Annals of Operations Research
https://doi.org/10.1007/s10479-023-05597-8

ORIGINAL RESEARCH

Shortening the project schedule: solving multimode chance-constrained critical chain buffer management using reinforcement learning

Claudio Szwarcfiter[1] · **Yale T. Herer**[2] · **Avraham Shtub**[2]

Received: 9 March 2022 / Accepted: 7 September 2023

Abstract
Critical chain buffer management (CCBM) has been extensively studied in recent years. This paper investigates a new formulation of CCBM, the multimode chance-constrained CCBM problem. A flow-based mixed-integer linear programming model is described and the chance constraints are tackled using a scenario approach. A reinforcement learning (RL)-based algorithm is proposed to solve the problem. A factorial experiment is conducted and the results of this study indicate that solving the chance-constrained problem produces shorter project durations than the traditional approach that inserts time buffers into a baseline schedule generated by solving the deterministic problem. This paper also demonstrates that our RL method produces competitive schedules compared to established benchmarks. The importance of solving the chance-constrained problem and obtaining a project buffer tailored to the desired probability of completing the project on schedule directly from the solution is highlighted. Because of its potential for generating shorter schedules with the same on-time probabilities as the traditional approach, this research can be a useful aid for decision makers.

Keywords Critical chain buffer management · Chance constraints · Project scheduling · Multimode project management · Reinforcement learning

1 Introduction

Real-life projects are fraught with uncertainties, many times leading to schedule and cost overruns. One report, based on a database of over 50,000 projects in 1000 organizations, reports that a staggering 56% of projects have cost overruns and 60% of projects are delayed (The Standish Group, 2015). More recent findings by the Project Management Institute indicate global figures of 38% and 45% for project budget and time overruns, respectively (PMI, 2021).

✉ Claudio Szwarcfiter
szwarcfiterc@hit.ac.il

1 Faculty of Industrial Engineering and Technology Management, Holon Institute of Technology, Holon, Israel

2 Faculty of Data and Decision Sciences, Technion—Israel Institute of Technology, Haifa, Israel

Published online: 20 September 2023

Critical chain buffer management (CCBM) has been accepted by the project management community as an effective tool in overcoming this problem (Ordoñez et al., 2019). Although CCBM was introduced in 1997 by Eliyahu Goldratt, the method has been extensively studied in recent years (refer to Sect. 2 for a list of recent publications). A central feature of CCBM is the utilization of time buffers to guarantee that the project deadline is met (Vanhoucke, 2016).

CCBM employs two types of time buffers: a project buffer (PB) and (usually more than one) feeding buffer (FB). To insert the buffers, we first identify the critical chain—the longest chain in the project that determines the project's duration. The concept is similar to the critical path; the main difference is that it takes into account resource constraints (Vanhoucke, 2016). The PB is inserted at the end of the project, and an FB is inserted wherever a non-critical chain merges into the critical chain.

The standard procedure for generating a schedule within the CCBM framework is to find an optimal or near-optimal baseline schedule for the problem with deterministic activity durations using an appropriate scheduling method and then apply a buffer sizing technique for the insertion of the PB and FBs. A crucial issue of CCBM is the correct sizing of the buffers; buffers that are too large may result in the loss of potentially valuable contracts (She et al., 2021) and buffers that are too short may result in nonrealistic deadlines and subsequent timeline overruns. Buffer sizing has, accordingly, received considerable attention.

Most previous work has been limited to proposing new methods for sizing the buffers and inserting them in a previously built baseline schedule. The uncertainty in the activity durations is frequently modeled with probability density functions (PDFs), and these are employed to simulate the buffered schedule and to verify, according to certain indicators, the schedule's robustness. Less attention, however, has been paid to integrating the stochastic activity durations into a chance-constrained model and sizing the buffers by directly solving this model instead of solving the deterministic model first.

Scheduling in a CCBM framework is an extension of the resource-constrained project scheduling problem (RCPSP), which is known to be NP-hard (Blazewicz et al., 1983); thus, heuristic methods are normally employed, especially in larger projects. Despite the importance of artificial intelligence (AI) based methods such as reinforcement learning (RL), which have been proved to effectively handle uncertain environments, few researchers have studied RL applications in project scheduling.

Finally, a neglected area in the field of CCBM is multimode project management, an important tool in project planning that enables the development and analysis of different baseline alternatives. It is applied in different project management areas such as construction (an example will be given in the literature review) and new product development (Solan & Shtub, 2021). In multimode projects, there are different alternatives or modes for the execution of each activity, and multimode project scheduling involves not only scheduling the activity start time, but also selecting the mode in which the activity is planned. Research on CCBM, however, has tended to focus on single-mode settings rather than on multimode CCBM projects.

This study takes a significant step toward filling in the above-mentioned gaps, focusing on solving the chance-constrained CCBM problem. We describe a mixed-integer linear programming (MILP) model for the multimode problem and propose an RL-based algorithm to solve it. We conduct experiments to prove two hypotheses:

Hypothesis H_1 Solving the chance-constrained CCBM problem generates shorter project durations compared to solving the deterministic-constrained problem and then inserting time buffers.

Hypothesis H_2 Our RL-based method is effective in the generation of CCBM schedules compared to established benchmarks.

Our results show that our technique has a clear advantage over inserting time buffers into a baseline schedule generated by the deterministic problem: for the same on-time probability, applying our model and solving the chance-constrained problem produces shorter project durations. We also confirm that our RL method produces competitive schedules compared to established benchmarks. To the best of our knowledge, this is the first time the CCBM problem has been formulated and solved with chance constraints and in a multimode setting. Our approach would lend itself well for use by project managers to improve project duration and robustness.

The rest of this paper is organized in the following way. Section 2 reviews recent literature on CCBM buffer sizing and scheduling, chance-constrained modeling and RL applications related to this paper. Section 3 describes the quantitative model. In Sect. 4, our RL solution approach is presented. Section 5 illustrates the problem with an example. Section 6 details the experimental setting. Section 7 shows our results, which are discussed in Sect. 8. Finally, in Sect. 9 we present our conclusions and lay out suggestions for future research.

2 Literature review

A detailed review of the literature on CCBM can be found in Ghaffari and Emsley (2015). Recent publications include Li et al. (2022), Aramesh et al. (2021), Chen and Hall (2020), Goto and Murray (2020), She et al. (2021), Tian et al. (2020), Zarghami et al. (2020), and Zhao et al. (2020). We focus our review on recent CCBM literature related to our research, namely CCBM buffer sizing and scheduling.

There is a considerable amount of literature on single-mode projects. Some papers utilize fuzzy numbers for modeling uncertainty. Zohrehvandi et al. (2022) introduce a fuzzy project buffer management algorithm that is a combination of the adaptive procedure with resource tightness and fuzzy failure mode and effects analysis methods. Zhang et al. (2017) employ resource constraints to size the PB, using fuzzy numbers to estimate the uncertainty of project resource usage. Zhang et al. (2015a, 2015b) calculate the PB using an uncertainty factor derived from fuzzy activity durations, resource tightness and network complexity factors. Ma et al. (2015) use fuzzy numbers to develop a probability matrix with all combinations of realized activity durations. All these papers describe buffering techniques based on first establishing a deterministic baseline schedule and then inserting the buffers.

Many studies represent activity duration uncertainty using PDFs. Poshdar et al. (2016) adopt an approximation technique for the convolution to combine activity-level probability density functions and model variability at the project level. Zhao et al. (2020) employ classical methods for sizing the FBs and propose a two-stage rescheduling approach for solving the resulting resource and precedence conflicts and avoiding critical chain breakdown and non-critical chain overflow. Ghoddousi et al. (2016) develop a multi-attribute buffer sizing method as an extension to the traditional root square error method (RSEM; Newbold, 1998). Bevilacqua et al. (2015) employ goal programming to minimize duration and variations in resource loads and insert the PB and FBs using RSEM. Ghaffari and Emsley (2016) show that allowing some multitasking—thus releasing resource capacity—can reduce buffer sizes. Hu et al. (2017) consider a modified CCBM approach in which an activity never starts before its planned time and that has two categories of resources: regular resources, available until a cut-off date, and irregular emergency resources, available afterwards. Peng et al. (2023) propose

an integrated proactive–reactive scheduling algorithm, employing priority rules for building the baseline schedule, RSEM for sizing the buffers, and adaptive dynamic programming for project monitoring and control.

Hu et al. (2016) offer a new project schedule monitoring framework. Their schedule is built using a branch-and-bound algorithm and the buffers are calculated using RSEM. Salama et al. (2018) combine location-based management with CCBM in repetitive construction projects, introducing a new resource conflict buffer. Zhang et al. (2018) calculate the PB using the duration rate and network complexity of each project phase, and monitor the buffer dynamically for each phase. Zarghami et al. (2020) develop a buffer sizing method based on the probability of the availability of resources assigned to project activities. Li et al. (2022) use machine learning algorithms to build a function that maps project features that affect the PB to the buffer size and evaluate their method using six performance indicators. She et al. (2021) decompose the project network and calculate each FB based on the uncertainties of all non-critical chains feeding it. Using the FBs, the PB is sized using the safety margins of critical tasks and remaining safety margins of feeding chains. Hoel and Taylor (1999) propose the use of Monte Carlo simulation to determine the cumulative probability distribution for the project completion time, and thus, the size of the PB (this method is also adopted in Project Management Institute, 2017; Tenera, 2008; Tysiak, 2017).[1]

As in the literature that employs fuzzy numbers, these approaches that use PDFs do not formulate and solve the chance-constrained problem; rather, the buffers are introduced into a previously defined baseline schedule.

Other researchers use information flow between project activities. Zhang et al. (2015a, 2015b) propose optimal sequencing of the critical chain activities based on the information flow to and from activities, aiming to minimize the total coordination cost and to reduce duration fluctuation and buffering. Zhang et al. (2016a, 2016b) use two factors to calculate the PB by means of a design structure matrix (DSM): physical resource tightness and information flow between activities. Information flows are also used in Zhang et al. (2016a, 2016b), whose work is extended in Ma et al. (2019a, 2019b). The schedules are developed taking into account rework risks and a rework buffer is proposed. The activity durations are represented by three-point estimates and beta distributions are used for the simulation runs.

In our research, we represented uncertainty in activity durations as three-point estimates as opposed to fuzzy numbers. This approach is common in the academic literature and is a standard in the Project Management Body of Knowledge (PMBOK) guide (Project Management Institute, 2017); thus, project managers are more familiar with estimating optimistic, most likely and pessimistic activity durations than other forms of uncertainty representation.[2]

Some works employ the free float as natural FBs (Chen et al., 2020; Hoel & Taylor, 1999; Peng & Huang, 2014). The activities are set to their earliest possible start and the FB is the free float of the activity that merges into the critical chain and, consequently, no new resource conflicts are created by the insertion of the FBs.

Few studies have been published on CCBM buffer sizing and scheduling with multimode projects. Tian and Demeulemeester (2014) compare CCBM schedules where activities start as soon as their predecessors finish with schedules in which activities never start earlier than planned and using work content in terms of resource-time units to generate activity modes.

