Robust Capacity Planning for Project Management

Antonio J. Conejo
Department of Integrated Systems Engineering, The Ohio State University, Columbus, OH 43210, conejo.1@osu.edu

Nicholas G. Hall
Department of Management Sciences, The Ohio State University, Columbus, OH 43210, hall.33@osu.edu

Daniel Zhuoyu Long
Department of Systems Engineering & Engineering Management, The Chinese University of Hong Kong, Shatin, Hong Kong, zylong@sse.cuhk.edu.hk

Runhao Zhang
Department of Systems Engineering & Engineering Management, The Chinese University of Hong Kong, Shatin, Hong Kong, zh.runhao@gmail.com

We consider a significant problem that arises in the planning of many projects. Project companies often use outsourced providers which require capacity reservations that must be contracted before task durations are realized. We model these decisions for a company which, given partially characterized distributional information, assumes the worst-case distribution for task durations. Once task durations are realized, the project company makes decisions about fast tracking and outsourced crashing, to minimize the total capacity reservation, fast tracking, crashing, and makespan penalty costs. We model the company’s objective using the target-based measure of minimizing an underperformance riskiness index. We allow for correlation in task performance, and for piecewise linear costs of crashing and makespan penalties. An optimal solution of the discrete, nonlinear model is possible for small to medium size projects. We compare the performance of our model against the best available benchmarks from the robust optimization literature, and show that it provides lower risk and greater robustness to distributional information. Our work thus enables more effective risk minimization in projects, and provides insights about how to make more robust capacity reservation decisions.

Key words: project management, capacity reservation, adjustable distributionally robust optimization, optimal algorithm

1. Introduction

The global economic value of projects exceeds $12 trillion annually, and represents over 20% of the world’s economic activity (Project Management Institute 2008). Moreover, in recent years, the range of applications that are managed as projects has expanded greatly, to include for example IT, research and development, new product and service development, pharmaceutical development, and change management. In response to the different characteristics of modern project management applications, new planning methodologies have been developed for project management. Wysocki
(2014) contrasts and compares traditional and modern project planning methods. Hall (2012, 2015, 2016) describes a variety of high level research questions for project management.

This work studies a financially significant problem at the planning stage of many projects. In order to respond to delays in task execution, crashing i.e. expediting, of tasks is needed. However, resources needed for crashing are outsourced, and their capacity must be reserved in advance. This is especially typical for high technology, pharmaceutical, and other specialized projects, due to long lead times in product certification and quality assurance. Once task durations are realized, the project company uses the previously reserved capacity to respond to any delays. For this problem, we develop and solve an adjustable distributionally robust optimization (ADRO) model. This two-stage model considers capacity reservation decisions, the possibility of delays to the tasks, and the use of previously reserved capacity to crash them. Decisions that are recommended by an optimal solution of our model provide robust performance against task time uncertainty.

Uncertainty in task time durations arises from three main sources in projects. The first is task time estimation errors due to, for example, limited data, poor estimation techniques, or deliberate overestimation (Okoro 2015). The second is variation in task performance due to, for example, miscommunication, unavailability of resources, unanticipated technical difficulty, scope changes by project owners, or quality problems leading to rework (Soni and Acharya 2015). The third is behavioral issues, such as Parkinson’s Law (Parkinson 1955), that affect task time performance.

Outsourcing is the procurement of products, services or capacity from external subcontractors. Organizations which run projects increasingly use outsourcing (Prahalad and Hamel 1990, Grossman and Helpman 2005, Oshri et al. 2015), for several reasons. First, they do not have the resources needed to support all the tasks in their projects, and it would be too expensive to maintain those resources in-house. Second, in many projects, it is essential that specialized tasks are performed by outside subcontractors. An example is creative or highly technical tasks to support new product development (Pisano and Verganti 2008). The use of outsourced capacity typically requires reservation, which has generated a substantial literature (for example, Aydinliyim and Vairaktarakis (2010), and Cai and Vairaktarakis (2012)). Further examples exist within options contracts (Cachon and Terwiesch (2006, p. 359)), as widely used in situations of uncertainty. To illustrate the practicality of capacity reservation contracts that support projects, we reference two examples from the business literature. In Securities and Exchange Commission (2007), p.109, an unspecified capacity reservation fee is used to require Cobra Biologics Limited to “reserve capacity and resources at its Facility”. In Securities and Exchange Commission (2014), p.20, Variation Biotechnologies, Inc.
agrees to pay Paragon Bioservices, Inc. $180,000, “in order to ensure sufficient development and manufacturing capacity / personnel resources ...”. These two contracts are taken from pharmaceutical development projects. In both cases, the pharmaceutical development company reserves capacity with an outsourced provider of product development and manufacturing services. These examples match the characteristics of our model, in that payment is made for capacity that eventually may or may not be needed. Also, the reservation of capacity provides the pharmaceutical development company with additional options.

Classical project planning methodologies such as PERT (DOD and NASA 1962) approximate a probability distribution for task time distributions, based on optimistic, pessimistic and most likely estimates. However, consistent industry and academic evidence (for example, Adler et al. 1995, Leach 1999, Williams 2003, Herroelen and Leus 2005) questions the reliability of this approach. Therefore, we assume only partially characterized task time distributions. Distributional ambiguity of this type suggests the use of robust optimization, which enables optimization under worst-case uncertainties (e.g., Soyster 1973, Ben-Tal and Nemirovski 1998, El Ghaoui et al. 1998, Bertsimas and Sim 2004). However, the simultaneous realization of all worst-case uncertainties is highly unlikely and results in overly conservative decision making, which motivates the use of distributionally robust optimization, as developed by Delage and Ye (2010), Goh and Sim (2010), and Wiesemann et al. (2014) for related applications.

Robust optimization has been applied to project management, to model crashing decisions during project execution (Cohen et al. 2007, Goh and Hall 2013) and project portfolio selection (Hall et al. 2015). However, the problem studied here involves a multi-stage setting and is called adjustable robust optimization (ARO). This problem is computationally intractable in general (Shapiro and Nemirovski 2005). Ben-Tal et al. (2004) propose the use of affine decision rules. For some special cases, this approach provides either optimal decisions (Bertsimas et al. 2010, Bertsimas and Goyal 2012, Iancu et al. 2013), or provably close to optimal decisions (Bertsimas and Goyal 2012).

The capacity reservation decision considered here is part of a planning, rather than an execution, process. Consequently, achieving a fast computation time is not the highest priority. A similar perspective is adopted by several related works. These works form another stream of literature on ARO, which designs algorithms for problems with a reasonable size. In particular, Bertsimas et al. (2013) describe an ARO model for energy unit commitment, using cuts generated from an inner minimization problem and a budget of uncertainty. Ruiz and Conejo (2015) study an energy
transmission problem using ARO. Zeng and Zhao (2013) develop a column-and-constraint generation algorithm for two-stage robust optimization problems; this algorithm dynamically generates constraints in the primal space of the problem. These works use only scenario information about uncertain factors. However, we also incorporate frequency information, which enables a more general characterization of uncertainties and more extensive use of historical data. Our uncertainty characterization generalizes typical information, such as mean, variance, and absolute deviation.

A two-stage setting is used in our paper because it models the scenario of an uncertain project environment that we are considering. Silver (1987), based on interviews with procurement managers in large-scale construction projects in the oil and gas industry, motivates capacity reservation at the first stage as a hedge against uncertainty about (a) requirements and/or timing, and (b) available supplier capacity, at a later stage. The optimized capacity reservation found by our model is purchased as a hedge against various possible external issues affecting the project environment that are potentially in play when the project needs to be executed (which is typically some time later). These issues include, for example, economic (such as a recession), financial (such as the availability of cheap credit), related to resources (such as shortages of expert labor or project-specific materials), legal (such as delays in obtaining planning permission), political (such as an unstable government), or societal (such as a public health emergency). Because of the substantial lead time required for capacity reservation, relative to the length of the project, we assume that the uncertain project environment factors do not change during the execution of the project. Indeed, those environmental issues naturally change only slowly over time, and we would not expect significant changes to occur during the execution of a single project. Further, since uncertainties in the project environment do not change during project execution, it is without loss of generality that we assume the task durations realize simultaneously and hence model the problem as having two stages. Nevertheless, there are also some practical problems where a subset of uncertain task duration realizes not before, but during the process of, deciding and implementing the fast tracking and crashing decisions. In that case, the decisions need to be made in a more dynamic way. Our model can be easily extended to a setting with multiple stages to incorporate this dynamic feature. However, this inevitably makes the problem much less computationally tractable and necessitates the use of heuristic methods, which is beyond the scope of this work.

Under the two-stage setting, the sequence of decisions and events in the project planning problem which we consider is as follows. First, the project company makes capacity reservation decisions of two types. For routine task work, such as standard business processes, providers of outsourcing
services offer blocks of capacity with announced prices, and the project company can reserve them. For technical components or specialized requirements of tasks, a request for a fixed amount of customized capacity is initiated by the project company. A task may require outsourced capacity of both types. Following capacity reservation decisions, we estimate the downside risk associated with the project environment and resulting in task delays, which may in turn delay the delivery of the project. The distribution of uncertain project environment factors is controlled by a set of constraints that model distributional robustness. The project company can respond by fast tracking (Passionate Project Management 2011, Monappa 2018), i.e. concurrent performance of tasks that are formally required to be performed in sequence, typically at the cost of additional rework. A typically more expensive option is crashing the tasks (Passionate Project Management 2011). However, this requires the use of previously reserved capacity. Therefore, the costs which we consider include the capacity reservation, fast tracking and crashing of tasks, and penalties for later makespan completion.

