Proc. of the European Safety and Reliability Conference (ESREL 2026)
Edited by José C. Matos, Paulo B. Lourenço, Michael Beer, Daniel Oliveira,
Edoardo Patelli, Hélder S. Sousa, Rui A. Silva and Monica Santamaria
 Published by Research Publishing, Singapore.
doi: 10.3850/ESREL2026061419_esrel26-p26532-cd

Project Risk Management using Deep Reinforcement Learning

Arne B. Huseby
University of Oslo, Norway. E-mail: arne@math.uio.no

Martine S. Falkeid
University of Oslo, Norway. E-mail: martsf@math.uio.no

Project scheduling under uncertainty involves balancing acceleration costs against the risk of delay. Due to the uncertainty in activity durations, the project manager must decide before each task whether to allocate extra resources to reduce expected lateness. This classical time-cost trade-off problem (TCTP), has been widely studied both in deterministic and stochastic settings. Recently, the problem has been formulated as a stochastic shortest-path decision process, providing a natural basis for learning-based control. Advances in reinforcement learning (RL) now enable adaptive scheduling without explicit stochastic models. Previous studies applied deep RL to resource-constrained scheduling and activity acceleration. Building on these foundations, the present paper formulates project scheduling as a decision process in which each activity has an uncertain duration described by a probability distribution, and an optional acceleration action with associated cost. A deep reinforcement learning approach is developed to determine, at each milestone of the project, whether acceleration is economically justified given the expected risk of overall project delay. The proposed framework thus combines classical time-cost optimisation with modern risk-aware learning to provide a flexible tool for managing project risk under uncertainty. In the paper both simple sequential projects as well as more complex projects with parallel activities are considered. The proposed methodology is illustrated by a few numerical examples.

Keywords: Project risk management; scheduling under uncertainty; deep reinforcement learning

1. Introduction

An important objection against traditional methodology for assessing project risk, is that the sequential nature of the management process is not addressed. During the different phases of a project, many decisions are made with significant impact on the progress. When the project risk is evaluated, one should incorporate not only uncertainties in the durations of the different activities, but also uncertainty due to future decisions.

Project scheduling under uncertainty involves balancing acceleration costs against the risk of delay. Due to the uncertainty in activity durations, the project manager must decide before each task whether to allocate extra resources to reduce expected lateness. This classical *time–cost trade-off problem (TCTP)* has been widely studied both in deterministic and stochastic settings Gutjahr and Strauss (2000); Cohen et al. (2007); Kang and Choi (2015). More recently, Bertazzi et al. (2024) formulated dynamic expediting as a stochastic shortest-path decision process, providing a natural basis for learning-based control. Advances in reinforcement learning (RL) now enable adaptive scheduling without explicit stochastic models. Previous studies Sung et al. (2020); Sallam et al. (2021); Cai et al. (2024) applied deep RL to resource-constrained scheduling and activity acceleration.

Building on these foundations, the present paper formulates project scheduling as a decision process in which each activity has an uncertain duration described by a probability distribution, and an optional acceleration action with associated cost. A deep reinforcement learning approach is developed to determine, at each milestone of the project, whether acceleration is economically justified given the expected risk of overall project delay. The proposed framework thus combines classical time-cost optimisation with modern risk-aware learning to provide a flexible tool for managing project risk under uncertainty.

2. Decision Processes

A *decision process* (DP) is a framework for analysing sequential decision-making under uncertainty. The decision maker is typically referred to as an *agent*, and the decisions are referred to as *actions*. Formally, a DP consists of the following elements $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, where $\mathcal{S}$ is a set of states observed by the agent, $\mathcal{A}$ is a set of actions available to the agent, P is a transition probability measure on the state space $\mathcal{S}$ governing the environment dynamics, R is a reward function, and $\gamma \in [0, 1]$ is a discount factor.

In this paper we only consider DPs with a *finite* number, n, of actions taken at some random points in time $t_0 < t_1 < t_2 < \cdots < t_{n-1}$, where $t_0 = 0$. We refer to these points of time as the *action points*. At action point t_i, the agent observes the current state $s(t_i) \in \mathcal{S}$, selects an action $a(t_i) \in \mathcal{A}$ according to some chosen *policy* and receives a reward $R = R(s(t_i), a(t_i))$. The process then transitions to a new state $s(t_{i+1})$ according to the transition probability measure P. Finally, the DP terminates at time $t_n > t_{n-1}$. No action is taken at the *termination point*.