[1] The PMBOK guide, Section 11.4.3.1, applies the cumulative probability distribution for determining the "amount of contingency reserve needed to provide a specified level of confidence.".

[2] The same can be said about robust optimization; the methods for representing it are not part of the PMBOK guide, and the modeling of uncertainty is less familiar to project managers.

Peng et al. (2015) combine mode selection rules and activity priority rules for scheduling multimode projects within a CCBM framework. Ma et al. (2014) add five metrics to the RSEM buffer calculation formula and employ three modes—urgent, normal and deferred—to perform multiple resource leveling. Moradi and Shadrokh (2019) investigate multimode projects with uncertain activity durations. To meet a given deadline, two options are considered, changing the mode of activities and hiring extra resources. A new resource buffer is proposed, allocating idle time for the resources on a critical chain to ensure their availability when needed. As in the case of single-mode research, chance-constrained models are not dealt with in multimode research.

Buffer management is also treated in contexts other than CCBM. Time buffers, or schedule reserves, are widely used in the 6th edition of the PMBOK guide to account for schedule uncertainty (Project Management Institute, 2017). Ma et al. (2019a, 2019b) evaluate new surrogate robustness measures using simulation with stochastic resource availabilities and activity durations, and analyze different buffering strategies. In Ning et al. (2018), time buffering seeking the minimization of the maximum cash flow gap for the contractor is investigated. Zheng et al. (2018) investigate the problem of maximizing the Net Present Value (NPV) with stochastic activity durations, employing two known time buffering methods and two reactive scheduling models. Davari and Demeulemeester (2019b) employ buffers after the completion times of the activities and examine two types of reactions to schedule disruptions. Torabi Yeganeh and Zegordi (2020) determine the schedule and distribute the buffers using a multi-objective model that employs the criticality index. Li and Demeulemeester (2016) build a buffered baseline schedule by minimizing deviations of resource utilizations and activity starting times. Bakry et al. (2016) use fuzzy numbers and the user's level of confidence for scheduling and buffer calculation. Ghoddousi et al. (2017) seek to minimize the duration and to maximize the aggregate free float for buffer insertion. The above papers consider single mode projects. In contrast, Wichmann et al. (2019) minimize the fuzzy overlap as a measure of robustness and examine the multimode resource-constrained discrete time/cost trade-off problem with fuzzy numbers for activity durations and buffer allocation.

Chance constraints have been applied in some related works aiming to create robust schedules. Here we review the main literature where the activity durations are stochastic and represented by probability distributions. Bruni et al. (2011, 2015) propose a two-part framework, first employing a stochastic dynamic version of the parallel schedule generation scheme using a set of priority rules and then solving a chance-constrained problem minimizing the makespan of the partial schedule generated up to the last decision point. Lamas and Demeulemeester (2016) introduce a formulation aiming to minimize the project makespan subject to a confidence level, defined as the joint probability that each activity starts exactly at its baseline starting time. A sample average approximation is adopted, where a sample is a set of realizations of the random activity durations vector. This formulation is used in subsequent works (Davari & Demeulemeester, 2019a; Tao et al., 2018; Zhou et al., 2022). In some works, the chance constraints express the project duration with a determined confidence level (Rahman et al., 2021a, 2021b); this is the concept that we adopt in the current paper. In some formulations, the chance constraints are converted into a deterministic counterpart (Chakrabortty et al., 2017; Sallam et al., 2021; Wang et al., 2015). The above works on chance-constrained formulation do not focus on buffer management methodologies, including CCBM.

In our research, we focus on buffer allocation under the CCBM framework; as was pointed out in the introduction, CCBM has been accepted as an effective tool to help project managers achieve their time goals, and although the method was introduced in 1997, it has been extensively studied in recent years.

From learning to play backgammon at near the level of the world's best players (Barto, 2019), through landing unmanned aerial vehicles (UAVs) (Polvara et al., 2019), beating the highest ranked players in Jeopardy! (Ferrucci et al., 2013), and human-level performance in Atari games (Mnih et al., 2015), RL has been successful in applications with uncertain environments. Project scheduling can be considered one such environment.

Wauters et al. (2011) use RL-based multi-agent algorithms on deterministic multimode projects, employing learning automata to select the activity list and modes. In a second paper, the authors apply the method to multi-project environments (Wauters et al., 2015). Another multi-agent RL method, the A-team approach, is applied to deterministic single-mode projects in Jędrzejowicz and Ratajczak-Ropel (2014) and extended to multi-mode projects in Jędrzejowicz and Ratajczak-Ropel (2015). A multi-project environment, working on a cloud manufacturing platform, is also tackled in Chen et al. (2019), where activity durations are deterministic and projects arrive randomly. Manufacturing tasks (MTs) are mapped to manufacturing services (MSs) and the authors employ an RL-based assigning policy to minimize the duration and logistic distance between MTs and MSs. Sallam et al. (2021) apply RL to select the most suitable algorithm between multi-operator differential evolution algorithms and discrete cuckoo search algorithms in a chance-constrained single-mode setting.

From this literature review, we can identify the gap in previous research that we address in this paper. None of the CCBM papers cited above employ chance-constrained models to schedule the projects and size the buffers. Moreover, research on multimode CCBM is extremely limited; thus, there are not many tools available to project managers. Finally, as far as we know, no other authors have applied RL to CCBM.

3 Quantitative model

Traditionally, CCBM scheduling starts with finding a minimum-duration baseline schedule that takes into consideration deterministic estimates for activity durations such as the median or mean (Herroelen et al., 2002). This means solving the resource-constrained project scheduling problem (RCPSP) or its multimode extension (MRCPSP). Since the problem is known to be NP-hard (Blazewicz et al., 1983), heuristic methods are normally employed, especially in larger projects.

After finding a baseline schedule, the PB and FBs are inserted according to a buffer-sizing technique, such as the more recent ones that we mentioned in the literature review. The aim is to create a stable schedule that can be evaluated by applying one or more of the robustness measures described in some works that we have cited—for example: standard deviation of the project length, stability cost and timely project completion probability (Tian & Demeulemeester, 2014).

On-time probability is not only an indicator of schedule robustness but can also be used for buffer calculation. In Sect. 2, we cited works that propose the use of Monte Carlo simulation to calculate the size of the PB. For instance, suppose we desire a 95% probability of completing the project on schedule: the PB is the difference between the 95% project duration and the baseline schedule duration.

Let us go one step further. If we search directly for the shortest time-buffered schedule that gives us the desired on-time probability, we open up a path to identifying a shorter schedule with this same probability. This is our chance-constrained CCBM problem—by contemplating multimode projects, we not only search for the schedule but also the activity

Table 1 Model notation

Parameters	
J	Number of project activities
K	Number of types of renewable resources
M_j	Number of modes for activity j
$\mathcal{R}^k$	Total availability of resource k
d_{jms}	Duration of activity j executed in mode m in scenario s
r^k_{jm}	Units of resource k needed for activity j executed in mode m
ES_{js}	Earliest start for activity j in scenario s
LS_{js}	Latest start for activity j in scenario s
β	Desired probability of completing the project on schedule
θ	Upper bound for delivery date overrun
S	Number of scenarios
Sets	
$\mathcal{P}(j)$	Set of immediate predecessors that precede activity j
Decision variables	
D	Project delivery
δ_{jm}	Binary variable indicating (value 1) if activity j is carried out in mode m
t_{js}	Starting time of activity j, $j = 0, ..., J+1$, for scenario s, where activities 0 and $J+1$ are dummy activities with a single mode, no duration and resources, and represent the start and end of the project, respectively
z_{ij}	Binary variable indicating (value 1) if activity j starts after activity i finishes
ϕ^k_{ij}	Flow variable modeling the amount of resource k transferred from activity i to activity j
τ_s	Binary variable indicating (value 1) whether, in scenario s, the project finishes by the delivery date

modes that provide the shortest project duration with the desired on-time probability. This duration includes the baseline schedule with the nominal (deterministic) activity durations and the PB. In this paper we define project delivery as this time-buffered project duration, i.e., the deadline that we can meet (deliver the project) with the desired on-time probability.

We model our chance-constrained CCBM problem as a MILP using the flow-based formulation described in Artigues et al. (2015), extending it to consider multimode projects. Additionally, we tackle the chance constraints using a scenario approach (SA), introduced in Calafiore and Campi (2005) and applied in recent project scheduling papers (Gutjahr, 2015; Sallam et al., 2021; Tian et al., 2019).[3] The idea is to take S samples, or scenarios, of the realization of the random variables in the constraints—in our case, the activity durations—and substitute the deterministic scenario constraints for the stochastic chance constraints. In Table 1 our notation is presented.

We now introduce the model, followed by an explanation of the objective function and constraints.

$$\text{Min } D, \tag{1}$$

[3] There are similar formulations for the RCPSP, but this particular combination is novel.

subject to:

$$t_{J+1,s} - \theta(1-\tau_s) \le D, \quad \forall s = 1,\ldots,S, \tag{2}$$

$$\sum_{s=1}^{S} \tau_s \ge \beta S, \tag{3}$$

$$z_{ij} + z_{ji} \le 1, \quad \forall i = 0,\ldots,J,\ \forall j = 1,\ldots,J+1,\ \forall i < j, \tag{4}$$

$$z_{ij} + z_{jh} - z_{ih} \le 1, \quad \forall i,j,h = 0,\ldots,J+1, \forall i \ne j \ne h, \tag{5}$$

$$z_{ij} = 1, \quad \forall i \in \mathcal{P}(j),\ \forall j = 1,\ldots,J+1, \tag{6}$$

$$t_{js} - t_{is} - M z_{ij} \ge \sum_{m=1}^{M_i} \delta_{im} d_{ims} - M, \quad \forall i,j = 0,\ldots,J+1,\ \forall i \ne j, \forall s = 1,\ldots S, \tag{7}$$

$$ES_{js} \le t_{js} \le LS_{js}, \quad \forall j = 0,\ldots,J+1,\ \forall s = 1,\ldots,S, \tag{8}$$

$$\phi_{ij}^k - \min\left(\tilde{r}_{im}^k, \tilde{r}_{jm'}^k\right) z_{ij} - \left(1-\delta_{im}\right)\left(\tilde{r}_{ij}^{\max,k} - \min\left(\tilde{r}_{im}^k, \tilde{r}_{jm'}^k\right)\right)$$

$$- \left(1-\delta_{jm'}\right)\left(\tilde{r}_{ij}^{\max,k} - \min\left(\tilde{r}_{im}^k, \tilde{r}_{jm'}^k\right)\right) \le 0, \text{ where } r_{ij}^{\max,k} = \max\left(\max_{m=1,\ldots,M_i} \tilde{r}_{im}^k, \max_{m'=1,\ldots,M_j} \tilde{r}_{jm'}^k\right),$$

$$\forall i = 0,\ldots,J,\ \forall j = 1,\ldots,J+1,\ \forall i \ne j,\quad \forall k = 1,\ldots,K,\ \forall m = 1,\ldots,M_i,\ \forall m' = 1,\ldots,M_j,$$

$$\text{where } \tilde{r}_{jm}^k = \begin{cases} r_{jm}^k & \text{if } 0 < j < n+1 \\ R^k & \text{if } j = 0 \text{ or } j = n+1, \end{cases} \tag{9}$$

$$\sum_{m=1}^{M_j} \delta_{jm} = 1, \quad \forall j = 0,\ldots,J+1, \tag{10}$$

$$\sum_{j \in \{1,\ldots,J+1\}\setminus\{i\}} \phi_{ij}^k = \sum_{m=1}^{M_i} \tilde{r}_{im}^k \delta_{im}, \quad \forall i = 0,\ldots,J,\ \forall k = 1,\ldots,K, \tag{11}$$

$$\sum_{i \in \{0,\ldots,J\}\setminus\{j\}} \phi_{ij}^k = \sum_{m=1}^{M_j} \tilde{r}_{jm}^k \delta_{jm}, \quad \forall j = 1,\ldots,J+1, \forall k = 1,\ldots,K, \tag{12}$$

$$0 \le \phi_{ij}^k \le \min\left(\max_{m=1,\ldots,M_i} \tilde{r}_{im}^k, \max_{m=1,\ldots,M_j} \tilde{r}_{jm}^k\right), \quad \forall i = 0,\ldots,J,\ \forall j = 1,\ldots,J+1,\ \forall i \ne j,\ \forall k = 1,\ldots,K. \tag{13}$$

