Solving the stochastic multimode resource-constrained project scheduling problem

Claudio Szwarcfiter[1], Avraham Shtub[2], and Yale T. Herer[3]

Faculty of Industrial Engineering and Management
Technion—Israel Institute of Technology, Haifa, Israel
[1]e-mail: claudioszw@campus.technion.ac.il

[2]e-mail: shtub@technion.ac.il

[3]e-mail: yale@technion.ac.il

Keywords: Stochastic resource-constrained project scheduling, multimode project management, stability and robustness in project management, reinforcement learning.

1. Introduction

The resource-constrained project scheduling problem (RCPSP) is a classic problem in project management, and its extensions, the multimode and the stochastic RCPSP (MRCPSP and SRCPSP), have received considerable attention. A standard procedure for solving these problems is the employment of heuristic methods, since the RCPSP is known to be NP-hard. However, less attention has been paid to the advances in artificial intelligence, particularly reinforcement learning (RL), and the opportunities they present for improving the search.

In this paper, we provide a novel RL-based approach for solving a version of the stochastic multimode RCPSP (SMRCPSP). This approach provides effective exploration of the search space, scanning a wide range of combinations of activity modes and start times, while simultaneously exploiting the learned knowledge. Our experiments currently being conducted suggest that the RL algorithm combines speed with performance close to the optimum.

2. Problem and solution approach

We model our SMRCPSP based on the flow-based formulation described in Artigues *et al.* (2015), expanding it for a multimode setting. Furthermore, we consider stochastic activity durations; therefore, the duration constraints cannot, in general, be guaranteed with certainty and thus we will model them as chance constraints. One way of solving the resulting stochastic program is by scenario optimization (SO), introduced in Calafiore and Campi (2005). 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. The result is a mixed-integer linear program (MILP).

We consider a project with *J* activities. Each activity *j* can be executed in one of M_j modes and is preceded by a set of immediate predecessors $\mathrm{P}(j)$. Each activity *j* executed in mode *m* in scenario *s* has a duration d_{jms}. There are *K* different renewable resources. Activity *j* executed in mode *m* needs r_{jm}^{k} units of resource *k*, which has a total availability of R^{k}. ES_{js} and LS_{js} are the earliest and latest start for activity *j* in scenario *s*, respectively. Decision variable *D* is the project delivery date. We set parameter β as the desired probability of the project finishing within the delivery date, and θ as an upper bound for delivery date overrun. Binary decision variable δ_{jm} indicates if activity *j* is carried out in mode *m* and decision variable t_{js} denotes, for scenario *s*, the starting time of activity $j, j = 0,..., J+1$, where $j = 0$ and $J = J+1$ are dummy activities with a single mode, no duration and resources, and represent the start and end of the project, respectively. Binary decision variable z_{ij} indicates (value 1) if activity *j* starts after activity *i* finishes. The amount of resource *k* transferred from activity *i* to activity *j* is modeled by the flow variable ϕ_{ij}^{k}.

τ_s is a binary decision variable indicating whether, in scenario *s*, the project finishes within the delivery date. The model is as follows.

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

Subject to:

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

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

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

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

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

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

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

$$\phi_{ij}^k - \min\left(\tilde{r}_{im}^k, \tilde{r}_{jm'}^k\right) z_{ij} - \left(1-\delta_{im}\right)\left(r_{ij}^{\max,k} - \min\left(\tilde{r}_{im}^k, \tilde{r}_{jm'}^k\right)\right) - \left(1-\delta_{jm'}\right)\left(r_{ij}^{\max,k} - \min\left(\tilde{r}_{im}^k, \tilde{r}_{jm'}^k\right)\right) \le 0,$$

$$\text{where } r_{ij}^{\max,k} = \max\left(\max_{m=1,\ldots,M_i} \tilde{r}_{im}^k, \max_{m'=1,\ldots,M_j} \tilde{r}_{jm'}^k\right) \text{ and } \tilde{r}_{j''m}^k = \begin{cases} r_{jm}^k & \text{if } 0 < j'' < n+1 \\ R^k & \text{if } j'' = 0 \text{ or } j'' = n+1, \end{cases} \tag{9}$$

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

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

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

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

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

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 it 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 with the maximum resource consumption modes.

2.1. Reinforcement learning solution approach

Reinforcement Learning (RL) has been shown to be successful in diverse applications with uncertain environments. This success is the factor motivating the application of RL to our stochastic environment. To the best of our knowledge, multimode problems involving stochastic activity duration have not yet been tackled with RL.

Our RL model starts with an agent at project activity *j*. The agent undertakes an action by choosing a mode $\hat{m}_j$ and start time $\hat{t}_j$ for activity *j* and then moves on to the next activity. After selecting modes and start times for all activities $j = 1,\ldots,J,$ she receives a reward $\mathrm{R}(j,m,t).$ The agent follows a policy $\pi(j,m,t)$ that tells her at each activity which action she should take. We further define an action-value function $q(j,m,t)$ as the estimated reward for taking an action at activity j and thereafter following policy $\pi(j,m,t)$. The RL problem's objective is to learn a policy that maximizes the agent's reward. We use Monte Carlo Control (MCC), based on Sutton and Barto (1998). Figure 1 presents our main MCC pseudocode.

```
Initialize action-values
while not stopping criterion:
    calculate policy
    choose mode and start time
    calculate reward
    update action-values RL1
    or update action-values RL2
```

Figure 1. MCC pseudocode.

