*, edited by P. Pulat, S. Sarin, and R. Uzsoy, Vol. 200, 219–247. Springer.

Parizi, Mahshid Salemi, Yasin Gocgun, and Archis Ghate. 2017. "Approximate Policy Iteration for Dynamic Resource-Constrained Project Scheduling." *Operations Research Letters* 45 (5): 442–447.

Powell, Warren B. 2011. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Vol. 842 of *Wiley series in probability and statistics*. Wiley.

Rostami, Salim, Stefan Creemers, and Roel Leus. 2018. "New Strategies for Stochastic Resource-Constrained Project Scheduling." *Journal of Scheduling* 21 (3): 349–365.

Satıç, Uğur. 2014. "Çok Kaynak Kısıtlı Projelerin Sezgisel Yöntemlerle Çizelgelenmesi." Yildiz Technical University. Accessed 2019-07-26. `https://tez.yok.gov.tr/UlusalTezMerkezi/TezGoster?key=gyLHMouPes-CvnhRcjQsKQIo1AYi4WhfoZxbAaK4INn4yvJXWUop3HXf7wZV_sh4`.

Satic, Ugur, Peter Jacko, and Christopher Kirkbride. 2019. "Performance evaluation of scheduling policies for the DRCMPSP." In *Proceedings of the 25th International Conference on Analytical & Stochastic Modelling Techniques & Applications*, Lecture Notes in Computer Science. to appear.

Sutton, R. S., and A. G. Barto. 2018. *Reinforcement Learning: An Introduction*. Second edition. ed., Adaptive Computation and Machine Learning. MIT Press.

Tijms, Henk C. 1994. *Stochastic Models: An Algorithmic Approach*. John Wiley & Sons.

Zarghami, Seyed Ashkan, Indra Gunawan, Graciela Corral de Zubielqui, and Bassam Baroudi. 2019. "Incorporation of resource reliability into critical chain project management buffer sizing." *International Journal of Production Research* 0 (0): 1–15.