The objective function (1) aims to minimize the project delivery time. Constraints (2) indicate whether a scenario finishes on time. Constraint (3) counts the fraction of scenarios that finish on time and forces this fraction to remain above the predetermined threshold. Constraints (4) and (5) avoid cycles of 2 and 3 or greater, respectively. Constraints (6) enforce the precedence constraints. Constraints (7) link the continuous activity start time variables with the binary sequencing variables. Constraints (8) give upper and lower bounds for the activity start times. Constraints (9), from Balouka and Cohen (2019), connect the continuous resource flow variables with the binary sequencing variables and the binary mode variables. Constraints (10) force the selection of only one mode per activity. Outflow constraints (11) ensure that all activities, except for $J+1$, send their resources to other activities. Inflow constraints (12) ensure that all activities, except for activity 0, receive their resources from other activities. Constraints (13) bound the flow variables by the maximum resource consumption modes.

Once the MILP is solved, a straightforward way to construct a schedule is as follows. Representing the project as an activity-on-node (AON) network, we have arcs connecting

activities j to their immediate predecessors $\mathcal{P}(j)$. If resources are flowing from activity i to j, j cannot start before i finishes; it is as if i is an immediate predecessor of j. Therefore, we add arcs between activities wherever $\phi_{ij}^k > 0$, constructing what some authors call resource flow networks (Lambrechts et al., 2011); this can be done by simply adding i to $\mathcal{P}(j)$. Now, all we have to do is create an earliest start schedule, where each activity starts when its immediate predecessor finishes.

The activity duration we employ in the baseline schedule is the most likely duration of a three-point estimate. After the conclusion of the last activity, we insert the PB, calculated as the difference between the delivery and the baseline duration. For an illustration, see the example section.

For the insertion of FBs, we adopt the method proposed in some works cited in Sect. 2 (e.g., Chen et al., 2020; Peng & Huang, 2014), where the activities are scheduled at their early start dates and the FBs correspond to the free float of the activity that merges into the critical chain. Thus, since we are starting all activities as early as possible, we can ignore FB sizing in our problem.

4 Reinforcement learning solution approach

The success of RL in applications with uncertain environments, pointed out in Sect. 2, is the factor motivating our application of RL to learning the activity modes and project schedules that minimize delivery in a stochastic environment. To the best of our knowledge, problems involving a combination of multimode projects and chance constraints have yet to be tackled using RL. Additionally, as far as we are aware, this is the first time that Monte Carlo control, the specific RL method used in this paper, has been applied to project scheduling.

Our RL model starts with an agent in a state $\mathcal{S}$. The agent undertakes action $\mathcal{A}$ and moves to state $\mathcal{S}'$, receiving reward $\mathcal{R}'$. It then executes action $\mathcal{A}'$, moving to state $\mathcal{S}''$, and receiving reward $\mathcal{R}''$, and so on. We can thus represent the agent's life trajectory as $\mathcal{S}, \mathcal{A}, \mathcal{R}', \mathcal{S}', \mathcal{A}', \mathcal{R}'', \mathcal{S}'', \mathcal{A}'', \mathcal{R}''', \mathcal{S}''', \mathcal{A}'''$, etc. The agent follows a policy $\pi(\mathcal{S}, \mathcal{A})$ that tells it at each state which action it should take. The RL problem's objective is to learn a policy that maximizes the agent's reward. We further define an action-value function $q(\mathcal{S}, \mathcal{A})$ as the estimated reward for taking action $\mathcal{A}$ on state $\mathcal{S}$ and thereafter, following policy $\pi(\mathcal{S}, \mathcal{A})$.

Applying the RL model to our problem, we define a state as project activity j. The agent undertakes an action by choosing a mode $\hat{m}_j$ and a start-time action (STA) $\hat{t}_j$ for activity j and then moves on to the next activity. After selecting modes and STAs for all activities $j = 1, ..., J$, it can calculate its reward $\mathcal{R}(j, m, t)$. As it receives rewards, it learns the action-value function $q(j, m, t)$ and which policy $\pi(j, m, t)$ to follow.

The RL method that we apply in this paper is Monte Carlo control (MCC), based on Sutton and Barto (2018). It makes full use of Monte Carlo simulation to determine the cumulative probability distributions for project completion times, used to calculate the rewards. We now present the pseudocode and an explanation of our MCC method. Table 2 summarizes our RL notation, in addition to the notation employed in the quantitative model.

The main procedure for MCC appears below (Algorithm 1). We then explain and present the pseudocode for each function shown in the main procedure.

Table 2 Additional notation for the RL method

π	ε-greedy policy, decision-making rule
$q(j,m,t)$	Value of choosing mode m and STA t for activity j under ε-greedy policy π
$\pi(j,m,t)$	Probability of choosing mode m and STA t for activity j under ε-greedy policy π
$\mathcal{R}(j,m,t)$	Reward for choosing mode m and STA t for activity j under ε-greedy policy π
ε	Probability of random action in ε-greedy policy
$\hat{m}_j$ or $\hat{m}$	Chosen mode for activity j; when $\hat{m}_j$ should appear as a subscript, we use $\hat{m}$ instead to avoid nesting subscripts
$\hat{t}_j$ or $\hat{t}$	Chosen STA for activity j; when $\hat{t}_j$ should appear as a subscript, we use $\hat{t}$ instead to avoid nesting subscripts
$d_{j\hat{m}}^{ML}$	Most likely duration of activity j in chosen mode $\hat{m}$
$r_{j\hat{m}}^{k}$	Quantity of resources of type k needed to execute activity j in chosen mode $\hat{m}$
$A(j)$	Set of activities executed in parallel to activity j
$N_{\hat{m}\hat{t}}$	Number of times mode $\hat{m}_j$ and STA $\hat{t}_j$ are chosen for activity j
α	Step-size parameter

Algorithm 1 Main procedure for MCC

initialize_action_values$\left(J, M_j, LS_j, \forall j = 1,...,J\right)$

while not stopping criterion:

calculate_policy$\left(J, M_j, q\left(j,m,t\right), \forall j = 1,...,J\right)$

choose_mode_start$\left(\pi\left(j,m,t\right), d_{j\hat{m}}^{ML}, \mathcal{P}\left(j\right), \forall j = 1,...,J\right)$

calculate_reward$\left(\text{sorted}\left(\hat{m}_j\right), \mathcal{P}\left(j\right), d_{j\hat{m}}^{ML}, r_{j\hat{m}}^{k}, R^k, \forall j = 1,...,J, \forall k = 1,...,K\right)$

update_action_values_RL$_1\left(J, \hat{m}_j, \hat{t}_j, \forall j = 1,...,J\right)$

or update_action_values_RL$_2\left(J, \hat{m}_j, \hat{t}_j, \forall j = 1,...,J\right)$

Our algorithm starts with the initialization of the action-value list (Algorithm 2). For each activity, the action taken is to choose the mode and the STA. We use ten STAs, equally spaced from zero to the latest start of the activity. These STAs are not the actual activity start times, rather they are RL actions in which the agent places the activity in a time position that will determine its sequencing vis-à-vis the other activities (this procedure is explained in Algorithm 4). Modeling actions as a selection of STAs is really a surrogate for choosing different combinations of precedencies among the activities. The range of possible STAs between zero and the upper bound offers ample possibilities of early start or postponement for each activity, providing a richer search space with the possibility of better solutions. We initialize the table with artificially high values, a technique known as optimistic initial values, in order to allow initial exploration of all actions.

The action-value list is then used to calculate the policy (Algorithm 3). To balance exploration and exploitation, we adopt an ε-greedy policy, meaning that in the policy table, we ascribe a probability ε of taking a random action and a probability $(1-\varepsilon)$ of taking a greedy action, i.e., the action with the highest action value.

Algorithm 2 Initialization of the action-value list

def initialize_action_values $\left(J, M_j, LS_j, \forall j = 1,...,J\right)$:

for activity $j = 1,...,J$:

for mode $m = 1,...,M_j$:

for start time $t = 0, \frac{LS_j}{9}, \frac{2LS_j}{9}, ..., LS_j$:

$$q\left(j,m,t\right) = \text{ large number}$$

return $q\left(j,m,t\right), \forall j = 1,...J, \forall m = 1,...,M_j, \forall t = 0, \frac{LS_j}{9}, \frac{2LS_j}{9}, ..., LS_j$

Algorithm 3 Policy calculation

def calculate_policy $\left(J, M_j, q\left(j,m,t\right), \forall j = 1,...,J\right)$:

for activity $j = 1,...,J$:

$$q^* = \max_{m,t} q\left(j,m,t\right)$$

$$x = \text{number of action-values for which } q\left(j,m,t\right) = q^*$$

$$\pi\left(j,m,t\right) = \begin{cases} \left(\frac{1}{x}\right)\left(1 - \frac{\varepsilon}{10M_j}\left(10M_j - x\right)\right), \forall m,t \middle| q\left(j,m,t\right) = q^* \\ \frac{\varepsilon}{10M_j} \forall m,t \middle| q\left(j,m,t\right) \neq q^* \end{cases}$$

return $\pi\left(j,m,t\right), \forall j = 1,...,J, \forall m = 1,...,M_j, \forall t = 0, \frac{LS_j}{9}, \frac{2LS_j}{9}, ..., LS_j$

Next, we take an action based on the policy (Algorithm 4), choosing for each activity the mode and STA according to the probabilities in the policy table. Then, by right-shifting the activities, adding to each start time the maximum finish times of its immediate predecessors, we adjust the start times to make them precedence-feasible. This means that if we choose an STA of $\hat{t}_j$ for activity j, we right-shift this activity to start at time $\hat{t}_j$ after its immediate predecessor finishes. The finish times are determined using the most likely duration of each activity in its chosen mode. Having done all this, we sort all activities according to their adjusted start time, obtaining a precedence-feasible activity list with the activities and their chosen modes.