For simplicity we do not include discounting by letting $\gamma = 1$. Moreover, the reward function is assumed to be deterministic.

A decision process where the transition probability measure satisfies the *Markov property*:

$$P(\cdot \mid s(t), a(t), 0 \le t \le t_i) = P(\cdot \mid s(t_i), a(t_i)), \quad i = 0, 1, \ldots n-1 \tag{1}$$

is called a Markov Decision process, or MDP.

When analysing a DP, the objective is to find a policy that maximises the total expected reward. For project management all rewards are in the form of costs, which represent negative rewards. Thus, the main objective can be stated as finding a policy that minimises the total expected cost.

A well-established method for solving such optimisation problems is *reinforcement learning*. *Q-learning* is a popular reinforcement learning algorithm used to train agents to maximise the expected reward. A key concept of Q-learning is the *Q-value function*, denoted $Q(a, s)$, which represents the expected cumulative reward of taking action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$, and then following the optimal policy thereafter. For MDPs where $\mathcal{S}$ and $\mathcal{A}$ are finite, the Q-value function can calculated *iteratively* using the Bellman equation. For more general cases where e.g., the state space is continuous, the Q-value function can be approximated by a neural network. By training this using e.g., an ϵ-greedy algorithm, a sufficiently good approximation can be obtained.

When the Markov property Eq. (1) is satisfied, all relevant information about the process up to and including time t_i is captured in the state $s(t_i)$. Moreover, given a sufficiently precise estimate, $\hat{Q}$, of the Q-function, the optimal action $a^*(t_i)$ at time t_i can be found by letting:

$$a^*(t_i) = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\, \hat{Q}(a, s(t_i))$$

Thus, the optimal action is essentially a function of the current state $s(t_i)$.

If Eq. (1) is not satisfied, this implies that current state, $s(t_i)$, does not contain all information needed for optimal decisions. As a result, the reinforcement learning process may converge to a suboptimal policy. *Non-Markovian* problems can often be described as *partially observable Markov decision processes* (POMDPs), where there is an underlying unobservable Markov state $x(t)$, and where the observable state $s(t)$ is a function of this underlying state, i.e., $s(t) = g(x(t))$. Fortunately, POMDPs can be analysed successfully in several different ways. For an extensive discussion of POMDPs see Oliehoek and Amato (2016). Note, however, that even when a reinforcement learning process does converge to a suboptimal policy, this policy may still work surprisingly well.

In order to estimate optimal policies, models were trained using deep reinforcement learning. That is, we approximated the Q-value function by a neural network. The input to this network is the current state. This is then passed through two hidden layers, each consisting of a linear (affine) transformation with 64 hidden units, followed by a so-called nonlinearity rectified linear unit, a simple, piecewise-linear activation function used in neural networks of the form $\mathrm{ReLU}(x) = \max(0, x)$. These hidden layers allow the network to learn nonlinear features of the state that are

useful for predicting long-term returns. The final linear layer maps the learned features to an output vector. Each component of this vector corresponds to the estimated Q-value of taking a particular action in the given state[a].

During the training and evaluation the environment models are simulated using python scripts. For project scheduling this is done efficiently using *discrete event simulations*. This is a flexible technique allowing the action points to be processed in the correct chronological order. In the examples we used $N_T = 500\,000$ training episodes and $N_E = 50\,000$ evaluation episodes. The neural network was optimised using a standard ϵ-greedy method and using the well-known `pytorch` and `gymnasium` libraries. The *decay factor* was chosen to be 0.99999, while the minimum ϵ-value was 0.01.

3. Project schedule models

A project schedule consists of a set of *activities* arranged in some logical order. A main goal of project scheduling is keeping the total project duration under control, and avoiding that a given time limit is exceeded. The durations of the activities are uncertain. Thus, even with a seemingly realistic time limit, there is still a risk that this may be exceeded, and if this happens, some *penalty cost* will occur. In order to minimise this risk, we will investigate available options in different phases of the project, and in particular what can be done in order to avoid that the time limit of the project is exceeded.

If it is possible to accelerate one or more of the activities, the risk of a penalty cost may be reduced. However, every such action typically has a cost as well. Thus, a manager typically wants to delay decisions regarding acceleration until updated progress information becomes available. In order to assess the overall project risk, such delayed decisions must be taken into account. The aim of the analysis is to develop a strategy minimising the total cost, i.e., the sum of the acceleration costs and the penalty cost.