Risk aversion is important in many projects, since dysfunctional projects tend to compromise others (Cooper et al. 2000). There may also be a loss of future business as a result of an unsuccessful project. Performance measures for evaluating downside risk include expected utility (von Neumann and Morgenstern 1944), value-at-risk (VaR, see, for example, Jorion 2006) and coherent risk measures including CVaR (Artzner et al. 1999, Rockafellar and Uryasev 2000, 2002). However, all these approaches require the specification of a risk aversion parameter. In general, this parameter is abstract and difficult for a decision maker to calibrate, which is especially the case in project management since most projects are unique. We observe that in business decisions generally and project management in particular, a prevalent criterion is whether operational cost can be maintained under a target level (Rappaport 1999). Compared with the risk aversion parameter in expected utility or risk measure approaches, the target level is clearly and exogenously given. These issues of practicality and clarity recommend the use of a target-oriented approach (Simon 1955). A target-oriented decision criterion, based on a set of axioms, is proposed by Brown and Sim (2009) and applied to project management (Goh and Hall 2013, Hall et al. 2015). However, two additional considerations complicate our work. The first is that we are considering a two-stage optimization problem that requires binary decisions, which imposes considerable computational challenges. The second is that the costs of project delay are frequently nonlinear due to penalty clauses in contracts, for example in local government construction projects. For these reasons, we use a target-oriented decision criterion based on general piecewise linear utility, which balances the
practical aspects of project costs against the issue of computational solvability. With this approach, our work addresses the problem of achieving robust performance for high uncertainty projects under limited availability of information about the distributions of task durations.

The main contributions of our work are as follows:

1. We model the capacity reservation problem, including the practical features of fast tracking, crashing, and nonlinear completion time cost.

2. Our information set for task durations is general enough to model the practical issues of correlation and disperse information.

3. We develop a target-based decision making framework that preserves linearity where the optimal solution provides minimum risk to the predetermined benchmark cost.

4. We develop a column and constraint generation algorithm, where each iteration requires solution of a bilinear problem, which we reformulate as a mixed integer program with special structure which improves solvability.

5. Our computational study shows that the algorithm is capable of solving the problem optimally for small to medium size projects.

6. Our results outperform those found by the best available benchmark procedures.

7. We provide project managers with a planning tool for effective risk minimization in projects with high uncertainty, and guidance towards achieving more effective use of outsourced capacity.

Section 2 provides our notation, formally describes the problem studied, and discusses how we model uncertainty and risk. In Section 3, we develop an algorithm for finding optimal solutions to the problem. Section 4 demonstrates the computational efficiency of our algorithm, and compares its performance against several benchmark approaches. Finally, Section 5 describes insights for project managers, and identifies suggestions for future research. All proofs appear in the e-companion.

2. Model

In Section 2.1, we provide our notation and formally describe the problem we consider. In Section 2.2, we describe our modeling of the uncertainties in the problem and how to evaluate the riskiness of the uncertain cost.

2.1. Problem description and motivation

We use bold characters, e.g., \( \mathbf{z} \) and \( \mathbf{A} \), to represent vectors and matrices, respectively. Given any vector \( \mathbf{z} \), we let \( z_i \) denote its \( i \)th element. Random variables are denoted using the tilde symbol, for example \( \tilde{z} \), where \( z \) is the corresponding realization. The inequality between two random variables
is statewise, i.e., $\tilde{w}_i \geq \tilde{w}_j$ implies that the probability that $(\tilde{w}_i \geq \tilde{w}_j)$ is 1. We use brackets to represent a set of running indices, e.g. $[N] = \{1, 2, \ldots, N\}$.

We consider a project that consists of a collection of $N$ individual tasks with precedence relationships between them. We let $n_i \rightarrow n_j$ denote that task $n_i$ must be completed before task $n_j$ starts. Decisions about capacity reservation are made at the planning stage, based on assumptions about future realizations of those uncertainties. The sequence of decisions and events is as follows.

1. The project manager needs to decide how much of various types of capacity to reserve.
2. Uncertain task durations are assumed to be realized, as a result of realization of a general project environment.

3. The project manager responds by fast tracking and/or crashing the project, in order to minimize the cost of fast tracking, crashing, and makespan penalty.

We consider two types of outsourced capacity that are typically observed in projects. The first type is general capacity that can help to perform any task, for example administrative support. This type of capacity is offered in a given quantity-price pair list $(q_x, u_x) \in \mathbb{R}^{2 \times O}$, where $O$ is the number of quantity-price pairs on the list. If the $j$th available capacity package is chosen, $q_{xj}$ units of the general resource are reserved and the reservation cost $u_{xj}$ is paid, for $j \in [O]$. We let $x \in \{0, 1\}^O$ denote the capacity decision for the general resource, where $x_j = 1$ if the $j$th package is reserved, and $x_j = 0$ otherwise. Hence, the total cost of general capacity reserved is $u^T_x x$.

The second type of capacity is customized resource, applicable to a specific task, and reserved by request of the project company. Examples of such resources include contract manufacturing and specialized engineering or chemical services. Since the details of the task are known, the reservation decision is modeled as binary. Let $y \in \{0, 1\}^N$ and $u_y \in \mathbb{R}^N$ represent the reservation decision and reservation cost, respectively, for the customized capacity. We let $y_n = 1$ mean that a decision has been made to reserve customized capacity for task $n$, and $y_n = 0$ otherwise. Hence, the total cost of customized capacity reserved is $u^T_y y$. We let $(x, y) \in \Lambda$ represent the feasible set for the capacity reservation decision $(x, y)$. We assume that all constraints in $(x, y) \in \Lambda$ are linear, except for $x \in \{0, 1\}^O$ and $y \in \{0, 1\}^N$.

We denote the uncertain delays for the tasks by $\tilde{z}$, which realizes to $z \in \mathbb{R}^N$ after the capacity reservation decision $(x, y)$. Then, the project manager makes fast tracking and crashing decisions. As described in Section 1, when making the fast tracking and crashing decisions, the uncertain delays $\tilde{z}$ have fully realized. Therefore, the fast tracking and crashing decisions are made to respond to the actual realization of delays for all tasks, $z$, with the following costs.
The first cost component is the cost incurred by fast tracking, which is the concurrent processing of tasks that have a formal precedence relationship. Fast tracking typically incurs additional rework cost and possibly greater management cost (Mochal 2006). Since such costs are not defined contractually, we model them as linear in the amount of fast tracking. The second cost component is the crashing cost. Crashing costs are typically considered to be convex, because cheaper crashing options are exhausted before more expensive ones are initiated; a reasonable approximation is given by a piecewise linear time-cost tradeoff function (Vrat and Kriengkrairit 1986). Similar comments apply to penalties, or equivalently lost incentives, for delays in project makespan. These can also be modeled as piecewise linear and convex (Pinedo, 2009, p. 70), representing both a possible bonus from early completion and a possible penalty from late completion, relative to a target delivery date specified in the project contract. We do not consider the time value of money through discounting, since doing so would obscure the main tradeoff we are studying, which is between capacity reservation costs and operational costs. The objective is to minimize the total of capacity reservation cost and the three operational costs.

We assume that fast tracking is only constrained by an exogenous upper limit for any pair of tasks linked by a precedence constraint. However, the crashing decision of each task is constrained by both an exogenous upper limit and the earlier capacity reservation decisions. We assume that, for any task \( n, n \in [N] \), crashing of \( w_n \) units requires \( \eta_n w_n \) units of the task’s customized capacity and also \( \theta_n w_n \) units of general capacity. Once reserved, the customized capacity is enough; hence, without loss of generality, \( \eta_n \) can be considered as a binary indicator, where \( \eta_n = 0 \) if task \( n \) does not require any customized capacity. On the other hand, \( \theta_n \) is a continuous rate where \( \theta_n = 0 \) if task \( n \) does not require any general capacity.

We list the decision variables and parameters of the operational problem as follows.

**Decision variables:**
\[
\begin{align*}
  c_n &= \text{completion time of task } n, \ n \in [N] \\
  w_{n_1,n_2}^F &= \text{concurrent processing of tasks } n_1 \text{ and } n_2 \text{ due to fast tracking}, \ n_1, n_2 \in [N], \ n_1 \rightarrow n_2 \\
  w_{j,n}^C &= \text{usage of the } j\text{th linear cost interval of crashing for task } n, \ j \in [Q_n], \ n \in [N] \\
  v_j &= \text{usage of the } j\text{th linear cost interval of project makespan}, \ j \in [Q].
\end{align*}
\]

Here \( Q_n, \ n \in [N] \), is the number of linear segments in the piecewise linear mapping of the crashing cost of task \( n \). Similarly, \( Q \) is the number of linear segments of the makespan cost.