Algorithm 4 Mode and start-time action selection

def choose_mode_start $\left(\pi(j,m,t), d_{j\hat{m}}^{ML}, \mathcal{P}(j), \forall j = 1,...,J\right)$:

for activity $j = 1,...,J$:

choose $\hat{m}_j, \hat{t}_j$ according to $\pi(j,m,t)$

if $\mathcal{P}(j) = \varnothing$:

$\hat{t}_j^* = \hat{t}_j$

else:

$\hat{t}_j^* = \hat{t}_j + \max\left(\hat{t}_i^* + d_{i\hat{m}}^{ML}, \forall\, i \in \mathcal{P}(j)\right)$

return sorted $\left(\hat{m}_j \,\middle|\, j \in \{1,...,J\}, \hat{m}_j \in \{1,...M_j\},\ \text{key} = \hat{t}_j^*\right)$

Note that the adjustment to generating only precedence-feasible activity lists avoids both wasting runtime with infeasible solutions and discarding potentially good solutions.

The next step in the algorithm is to calculate the reward for the actions taken (Algorithm 5). First, the early-start baseline schedule is determined by sequentially taking each activity from the list and scheduling it at the earliest possible time. We make this the finish time of the most immediate predecessor, and if there are not enough resources, we repeatedly right-shift the activity to the next scheduled activity finish time until there are enough resources. With the schedule now in place, we calculate the project delivery date. For example, if the decision makers desire a 95% probability of completing the project on schedule, the baseline schedule is simulated, say, 1000 times, the realized finish times are sorted, and the 950$^{\text{th}}$ element of the finish time list is the delivery date. Since in our problem we are interested in minimizing the delivery date, to maintain the RL idea of maximizing the reward, we define the reward as $1/D$.

The last step in the algorithm is to update the action-value list using the reward. We can choose from two update methods, RL_1 (Algorithm 6) and RL_2 (Algorithm 7). For both, we only update the action values corresponding to the chosen modes and STAs. RL_1 learns an action value by averaging all the rewards this action (mode and STA) has received each time it was taken. Doing this implies that the more actions that are taken, the smaller the impact of new rewards. The means are calculated incrementally to speed up the process and save memory. RL_2 updates the action values using a formula very similar to the incremental mean from RL_1, but instead of using the decreasing step $1/N_{\hat{m}\hat{t}}$, it uses a constant step α, giving an exponentially large weight to the last action.

Algorithm 5 Reward calculation

def calculate_reward $\left(\text{sorted}\left(\hat{m}_j\right), \mathcal{P}(j), d_{j\hat{m}}^{ML}, r_{j\hat{m}}^k, R^k, \forall j=1,...,J, \forall k=1,...,K\right)$:

$\hat{t}_j^*\left(\text{1st activity mode in sorted}\left(\hat{m}_j\right)\right)=0$

for activity mode $\hat{m}_j$ in sorted$\left(\hat{m}_j\right)$ except 1st activity mode:

$$\hat{t}_j^* = \min\left(t_j \middle| t_j \geq \hat{t}_i^* + d_{i\hat{m}}^{ML}, \forall i \in \mathcal{P}(j) \text{ and } r_{j\hat{m}}^k \leq R_{\text{surplus}}^k, \forall k=1,...,K\right),$$

$$\text{where } R_{\text{surplus}}^k = \min_{\left[t_j, t_j + d_{j\hat{m}}^{ML}\right)} \left(R^k - \sum_{i \in A(j)} r_{i\hat{m}}^k\right)$$

return $\mathcal{R}\left(j, \hat{m}_j, \hat{t}_j\right) = 1/D \left| \Pr\left[t_{J+1}^* \leq D\right] \geq \beta, \forall j=1,...,J\right.$

Algorithm 6 Action-values update using average rewards

def update_action_values_RL$_1$ $\left(J, \hat{m}_j, \hat{t}_j, \forall j=1,...,J\right)$:

for activity $j=1,...,J$:

$$q\left(j, \hat{m}_j, \hat{t}_j\right) = q\left(j, \hat{m}_j, \hat{t}_j\right) + \frac{1}{N_{ij}}\left(\mathcal{R}\left(j, \hat{m}_j, \hat{t}_j\right) - q\left(j, \hat{m}_j, \hat{t}_j\right)\right)$$

return $q\left(j, \hat{m}_j, \hat{t}_j\right), \forall j=1,...,J$

We repeat the process, calculating the policy based on the updated action values, choosing the modes and STAs based on the policy, calculating the reward and updating the action values, until a stopping criterion. The solution takes the form of a baseline schedule consisting of the chosen activity modes and start times, the PB and the delivery date.

Algorithm 7 Action-values update using constant step

def update_action_values_RL$_2$ $\left(J, \hat{m}_j, \hat{t}_j, \forall j=1,...,J\right)$:

for activity $j=1,...,J$:

$$q\left(j, \hat{m}_j, \hat{t}_j\right) = q\left(j, \hat{m}_j, \hat{t}_j\right) + \alpha\left(\mathcal{R}\left(j, \hat{m}_j, \hat{t}_j\right) - q\left(j, \hat{m}_j, \hat{t}_j\right)\right)$$

return $q\left(j, \hat{m}_j, \hat{t}_j\right), \forall j=1,...,J$

5 Example: a radar development project

To illustrate our multimode chance-constrained CCBM problem and RL solution approach, let us consider a small five-activity radar development project. The project network diagram is shown in Fig. 1. Each activity can be executed in two modes. There are two types of resources, engineers and technicians, and the duration of each activity mode is represented

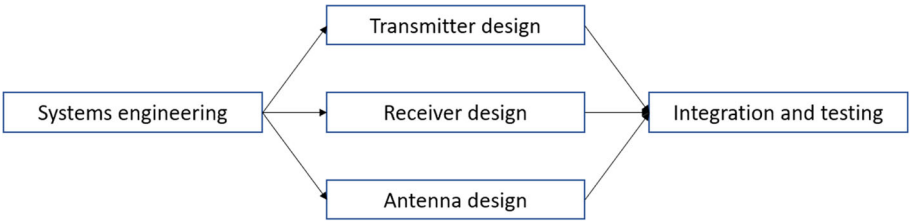


Fig. 1 Project network diagram

Table 3 Summary of data for radar development activity modes

Activity	Mode	Duration			Resources	
		O	ML	P	Engineers	Technicians
1. Sys eng	1. Small team	5	7	10	1	0
	2. Large team	3	4	4	3	1
2. T design	1. Reengineer	3	5	8	4	2
	2. New design	7	9	11	2	1
3. R design	1. Reengineer	3	5	9	3	1
	2. New design	8	10	11	2	1
4. A design	1. Reengineer	3	7	9	5	2
	2. New design	6	7	9	3	2
5. I & T	1. In-house	3	4	4	1	0
	2. Subcontract	2	2	5	3	3

by a three-point estimate: optimistic (O), most likely (ML) and pessimistic (P). The data for the activity modes is summarized in Table 3.

There are a total of 11 engineers and four technicians available. We want to find the shortest project delivery with a probability β of finishing on time.

Note that because this is a resource-constrained problem, we would not be able to simply enumerate early-start schedules; for example, if we selected mode 1 for activities 2, 3, and 4, they could not be scheduled to start in parallel because of an insufficient number of engineers and technicians.

We now demonstrate our RL approach by running one iteration of Algorithm 1 using the RL_1 action-values update method (Algorithm 6). Since it is a small example project, we use three STAs.

First, we initialize the action-value table with optimistic initial values (Algorithm 2). We use unit values for all initial action values (Table 4).

Next, we calculate the policy according to the action-value table (Algorithm 3). We use the probability of random action $\varepsilon = 0.1$. In this first iteration, since all action values are the same, all action values are ascribed the same probability (Table 5).

We set a latest start upper bound for each activity considering all activities sequentially (none of them in parallel), using the pessimistic durations and longest modes, and the current activity as the last one. Table 6 shows the possible STAs for each activity from zero to the latest start.

Table 4 Initial action values for the radar example project

j	$m=1,t=0$	$m=1,t=LS_j/2$	$m=1,t=LS_j$	$m=2,t=0$	$m=2,t=LS_j/2$	$m=2,t=LS_j$
1	1	1	1	1	1	1
2	1	1	1	1	1	1
3	1	1	1	1	1	1
4	1	1	1	1	1	1
5	1	1	1	1	1	1

Table 5 Initial policy for the radar example project

j	$m=1,t=0$	$m=1,t=LS_j/2$	$m=1,t=LS_j$	$m=2,t=0$	$m=2,t=LS_j/2$	$m=2,t=LS_j$
1	0.1666	0.1666	0.1666	0.1666	0.1666	0.1666
2	0.1666	0.1666	0.1666	0.1666	0.1666	0.1666
3	0.1666	0.1666	0.1666	0.1666	0.1666	0.1666
4	0.1666	0.1666	0.1666	0.1666	0.1666	0.1666
5	0.1666	0.1666	0.1666	0.1666	0.1666	0.1666

Table 6 Possible start-time actions for the radar example project

j	$t=0$	$t=LS_j/2$	$t=LS_j$
1	0	18	36
2	0	17.5	35
3	0	17.5	35
4	0	18.5	37
5	0	20.5	41

The agent now takes an action based on the policy, choosing the modes and STAs (Algorithm 4). The selected actions and adjusted start times (following Algorithm 4) are shown in Table 7. Note that these are not the actual start times; rather, they are used to generate the activity list according to the time sequence of the adjusted start times. The resulting activity list is shown in Table 8.

Table 7 Chosen modes and start times for the radar example project in the first iteration

j	1	2	3	4	5
$\hat{m}_j, \hat{t}_j$	2, 0	2, 35	2, 35	1, 18.5	1, 20.5
$\hat{t}_j^*$	0	39	39	22.5	49

Table 8 Activity-mode list for the radar example project in the first iteration

j	1	4	2	3	5
$\hat{m}_j$	2	1	2	2	1

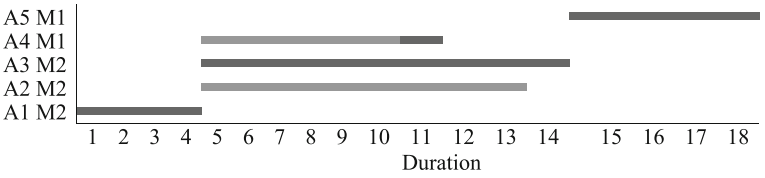
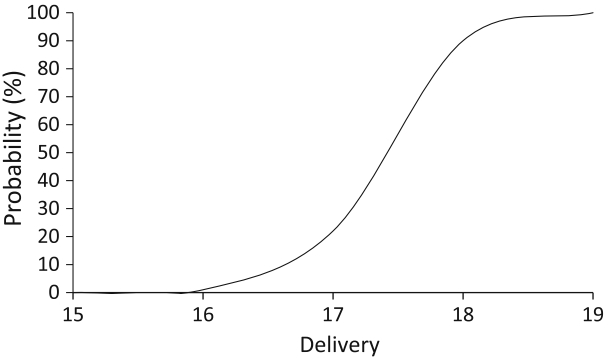


Fig. 2 Gantt chart for the radar example project in the first iteration

Fig. 3 Simulation runs for the radar example project in the first iteration



Now, we calculate the agent's reward (Algorithm 5). First, we use the activity-mode list to build the baseline schedule, shown in Fig. 2. Then, the chance constraints are enforced: if the decision makers set $\beta = 90\%$, we simulate 1000 project runs and obtain the cumulative probability data displayed in Fig. 3. For 90% probability, the project delivery is 18, meaning that there is a probability of 90% to deliver the project in up to 18 time periods. Thus, the reward $\mathcal{R}\left(j, \hat{m}_j, \hat{t}_j\right) = 1/18 \approx 0.0555$.