More specifically, we consider a project consisting of activities, denoted $A_1, A_2 \ldots, A_n$, and milestones, denoted $M_0, M_1, M_2, \ldots, M_n$, where M_0 represents the start of the project and M_n the end of the project. A common technique for managing such projects is the PERT[b]-method, where a project is represented as an acyclic directed graph $G = (V, E)$ where V is the set of nodes, and E is the set of edges. The nodes are the *milestones* of the projects while the directed edges are the *activities*. Thus, we let $V = \{M_0, M_1, \ldots, M_n\}$, while $E = \{A_1, \ldots, A_n\}$. Being acyclic and directed, the graph G defines a *partial ordering*, denoted $\prec$ of the project milestones. More specifically, $M_j \prec M_k$ if the graph G contains a directed path from M_j to M_k. As the project progresses, the milestones are reached in an order which is consistent with this partial ordering. Thus, if $M_j \prec M_k$, this implies that M_j is reached before M_k. A milestone is said to be reached as soon as all activities preceding it are completed. Moreover, we assume that an activity starts as soon as the milestone immediately preceding it is reached[c].

The milestones correspond to the action points of the DP, i.e., the points where decisions are made regarding acceleration of the activities. In particular, the start of the project, represented by M_0, corresponds to the action point t_0, while the end of the project, represented by M_n corresponds to the termination point t_n. In general, however, the milestones may be reached in an unpredictable order. Thus, the action point t_j does not necessary correspond to the milestone M_j.

In order to model the uncertainties related to the durations of the activities we introduce the following random variables for $i = 1, \ldots, n$:

$X_i =$ The duration of A_i given no acceleration,

$T_i =$ The actual duration of A_i.

[a] The chosen structure of the neural network is arguably somewhat arbitrary. In this brief study no attempt has been made to optimise this structure. Still the chosen structure is rich enough to produce a good approximation to the Q-value function, and still allow for efficient training.

[b] PERT: Project Evaluation and Review Technique

[c] More general project models allow activities to be postponed, so that the milestones just represent *earliest* point of time when the activity can start.

We assume that $X_1, \ldots, X_n$ are independent random variables, and that $X_i \sim F_i$, $i = 1, \ldots, n$, where $F_1, \ldots, F_n$ are known distribution functions. We also introduce action variables for $i = 1, \ldots, n$:

$$\delta_i = \begin{cases} 1 & \text{if activity } A_i \text{ is accelerated} \\ 0 & \text{otherwise} \end{cases}$$

If an activity is accelerated, its duration is reduced by a certain (known) factor α. Thus, the relation between X_i, T_i, δ_i and α can be expressed as:

$$T_i = \alpha^{\delta_i} X_i, \quad i = 1, \ldots, n.$$

In order to find the total duration of the project, it is convenient to identify the *directed paths* in G from the start M_0 to the finish M_n of the project. A directed path $\mathcal{P}$ is interpreted as the *index set* of a subset of E such that the subgraph spanned by the set $\{A_i : i \in \mathcal{P}\}$ forms a directed path from M_0 to M_n. Assuming that these directed paths are $\mathcal{P}_1, \ldots, \mathcal{P}_p$, the duration of the project, denoted T, is given by:

$$T = \max_{1 \le j \le p} \sum_{i \in \mathcal{P}_j} T_i = \max_{1 \le j \le p} \sum_{i \in \mathcal{P}_j} \alpha^{\delta_i} X_i \qquad (2)$$

Thus, the total duration of the project is equal to the duration of the so-called *critical path* of the project, i.e., the path with the longest duration.