**Parameters:**
\[
\begin{align*}
  t_n &= \text{standard duration of task } n \text{ without random delay or crashing}, \ n \in [N]
\end{align*}
\]
\[ z_n = \text{uncertain duration of task } n \text{ without fast tracking and crashing, for } n \in [N] \]
\[ z_n = \text{realization of } \hat{z}_n, \ n \in [N] \]
\[ g_{n_1 n_2}^F = \text{linear rework cost of fast tracking of task } n_2 \text{ with task } n_1, \text{ for } n_1, n_2 \in [N], \ n_1 \rightarrow n_2 \]
\[ \gamma_{n_1 n_2} = \text{upper limit on concurrent processing of tasks } n_1 \text{ and } n_2, \text{ for } n_1, n_2 \in [N], \ n_1 \rightarrow n_2 \]
\[ g_{j n}^C = \text{linear cost of crashing task } n \text{ in the } j\text{th linear cost interval, for } j \in [Q_n], n \in [N], \text{ where} \]
\[ g_{1 n}^C < \cdots < g_{Q_n n}^C \text{ for } n \in [N] \]
\[ e_{j n}^C = \text{upper limit on the } j\text{th linear cost interval for crash cost of task } n, \text{ for } j \in [Q_n], n \in [N] \]
\[ \eta_n = \text{a binary indicator of whether a customized capacity is needed to crash task } n, \text{ for } n \in [N] \]
\[ \theta_n = \text{a usage rate such that general capacity } \theta_n w_n \text{ is needed to crash task } n \text{ by } w_n, \text{ for } n \in [N] \]
\[ d_j = \text{linear cost of project completion time penalty in the } j\text{th linear cost interval, for } j \in [Q], \text{ where} \]
\[ d_1 < \cdots < d_Q \]
\[ l_j = \text{upper limit on the } j\text{th linear cost interval for completion time penalty, for } j \in [Q - 1] \text{ (the last interval is assumed to be unbounded to ensure feasibility)} \]
\[ \Lambda = \text{the set of feasible capacity reservation decisions } (x, y). \]

To simplify the notation, we define the following vectors:

vectors \( w_F, g_F: w_F = (w_F^1, \ldots, w_F^N) \) and \( w_{n_2}^F = [w_{n_1 n_2}^F]_{n_1 \rightarrow n_2}, n_2 \in [N]; g_F \) is defined similarly;
vectors \( w_C, g_C: w_C = (w_C^1, \ldots, w_C^N), w_n^C = (w_{1 n}^C, \ldots, w_{Q_n n}^C), n \in [N]; g_C \) is defined similarly.

Given capacity reservation decisions \( x \) and \( y \), and uncertainty realizations \( z \), the project management planning problem we consider is

\[
\begin{align*}
\min & \quad g_F^T w_F + g_C^T w_C + d^T v \\
\text{s.t.} & \quad (w_F, w_C, v, c) \in S(x, y, z). \tag{1}
\end{align*}
\]

Here, \( S(x, y, z) \) is the set of feasible second stage decisions for \( (w_F, w_C, v, c) \), where the first stage decision is \( (x, y) \) and the uncertainties realize as \( z \), as defined by the following constraints.

\[
c_{n_2} - c_{n_1} + w_{n_1 n_2}^F + \sum_{j=1}^{Q_n} w_{j n_2}^C \geq t_{n_2} + z_{n_2}, \quad n_1, n_2 \in [N], \quad n_1 \rightarrow n_2 \tag{2}
\]
\[
\sum_{j=1}^{Q} v_j - c_n \geq 0, \quad n \in [N] \tag{3}
\]
\[
-\sum_{n=1}^{N} \theta_n \sum_{j=1}^{Q_n} w_{j n}^C \geq -q_x^T x \tag{4}
\]
\[
-\sum_{j=1}^{Q_n} w_{j n}^C \geq -e_{Q_n n}^C y_n, \quad n \in [N] \text{ if } \eta_n \neq 0 \tag{5}
\]
\[
-w_{n_1 n_2}^F \geq -\gamma_{n_1 n_2}, \quad n_1, n_2 \in [N], \quad n_1 \rightarrow n_2 \tag{6}
\]
\[-w^c_{jn} \geq -e^c_{jn}, \quad j \in [Q_n], \quad n \in [N]\] (7)
\[-v_j \geq -l_j, \quad j \in [Q - 1]\] (8)
\[c_n \geq 0, \quad n \in [N]\]
\[w^F_{n_1 n_2} \geq 0, \quad n_1, n_2 \in [N], \quad n_1 \rightarrow n_2\]
\[w^C_{jn} \geq 0, \quad j \in [Q_n], \quad n \in [N]\]
\[v_j \geq 0, \quad j \in [Q].\]

The objective in (1) minimizes the total of fast tracking cost, crashing cost and project makespan penalty. Constraints (2) in \(S(x, y, z)\) enforce precedence requirements between tasks, allowing for random durations, fast tracking and crashing. Constraints (3) ensure that all tasks finish by the project completion time, \(\sum_{j=1}^{Q} v_j\). Constraint (4) ensures that the total general capacity used for crashing is no greater than that reserved. Constraints (5) require that any customized capacity used to crash task \(n\) has been reserved, as required if \(\eta_n \neq 0\). Constraints (6) ensure that the fast tracking between any pair of jobs does not exceed the given limit. Constraints (7) and (8) define piecewise linear cost functions for crashing task \(n\) and the project makespan, respectively.

Apart from the cost of fast tracking, crashing, and penalty for makespan delay, the total cost also includes the cost of reserving capacity. Hence, the problem of minimizing the second stage cost is
\[
\hat{h}(x, y, z) = u^T_x x + u^T_y y + \min_{(w_F, w_C, v, c) \in S(z, x, y)} \left\{ g^T_F w_F + g^T_C w_C + d^T v \right\},
\]
where \(u_x\) and \(u_y\) denote the capacity reservation costs corresponding to \(x\) and \(y\), respectively.

Then, the overall cost minimization problem of the project company is
\[
\min_{(x, y) \in X} \hat{h}(x, y, z).
\]

We note that (10) is not yet well defined, because when the capacity reservation decision \(x, y\) needs to be made the actual value of task duration \(z\) is still uncertain. We now discuss how to model the uncertainties embedded in \(z\) and the resulting risk.

### 2.2. Uncertainties and risk

We model the uncertain duration \(\tilde{z}\) of each task as an affine function of bounded uncertain factors
\[
\tilde{\delta} = (\tilde{\delta}_1, \ldots, \tilde{\delta}_K), \text{ i.e.,}
\]
\[
\bar{z}_n = z^0_n + \sum_{k=1}^{K} \frac{\bar{z}^0_n}{\delta_k},
\]
where the factor coefficients $z_n^0, \ldots, z_n^K$ are known. We describe correlation between uncertain delays from different tasks using a factor-based model. As Hall et al. (2015) discuss, this model has particular value in project management applications, since (a) estimating the covariance matrix is difficult for unique projects due to a lack of historical data, (b) uncertain durations of different tasks often arise from common project management environment issues as discussed in Section 1, and (c) the project management problem itself is computationally challenging and this model preserves linear structure, which limits complexity.

We denote the probability space as $(\mathbb{P}, \Omega, \mathcal{F})$, where $\Omega$ is the set of all possible outcomes, $\mathcal{F}$ is the set of events that describe task durations, and $\mathbb{P}$ is a function that assigns probabilities to events. Due to limited historical data about similar projects (Ward and Chapman 2003), we assume that full distributional knowledge, i.e. $\mathbb{P}$, is not available. Instead, the distributional information $\mathbb{P}$ is only partially characterized. In particular, we describe the probability by $\mathbb{P} \in \mathcal{P}$, where

$$
\mathcal{P} = \left\{ \mathbb{P} : \begin{array}{l}
\mathbb{P} \left( \delta_k \in \{\delta_{k1}, \ldots, \delta_{kL_k}\} \right) = 1, \ k = 1, \ldots, K \\
\mathbb{E}_{\mathbb{P}} \left[ \chi_j^{S,k}(x) \right] \leq \sigma_j^{S,k}, \ k = 1, \ldots, K, \ j = 1, \ldots, J_{S,k} \\
\mathbb{E}_{\mathbb{P}} \left[ \chi_j^{C}(x) \right] \leq \sigma_j^{C}, \ j = 1, \ldots, J_C
\end{array} \right\}.
$$

(12)

We assume $\mathcal{P} \neq \emptyset$ to avoid a trivial case. In (12), $\delta_{k1}, \ldots, \delta_{kL_k}$ represent all potential realizations of $\delta_k$ from a discrete distribution. The use of a discrete distribution of task times is consistent with resource planning in projects using discrete time units (Ranjbahr et al. 2009, Artigues 2017, Martins 2017, PM Knowledge Center 2019). In practice, most project management software, for example Microsoft Project, uses discrete time units. We incorporate the information about any estimator of an individual uncertain factor in the second constraint in (12). Specifically, $\chi_j^{S,k} : \mathbb{R} \rightarrow \mathbb{R}$ can be any general function to estimate the uncertain factor $\delta_k$, and $\sigma_j^{S,k}$ is an upper bound on the estimation. The third constraint in (12) takes into account correlation information among any subset of uncertain factors. In particular, the function $\chi_j^{C} : \mathbb{R} \rightarrow \mathbb{R}$ is used to estimate the linear combination $(\xi_j^T \tilde{\delta} + \epsilon_j)$, where $\xi_j$ and $\epsilon_j$ are given parameters. Also, $\sigma_j^{C}$ is a given upper bound on the estimation. For tractability, we assume $\chi_j^{C}$ to be convex. The information set defined in (12) includes a broad class of practical information. We now show three relevant examples which can be represented as special cases of $\mathcal{P}$ as defined in (12).