Finally, we update the action values with the new rewards (Algorithm 6). In this first iteration, the reward substitutes for the artificial optimistic initial values. In subsequent iterations, each time the same action is taken, the new reward is averaged with all the previous ones, and this is how the agent learns the action values. The updated action values are shown in Table 9.

The next iteration would start with calculating the new policy (Algorithm 3). With $\varepsilon = 0.1$, we would choose a greedy action with a probability of 90% (at this stage this would mean choose one of the actions with the optimistic initial values randomly; in later iterations, there will usually be one action associated with the highest action value) and a random action with a probability of 10%.

After running the algorithm, Fig. 4 shows three Gantt charts with the activities (A), selected modes (M), FBs and a PB for solutions with $\beta = 90\%$, 95% and 100%, respectively. The

Table 9 Updated action values for the radar example project

j	$m=1,t=0$	$m=1,t=LS_j/2$	$m=1,t=LS_j$	$m=2,t=0$	$m=2,t=LS_j/2$	$m=2,t=LS_j$
1	1	1	1	0.0555	1	1
2	1	1	1	1	1	0.0555
3	1	1	1	1	1	0.0555
4	1	0.0555	1	1	1	1
5	1	0.0555	1	1	1	1

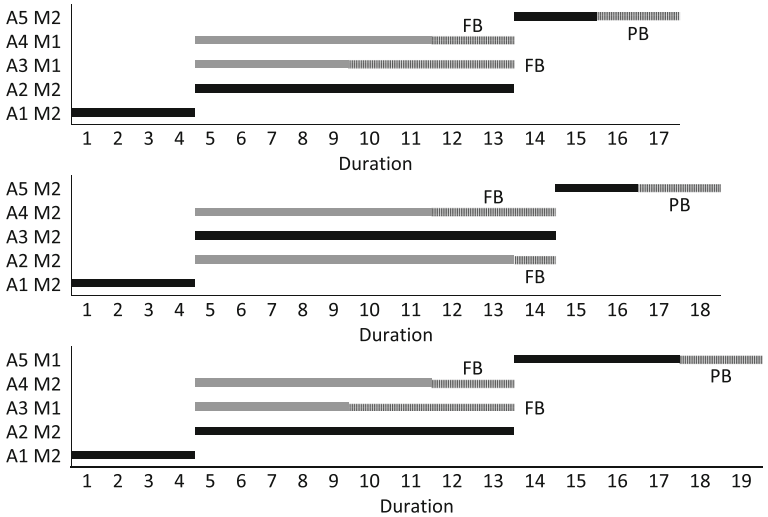


Fig. 4 Radar example Gantt charts for 90%, 95% and 100% on-time probabilities, respectively

baseline schedule activity durations are the most likely durations from Table 3. The critical chain activities are shown in black.

From the Gantt charts, we learn that the project plan, as each activity mode is selected, changes according to the risk that we are willing to accept. This contrasts with the traditional method, described in the quantitative model section, where first a mode selection that minimizes the deterministic duration is found, then a buffering method is adopted.

We reached the same delivery dates using RL_1 and RL_2, and by solving the MILP from the quantitative model section. The number of MILP scenarios and the value of the RL parameters used are the same as those in the experimental setting section. As expected, the smaller the risk we wish to accept, i.e., a higher on-time probability, the longer is the buffered project duration.

It is interesting to visualize how our RL agent learns better solutions. Recall from the RL solution approach that the agent wants to learn the best actions, i.e., select activity modes and STAs that will maximize the agent's reward. In our RL model, we defined the reward as the inverse of the project delivery; thus, the agent will learn the action values, generate

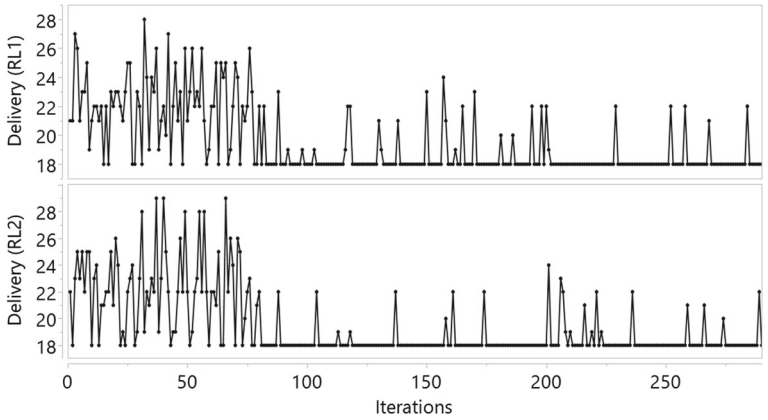


Fig. 5 Radar example learning curves for RL_1 and RL_2: 95% on-time probability

policies from the action values, and take actions based on the policies, seeking to minimize the project delivery.

Figure 5 exhibits the learning curves for both action-value updating variants, RL_1 and RL_2, with 95% on-time probability. We see at the beginning of the curves the effect of the optimistic initial values (see the RL solution approach section): even though the minimum delivery was found early on, the agent kept searching haphazardly, "thinking" that they could receive a better reward taking other actions, since the action-value list was initialized with artificially high values. The delivery eventually stabilized on 18 time periods. Since we used ε-greedy policies (the RL solution approach), sometimes the agent still wanted to explore, so the project delivery never settled completely on the minimum, jumping occasionally to longer durations.

The agent learned the shortest delivery before the 100th iteration; after less than 300 iterations, the median and mode for RL_1 and RL_2 equaled 18.

6 Factorial experimental design

In this section, we explain our factorial experiment, summarized in Table 10. We examined three project sizes, each with three modes per activity: 10-activity projects, which represent small problems with more potential of quickly covering a wider search space and obtaining faster solutions, and 50- and 100-activity projects, closer to real-life projects. For the 10-activity projects we used the PSPLIB datasets (Kolisch & Sprecher, 1997), and for the 50-

Table 10 Fractional factorial design

Project size	Algorithm	Constraints
10	RL_1	Chance constraints
50	RL_2	Deterministic
100	Solver	
	PR	

and 100-activity projects, the MMLIB datasets (Van Peteghem & Vanhoucke, 2014). Both datasets are the standard in the multimode project management literature (Vanhoucke & Coelho, 2018).

We ran our RL algorithm using both methods for updating the action values described in the RL solution approach: RL_1 and RL_2. We chose two additional methods as benchmarks. First, we solved our MILP from the quantitative model section using the Python interface for the commercial solver Gurobi version 9.0 and second, we employed the best combination of mode-selecting and activity-selecting priority rules found in Peng et al. (2015) (see the Appendix for more details about the method).[4]

We used two types of constraints: chance and deterministic constraints. For the chance constraints using the solver (*CS*), we employed 1000 scenarios. *CS* was evaluated only for the 10-activity projects. For the 50-activity projects, the solver could not generate a single incumbent solution for a sample of four projects after 48 h of runtime. For the chance-constrained RL_1 and RL_2 (CRL_1 and CRL_2) approaches, we used 1000 simulation runs to calculate each reward using the calculate_reward function. The published priority rule is not applicable to chance-constrained problems (henceforth we use an abbreviation for deterministic priority rules, *DPR*).

We also set up the experiments with the solver using deterministic constraints (DS), removing from the model in Sect. 3 all the references to the chance constraints. We removed constraints (2) and (3), and removed the scenario indexes from the relevant constraints as follows: Constraints (7) became $t_j - t_i - Mz_{ij} \geq \sum_{m=1}^{M_i} \delta_{im} d_{im} - M, \forall i, j = 0, \ldots, J + 1, \ \forall i \neq j$, and constraints (8) became $ES_j \leq t_j \leq LS_j, \ \forall j = 0, \ldots, J + 1$. For the deterministic-constrained RL_1 and RL_2 (DRL_1 and DRL_2), the calculate_reward function returned the inverse of the project duration with no simulation runs: $\mathcal{R}\left(j, \hat{m}_j, \hat{t}_j\right) = 1/D, \forall j = 1, \ldots, J$.

Initial tests showed that the runtime to obtain *DS* solutions for the 50- and 100-activity projects could vary from 2 min to 2 days or more, so we set a 30-min runtime cap using Gurobi's timeLimit parameter. If an optimal solution was not produced in this period, the best incumbent was registered. Likewise, due to the runtimes, we used a random sample of 100 instances for 50-activity projects and 50 instances for 100-activity projects. For 10-activity projects, we used the full sample of 535 projects.[5] To find the delivery date (baseline duration + PB) for all solutions of deterministic-constrained problems, we ran 1000 simulations for each solution.

For scenario generation and simulation runs, we used three-point estimates for the activity mode durations, defining the dataset's duration as the most likely duration, the pessimistic duration as 2.25 times this value, and the optimistic duration as half this value. These multipliers are found in Van de Vonder et al. (2006) and are in line with the well-known fact that in project management, the distribution of activity durations is characteristically skewed to the right (see, for example, Ma et al., 2019a, 2019b). Realized durations were drawn from a triangular distribution, very common in project simulation (see, for example, Batselier & Vanhoucke, 2016), and rounded to the nearest integer.

We set the desired on-time probability $\beta = 0.95$ for all chance-constrained runs and for the deterministic solutions' simulation runs. The stopping criterion for RL_1 and RL_2 was 1000 iterations after visiting all states with optimistic initial values for the 10- and 50-activity

[4] We chose these benchmarks because the MILP gives an optimal solution for both the deterministic problem and the sampled constraints in the chance-constrained problem; and priority rules are attractive because they are very fast.

[5] For *CS*, owing to the increased runtime, we set a 30-min limit.

projects and 2000 iterations for the 100-activity projects. We used the probability of random action $\varepsilon = 0.1$ and, for RL_2, the constant step-size parameter $\alpha = 0.1$ as in Sutton and Barto (2018).

