Maximizing value—Modeling and solving lean project management

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: Lean project management, project scheduling, multimode project management, stability and robustness in project management.

1. Introduction

Lean Project Management (LPM) is a comprehensive framework with the goal of creating value and minimizing waste, in a minimum of time (Oehmen (Ed.), 2012). A key feature of LPM is the integration of two disciplines, which until now have been kept separate: program management and systems engineering. The former relates to project scope and the latter, to product scope.

The main contribution of our research is to provide a decision support tool for project managers by modeling and solving a novel LPM application. The model's objective is to maximize project value subject to due date and budget constraints. Moreover, our model tackles project risk, which often plays out in projects by causing unforeseen delays in activity durations, ultimately leading to due date and budget overruns. Therefore, our model considers stochastic activity durations, and the way we proactively manage risk is by generating a stable project plan in which the due-date and budget violation probabilities are kept within a desired threshold.

Project scope and product scope integration—which is key to LPM—is achieved in our model using a multimode approach, in which each project activity can be executed in one or more modes or alternatives. The integrative feature is that each activity mode, apart from containing information on the project scope, such as stochastic duration parameters, fixed and resource costs, also embodies data on the product scope, i.e., value parameters. Thus, when a mode is chosen, not only does the choice impact the project duration, cost, and associated risks, but also the project value. For example, for the activity "antenna design" in a radar project, there could be two modes: "re-engineer" and "new design"; associated with each mode we could have a radar range parameter, so that the mode choice would affect the overall radar range.

To the best of our knowledge, this is the first attempt to assist lean project managers and their teams with a decision support tool that models and solves the project planning problem focusing not only on the project scope (the work to be done) but also on the product scope (the features and functions of the product).

2. Problem and solution approach

We consider a project with *J* activities. Each activity *j* can be executed in one of M_j modes. The parameter V_{jmv} designates the value of attribute *v* for activity *j* executed in mode *m*. Decision variable V_{jv}' corresponds to the value of attribute *v* for activity *j* executed in its chosen mode. We define a function $F_v\left(V_{1v}',...,V_{Jv}'\right)$ that determines the project value for each attribute *v* given the individual attributes V_{jv}', and a function $V''\left(F_1,...,F_V\right)$ that calculates the project value given the values for each attribute. Binary decision variable δ_{jm} indicates if activity *j* is carried out in mode *m*. The objective of our model is to maximize the project value, which is a project-specific function of the chosen modes and can be a non-linear function:

$$\text{Maximize } V''\left(F_1\left(V_{11}',...,V_{J1}'\right),...,F_V\left(V_{1V}',...,V_{JV}'\right)\right),$$

where

$$V_{jv}' = \sum_{m=1}^{M_j} \delta_{jm} V_{jmv}, \ \forall v = 1,...,V, \ \forall j = 1,...,J.$$

We formulate the deterministic version of our problem as a mixed integer program subject to duration and cost constraints. If we now consider stochastic activity durations, the duration constraints cannot, in general, be guaranteed with certainty and thus we model them as chance constraints. One way of solving the resulting stochastic program is by Scenario Optimization (SO), introduced in Calafiore and Campi (2005) and applied in recent project scheduling papers. 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. Thus, if our objective function is linear, the new SO program is a mixed integer linear program (MILP), and can be solved with a commercial solver. We use this method as a benchmark in the computational experiments.

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 our application of RL to learning the activity modes that maximize value in a stochastic environment. RL-based heuristics have also been applied to project scheduling, but to the best of our knowledge, problems involving stochastic activity duration or project value have not yet been tackled with RL. Hence, another contribution of our research is applying RL to this problem.

The RL model starts with an agent in a state Σ. The agent undertakes action A and moves to state S', receiving a reward R'. She then executes action A', moving to state S'' and receiving a reward R'', and so on. We can thus represent the agent's life trajectory as $\mathrm{S},\mathrm{A},\mathrm{R}',\mathrm{S}',\mathrm{A}',\mathrm{R}'',\mathrm{S}'',\mathrm{A}'',\mathrm{R}''',\mathrm{S}''',\mathrm{A}'''$, etc. The agent follows a policy $\pi(\mathrm{S},\mathrm{A})$ that tells her at each state which action she 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 $q(\mathrm{S},\mathrm{A})$ as the estimated reward for taking action A on state Σ and thereafter following policy $\pi(\mathrm{S},\mathrm{A})$.