**Example 1** The range of mean for $\tilde{\delta}_k$. We let $\chi_j^{S,k}(x) = x \ \forall x \in \mathbb{R}$ for certain $j$, and $\sigma_j^{S,k} = \mu_k$. Then, the second constraint in (12) for this $k$ and $j$ becomes

$$
\mathbb{E}_{\mathbb{P}} \left[ \tilde{\delta}_k \right] \leq \mu_k,
$$


which defines the upper bound for the mean of $\delta_k$. Similarly, if we let $\chi_{j}^{S,k}(x) = -x \forall x \in \mathcal{R}$ and $\sigma_{j}^{S,k} = -\mu_k$, the constraint becomes

$$\mathbb{E}_{\mathcal{P}} \left[ \hat{\delta}_k \right] \geq \mu_k,$$

which defines the lower bound for the mean of $\delta_k$.

**Example 2** The range of any estimator for $\delta_k$, which generalizes the previous example. We let $\chi_{j}^{S,k}$ denote the underlying function of any estimator, such as $\chi_{j}^{S,k}(x) = x^2$, for $x \in \mathcal{R}$. We then incorporate the information about upper and lower bounds, for the estimation of uncertain factor $\delta_k$. Indeed, since there is no restrictions on the function $\chi_{j}^{S,k}$; we can also incorporate information for some nonconvex estimators, such as $\chi_{j}^{S,k}(x) = x^3$ for the third moment.

**Example 3** Absolute deviation. For a given $j$, we let the function $\chi_{j}^{C}(x) = |x|$, and let $\xi_j$ and $\epsilon_j$ be such that $\xi_j^T \delta + \epsilon_j = \sum_{k \in K_j} \frac{\hat{\delta}_k - \mu_k}{\sigma_k}$. Here $K_j \subseteq [K]$ is a given subset which contains the indices of the related uncertain factors. The third constraint in (12) for this $j$ is then

$$\mathbb{E}_{\mathcal{P}} \left[ \sum_{k \in K_j} \frac{\hat{\delta}_k - \mu_k}{\sigma_k} \right] \leq \sigma_{j}^{C}.$$

This design emulates the budget of uncertainty in Bertsimas and Sim (2004), where uncertain factors are unlikely to realize in the extreme case simultaneously. The special case where $K_j$ is a singleton is recently studied by Postek et al. (2015).

**Remark 1** Many alternative descriptions of the set of available distributions $\mathcal{P}$ are possible, including information on moments (Popescu 2007, Delage and Ye 2010, Zymler et al. 2013), $\phi$-divergence (Ben-Tal et al. 2013, Wang et al. 2016, Jiang and Guan 2016), Wasserstein distance (Esfahani and Kuhn 2017, Gao and Kleywegt 2018), and a general framework (Wiesemann et al. 2014). However, the requirement for optimal solvability disqualifies several of these, since they typically lead to conic programming and add too much complexity to a two-stage problem with binary variables. We use discrete support for the task durations to be consistent with project management practice and obtain a finite-dimensional problem; the second and third constraints in (12) model individual and joint dispersion while maintaining a linear structure. We show below that the problem can be solved efficiently for small to medium size projects.

We demonstrate the dependence of the total cost on uncertain factors. We define the function

$$h(x, y, \delta) = \hat{h}(x, y, z^0 + Z\delta)$$

(13)
to represent the total cost corresponding to capacity reservation decisions \((x, y)\), and uncertain factor realizations \(\delta\). We define the vector \(z^0 = (z^0_1, \ldots, z^0_N)\), and the matrix \(Z \in \mathbb{R}^{N \times K}\) such that the element in the \(n\)th row and \(k\)th column is \(z^0_n, n \in [N], k \in [K]\).

Given any capacity reservation decisions \((x, y)\), the corresponding total cost is \(h(x, y, \delta)\), which is a random variable depending on the elementary uncertainties \(\delta\). As discussed in Section 1, we assume that the decision maker has a benchmark cost, \(\tau\), and her objective is to minimize the risk that the cost exceeds \(\tau\). In order to incorporate the effect of the target \(\tau\), we evaluate a random cost by its loss relative to this target. We define \(\mathcal{W} = \{\hat{w} : \hat{w} = \tilde{v} - \tau, \tilde{v} \in \mathcal{V}\}\) as the set of all possible losses with respect to the target, where \(\mathcal{V}\) is the set of all possible costs. Therefore, a loss \(\hat{w}\) with \(P(\hat{w} \leq 0) = 1\) indicates that the cost is never higher than the target \(\tau\).

Following Brown and Sim (2009), and Hall et al. (2015), the target-oriented objective is as follows.

**Definition 1** A function \(\rho : \mathcal{W} \to [0, \infty]\) is an Underperformance Riskiness Index (URI) if, for all \(\hat{w}, \hat{w}^o \in \mathcal{W}\), it satisfies the following properties:

1. **Monotonicity**: if \(\hat{w} \leq \hat{w}^o\), then \(\rho(\hat{w}) \leq \rho(\hat{w}^o)\).

2. **Satisficing**:
   
   (a) **Attainment Content**: if \(\hat{w} \leq 0\), then \(\rho(\hat{w}) = 0\);

   (b) **Starvation Aversion**: if \(\hat{w} > 0\), then \(\rho(\hat{w}) = \infty\).

3. **Convexity**: \(\rho(\lambda \hat{w} + (1 - \lambda)\hat{w}^o) \leq \lambda \rho(\hat{w}) + (1 - \lambda)\rho(\hat{w}^o)\), for all \(\lambda \in [0, 1]\).

4. **Positive Homogeneity**: \(\rho(\lambda \hat{w}) = \lambda \rho(\hat{w})\), for all \(\lambda > 0\).

Monotonicity indicates that if the cost \(\hat{w}^o\) is always higher than \(\hat{w}\), then it is associated with higher risk. Satisficing implies that a random cost has no risk if it is always lower than the target; by contrast, it has infinite risk if the cost is always strictly greater than the target. Convexity is motivated by a preference for diversification. Positive Homogeneity establishes the cardinal nature of risk, e.g. \(2 \hat{w}\) is as twice as risky as \(\hat{w}\). Apart from its justification from the above properties, the URI can be dually represented by classical convex risk measures, as we now show.

**Theorem 1** (Hall et al. 2015) A function \(\rho : \mathcal{W} \to [0, \infty]\) is a URI if and only if it has the representation

\[
\rho(\hat{w}) = \inf \left\{ \alpha : \psi \left( \frac{\hat{w}}{\alpha} \right) \leq 0, \ \alpha > 0 \right\},
\]

(14)
where we define \( \inf \emptyset = \infty \) by convention and \( \psi \) is a normalized convex risk measure. That is, it satisfies the properties of Monotonicity, Cash Invariance, Convexity, and has \( \psi(0) = 0 \). Conversely, given a URI \( \rho \), the underlying normalized convex risk measure is given by

\[
\psi(\bar{w}) = \min\{ a : \rho(\bar{w} + a) \leq 1 \}.
\]  

We consider a special class of convex risk measure, which is a shortfall risk measure (Föllmer and Schied 2002). We redefine it under distributional ambiguity.

**Definition 2** A shortfall risk measure is any function \( \psi_u : W \to \mathbb{R} \) defined by

\[
\psi_u(\bar{w}) = \inf_{\iota \in \mathbb{R}} \left\{ \sup_{\bar{p} \in \mathcal{P}} \mathbb{E}_\bar{p}[u(\bar{w} - \iota)] \leq 0 \right\},
\]

where \( u : \mathbb{R} \to \mathbb{R} \) is an increasing convex differentiable function normalized by \( u(0) = 0 \) and \( u'(0) = 1 \). Correspondingly, using the dual representation in Theorem 1, we define the utility-based URI, \( \rho_u : W \to [0, \infty] \), as one based on a shortfall risk measure as follows:

\[
\rho_u(\bar{w}) = \inf \left\{ \alpha : \sup_{\bar{p} \in \mathcal{P}} \mathbb{E}_\bar{p}\left[ u\left(\frac{\bar{w}}{\alpha}\right)\right] \leq 0, \alpha > 0 \right\}.
\]  

It can easily be shown that \( \psi_u \) defined by (16) is a convex risk measure. To preserve linearity of the problem, we assume

\[
u(w) = \max_{m \in [M]} \{ a_m w + b_m \}
\]

with \([M] = \{1, 2, \ldots, M\}\), and given parameters \( 0 \leq a_1 \leq a_2 \leq \cdots \leq a_M \), \( b_1, \ldots, b_M \in \mathbb{R} \). We observe that this piecewise-linear function can be used to approximate any general utility function. However, we remark that there are cases where the underlying utility is piecewise linear, as in Equation (18), and in those cases we are finding the exact optimal solution.