The algorithms were coded in Python. We ran all experiments on a computer with an Intel(R) Core(TM) i7-7700 CPU 3.60 GHz, 8 GB RAM. To analyze the data, we conducted pairwise comparisons of the project delivery generated by each method and used JMP to calculate the p values (p) for the Wilcoxon signed rank (WSR) tests with a significance level of 0.05. Pareto analysis was also used to gain more insight into the results.

7 Results

As stated in the introduction, the research was conducted to prove two hypotheses:

Hypothesis H_1 Solving the chance-constrained CCBM problem directly generates shorter project deliveries than the traditional method of solving the deterministic problem and then inserting the time buffers.

Hypothesis H_2 Our RL method produces project deliveries at least as short as the benchmarks.

The hypotheses were tested using WSR tests for the project delivery pairwise comparisons; for further insight, we performed Pareto analysis of the rate at which each method generated the shortest delivery.

7.1 Hypothesis H_1

The chance-constrained methods generated shorter project durations than their deterministic-constrained counterparts.

For 10-activity projects, WSR tests for CRL_1–DRL_1 and CRL_2–DRL_2 generated highly significant differences, with $p < 0.0001$. For CS–DS we divided the results between those where the solver found an optimal solution for CS within the 30-min time limit (hereafter optimal group) and those where it did not (hereafter nonoptimal group).[6] For the optimal group, again we saw overwhelmingly negative differences, with $p < 0.0001$. For the nonoptimal group, the opposite was observed, also with $p < 0.0001$.

For 50-activity projects, CRL_1–DRL_1 and CRL_2–DRL_2 gave highly significant negative differences, with $p < 0.0001$. Likewise, for 100-activity projects, CRL_1–DRL_1 and CRL_2–DRL_2 gave $p < 0.0001$.

7.2 Hypothesis H_2

The performance of CRL_1 was at least equivalent to the benchmarks for all project sizes.

For 10-activity projects, CRL_1 produced the largest count of shortest durations, followed by CRL_2, even for the optimal group (Table 11). WSR tests for pairwise comparisons between CRL_1 and all the other methods, including CS for the optimal group, showed that CRL_1 generated shorter deliveries with $p < 0.0001$. CRL_2 also outperformed all the other methods, except CRL_1 and CS for the optimal group, with $p < 0.0001$.

[6] For DS, the solver found an optimal solution for all project instances without needing the 30-min limit.

Table 11 Pareto count of shortest deliveries

J		CRL_1	CRL_2	CS	DRL_1	DRL_2	DS	DPR
10	Optimal group	306	175	144	59	53	49	21
10	Nonoptimal group	156	66	2	51	22	1	11
50	Optimal group	22	2	–	6	0	56	4
50	Nonoptimal group	13	0	–	1	1	7	8
100	Optimal group	6	1	–	0	0	4	2
100	Nonoptimal group	5	0	–	0	0	0	10

For 50-activity projects, CRL_1 rendered shorter project deliveries than DPR but similar to DS[7]: DS generated a greater frequency of shortest deliveries for the optimal group (with CRL_1 in second place), and CRL_1 came first in the nonoptimal group (Table 11). Likewise, DS outperformed CRL_1 in the WSR tests for the optimal group ($p < 0.0001$) and CRL_1 rendered smaller durations than DS in the nonoptimal group ($p = 0.0001$). Shorter durations for CRL_1 compared to DPR were confirmed by WSR tests with $p < 0.0001$. Note that DPR performed relatively better in the Pareto count for 50 activities than for 10 activities (Table 11).

For 100-activity projects, no significant difference was revealed with the two-sided WSR tests, giving $p = 0.8587$ for CRL_1–DS and $p = 0.3181$ for CRL_1–DPR. Dividing DS into the optimal and nonoptimal groups, CRL_1 generated a greater rate of shortest deliveries in the former and generated the second greatest frequency of shortest deliveries in the latter (Table 11). DS, however, outperformed CRL_1 in the WSR tests in the optimal group ($p < 0.0001$); as expected, CRL_1 gave smaller durations than DS in the nonoptimal group ($p = 0.0007$).

Our results lead us to accept Hypotheses H_1 and H_2, validating both our proposal of directly solving the chance-constrained CCBM problem to obtain shorter schedules and our RL method as an effective way to solve the problem.

8 Discussion

The most striking result to emerge from the data is that shorter schedules are obtained when the chance-constrained model is solved directly instead of solving the deterministic model and subsequently inserting time buffers (Hypothesis H_1). When we tackle the chance-constrained problem directly, considering how the activities' durations actually play out, we are choosing the modes and start times that satisfy the real objective, i.e., minimizing the delivery date, which we defined as the project duration, including the PB, which delivers the project with the desired on-time probability.

Importantly, we have also found that the RL approach generates competitive schedules compared to accepted benchmarks (Hypothesis H_2). Interestingly, for 10-activity projects, CRL_1 generated shorter durations than CS, even where CS found an optimal solution. The explanation seems to be that CS finds the optimal solution for a given sample of scenarios; a solution for a different sample of realized durations may generate a shorter schedule. 10-activity projects, being small, permit a faster and more thorough exploration of the search space, and CRL_1 achieved better start-time and mode combinations and explored

[7] In a pairwise comparison considering all DS solutions, the WSR test generated $p = 0.0408$, giving evidence that DS rendered shorter deliveries than CRL_1.

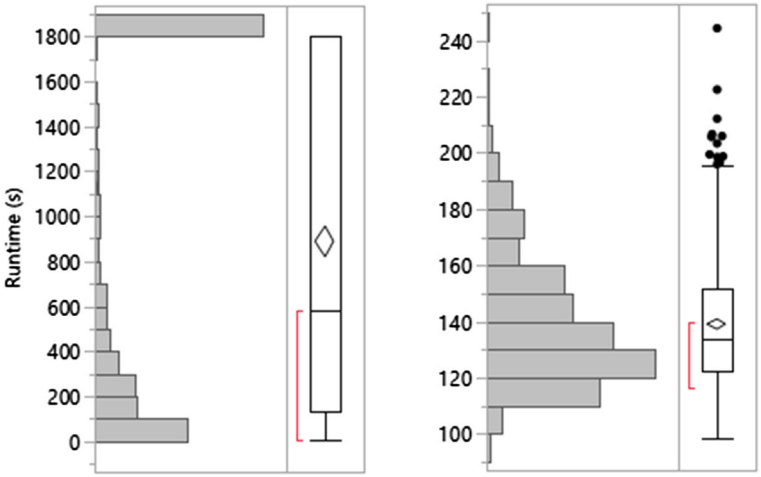


Fig. 6 10-activity projects runtime box plots. Left: CS (the maximum corresponds to the 30-min limit); right: CRL1

more instances of realized durations. Note that for 10 activities, the runtimes for *CS* tend to be much longer than those for CRL_1 as shown by the distributions in Fig. 6, further suggesting that MILP solver-based solutions tend to be a less interesting option.[8]

When we increased the project size to 50 and 100 activities, the expanded search space volume, on one hand, prevented the application of *CS* and, on the other, weakened the CRL_1 solutions vis-à-vis *DS* when the latter found an optimal solution. Increasing the number of CRL_1 iterations could improve the results, but at the expense of more processing time. Still, considering the whole sample, we showed that CRL_1 fared better than *DS*, which could not find an optimal solution in 30 min in 28% of the cases for 50 activities and 52% for 100 activities. The attainment of optimal solutions for *DS* in these cases could take prohibitively long runtimes for business decisions.

The option of working with nonoptimal *DS* solutions does not seem to pay off in 50- and 100-activity projects. We showed that nonoptimal *DS* solutions were generally of inferior quality, suggesting that *DS* is only competitive if an optimal solution is found. Branch and bound algorithms, which are the basis of commercial solvers, invest runtime in finding an incumbent, and in 50- and 100-activity projects considerable time may elapse until the first incumbent appears. In contrast, our RL algorithms do not waste runtime on infeasible solutions.

We have succeeded in verifying that *DPR* exhibited relatively low-quality results for 10 activities and generated improved deliveries for 50 and 100 activities. Because *DPR* does not search or learn a solution, for 10-activity projects it is easier to find a better RL or MILP solution. For larger search spaces, where we need longer runtimes to find better solutions, *DPR* will perform relatively better. Although CRL_1 surpassed *DPR* in the 10-activity and 50-activity projects, *DPR* may be an attractive alternative for the generation of instant schedules in larger projects.

An interesting outcome was the superior results obtained with CRL_1 vis-à-vis CRL_2. We conducted WSR tests for the pairwise project delivery differences CRL_1–CRL_2 for projects

[8] The 75% quartile for *CS* is actually masked by the time limit and is probably much higher.

with 10, 50 and 100 activities. For all project sizes, CRL_1 generated shorter schedules with $p < 0.0001$. Since the constant-step action-value update gives larger weight to the last actions and exponentially less weight to previous actions, we would think that CRL_2 could be more promising: the last decisions tend to be better because of the learning and ascribing them more weight could more quickly point to better policies. Based on our experience,[9] we believe that CRL_2 tends to find better results if we restrict the learning to fewer iterations. After repeated iterations, however, CRL_1 tends to stabilize in more promising regions of the search space, while CRL_2 keeps oscillating because of the greater weight ascribed to the last decisions.

The importance of solving the chance-constrained CCBM problem directly cannot be stressed too much. The earlier project delivery date in tandem with the desired on-time probability could mean the difference between winning or not winning a contract, and our RL-based algorithm is capable of tackling the problem and generating attractive solutions.

9 Conclusions and future research

This paper has investigated a formulation of CCBM combining multimode projects and chance constraints. We have presented a MILP formulation for the problem, tackling the chance constraints using SA. We have found an innovative solution using RL and conducted experiments to validate the formulation and the solution.

Our research underlined the importance of solving the chance-constrained problem and obtaining directly from the solution a PB tailored to the desired probability of completing the project on schedule. The results of this study show that solving the chance-constrained CCBM problem produces shorter project deliveries than inserting time buffers into a baseline schedule generated by the deterministic problem. We have also confirmed that our RL method produces competitive schedules compared to the priority rules and MILP solutions.

Our research can be a useful tool for decision makers because of its potential for generating shorter schedules with the same on-time probabilities. Furthermore, this work has revealed that our RL method is able to tackle large multimode projects with 50 and 100 activities, where the search space is very large.

We think that our findings could be useful for solving project scheduling problems with nonregular scheduling objectives, such as maximizing the net present value. Research into solving this problem is already underway. Additionally, further work on other RL techniques such as function approximation applied to project scheduling could help us enhance the search for better schedules in larger projects.

Acknowledgements This study has received funding from EIT Food, the Innovation Community on Food of the European Institute of Innovation and Technology (EIT), a body of the EU under Horizon 2020, the EU Framework Programme for Research and Innovation, project number 19147, and from the Bernard M. Gordon Center for Systems Engineering at the Technion.

Funding This study has received funding from EIT Food, the innovation community on Food of the European Institute of Innovation and Technology (EIT), a body of the EU under the Horizon 2020, the EU Framework Programme for Research and Innovation, project number 19147, and from the Bernard M. Gordon Center for Systems Engineering at the Technion.