Our algorithm starts with the initialization of the action-values table with artificially high values. The action-values table is then used to calculate the policy. 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. Next, we take an action based on the policy, choosing for each activity the mode and start time according to the probabilities in the policy table. Then, we calculate the reward for the actions taken as $(1/D)$, where D is the delivery date for on-time probability β. The last step in the algorithm is to update the action-value table using the reward. We can choose from two update methods, RL_1 and RL_2: RL_1 learns an action-value by averaging all the rewards this action has received each time it was taken. RL_2 updates the action-values giving an exponentially large weight to the last action.

3. Experimental setting and partial results

To validate the RL procedure we propose a factorial experiment, summarized in Table 1, as follows. We will compare three project sizes, each with three modes per activity. For the 10-activity projects we will use the PSPLIB datasets (Kolisch and Sprecher, 1997), and for the 50 and 100-activity projects, the MMLIB datasets (Van Peteghem and Vanhoucke, 2014), generating additional data for the stochastic activity durations. We will run our RL algorithm using both methods for updating the action-values: RL_1 and RL_2, as described in Section 2.1. The delivery dates obtained with both variants will be compared to those from two benchmarks: the best combination of mode and activity priority rules (Peng, Huang and Yongping, 2015) and a solution for our MILP, using the Gurobi 8.1 solver. We will compare two types of constraints: solving the deterministic problem and then simulating realized durations to generate the delivery date, and solving directly the chance-constrained problem; in both cases, we will set the desired probability of the project finishing within the delivery date $\beta = 0.95$.

Table 1. Partial factorial design.

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

The algorithm is currently being executed and evaluated and we will be reporting the results in the conference. We present here partial results for 10-activity projects. Chance-constrained RL_1 (CRL_1) outperformed the other algorithms. Figure 2 provides a comparative view of project delivery for 10-activity projects; for clarity, we show only three curves: CRL_1, chance-constrained solver (CS) and deterministic-constrained priority rules (DPR). CRL_1, represented by the solid line, is consistently below the other curves. In fact, Wilcoxon signed rank tests for pairwise comparisons between CRL_1 and all the other methods, showed that CRL_1 generated shorter deliveries with p-value < 0.0001.

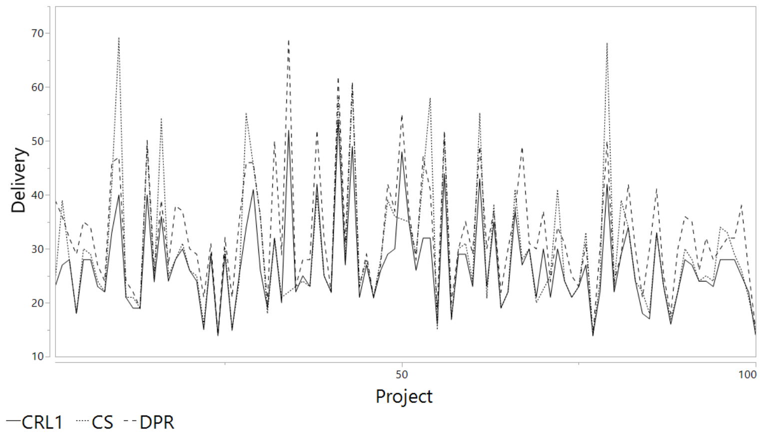

Figure 2. 10-activity projects: Overlay plot comparing CRL_1 with DPR and CS; for clarity, we show here a random subsample of 100 projects from the 535-project sample

4. Conclusions

In this paper, we presented a flow-based formulation of a variant of the SMRCPSP. The objective is to minimize the project delivery date and we introduce a constraint imposing a lower bound on the probability of finishing within this date. We described a novel RL-based approach for solving the problem and proposed a partial factorial design for the evaluation of our method. We have completed experiments for 10-activity projects and have concluded, with statistical significance, that for this project size, our RL approach renders shorter schedules than both the best priority rules, and the MILP solutions obtained with the solver using SO. We will be reporting the main results for the full experiment at the conference.

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 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.

References

Artigues, C. *et al.* (2015), 'Mixed-integer linear programming formulations', in *Handbook on Project Management and Scheduling Vol.1*. Cham: Springer International Publishing, pp. 17–41.

Balouka, N. and Cohen, I. (2019), 'A robust optimization approach for the multi-mode resource-constrained project scheduling problem', to be published in *European Journal of Operational Research* [Preprint].

Calafiore, G. and Campi, M. C. (2005), 'Uncertain convex programs: Randomized solutions and confidence levels', *Mathematical Programming*, 102(1), pp. 25–46.

Kolisch, R. and Sprecher, A. (1997), 'PSPLIB – A project scheduling problem library', *European Journal of Operational Research*, 96(1), pp. 205–216.

Peng, W., Huang, M. C. and Yongping, H. (2015), 'A multi-mode critical chain scheduling method based on priority rules', *Production Planning and Control*, 26(12), pp. 1011–1024.

Van Peteghem, V. and 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*. North-Holland, 235(1), pp. 62–72.

Sutton, R. S. and Barto, A. G. (1998), *Reinforcement learning: An introduction*. MIT Press.