Then, from (17) and (18), the problem of minimizing the URI of overall uncertain cost is

\[
\alpha^* = \inf \alpha \\
\text{s.t.} \sup_{\bar{p} \in \mathcal{P}} \mathbb{E}_\bar{p}\left[ \max_{m \in [M]} \left\{ \frac{h(x, y, \delta)}{\alpha} - \frac{a_m}{\alpha} + b_m \right\} \right] \leq 0
\]

\[
\alpha > 0 \\
(x, y) \in \Lambda.
\]

While the construction of the utility-based URI also relies on a utility function, it is worth mentioning that it still simplifies the parameter calibration. For example, if using piecewise linear utility \( u(w) = \max\{w/\alpha, -1\} \) for some constant \( \alpha > 0 \), then in the expected utility approach we
need to calibrate \( \alpha \). By contrast, in the URI approach, \( \alpha \) is a decision variable and is optimized according to the target. In general, in using the expected utility approach, we need to calibrate both the shape and the level of convexity of the utility function, while the latter is actually implied by the target in the URI approach. The general form of a URI is proposed by Brown and Sim (2009). Hall et al. (2015) adopt a specific URI, which is based on an exponential utility function, and exploit special properties of the exponential function to solve a project portfolio selection problem. Qi (2017) studies an appointment scheduling problem using the URI based on CVaR since it approximates the chance constraint in appointment scheduling. However, neither the general form nor the specific form of URI mentioned above can be efficiently optimized in our two-stage capacity reservation problem. Therefore, we propose a URI that is based on a piecewise linear function. As a result, it incorporates the impact of target and risk aversion and preserves a linear structure. Practically, the decision makers can determine the underlying utility function based on their general perception of cost. Alternatively, the utility function can be determined via the connection between the utility-base URI and the probability guarantee (e.g., Hall et al. 2015). If the underlying utility function is not piecewise linear, we can approximate it with a piecewise linear one based on a tradeoff between accuracy of approximation and the computational solvability.

The following two results discuss the formal tractability of problem (10) in the deterministic case, i.e. where all probability distributions are single-point.

**Theorem 2** The recognition version of problem (10) with no uncertainty is unary NP-complete.

Theorem 2 establishes that, even without uncertainty, the existence of an efficient algorithm for problem (10) is highly unlikely (Garey and Johnson 1979). Recall that the precedence constraints of a project are \emph{chain} if each task has exactly one direct predecessor and one direct successor. Chain precedence constraints can be used to characterize new product development processes (Roemer et al. 2000). For this special case, we have the following result.

**Remark 2** Problem (10) without uncertainty is solvable in pseudopolynomial time if the precedence constraints are chain. Further, the recognition version of this problem is binary NP-complete.

3. Solution Procedure

It is difficult to solve problem (19) directly for several reasons. First, the problem involves several levels of optimization; hence, given any feasible reservation decision \( (x, y) \), even evaluating
the objective value is problematic. Second, the evaluation of $\sup_{\varphi \in \mathcal{P}}$ requires a high-dimensional optimization. Third, as shown in Theorem 2, the underlying deterministic problem is intractable.

We develop a procedure to solve problem (19). The first step is to ensure the closedness of the feasible set of the optimization problem, which is not guaranteed in problem (19) since there is a constraint of $\alpha > 0$. Hence, we analyze an $\epsilon$-closure of problem (19) instead.

**Proposition 1** For any $\epsilon > 0$, we have $\alpha^* \leq \alpha^*_\epsilon \leq \alpha^* + \epsilon$, where

$$
\alpha^*_\epsilon = \min \alpha \\
\text{s.t. } \sup_{\varphi \in \mathcal{P}} \mathbb{E}_\varphi \left[ \max_{m \in [M]} \left\{ a_m h \left( x, y, \tilde{\delta} \right) - a_m \tau + b_m \alpha \right\} \right] \leq 0
$$

From Proposition 1, the optimal objective value of problem (20) is arbitrarily close to that of problem (19). We therefore focus on solving problem (20).

In solving problem (20), it is necessary to evaluate $\sup_{\varphi \in \mathcal{P}} \mathbb{E}_\varphi \left[ \max_{m \in [M]} \left\{ a_m h \left( x, y, \tilde{\delta} \right) - a_m \tau + b_m \alpha \right\} \right]$. That is, given capacity reservation decisions $(x, y)$, we need to evaluate performance under the worst-case distribution. We consider this problem as follows.

**Lemma 1**

$$
\begin{align*}
&\sup_{\varphi \in \mathcal{P}} \mathbb{E}_\varphi \left[ \max_{m \in [M]} \left\{ a_m h \left( x, y, \tilde{\delta} \right) - a_m \tau + b_m \alpha \right\} \right] \\
= &\min \tau_0 + \sum_{k=1}^{K} \sigma^{S,k} \tau^{S,k} + \sigma^{CT} \tau^C \\
\text{s.t. } &a_m u^m_y x + a_m u^m_y y + a_m g^F_k w^i + a_m g^C_k w^i_C + a_m d^T v - a_m \tau + b_m \alpha \\
&\leq \tau_0 + \sum_{k=1}^{K} \sum_{j=1}^{J_{S_k}} \omega^{S,k}_{j} \tau^{S,k}_j + \sum_{j=1}^{J_{C}} \omega^{C,i}_{j} \tau^C_j, \quad i \in [I], \ m \in [M] \\
&\tau^C, \ \tau^{S,k} \geq 0, \ k \in [K],
\end{align*}
$$

where we let $\delta^i, i = 1, \ldots, I$ represent all arbitrarily indexed elements of the set $Q$ defined as

$$Q = \{ \delta : \delta_k \in \{ \delta_{k1}, \ldots, \delta_{kL_k} \}, \ k \in [K] \}, \quad (22)$$

and correspondingly,

$$
\omega^{S,k}_{j} = \chi^{S,k}_{j} (\delta^i), \\
\omega^{C,i}_{j} = \chi^{C}_{j} (\xi^T \delta^i + \epsilon_j),
$$

are given parameters and $\tau_0, \tau^{S,k}, \tau^C, w^i, w^i_C, v^i, c^i, k \in [K], \ i \in [I]$, are decision variables.
Lemma 1 enables the following reformulation of the overall problem.

**Theorem 3** The overall problem (20) is equivalent to the following mixed-integer program

\[
\begin{align*}
\min \ & \alpha \\
\text{s.t.} \ & \tau_0 + \sum_{k=1}^{K} \sigma^{S,k} \tau^{S,k} + \sigma^{C,T} \tau^C \leq 0 \\
& a_m u^T z + a_m u^T y + a_m g_f^T w_f + a_m g_c^T w_c + a_m d^T v - a_m \tau + b_m \alpha \\
& \leq \tau_0 + \sum_{k=1}^{K} \sum_{j=1}^{J_{S,k}} \omega_j^{S,k} n_j \tau^{S,k} + \sum_{j=1}^{J_{C}} \omega_j^{C,i} n_j \tau^C, \ i \in [I], \ m \in [M] \\
& (w_f, w_c, v^i, c^i) \in S(z^0 + Z \delta^i, x, y), \ i \in [I] \\
& \tau^C, \ \tau^{S,k} \geq 0, \ k \in [K] \\
& \alpha \geq \epsilon \\
& (x, y) \in \Lambda.
\end{align*}
\]  

(23)  
(24)  
(25)  
(26)  
(27)  
(28)  
(29)

Since \( I \) is the number of elements in \( Q \) which is exponentially large, problem (23)–(29) has many decision variables and constraints. Hence, we apply a column and constraint generation method.

**Definition 3** Let problem (\( R \)) denote problem (23)–(29) where the constraints defined over the set \( [I] \) are replaced by a subset of constraints defined by \([I^o] \) with \( I^o \leq I \).

Note that, with fewer constraints, problem (\( R \)) has a feasible set containing that of the original problem (23)–(29) as a subset. If the solution of problem (\( R \)) is feasible for the original problem (23)–(29), then we stop. Otherwise, we increase the value of \( I^o \), and solve the relaxed problem again.

We consider the case where the function \( \chi_j^C \) in the distributional information set defined in (12) is piecewise linear and convex, for all \( j \). Specifically, let

\[
\chi_j^C (x) = \max_{m \in [M_j]} \{ r_j^m x + \vartheta_j^m \},
\]  

(30)

where \( r_j^m \) and \( \vartheta_j^m \) are given parameters.

Now, we demonstrate how to check the feasibility of constraints (25) and (26). We define a bilinear programming problem which is used for this purpose. Observe that \( S(z, x, y) \) defines a polyhedron of \( (w_f, w_c, v, c) \), where the constraints are also linear in \((x, y)\). Following the definition of \( S(z, x, y) \), we find matrices \( A_F, A_C, A_v, A_c, A_x, A_y, A_z \), and vector \( b^o \) such that

\[
S(z, x, y) = \left\{ (w_f, w_c, v, c) : A_F w_f + A_C w_c + A_v v + A_c c \geq A_x x + A_y y + A_z z + b^o \right\}
\]  

(31)
Based on these matrices, we let $P$ denote the polyhedron defined by

$$P = \{ p \geq 0 : p^T (A_F A_C A_c) \leq (g_F g_C d^T 0^T) \}.$$ 

Further, for any $m \in [M]$, we define a function $q^*_m(x, y, \tau^C, \tau^{S,1}, \ldots, \tau^{S,K})$ as the optimal value of an optimization problem parameterized by $(x, y, \tau^C, \tau^{S,1}, \ldots, \tau^{S,K})$ as follows:

$$q^*_m(x, y, \tau^C, \tau^{S,1}, \ldots, \tau^{S,K}) = \max_{p \in P, \delta \in \mathcal{Q}} q_m(p, \delta, x, y, \tau^C, \tau^{S,1}, \ldots, \tau^{S,K})$$

where

$$q_m(p, \delta, x, y, \tau^C, \tau^{S,1}, \ldots, \tau^{S,K}) = -\sum_{k=1}^K \sum_{j=1}^{J_C} \tau_j^{S,k} x_j^k (\delta_k) - \sum_{j=1}^{J_C} \tau_j^C x_j (\xi_j^T \delta + \epsilon_j) + a_m \delta^T (A_z Z)^T p + a_m (A_x x + A_y y + A_z z^0 + b^*)^T p$$

has a bilinear term in the decision variables $p$ and $\delta$.