As already mentioned, the main goal for the project manager is to make sure that the project is completed within a certain time limit, which we denote by τ. If $T > \tau$, a *penalty cost*, denoted $\Psi(T)$ must be paid. Several different penalty cost functions may be considered. Typically, Ψ is assumed to be nonnegative and nondecreasing, and with the property that $\Psi(T) = 0$ if $T \le \tau$. In this paper, we focus on the following penalty function:

$$\Psi(T) = C \cdot \max[T - \tau, 0], \qquad (3)$$

where $C > 0$ is a suitable constant. Concerning the cost of acceleration cost we assume simply that is proportional to the expected duration of the activity given no acceleration. Denoting $E[X_i]$ by ξ_i, $i = 1, \ldots, n$, and the proportionality factor by D, we get that the total cost, denoted K, is given by:

$$\begin{aligned} K &= D \sum_{i=1}^{n} \delta_i \xi_i + \Psi(T) \qquad (4) \\ &= D \sum_{i=1}^{n} \delta_i \xi_i + C \cdot \max[T - \tau, 0] \end{aligned}$$

3.1. *Linear projects*

In this subsection, we consider *linear* projects, i.e., projects where all activities are carried out in a fixed order, and no activities are carried out in parallel. Such a project is illustrated by the directed graph shown in Figure 1. Milestone M_i is reached when the activity immediately preceding M_i, i.e., A_i, is completed, $i = 1, \ldots, n$. Moreover, activity A_i starts immediately after milestone M_{i-1} is reached, $i = 1, \ldots, n$.

As one can see in Figure 1, the directed graph of a linear project only has one directed path from M_0 to M_n, i.e., $\mathcal{P} = \{1, 2, \ldots, n\}$. Hence, the formula Eq. (2) for calculating the total project duration simplifies to:

$$T = \sum_{i=1}^{n} T_i = \sum_{i=1}^{n} \alpha^{\delta_i} X_i \qquad (5)$$

The action space $\mathcal{A} = \{a_0, a_1\}$ consists of two possible actions, a_0: *do not accelerate* the next activity, and a_1: *do accelerate* the next activity. At milestone M_i, i.e., at action point t_i, an action $a \in \mathcal{A}$ is chosen. If $a = a_1$, the next activity, A_{i+1} is accelerated and $\delta_{i+1} = 1$. If $a = a_0$, the next activity, A_{i+1} is *not* accelerated and $\delta_{i+1} = 0$.

The state space $\mathcal{S}$ can be defined in various ways. However, in order to ensure that the Markov property Eq. (1) holds, we recall that the state at a given action point t_i should contain all relevant information at this point of time. A natural choice is letting the state $s(t_i) \in \mathcal{S}$ at time t_i be defined as the time t_i together with the index i of current

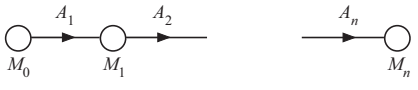

Fig. 1. A linear project

milestone, i.e., $s(t_i) = (t_i, i)$. The transition from $s(t_i)$ to the next state $s(t_{i+1})$ can then be described by the difference between the two states:

$$\begin{aligned} s(t_{i+1}) - s(t_i) &= (t_{i+1} - t_i, 1) \\ &= (T_{i+1}, 1) = (\alpha^{\delta_{i+1}} X_{i+1}, 1) \end{aligned}$$

Since $X_1, \ldots, X_n$ are assumed to be independent, it follows that $(T_{i+1}, 1)$ only depends on the decision δ_{i+1}. This implies that the Markov property Eq. (1) holds, and the process is an MDP.

We now illustrate this case by an example where $n = 8$, and where $X_i \sim R[0, 2\xi_i]$ (the uniform distribution), $i = 1, \ldots, 8$. Hence, it follows that $\mathrm{E}[\boldsymbol{X}] = \boldsymbol{\xi} = (\xi_1, \ldots, \xi_8)$, where:

$$\boldsymbol{\xi} = (1.0, 2.0, 1.0, 2.0, 1.0, 2.0, 1.0, 2.0)$$

We calculate the total duration of the project assuming that $X_i = \xi_i$, $i = 1, \ldots, 8$, and given no acceleration to be:

$$\sum_{i=1}^{8} X_i = \sum_{i=1}^{8} \xi_i = 12$$

The time limit of the project, τ, is 12.6. This includes a 5% buffer on top of the calculated duration. The cost factors used in Eq. (4) are $C = 20$ and $D = 1$ (monitary units).

After the agent is trained, it is evaluated using an independent set of episodes. Based on these episodes the performance of the agent can be calculated. Table 1 lists the observed frequency of accelerations (i.e., the action a_1) among the evaluation episodes. We observe that at milestones M_0, M_1 and M_2 the following activities, i.e. A_1, A_2 and A_3, were never accelerated. Later in the project, the agent has a stronger tendency towards acceleration. This is reasonable as at these milestones more information about the progress is available, and the risk of not finished before the deadline is more imminent. Activity A_6 following after milestone M_5, which is accelerated in 46.5% of the episodes, and A_8 following after milestone M_7 which is accelerated in 44.0% of the episodes, are the two most frequently accelerated activities. Both of these activities has expected durations of 2 given no accelerations. This implies that the impacts of accelerations are more noticeable compared to the activities with shorter expected durations.