Availability of data and materials The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

[9] We applied CRL_2 trying to maximize a project value function and in some cases CRL_2 outperformed CRL_1 (Szwarcfiter et al., 2022).

Code availability Not applicable.

Declarations

Conflict of interest The authors have no conflict of interest.

Appendix

Mode and activity priority rules employed in the experiment

Peng et al. (2015) evaluate a total of 60 mode-selecting and activity-selecting priority rules on different datasets. The combination of mode-activity priority rules that generated the shortest project durations was found to be the least total resource usage (LTRU) formula for selecting modes and the greatest resource demand (GRD) formula for selecting activities (Peng et al., 2015). We first transformed the multimode problem into a single-mode one by applying the LTRU priority rule. We then applied the GRD priority rule together with the well-known serial schedule generation scheme (Kolisch, 1996) to generate the schedule.

References

Aramesh, S., Mousavi, S. M., Mohagheghi, V., Zavadskas, E. K., & Antucheviciene, J. (2021). A soft computing approach based on critical chain for project planning and control in real-world applications with interval data. *Applied Soft Computing, 98*, 106915. https://doi.org/10.1016/j.asoc.2020.106915

Artigues, C., Koné, O., Lopez, P., & Mongeau, M. (2015). Mixed-integer linear programming formulations. In C. Schwindt & J. Zimmermann (Eds.), *Handbook on project management and scheduling* (Vol. 1, pp. 17–41). Springer. https://doi.org/10.1007/978-3-319-05443-8_2

Bakry, I., Moselhi, O., & Zayed, T. (2016). Optimized scheduling and buffering of repetitive construction projects under uncertainty. *Engineering, Construction and Architectural Management, 23*(6), 782–800. https://doi.org/10.1108/ECAM-05-2014-0069

Balouka, N., & Cohen, I. (2019). A robust optimization approach for the multi-mode resource-constrained project scheduling problem. *European Journal of Operational Research*. https://doi.org/10.1016/j.ejor.2019.09.052

Barto, A. G. (2019). Reinforcement learning: Connections, surprises, challenges. *AI Magazine, 40*(1), 3–15. https://doi.org/10.1609/aimag.v40i1.2844

Batselier, J., & Vanhoucke, M. (2016). Practical application and empirical evaluation of reference class forecasting for project management. *Project Management Journal, 47*(5), 36–51. https://doi.org/10.1177/875697281604700504

Bevilacqua, M., Ciarapica, F. E., Mazzuto, G., & Paciarotti, C. (2015). Robust multi-criteria project scheduling in plant engineering and construction. In C. Schwindt & J. Zimmermann (Eds.), *Handbook on project management and scheduling* (Vol. 2, pp. 1291–1305). Springer. https://doi.org/10.1007/978-3-319-05915-0_28

Blazewicz, J., Lenstra, J. K., & Kan, A. H. G. R. G. R. (1983). Scheduling subject to resource constraints: Classification and complexity. *Discrete Applied Mathematics, 5*(1), 11–24. https://doi.org/10.1016/0166-218X(83)90012-4

Bruni, M. E., Beraldi, P., & Guerriero, F. (2015). The stochastic resource-constrained project scheduling problem. In C. Schwindt & J. Zimmermann (Eds.), *Handbook on project management and scheduling* (Vol. 2, pp. 811–835). Springer. https://doi.org/10.1007/978-3-319-05915-0_7

Bruni, M. E., Beraldi, P., Guerriero, F., & Pinto, E. (2011). A heuristic approach for resource constrained project scheduling with uncertain activity durations. *Computers and Operation Research, 38*, 1305–1318. https://doi.org/10.1016/j.cor.2010.12.004

Calafiore, G., & Campi, M. C. (2005). Uncertain convex programs: Randomized solutions and confidence levels. *Mathematical Programming, 102*(1), 25–46. https://doi.org/10.1007/s10107-003-0499-y

Chakrabortty, R. K., Sarker, R. A., & Essam, D. L. (2017). Resource constrained project scheduling with uncertain activity durations. *Computers and Industrial Engineering*. https://doi.org/10.1016/j.cie.2016.12.040

Chen, B., & Hall, N. G. (2020). Incentive schemes for resolving Parkinson's Law in project management. *European Journal of Operational Research, 7*, 56. https://doi.org/10.1016/j.ejor.2020.06.006

Chen, S., Fang, S., & Tang, R. (2019). A reinforcement learning based approach for multi-projects scheduling in cloud manufacturing. *International Journal of Production Research, 57*(10), 3080–3098. https://doi.org/10.1080/00207543.2018.1535205

Chen, W., Zhao, Y., Yu, Y., Chen, K., & Arashpour, M. (2020). Collaborative scheduling of on-site and off-site operations in prefabrication. *Sustainability (switzerland), 12*(21), 1–23. https://doi.org/10.3390/su12219266

Davari, M., & Demeulemeester, E. (2019a). A novel branch-and-bound algorithm for the chance-constrained resource-constrained project scheduling problem. *International Journal of Production Research, 57*(4), 1265–1282. https://doi.org/10.1080/00207543.2018.1504245

Davari, M., & Demeulemeester, E. (2019b). Important classes of reactions for the proactive and reactive resource-constrained project scheduling problem. *Annals of Operations Research, 274*(1–2), 187–210. https://doi.org/10.1007/s10479-018-2899-7

Ferrucci, D., Levas, A., Bagchi, S., Gondek, D., & Mueller, E. T. (2013). Watson: Beyond jeopardy! *Artificial Intelligence, 199–200*, 93–105. https://doi.org/10.1016/j.artint.2012.06.009

Ghaffari, M., & Emsley, M. W. (2015). Current status and future potential of the research on critical chain project management. *Surveys in Operations Research and Management Science, 20*(2), 43–54. https://doi.org/10.1016/j.sorms.2015.10.001

Ghaffari, M., & Emsley, M. W. (2016). The impact of good and bad multitasking on buffer requirements of CCPM portfolios. *Journal of Modern Project Management, 4*(2), 91–95. https://doi.org/10.19255/JMPM01108

Ghoddousi, P., Ansari, R., & Makui, A. (2016). A risk-oriented buffer allocation model based on critical chain project management. *KSCE Journal of Civil Engineering, 21*, 1–13. https://doi.org/10.1007/s12205-016-0039-y

Ghoddousi, P., Ansari, R., & Makui, A. (2017). An improved robust buffer allocation method for the project scheduling problem. *Engineering Optimization, 49*(4), 718–731. https://doi.org/10.1080/0305215X.2016.1206534

Goto, H., & Murray, A. T. (2020). Exact and flexible solution approach to a critical chain project management problem. *Constraints, 25*, 280–297. https://doi.org/10.1007/s10601-020-09314-1

Gutjahr, W. J. (2015). Bi-objective multi-mode project scheduling under risk aversion. *European Journal of Operational Research, 246*(2), 421–434. https://doi.org/10.1016/j.ejor.2015.05.004

Herroelen, W., Leus, R., & Demeulemeester, E. (2002). Critical chain project scheduling: Do not oversimplify. *Project Management Journal, 33*(4), 48–60.

Hoel, K., & Taylor, S. G. (1999). Quantifying buffers for project schedules. *Production and Inventory Management Journal, 40*(2), 43–47.

Hu, X., Cui, N., Demeulemeester, E., & Bie, L. (2016). Incorporation of activity sensitivity measures into buffer management to manage project schedule risk. *European Journal of Operational Research, 249*(2), 717–727. https://doi.org/10.1016/j.ejor.2015.08.066

Hu, X., Demeulemeester, E., Cui, N., Wang, J., & Tian, W. (2017). Improved critical chain buffer management framework considering resource costs and schedule stability. *Flexible Services and Manufacturing Journal, 29*, 159–183. https://doi.org/10.1007/s10696-016-9241-y

Jędrzejowicz, P., & Ratajczak-Ropel, E. (2014). Reinforcement learning strategies for A-team solving the resource-constrained project scheduling problem. *Neurocomputing, 146*, 301–307. https://doi.org/10.1016/j.neucom.2014.05.070

Jędrzejowicz, P., & Ratajczak-Ropel, E. (2015). Reinforcement learning strategy for solving the MRCPSP by a team of agents. In R. Neves-Silva, L. Jain, & R. Howlett (Eds.), *Intelligent decision technologies* (pp. 537–548). Springer. https://doi.org/10.1007/978-3-319-19857-6_46

Kolisch, R. (1996). Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. *European Journal of Operational Research, 90*(2), 320–333. https://doi.org/10.1016/0377-2217(95)00357-6

Kolisch, R., & Sprecher, A. (1997). PSPLIB—A project scheduling problem library. *European Journal of Operational Research, 96*(1), 205–216. https://doi.org/10.1016/S0377-2217(96)00170-1

Lamas, P., & Demeulemeester, E. (2016). A purely proactive scheduling procedure for the resource-constrained project scheduling problem with stochastic activity durations. *Journal of Scheduling, 19*(4), 409–428. https://doi.org/10.1007/s10951-015-0423-3

Lambrechts, O., Demeulemeester, E., & Herroelen, W. (2011). Time slack-based techniques for robust project scheduling subject to resource uncertainty. *Annals of Operations Research, 186*, 443–464. https://doi.org/10.1007/s10479-010-0777-z

Li, H., Cao, Y., Lin, Q., & Zhu, H. (2022). Data-driven project buffer sizing in critical chains. *Automation in Construction, 135*, 104134. https://doi.org/10.1016/j.autcon.2022.104134

Li, H., & Demeulemeester, E. (2016). A genetic algorithm for the robust resource leveling problem. *Journal of Scheduling, 19*(1), 43–60. https://doi.org/10.1007/s10951-015-0457-6

Ma, G., Gu, L., & Li, N. (2015). Scenario-based proactive robust optimization for critical-chain project scheduling. *Journal of Construction Engineering and Management, 141*(10), 1–12. https://doi.org/10.1061/(ASCE)CO.1943-7862

Ma, G., Hao, K., Xiao, Y., & Zhu, T. (2019a). Critical chain design structure matrix method for construction project scheduling under rework scenarios. *Mathematical Problems in Engineering*. https://doi.org/10.1155/2019/1595628

Ma, G., Wang, A., Li, N., Asce, M., Gu, L., & Ai, Q. (2014). Improved critical chain project management framework for scheduling construction projects. *Journal of Construction Engineering and Management, 140*(12), 04014055. https://doi.org/10.1061/(ASCE)CO.1943-7862.0000908

Ma, Z., Demeulemeester, E., He, Z., & Wang, N. (2019b). A computational experiment to explore better robustness measures for project scheduling under two types of uncertain environments. *Computers and Industrial Engineering, 131*, 382–390. https://doi.org/10.1016/j.cie.2019.04.014

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature, 518*(7540), 529–533. https://doi.org/10.1038/nature14236

Moradi, H., & Shadrokh, S. (2019). A robust scheduling for the multi-mode project scheduling problem with a given deadline under uncertainty of activity duration. *International Journal of Production Research, 57*(10), 3138–3167. https://doi.org/10.1080/00207543.2018.1552371