The next result establishes necessary and sufficient conditions for a solution to the relaxed problem (R) to be feasible for constraints (25) and (26).

**Lemma 2** Consider any feasible solution $x, y, \tau^{S,1}, \ldots, \tau^{S,K}, \tau_C, \tau_0, \alpha$ for the relaxed problem (R). If $\forall m \in [M],$

$$q^*_m(x, y, \tau^C, \tau^{S,1}, \ldots, \tau^{S,K}) \leq \tau_0 + a_m \tau - b_m \alpha - a_m u_x^T x - a_m u_y^T y,$$

then this solution is feasible for constraints (25) and (26). Otherwise, this solution is not feasible for constraints (25) and (26), since for any $m_* \in [M]$ and $i^* \in [I]$ with

$$\max_{p \in P} q_{m_*}(p, \delta^{i^*}, x, y, \tau^C, \tau^{S,1}, \ldots, \tau^{S,K}) > \tau_0 + a_m \tau - b_m \alpha - a_m u_x^T x - a_m u_y^T y,$$

there does not exist $(w^x_F, w^y_C, v^{i^*}, c^{i^*}) \in \mathcal{S}(z^0 + Z \delta^{i^*}, x, y)$ with

$$a_m u_x^T x + a_m u_y^T y + a_m g_F^T w^x_F + a_m g_C^T w^y_C + a_m d^T v^{i^*} - a_m \tau - b_m \alpha$$

$$\leq \tau_0 + \sum_{k=1}^K \sum_{j=1}^{J_C} \omega_j^{S,k,i^*} \tau_j^{S,k} + \sum_{j=1}^{J_C} \omega_j^C \tau_j^C,$$

i.e., constraints (25) and (26) are not satisfied for $m = m_*$ and $i = i^*$.

Lemma 2 provides a stopping rule and an improvement direction, for our Column and Constraint Generation (CCG) algorithm below. Every time after solving the relaxed problem, we check whether $q^*_m \leq \tau_0 + a_m \tau - b_m \alpha - a_m u_x^T x - a_m u_y^T y$ for all $m \in [M]$. If so, then we stop. Otherwise, we need to find $m \in [M]$ such that any solution of the corresponding bilinear problem (32) with objective value
greater than \((\tau_0 + a_m \tau - b_m \alpha - a_m u_x^T x - a_m u_y^T y)\) and add it to the constraints. It now remains to solve the bilinear problem (32).

While bilinear problems are in general NP-hard, the complexity of problem (32) can be reduced and hence we can solve it optimally, as we discuss below.

**Lemma 3** The bilinear programming problem (32) is equivalent to the following mixed-integer programming problem,

\[
\begin{align*}
\max \quad & a_m c^T p - \sum_{k=1}^{K} \sum_{j=1}^{J_S} \sum_{l=1}^{L_k} \sum_{k'=1}^{L_{k'}} \sum_{k''=1}^{L_{k''}} \chi_{j}^{S,k} (\delta_{kl}) \phi_{kl} - \sum_{j=1}^{J_C} \tau_j^{C} \Omega_j + a_m \sum_{k=1}^{K} \sum_{l=1}^{L_k} \zeta_{kl} \\
\text{s.t.} \quad & p \in P \\
& \sum_{l=1}^{L_k} \phi_{kl} = 1, \quad k \in [K] \\
& \phi_{kl} \in \{0, 1\}, \quad k \in [K], \quad l \in [L_k] \\
& \zeta_{kl} \leq \delta_{kl} b_k p + \Delta (1 - \phi_{kl}), \quad k \in [K], \quad l \in [L_k] \\
& \zeta_{kl} \leq \Delta \phi_{kl}, \quad k \in [K], \quad l \in [L_k] \\
& \Omega_j \geq \tau_j^m \left( \sum_{k=1}^{K} \sum_{l=1}^{L_k} \phi_{kl} \delta_{kl} + \epsilon_j \right) + \varphi_j^m, \quad m \in [M_j], \quad j \in [J_C], \\
\end{align*}
\]

where we let \(B = (A_z Z)^T\), let \(b_k\) denote its \(k\)th row vector, \(c = A_x x + A_y y + A_z z^0 + b^o\), and let \(\Delta\) be a sufficiently large constant.

The value of \(\Delta\) in (35) is set sufficiently large to enforce the constraints on \(\zeta_{kl}\). Practically, we can adopt a similar approach to Gabrel et al. (2014). Thus, we solve \(\Delta_{h} = \max_{p \in P} b_k p\), which is a linear optimization problem, and then \(\Delta\) can be chosen as \(\max\{\delta_{kl} \Delta_k : l = \Delta_k [L_k], k \in [K]\}\).

**Remark 3** Recall that all bilinear optimization problems have an optimal solution that is an extreme point of the feasible region, hence they can be equivalently formulated as a mixed integer program using the approach in Lemma 3. In general, this approach requires one auxiliary binary decision variable for each extreme point, the number of which is exponentially large. Hence, the corresponding mixed integer program is still computationally intractable and the reformulation does not greatly help to solve the bilinear optimization problem. However, in problem (32), since the constraints on \(\delta\) can be decomposed into one constraint for each uncertain factor \(\delta_k\), \(k \in [K]\), the number of auxiliary binary decision variable reduces to \(L_1 + \ldots + L_K\). When the distributional information \(\mathcal{P}\) contains only bound information, a similar approach is proposed to formulate the bilinear problem into a mixed integer program; see, for example, Jiang et al. (2012), and Zeng and Zhao (2013). However, in Lemma 3, we extend this approach to allow \(\mathcal{P}\) to contain frequency and scenario information.
We are now ready to present our algorithm, CCG, for solving the overall problem.

Algorithm CCG

1. Initialize $I^o = 1$, and $\mathbf{\delta}^1 = (\delta_{11}, \ldots, \delta_{KL})$.
2. Solve the relaxed problem $(R)$, and denote the solution by $\mathbf{x}, \mathbf{y}, \mathbf{\tau}^{S,1}, \ldots, \mathbf{\tau}^{S,K}, \mathbf{\tau}^C, \tau_0$, and $\alpha$. In addition, set $m = 1$.
3. Define an instance of the bilinear problem (32) using the solution found in Step 2 and the current value of $m$. Solve problem (32) by solving the equivalent mixed integer programming problem (35); output the objective value $O_b$ and an optimal solution of $\mathbf{\delta}$ as $\mathbf{\delta}^*$. 
4. If $O_b > \tau_0 + a_mT - b_m\alpha - a_m\mathbf{u}_T^T\mathbf{x} - a_m\mathbf{u}_T^T\mathbf{y}$, then set $I^{m+1} = \mathbf{\delta}^*, I^o = I^o + 1$, and go to Step 2.
5. If $m < M$, then set $m = m + 1$, and go to Step 3.
6. Output $\mathbf{x}, \mathbf{y}$ as optimal decisions, and stop.

Theorem 4 Algorithm CCG solves the overall problem optimally in a finite number of steps.

Remark 4 When the function $\chi^C_j$ is not piecewise linear but an arbitrary convex function, the problem can still be solved using a procedure similar to Algorithm CCG. The only difference is that, given a solution to the relaxed problem $(R)$, we would need a more complex procedure to verify its feasibility for the original problem (23)-(29). In particular, when the function $\chi^C_j$ is not piecewise linear, we can no longer use the MILP (35) since the last constraint in (35) is replaced by $\Omega_j \geq \chi^C_j(\sum_{k=1}^K \xi_k \sum_{l=1}^{I_k} \phi_{kl} \delta_{kl} + \epsilon_j)$, which has nonlinear structure since $\chi^C_j$ is not piecewise linear. However, as $\mathbf{\delta}$ can take only a finite number of values due to the use of a discrete distribution, we can still linearize this constraint by enumerating all possible $\mathbf{\delta}$ values; hence, we still obtain a MILP. This MILP can also be solved using a column and constraint generation method.

4. Computational Study

In this section, we describe a computational study of Algorithm CCG. The objectives of this study are as follows. In Section 4.1, we test the computational efficiency of Algorithm CCG, to establish the size of projects for which it can provide optimal solutions. In Section 4.2, we compare the performance of decisions recommended by our algorithm against those from four alternative procedures that serve as benchmarks. Section 4.3 includes an analysis of sensitivity to the most important performance factors. Section 4.4 studies the robustness of the performance of those decisions against incorrect assumptions about the distribution of task time durations. Section 4.5
demonstrates the computational advantage of Algorithm CCG over direct solution. All data and results in this section are available on GitHub\textsuperscript{1}.

4.1. Computational efficiency

We study the computational efficiency of Algorithm CCG for projects with various characteristics. As recommended by Hall and Posner (2001), our experiment includes a number of instances with different parameters that may affect performance. Specifically, we use the 1250 project management instances from the webpage of Operations Research & Scheduling Research Group at http://www.projectmanagement.ugent.be. We study computational efficiency as a function of the number of tasks, \( N \), and the depth indicator of the network, \( \Gamma = (\varrho - 1)/(N - 1) \), where \( \varrho \) is the length of the longest chain. In the dataset of 1250 project instances, the number of tasks in a project takes values in \{12, 22, 32, 42, 52\}. Also, \( \Gamma \) takes a wide range of values. We divide the instances with the same \( N \) value into two similar size groups, with \( \Gamma < \Gamma_0 \) and \( \Gamma \geq \Gamma_0 \). All instances are solved using CPLEX 12 on a computer with an Intel Xeon E5 processor and 64GB RAM.