Table 1. Frequency of accelerations

Milestone	Acc. freq.
0	0.0 %
1	0.0 %
2	0.0 %
3	24.0 %
4	16.2 %
5	46.5 %
6	32.8 %
7	44.0 %

In order to show the advantage of having an agent (DRL-agent) utilising progress information along the way, we compare the performance of this agent to eight agents using deterministic poli-

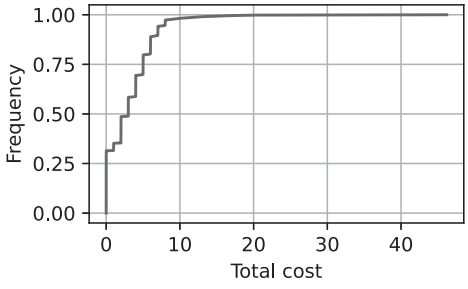

Fig. 2. Cumulative distribution of the total project cost

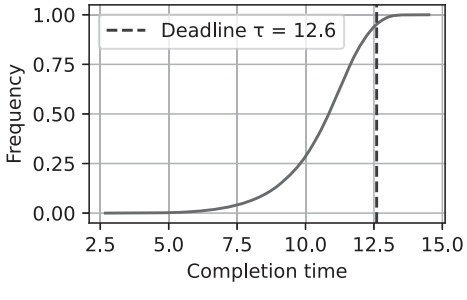

Fig. 3. Cumulative distribution of the total project duration

cies where the number of accelerated activities is determined at the start of the project. The first deterministic agent (D-agent 1) accelerates only the last activity, the second deterministic agent (D-agent 2) accelerates the two last activities, and so on up to the last deterministic agent (D-agent 8) who accelerates all activities.

In Table 2 the 95%-percentiles and mean values of the total project cost for all the agents are listed. We observe that the DRL-agent by far has the lowest mean total cost (3.06). At 95%-level D-agent 4, with four accelerated activities, has a lower value, 6.00 compared to 8.00 for the DRL-agent. Still the mean total cost of D-agent 4 (6.55) is much higher than mean total cost of the DRL-agent. For the DRL-agent the number of accelerated activities varies between the episodes. On average 1.64 activities are accelerated. This is lower than for all the other agents except D-Agent 1. Still since the DRL-agent chooses the activities to accelerate dynamically, the mean total cost is less than a third of the mean total cost for D-Agent 1. In Figure 2 the cumulative distribution of the total project cost of the DRL-agent is shown.

Table 2. Total cost statistics for the DRL-agent and 8 deterministic agents (D-agents 1-8)

Agent	p95	Mean	# Accel.
DRL-agent	8.00	3.06	1.64
D-agent 1	48.91	9.31	1
D-agent 2	37.37	7.63	2
D-agent 3	12.22	6.21	3
D-agent 4	6.00	6.55	4
D-agent 5	8.00	8.02	5
D-agent 6	9.00	9.00	6
D-agent 7	11.00	11.00	7
D-agent 8	12.00	12.00	8

In Table 3 the 95%-percentiles and mean values of the total project time for all the agents are listed. The main goal is to make sure that the project finishes before the time limit, but as close as possible to this. If a project finishes much earlier than the time limit, this indicates that the agent has spent too much on acceleration. On the other hand, if a project finishes after the deadline, this means that the agent should have accelerated more activities.

In terms of mean values the DRL-agent, with a mean value of 10.6 comes fairly close to the time limit of $\tau = 12.6$ which indicates a good balance between the actions a_1 and a_0. At the same time the frequency of failures, i.e., projects not being finished within the time limit, is as low as 4.77%. In Figure 3 the cumulative distribution of the total project time of the DRL-agent is shown. The only agent with a mean value closer to τ is D-Agent 1 with a mean value of 11.01. However the frequency of failures for this agent is as high as 25.73%. We observe that D-Agent 7 and D-Agent 8 have 0% failed projects. However, the agents also have very low mean values (6.51 and 6.00). Thus, these agents clearly accelerate too many activities.