Applying the RL model to our problem, we define a state as project activity *j*. The agent undertakes an action by choosing a mode $\hat{m}_j$ for activity *j* and then moves on to the next activity. After selecting modes for all activities $(\hat{m}_j, \forall j = 1,...,J)$, she receives a reward, which we define as the project value V'' if, after a number of simulation runs with the chosen modes, the proportion of projects on time and on budget is more than or equal to the pre-defined on-time and on-budget probabilities; otherwise, the reward is zero. To balance exploration and exploitation, we employ ε-greedy policies, in which we set a probability ε of choosing a random mode for an activity; otherwise, a mode is chosen greedily, i.e., the one that has the highest action-value. We employ two alternative methods for updating the action-values. The first is Average Rewards (RL_1), in which the action-values are calculated by averaging the rewards accrued every time a certain mode is chosen for a certain activity. The second method is Constant Step (RL_2), which tries to leverage the learning by giving an exponentially larger weight to the last actions. We use a RL procedure known as Monte Carlo Control (MCC; based on Sutton and Barto, 1998), in which first we initialize the action-values; then, we calculate the policy, choose the activity modes based on the policy, calculate the reward accrued from this choice, and update the action-values using RL_1 or RL_2; we then once again calculate the policy using the updated action-values, and so on until the stopping criterion is met.

3. Experimental setting and main results

To validate the RL procedure we designed and conducted a factorial experiment, summarized in Table 1, as follows. We ran the algorithms with deterministic (zero risk) and stochastic activity durations and compared three project sizes, each with three modes per activity. For the 10-activity projects we used the PSPLIB datasets (Kolisch and Sprecher, 1997), and for the 50 and 100-activity projects, the MMLIB datasets (Van Peteghem and Vanhoucke, 2014). Both datasets are the standard in the multimode project management literature. We ran our RL algorithm using both

methods for updating the action-values: RL_1 and RL_2, as described in Section 2.1. The project values obtained with both variants were compared to those from two benchmarks, a genetic algorithm (GA; Balouka, Cohen and Shtub, 2016) that seeks to maximize the project value for deterministic problems, with populations of 500, 1000 and 10,000, and a solution for our mixed integer program, in the case of linear objective functions (MILP), using the Gurobi 8.1 solver. For our stochastic settings, we developed a new fitness function for the GA:

$$f(I)=\begin{cases}V''(I), \text{ if } E(I)=0\\ V''(I)-E(I)+Min\_V''(\text{feasible solutions})-Max\_V''(\text{all solutions}), \text{otherwise,}\end{cases}$$

where $E(I)=\max\left(0,\hat{\beta}-\text{ proportion of on-budget runs}\right)$ for a solution *I*, i.e., the selected mode for each activity, and $\hat{\beta}$ is the desired probability of the project finishing within the budget.

Table 1. Partial factorial design: All the combinations were tested, except $GA_{10,000}$ for the stochastic trials and Solver for nonlinear objective functions.

Project risk	Project size	Algorithm	Objective function	Stopping criterion
Deterministic	10	GA_{500}	Linear	GA_S
Stochastic	50	GA_{1000}	Nonlinear	5 min
	100	$GA_{10,000}$		Max/near max
		RL_1		
		RL_2		
		Solver		

For each project, we generated two objective functions: a linear one, $0.6\sum_{j=1}^{J}V_{j1}'+0.4\sum_{j=1}^{J}V_{j2}'$ and a nonlinear one, $0.6\prod_{j=1}^{J}V_{j1}'+0.4\prod_{j=1}^{J}V_{j2}'$. For the linear objectives, we drew uniformly random value parameters V_{jmv} from the sets $\{0,1,2,...,10\}$, $\{0,0.2,0.4,...,2\}$, and $\{0,0.1,0.2,...,1\}$ for projects with $J=$ 10, 50 and 100 activities, respectively. For the nonlinear objectives, we generated uniformly random parameters from the sets $\left\{1+a\left(100^{1/J}-1\right)/5 \middle| a=0,1,...,5\right\}$ for experiments with $J=$ 10, 50 and 100 activities. Finally, we compared three stopping criteria: stopping RL_1 and RL_2 at the same execution time it took for the GA to converge (according to its published stopping criterion, two generations with the same best value; GA_S in Table 1);[1] five minutes, which is a reasonable running time for applications in industry; and the time it takes for the algorithms to give their best performance and produce a near-optimal solution ("max/near max" in Table 1) – by simulation, we estimated this time to be 20 seconds for the 10-activity projects and 2 hours for 50 and 100 activities.