Table 1 reports the mean and maximum computation time in seconds for the instances corresponding to each group. Within a few hours of computation, Algorithm CCG solves project management instances with up to 52 tasks. When \( N \) takes a relatively large value such as 52, the computation time increases with the depth indicator. But this effect is less obvious for small values of \( N \). Since Algorithm CCG is designed to be used at the planning stage of projects, rather than in real time, these computation times are reasonable enough to be useful in practice.

<table>
<thead>
<tr>
<th>( N )</th>
<th>Average time (in seconds)</th>
<th>Maximum time (in seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( \Gamma &lt; \Gamma_0 )</td>
<td>( \Gamma \geq \Gamma_0 )</td>
</tr>
<tr>
<td>12</td>
<td>58</td>
<td>37</td>
</tr>
<tr>
<td>22</td>
<td>86</td>
<td>84</td>
</tr>
<tr>
<td>32</td>
<td>80</td>
<td>47</td>
</tr>
<tr>
<td>42</td>
<td>89</td>
<td>83</td>
</tr>
<tr>
<td>52</td>
<td>192</td>
<td>349</td>
</tr>
</tbody>
</table>

Table 1  Computation time for various projects.

4.2. Comparison with benchmarks

We first describe four approaches to risk minimization that can be used as alternatives to Algorithm CCG. These approaches are apparently the best available alternatives, and are used as benchmarks in our computational study. The first benchmark is to minimize the expected cost. The second is to use the mean value of task durations deterministically, and make capacity reservation

\textsuperscript{1}https://github.com/danielz-long/robust-capacity-planning
decisions using that information. The third and fourth approaches are both to optimize the cost in the worst scenario of the uncertainty realization. Under this criterion, the third approach of Zeng & Zhao (2013) solves the exact optimal solution, whereas the fourth approach of Bertsimas et. al. (2013) uses a heuristic to obtain a sub-optimal solution with higher computational efficiency.

In order to provide an objective comparison between our approach and the benchmark approaches, we use the 240 project management instances from the webpage of Operations Research & Scheduling Research Group. The instances are generated as described there. In each project instance, the number of tasks is either 27 or 52, and the number of immediate precedence relationships varies from 56 to 257. The underlining task duration, $t_i$, is specified by the instance data as well. The other parameters are specified as follows.

To study the role of capacity reservation, we do not incorporate fast tracking decisions, since they require no capacity. Thus, we set the upper limit of fast tracking, $\gamma$, to 0. Similarly, we let $\Lambda$, the feasible set for the capacity reservation decisions, be the set of all binary vectors.

For crashing, let $g_e = 0$ and $e_c$ be large values that make their respective constraints redundant. For the general capacity, we define a price list containing five prices with quantity discounts, such that when the purchasing quantity is $q_{xi} = i \times 1't$, the per-unit price is $0.01 \times (1 - 0.05 \times (i - 1))$ and hence $u_{xi} = q_{xi} \times 0.01 \times (1 - 0.05 \times (i - 1))$, $i = 1, \ldots, 5$. For the customized capacity reservation, we let the cost be $u_p = 10 \times 1$, and let $\eta = 1$ such that all crashing requires customized capacity.

We assume there are three uncertain factors, $\tilde{\delta}_i, i = 1, 2, 3$, which follow an independent and identical beta-binomial distribution with $n = 6, \alpha = 1, \beta = 3$. We let the delay of each task be $\tilde{z}_j = t_j \tilde{\delta}_1 + t_j \tilde{\delta}_2 + t_j \tilde{\delta}_3$, where $t_j$ is the underlining task duration in the instance. Therefore, we have three underlying uncertain factors, and each of them can take seven different values.

Finally, we define the cost for delay in makespan to have a two-piece linear structure, i.e., $Q = 2$. Specifically, we let the breakpoint $l_1$ be four times the makespan when there is no delay; the cost rate for makespan lower than $l_1$ is $d_1 = 1$, and beyond $l_1$ it is $d_2 = 3$.

For each instance, we perform the following steps:

a. We estimate the information set for uncertain factors. For individual information, we use scenario, mean, and variance. For the joint information, we use the mean dispersion. Specifically,
We solve the first stage solutions using five solution approaches: ADRO, minimization of expected cost, use of expected parameter values as deterministic, the two worst-scenario robust optimization approaches (Zeng & Zhao 2013, Bertsimas et al. 2013). For the ADRO approach, the target is pre-determined as 120% of the minimal expected cost, the underlying utility function is chosen as \( u(w) = \max\{x, -1\} \). The solutions define the capacity reservation.

b. To compare the resulting costs, we simulate the realization of the uncertainty based on the underlying beta-binomial distribution. We solve the operational stage problem over 10,000 scenarios to compute the performance of each solution.

c. For each approach, among the cost for all scenarios, we calculate their best (minimal) value, worst (maximal) value, average, variance, VaR and CVaR at 5%, probability that the cost exceeds the target, and expected loss with respect to the target. Across all the criteria, a low value is preferable. We normalize the values by the corresponding performance from the ADRO approach.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Best</th>
<th>Worst</th>
<th>Average</th>
<th>Variance</th>
<th>VaR5%</th>
<th>CVaR5%</th>
<th>EP</th>
<th>EL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADRO</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Robust-Z</td>
<td>1.2108</td>
<td>0.9369</td>
<td>1.1110</td>
<td>0.8734</td>
<td>1.0269</td>
<td>1.0056</td>
<td>3.1113</td>
<td>2.8687</td>
</tr>
<tr>
<td>Robust-B</td>
<td>1.2108</td>
<td>0.9369</td>
<td>1.1112</td>
<td>0.8787</td>
<td>1.0288</td>
<td>1.0063</td>
<td>3.1430</td>
<td>2.7878</td>
</tr>
<tr>
<td>RN</td>
<td>0.6269</td>
<td>1.3723</td>
<td>0.8685</td>
<td>5.3727</td>
<td>1.1599</td>
<td>1.2432</td>
<td>1.6510</td>
<td>4.3490</td>
</tr>
<tr>
<td>D</td>
<td>0.5636</td>
<td>1.7165</td>
<td>0.8998</td>
<td>11.1462</td>
<td>1.3953</td>
<td>1.5617</td>
<td>2.3266</td>
<td>9.7579</td>
</tr>
</tbody>
</table>

Robust-Z=Robust approach by Zeng & Zhao (2013); Robust-B=Robust approach by Bertsimas et al. (2013); RN=Risk Neutral; D=Deterministic; EP = Exceeding probability; EL = expected loss.

**Table 2  Comparison with benchmark approaches.**

Table 2 presents the average performance from the 240 instances for each approach. Due to normalization, each performance measure for the ADRO approach is 1. For the other approaches, a number larger than 1 indicates that the ADRO approach outperforms them. From the table, ADRO is not as good as the risk neutral and deterministic approaches in the best scenario and average, due to risk aversion. In terms of worst scenario and variance, ADRO is worse than the two robust benchmark approaches, since the two robust benchmark approaches are extremely risk averse and indeed optimize the worst scenario performance. Even so, the two robust approaches only slightly outperform the ADRO approach in the worst case. Their advantage over the ADRO approach on variance is more significant, that is indeed due to that their uncertain performance concentrates on the worst-case situation. For all other performance measures, the results from ADRO are always better, and typically much better, than those from both the benchmark approaches. The ADRO approach requires more computation time than the benchmark approaches since more information
is used. While our overall solution time is 62 seconds, that of Zeng & Zhao (2013) is only 0.6 seconds. However, the approach by Zeng & Zhao (2013) finds the worst-case scenario, which would be trivial here since the worst-case scenario is one where all uncertain factors realize at maximum value. From Table 2, the ADRO approach results in better performance. In addition, the result in Section 4.1 indicates that for instances with reasonable size, the ADRO approach is efficient enough for use in project planning. To illustrate the performance over all 240 instances, we also provide, in Figure 1, a histogram of the performances from the 240 instances. As in Table 2, the results are normalized against a value of 1 for the ADRO approach.