Table 3. Total time statistics for the DRL-agent, and 8 deterministic agents (D-agent 1-8)

Agent	p95	Mean	% Failure
DRL-agent	12.58	10.60	4.77
D-agent 1	14.95	11.01	25.73
D-agent 2	14.32	10.49	18.76
D-agent 3	12.96	9.49	0.70
D-agent 4	12.38	9.01	3.94
D-agent 5	10.93	8.00	0.31
D-agent 6	10.31	7.51	0.04
D-agent 7	8.79	6.51	0.00
D-agent 8	8.14	6.00	0.00

3.2. *Non-linear projects*

In this subsection we will consider the non-linear project illustrated in Figure 4. In this project activities A_1 and A_4 start at the same time. Milestone M_1 is reached when A_1 is completed, and then both A_2 and A_3 start. Milestone M_2 is reached when A_2 is completed, and then A_5 starts. Milestone M_3 is reached when A_3 and A_4 are completed, and then A_6 starts. Finally, the project ends when milestone M_4 is reached, i.e., when A_5 and A_6 are completed. The durations given no acceleration, i.e., $X_1, \ldots, X_8$ have the same distributions as in the previous subsection.

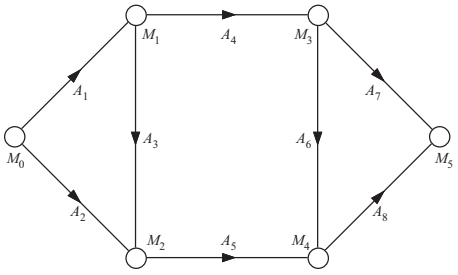

Fig. 4. A non-linear project

The minimal paths of this network are $\mathcal{P}_1 = \{1, 4, 7\}$, $\mathcal{P}_2 = \{1, 4, 6, 8\}$, $\mathcal{P}_3 = \{1, 3, 5, 8\}$, and $\mathcal{P}_4 = \{2, 5, 8\}$. If $X_i = \xi_i$, $i = 1, \ldots, 8$ and no activities are accelerated, it follows that the calculated total duration of the project is:

$$T = \max_{1 \le j \le 4} \sum_{i \in \mathcal{P}_j} X_i = \max\{4, 7, 5, 5\} = 7$$

The time limit of the project, τ, is 7.35. As in the linear case, this includes a 5% buffer on top of the calculated duration. The cost factors used in Eq. (4) are also the same as in the linear case.

As before the action space $\mathcal{A} = \{a_0, a_1\}$ consists of two possible actions, a_0: *do not accelerate* and a_1: *do accelerate*. However, in this case the actions apply to *all activities* immediately following the given milestone[d]. Thus, at milestone M_0, the action a_1 means that both activity A_1 and A_2 are accelerated, while if action a_0 is chosen, A_1 and A_2 are not accelerated. Similarly, at milestone M_2, the action a_1 means that both activity A_3 and A_4 are accelerated, and so on.

For simplicity we use the same state space as in the linear case. That is, $s(t_i) = (t_i, i)$, $i = 0, 1, \ldots, 5$. Since this project is non-linear, the resulting decision process does not satisfy the Markov property Eq. (1). By inspecting the project closer, however, it turns out that it is *almost* an MDP. That is, Eq. (1) holds for all action points except t_2 and t_3. Thus, training an agent using `pytorch` and `gymnasium` still works very well.

[d]Letting actions apply to all activities immediately following a milestone, is of course a simplification. In a more advanced model one could allow for more complicated action spaces where only a selected group of activities are accelerated. In this brief study, however, we have not considered such cases.

Table 4. Frequency of accelerations

Milestone	Acc. freq.
0	0.0 %
1	0.0 %
2	0.0 %
3	30.8 %
4	74.1 %

Table 4 shows the observed frequency of accelerations among the evaluation episodes. At milestones M_0, M_1 and M_2 the following activities, i.e. A_1, A_2, A_3, A_4, A_5, were never accelerated. At milestone M_3 the activities A_6 and A_7 are accelerated in 30.8% of the evaluation episodes, while at milestone M_4 activity A_8 is accelerated in 74.1% of the evaluation episodes. As in the linear case, the agent has a stronger tendency towards acceleration closer to the end of the project. As argued previously, this is reasonable. The average number of accelerated activities is 1.05.