Table 2 shows the results for the deterministic experiments with nonlinear objectives and GA_S stopping criterion (we show here only GA population 1000), and Table 3 shows the same results for the stochastic experiments. The columns show the number of projects tested, number of activities, GA population, average running time for the GA, and the average percent differences of the objective values; the alternative hypotheses H_1 are tested using one-tailed sign tests to evaluate whether one algorithm produces better solutions than the other and the p-values are presented. If no significant difference is found, a two-tailed sign test is conducted and the alternative hypothesis $a\neq b$ and p-value are recorded.

Table 2. Sign tests comparing RL to GA: Deterministic, nonlinear objective, stopping criterion GA_s.

Proj	Act	Pop_{GA}	T_{GA}(s)	RL_1-GA	H_1	P_V	RL_2-GA	H_1	P_V	RL_1-RL_2	H_1	P_V
535	10	1000	0.65	7.52	RL_1>GA	0.000	8.91	RL_2>GA	0.000	-1.23	RL_2>RL_1	0.000
540	50	1000	3.26	25.93	RL_1>GA	0.000	5.63	RL_2>GA	0.000	-0.94	RL_2>RL_1	0.003
540	100	1000	7.05	30.21	RL_1>GA	0.000	27.97	RL_2>GA	0.000	3.34	RL_1>RL_2	0.000

[1] This stopping criterion generated near-optimal solutions for 10- and 20-activity projects using populations of 500 and 1000 in Balouka, Cohen and Shtub, (2016).

Table 3. Sign tests comparing RL to GA: Stochastic, nonlinear objective, GA_S stopping criterion.

Proj	Act	Pop_{GA}	T_{GA}(s)	RL_1–GA	H_1	P_V	RL_2–GA	H_1	P_V	RL_1–RL_2	H_1	P_V
535	10	1000	119.18	7.70	RL_1>GA	0.000	7.44	RL_2>GA	0.000	1.80	$RL_1 \neq RL_2$	0.166
120	50	1000	479.15	0.64	$RL_1 \neq$GA	0.519	13.06	RL_2>GA	0.000	-9.99	RL_2>RL_1	0.000
71	100	1000	798.01	-3.43	$RL_1 \neq$GA	1.0	4.50	$RL_2 \neq$GA	1.0	-4.13	$RL_1 \neq RL_2$	0.453

Considering a significance level of 0.05, we can conclude that for the GA_S stopping criterion in the deterministic experiments, RL_1 and RL_2 render better solutions than GA across the board, and the results are strongly significant. For the stochastic experiments, where the null hypothesis was rejected, similar results were obtained.

4. Conclusions

In this paper, we presented a new model for LPM, maximizing value and providing stability by complying with minimum on-time and on-budget probabilities set by the decision makers. We developed a stochastic programming model with a SO formulation. We introduced a new RL method to solve the problem, with two variants for action-value updates. We conducted a partial factorial experiment, both with small 10-activity and with larger 50- and 100-activity projects, comparing both RL variants with two benchmarks, a GA and, for linear objectives, a solution using a commercial solver, reaching the following conclusions, with statistical significance:

1. The RL methods are the best option when we want to find good solutions in less time, as they are much faster than the GA.
2. When given enough time to perform its best, the GA can outperform both RL variants, and for a fixed running time, the GA achieves better results with smaller populations (e.g., 500).
3. RL2 generally reaches good results faster than RL1, but when both variants are given enough time to perform their best, RL1 tends to give better results.
4. Although SO provides higher objective values for linear problems, it generates a higher proportion of infeasible solutions when these are simulated with test sets, apart from also resulting in long running times, typical of large MILP problems.

Our LPM modelling using RL opens avenues for new research. One research track could be the enhancement of the problem setting, introducing, for example, resource constraints and redefining the agent's actions accordingly. Another track is the application of different RL methods, such as Q-Learning and function approximation.

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

Balouka, N., Cohen, I. and Shtub, A. (2016) 'Extending the multimode resource-constrained project scheduling problem by including value considerations', *IEEE Transactions on Engineering Management*, 63(1), pp. 4–15.

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.

Oehmen (Ed.), J. (2012) *The guide to lean enablers for managing engineering programs*. Joint MIT-PMI-INCOSE Community of Practice on Lean in Program Management.

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.