Newbold, R. C. (1998). Project management in the fast lane. In R. C. Newbold (Ed.), *Project management in the fast lane.* St. Lucie Press. https://doi.org/10.1201/b18205

Ning, M., He, Z., Wang, N., & Liu, R. (2018). Metaheuristic algorithms for proactive and reactive project scheduling to minimize contractor's cash flow gap under random activity duration. *IEEE Access, 6*, 30547–30558. https://doi.org/10.1109/ACCESS.2018.2828037

Ordoñez, R. E. C., Vanhoucke, M., Coelho, J., Anholon, R., & Novaski, O. (2019). A study of the critical chain project management method applied to a multiproject system. *Project Management Journal, 50*(3), 322–334. https://doi.org/10.1177/8756972819832203

Peng, W., & Huang, M. (2014). A critical chain project scheduling method based on a differential evolution algorithm. *International Journal of Production Research, 52*(13), 3940–3949. https://doi.org/10.1080/00207543.2013.865091

Peng, W., Huang, M. C., & Yongping, H. (2015). A multi-mode critical chain scheduling method based on priority rules. *Production Planning and Control, 26*(12), 1011–1024. https://doi.org/10.1080/09537287.2014.1002020

Peng, W., Lin, X., & Li, H. (2023). Critical chain based proactive-reactive scheduling for resource-constrained project scheduling under uncertainty. *Expert Systems with Applications, 214*, 119188. https://doi.org/10.1016/j.eswa.2022.119188

PMI. (2021). Beyond agility: Flex to the future. In *Pulse of the profession*. https://www.pmi.org/-/media/pmi/documents/public/pdf/learning/thought-leadership/pulse/pmi_pulse_2021.pdf?v=b5c9abc1-e9ff-4ac5-bb0d-010ea8f664da&sc_lang_temp=en

Polvara, R., Sharma, S., Wan, J., Manning, A., & Sutton, R. (2019). Autonomous vehicular landings on the deck of an unmanned surface vehicle using deep reinforcement learning. *Robotica*. https://doi.org/10.1017/S0263574719000316

Poshdar, M., González, V., Raftery, G., Orozco, F., Romeo, J., & Forcael, E. (2016). A probabilistic-based method to determine optimum size of project buffer in construction schedules. *Journal of Construction Engineering and Management, 142*(10), 4016046. https://doi.org/10.1061/(ASCE)CO.1943-7862.0001158

Project Management Institute. (2017). *A guide to the project management body of knowledge (PMBOK guide)* (6th ed.). Project Management Institute.

Rahman, H. F., Chakrabortty, R. K., & Ryan, M. J. (2021a). Scheduling project with stochastic durations and time-varying resource requests: A metaheuristic approach. *Computers and Industrial Engineering, 157*, 107363. https://doi.org/10.1016/J.CIE.2021.107363

Rahman, M. H. F., Chakrabortty, R. K., & Ryan, M. J. (2021b). Managing uncertainty and disruptions in resource constrained project scheduling problems: A real-time reactive approach. *IEEE Access, 9*, 45562–45586. https://doi.org/10.1109/ACCESS.2021.3063766

Salama, T., Salah, A., & Moselhi, O. (2018). Integration of linear scheduling method and the critical chain project management. *Canadian Journal of Civil Engineering, 45*(1), 30–40. https://doi.org/10.1139/cjce-2017-0020

Sallam, K. M., Chakrabortty, R. K., & Ryan, M. J. (2021). A reinforcement learning based multi-method approach for stochastic resource constrained project scheduling problems. *Expert Systems with Applications, 169*, 114479. https://doi.org/10.1016/j.eswa.2020.114479

She, B., Chen, B., & Hall, N. G. (2021). Buffer sizing in critical chain project management by network decomposition. *Omega (united Kingdom)*. https://doi.org/10.1016/j.omega.2020.102382

Solan, D., & Shtub, A. (2021). The Influence of competition on new product development project planning decisions. *IEEE Transactions on Engineering Management, 68*(5), 1398–1405. https://doi.org/10.1109/tem.2019.2910207

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). The MIT Press. https://doi.org/10.1108/k.1998.27.9.1093.3

Szwarcfiter, C., Herer, Y. T., & Shtub, A. (2022). Project scheduling in a lean environment to maximize value and minimize overruns. *Journal of Scheduling*. https://doi.org/10.1007/s10951-022-00727-9

Tao, S., Wu, C., Sheng, Z., & Wang, X. (2018). Stochastic project scheduling with hierarchical alternatives. *Applied Mathematical Modelling, 58*, 181–202. https://doi.org/10.1016/j.apm.2017.09.015

Tenera, A. B. (2008). Critical chain buffer sizing: a comparative study. In *Proceedings of PMI research conference. July 13*. https://www.pmi.org/learning/library/critical-chain-project-management-theory-7118

The Standish Group. (2015). *CHAOS report 2015 edition*. https://www.standishgroup.com/sample_research_files/CHAOSReport2015-Final.pdf

Tian, J., Hao, X., & Gen, M. (2019). A hybrid multi-objective EDA for robust resource constraint project scheduling with uncertainty. *Computers and Industrial Engineering, 130*, 317–326. https://doi.org/10.1016/j.cie.2019.02.039

Tian, M., Liu, R. J., & Zhang, G. J. (2020). Solving the resource-constrained multi-project scheduling problem with an improved critical chain method. *Journal of the Operational Research Society, 71*(8), 1–16. https://doi.org/10.1080/01605682.2019.1609883

Tian, W., & Demeulemeester, E. (2014). Railway scheduling reduces the expected project makespan over roadrunner scheduling in a multi-mode project scheduling environment. *Annals of Operations Research, 213*(1), 271–291. https://doi.org/10.1007/s10479-012-1277-0

Torabi Yeganeh, F., & Zegordi, S. H. (2020). A multi-objective optimization approach to project scheduling with resiliency criteria under uncertain activity duration. *Annals of Operations Research, 285*(1–2), 161–196. https://doi.org/10.1007/s10479-019-03375-z

Tysiak, W. (2017). Monte Carlo simulation and critical chains. In *Proceedings of the 2017 IEEE 9th international conference on intelligent data acquisition and advanced computing systems: Technology and applications, IDAACS 2017*, *1* (pp. 471–474). https://doi.org/10.1109/IDAACS.2017.8095125

Van De Vonder, S., Demeulemeester, E., Herroelen, W., & Leus, R. (2006). The trade-off between stability and makespan in resource-constrained project scheduling. *International Journal of Production Research, 44*(2), 215–236. https://doi.org/10.1080/00207540500140914

Van Peteghem, V., & Vanhoucke, M. (2014). An experimental investigation of metaheuristics for the multi-mode resource-constrained project scheduling problem on new dataset instances. *European Journal of Operational Research, 235*(1), 62–72. https://doi.org/10.1016/J.EJOR.2013.10.012

Vanhoucke, M. (2016). *Integrated project management sourcebook*. Springer. https://doi.org/10.1007/978-3-319-27373-0

Vanhoucke, M., & Coelho, J. (2018). A tool to test and validate algorithms for the resource-constrained project scheduling problem. *Computers and Industrial Engineering, 118*, 251–265. https://doi.org/10.1016/j.cie.2018.02.001

Wang, L., Huang, H., & Ke, H. (2015). Chance-constrained model for RCPSP with uncertain durations. *Journal of Uncertainty Analysis and Applications, 3*(1), 1–10. https://doi.org/10.1186/s40467-015-0034-8

Wauters, T., Verbeeck, K., de Causmaecker, P., & van den Berghe, G. (2015). A learning-based optimization approach to multi-project scheduling. *Journal of Scheduling, 18*(1), 61–74. https://doi.org/10.1007/s10951-014-0401-1

Wauters, T., Verbeeck, K., van den Berghe, G., & de Causmaecker, P. (2011). Learning agents for the multi-mode project scheduling problem. *Journal of the Operational Research Society, 62*(2), 281–290. https://doi.org/10.1057/jors.2010.101

Wichmann, M. G., Gäde, M., & Spengler, T. S. (2019). A fuzzy robustness measure for the scheduling of commissioned product development projects. *Fuzzy Sets and Systems, 377*, 125–149. https://doi.org/10.1016/j.fss.2019.02.015

Zarghami, S. A., Gunawan, I., Corral de Zubielqui, G., & Baroudi, B. (2020). Incorporation of resource reliability into critical chain project management buffer sizing. *International Journal of Production Research, 58*(20), 6130–6144. https://doi.org/10.1080/00207543.2019.1667041

Zhang, J., Jia, S., & Diaz, E. (2015a). A new buffer sizing approach based on the uncertainty of project activities. *Concurrent Engineering, 23*(1), 3–12. https://doi.org/10.1177/1063293X14561871

Zhang, J., Jia, S., & Diaz, E. (2018). Dynamic monitoring and control of a critical chain project based on phase buffer allocation. *Journal of the Operational Research Society, 69*(12), 1966–1977. https://doi.org/10.1080/01605682.2017.1415641

Zhang, J., Song, X., Chen, H., & Shi, R. (2015b). Optimisation of critical chain sequencing based on activities information flow interactions. *International Journal of Production Research, 53*(20), 6231–6241. https://doi.org/10.1080/00207543.2015.1043157

Zhang, J., Song, X., Chen, H., & Shi, R. (2016a). Determination of critical chain project buffer based on information flow interactions. *Journal of the Operational Research Society, 16*(109), 1–12. https://doi.org/10.1057/jors.2016.9

Zhang, J., Song, X., & Díaz, E. (2016b). Project buffer sizing of a critical chain based on comprehensive resource tightness. *European Journal of Operational Research, 248*(1), 174–182. https://doi.org/10.1016/j.ejor.2015.07.009

Zhang, J., Song, X., & Díaz, E. (2017). Critical chain project buffer sizing based on resource constraints. *International Journal of Production Research, 55*(3), 671–683. https://doi.org/10.1080/00207543.2016.1200151

Zhao, Y., Cui, N., & Tian, W. (2020). A two-stage approach for the critical chain project rescheduling. *Annals of Operations Research, 285*(1–2), 67–95. https://doi.org/10.1007/s10479-019-03347-3

Zheng, W., He, Z., Wang, N., & Jia, T. (2018). Proactive and reactive resource-constrained max-NPV project scheduling with random activity duration. *Journal of the Operational Research Society, 69*(1), 115–126. https://doi.org/10.1057/s41274-017-0198-3

Zhou, T., Long, Q., Law, K. M. Y., & Wu, C. (2022). Multi-objective stochastic project scheduling with alternative execution methods: An improved quantum-behaved particle swarm optimization approach. *Expert Systems with Applications, 203*, 117029. https://doi.org/10.1016/j.eswa.2022.117029

Zohrehvandi, S., Vanhoucke, M., Khalilzadeh, M., Amiri, M., & Shadrokh, S. (2022). A fuzzy project buffer management algorithm: A case study in the construction of a renewable project. *International Journal of Construction Management*. https://doi.org/10.1080/15623599.2022.2045860