The cumulative distribution of the total cost is shown in Figure 5. We observe that this has long upper tail indicating that high costs may occur as a result of exceeding the time limit. In particular, the 95%-percentile is as high as 16.29, while the mean is just 4.11.

The cumulative distribution of the total project

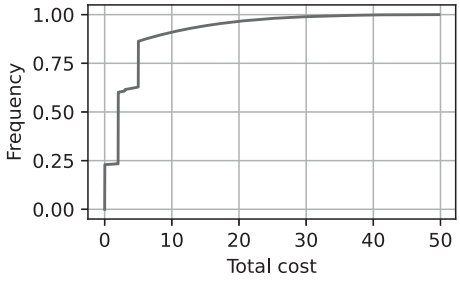

Fig. 5. Cumulative distribution of the total project cost

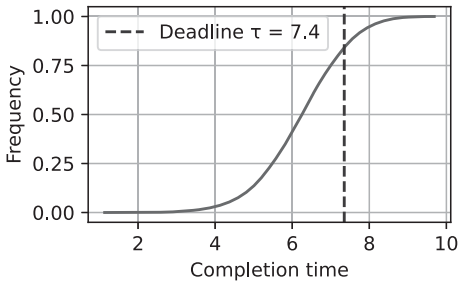

Fig. 6. Cumulative distribution of the total project duration

duration is shown in Figure 6. The mean value of the total project duration 6.22, i.e., less than the time limit $\tau = 7.35$. The 95%-percentile is 8.01, and the frequency of episodes with projects exceeding the time limit is 16.02%.

Just as in the linear case the DRL-agent outperformed the deterministic agents. See Table 5 and Table 6.

Table 5. Total cost statistics for the DRL-agent and 5 deterministic agents (D-agent 1-5)

Agent	p95	Mean	# Accel.
DRL-agent	16.29	4.11	1.05
D-agent 1	35.56	7.01	1
D-agent 2	8.34	5.68	2
D-agent 3	9.52	6.70	3
D-agent 4	9.00	9.00	4
D-agent 5	12.00	12.00	5

Table 6. Total time statistics for the DRL-agent and 5 deterministic agents (D-agent 1-5)

Agent	p95	Mean	% Failure
DRL-agent	8.01	6.22	0.1602
D-agent 1	9.03	6.28	0.2414
D-agent 2	7.52	5.48	0.0652
D-agent 3	7.53	5.29	0.0656
D-agent 4	6.08	4.44	0.0007
D-agent 5	5.23	3.64	0.0000

4. Conclusions and final remarks

In this paper we have shown how project risk can be managed efficiently using deep reinforcement learning. By utilising information about the progress of the different activities, the DRL-agent outperforms simpler deterministic agents with respect to stability and cost management. The study presented here, should be regarded as a preliminary proof of concept study. There are lots of rooms for improvement in the methodology, especially related to non-linear projects as well as for the structure of the trained neural networks. In an extended study we will investigate more advanced action spaces, state spaces and neural network structures.

Acknowledgement

The author is grateful to the Department of Mathematics at the University of Oslo for funding this project.

References

Bertazzi, L., R. Mogre, and N. Trichakis (2024). Dynamic project expediting: A stochastic shortest-path approach. *Management Science 70*(6), 3748–3768.

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

Cohen, I., B. Golany, and A. Shtub (2007). The stochastic time–cost tradeoff problem: A robust optimization approach. *Networks 49*(2), 175–188.

Gutjahr, W. J. and C. Strauss (2000). A stochastic branch-and-bound approach to activity crashing in pert networks. *INFORMS Journal on Computing 12*(2), 125–135.

Kang, C. and B.-C. Choi (2015). An adaptive crashing policy for stochastic time-cost tradeoff problems. *Computers & Operations Research 63*, 1–6.

Oliehoek, F. A. and C. Amato (2016). *A Concise Introduction to Decentralized POMDPs*. Springer.

Sallam, K. M., R. K. Chakrabortty, and M. J. Ryan (2021). A reinforcement learning based multi-method approach for stochastic resource constrained project scheduling problems. *Expert Systems with Applications 169*, 114479.

Sung, I., B. Choi, and P. Nielsen (2020). Reinforcement learning for resource constrained project scheduling problem with activity iterations and crashing. In *IFAC-PapersOnLine*, Volume 53, pp. 10493–10497.