![Figure 1](image_url)  
**Figure 1**  Comparison with benchmark approaches

### 4.3. Sensitivity analysis

We study the sensitivity of the results from our ADRO approach to the size and depth of projects. When we have fewer tasks in the project, we have less flexibility to adjust operational decisions. Hence, accurate capacity planning is more important, and the advantage of using the ADRO approach should be more significant. Similarly, when the value of the depth indicator $\Gamma$ is high, the longest chain is longer and the uncertain delay is larger. In that case, the ADRO approach should provide a more significant advantage, since it incorporates risk, whereas the risk neutral and deterministic benchmark approaches do not. The above insights are supported by the comparison between our ADRO approach and those two benchmark approaches.
We divide the 240 instances into two groups based on the number of tasks, \( N \). We let the instances with \( N = 27 \) form one group, and those with \( N = 52 \) form another. We then use ANOVA to study whether the average performance from these two groups is different. The results are summarized in Table 3. They show that the advantage of the ADRO approach over both the benchmark approaches is significant in both groups, but more significant in the group with small \( N \). Similarly, we provide sensitivity analysis to the depth indicator \( \Gamma \), which varies from 0.0784 to 0.4231 in the 240 instances. We also compare the group of the 120 instances with \( \Gamma \leq 0.1538 \) and the group of the 120 instances with \( \Gamma > 0.1538 \). The results in Table 4 indicate that in the group with large \( \Gamma \), the advantage from our ADRO approach is more significant, although again there is a big advantage from the use of ADRO in both groups of data.

| Performance criteria | RN |  |  | D |  |  |
|----------------------|----------------|----------------|----------------|----------------|----------------|
|                      | Average from small \( N \) | Average from large \( N \) | \( p \)-value | Average from small \( N \) | Average from large \( N \) | \( p \)-value |
| Best                 | 0.6609 | 0.5928 | 8.19E-06(\*\*) | 0.5793 | 0.5478 | 0.01629(\*\*) |
| Worst                | 1.4004 | 1.3441 | 0.02950(\*\*) | 1.8486 | 1.5843 | 2.42E-10(\*\*) |
| Mean                 | 0.8607 | 0.8762 | 0.0237(\*\*) | 0.8859 | 0.9137 | 0.0066(\*\*) |
| Variance             | 6.2193 | 4.5261 | 0.0177(\*\*) | 14.5446 | 7.7479 | 3.96E-05(\*\*) |
| VaR\( @5\% \)       | 3.8849 | 3.1914 | 0.8485 | 5.7217 | 7.2654 | 0.7511 |
| CVaR\( @5\% \)      | 3.5642 | 2.2200 | 0.1452 | 7.6943 | 3.2624 | 0.0596 |
| EP                   | 1.9487 | 1.3534 | 0.0175(\*\*) | 3.0520 | 1.6012 | 0.0004(\*\*) |
| EL                   | 5.7004 | 2.9975 | 0.0003(\*\*) | 14.2533 | 5.2626 | 1.77E-06(\*\*) |

\( (\*\*) \) : significant difference between the group with small \( N \) and the group with large \( N \) at 0.05 level.

| Performance criteria | RN |  |  | D |  |  |
|----------------------|----------------|----------------|----------------|----------------|----------------|
|                      | Average from small \( \Gamma \) | Average from large \( \Gamma \) | \( p \)-value | Average from small \( \Gamma \) | Average from large \( \Gamma \) | \( p \)-value |
| Best                 | 0.6034 | 0.6503 | 0.0024(\*\*) | 0.5573 | 0.5699 | 0.3387 |
| Worst                | 1.3247 | 1.4199 | 0.0002(\*\*) | 1.5610 | 1.8720 | 3.32E-14(\*\*) |
| Mean                 | 0.8788 | 0.8582 | 0.0027(\*\*) | 0.9077 | 0.8919 | 0.1265 |
| Variance             | 4.1715 | 6.5738 | 0.0007(\*\*) | 6.4934 | 15.7991 | 1.07E-08(\*\*) |
| VaR\( @5\% \)       | 3.3967 | 3.6796 | 0.9379 | 5.0527 | 7.9344 | 0.5536 |
| CVaR\( @5\% \)      | 2.0485 | 3.7357 | 0.0671 | 2.8768 | 8.0799 | 0.0268(\*\*) |
| EP                   | 1.3985 | 1.9036 | 0.0442(\*\*) | 1.5738 | 3.0793 | 0.0003(\*\*) |
| EL                   | 2.7936 | 5.9044 | 2.42E-05(\*\*) | 4.1502 | 15.3657 | 1.50E-09(\*\*) |

\( (\*\*) \) : significant difference between the group with small \( \Gamma \) and the group with large \( \Gamma \) at 0.05 level.

We also consider an interaction between the impact of the number of tasks and the impact from the depth indicator. However, a two-way ANOVA for \( N \) and \( \Gamma \) shows no significant effect.

Next, we study how the amount of capacity reservation varies with different project characteristics. We identify no clear relationship between the number of tasks and the percentage of available
general or customized capacity in an optimal solution. However, as shown in Figure 2, both types of capacity show a strong positive relationship with the network depth indicator, $\Gamma$. Here, there are typically more tasks in series, hence delays in completing tasks may accumulate and delay the overall project completion time significantly. Similarly, as network depth increases, there is a growing preference for the use of general over customized capacity, because outsourcing is likely to be needed for more tasks. This explains the stronger relationship with $\Gamma$ for general capacity reservation in Figure 2(a) than for customized capacity reservation in Figure 2(b).

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure2.png}
\caption{Sensitivity of capacity reservation}
\end{figure}

\begin{itemize}
\item[] : reservation for instance with depth indicator $\Gamma$.
\item[] : average reservation for instances with depth indicator no greater than $\Gamma$.
\end{itemize}

\section{4.4. Robustness}

We study the robustness of the solutions delivered by Algorithm CCG to incorrect assumptions about task time durations, for example different parameter settings in the beta distribution. To do so, we compare the solutions delivered by Algorithm CCG with those obtained by assuming an exact distribution. Specifically, for any given project management instance, we assume that we have certain sampling data for the uncertain factors. On one hand, based on the sampling data, we extract the scenario, mean, variance, and joint dispersion information, and solve the solution from Algorithm CCG; we call this solution the robust solution. On the other hand, we use the sampling data as the exact sampling distribution and solve another solution to minimize the URI criterion under that sampling distribution; we call this solution the sampling solution. For both solutions,
conditioning on the scenarios, mean, variance, and joint dispersion information, we solve the worst-case distribution under which the worst-case URI is achieved. We compare the two solutions under both worst-case distributions, by analyzing their uncertain cost under the eight criteria described in Section 4.2. We conduct this comparison for the same 240 instances as in Section 4.2. The average comparison between the robust solution and the sampling solution is presented in Table 5.

<table>
<thead>
<tr>
<th>Distribution 1</th>
<th>Solution</th>
<th>Best</th>
<th>Worst</th>
<th>Average</th>
<th>Variance</th>
<th>VaR5%</th>
<th>CVaR5%</th>
<th>EP</th>
<th>EL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robust solution</td>
<td>213</td>
<td>820</td>
<td>286</td>
<td>26806</td>
<td>820</td>
<td>820</td>
<td>8%</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td>Sampling solution</td>
<td>218</td>
<td>1054</td>
<td>356</td>
<td>80611</td>
<td>964</td>
<td>1021</td>
<td>12%</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Distribution 2</th>
<th>Solution</th>
<th>Best</th>
<th>Worst</th>
<th>Average</th>
<th>Variance</th>
<th>VaR5%</th>
<th>CVaR5%</th>
<th>EP</th>
<th>EL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Robust solution</td>
<td>212</td>
<td>756</td>
<td>277</td>
<td>16619</td>
<td>597</td>
<td>664</td>
<td>11%</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>Sampling solution</td>
<td>214</td>
<td>968</td>
<td>304</td>
<td>35527</td>
<td>748</td>
<td>841</td>
<td>19%</td>
<td>46</td>
<td></td>
</tr>
</tbody>
</table>

Distribution 1: worst-case distribution for the robust solution; Distribution 2: worst-case distribution for the sampling solution.

**Table 5**  Comparison with benchmarks for various projects.

From Table 5, the robust solution outperforms the sampling solution in all eight criteria. Hence, the robust solution is more robust in the sense that when the worst-case distribution is realized, the robust solution has a cost that is lower than that of the sampling solution.

### 4.5. Advantage of the CCG Algorithm

We demonstrate the computational advantage from using the CCG algorithm. The overall problem (23)-(29) can be solved directly, but this is computationally challenging, whereas it can be solved using the CCG algorithm. We vary the number of uncertain factors and the number of potential scenarios for each uncertain factor, such that \( I \) is changed. We test 20 out of the 240 instances in Section 4.2, and report the average solution time in Table 6.

<table>
<thead>
<tr>
<th>5 factors, each of them have 5 scenarios, ( I = 5^5 = 3125 )</th>
<th>CCG</th>
<th>direct solving</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>3.384</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>7 factors, each of them have 7 scenarios, ( I = 7^7 = 823,543 )</th>
<th>CCG</th>
<th>direct solving</th>
</tr>
</thead>
<tbody>
<tr>
<td>1425</td>
<td>&gt; 48,000</td>
<td></td>
</tr>
</tbody>
</table>

**Table 6**  Comparison between CCG and direct solving computation time in seconds.

Table 6 shows that Algorithm CCG solves problems with large values of \( I \), and solution times do not increase significantly with \( I \). However, direct solution of the problem is challenging. For example, when \( I = 823,543 \), the problem still cannot be solved directly after 48,000 seconds.
5. Concluding Remarks

Summary and insights

This work presents an ADRO model to achieve robust performance in uncertain projects with limited information about the distributions of task durations. Improved performance is achieved through more robust capacity reservation decisions at the planning stage of projects. We incorporate various practical realities of project management, including both general and customized outsourced capacity, correlation between task durations, worst-case distribution of delays to tasks, fast tracking of tasks, and piecewise linear crashing costs and project makespan penalties. A column and constraint generation algorithm is developed to minimize the URI of the total cost, which includes capacity reservation, fast tracking, crashing, and makespan penalty costs. In part due to the decomposition property described in Remark 3, the algorithm is capable of solving the problem optimally for small to medium size projects. The decisions recommended by our model deliver results that provide significantly improved performance, relative to the best available benchmarks, and are also robust against incorrect distributional assumptions. In summary, our work provides project managers with a planning tool for effective risk minimization in highly uncertain projects, and provides insights about how to make better capacity reservation decisions.

Project managers who work with outsourced resources on highly uncertain projects should benefit from several insights in our work. First, the optimization of capacity reservation decisions should enable a reduction in unnecessary capacity reservations, which will reduce costs for project companies. This helps providers of outsourcing services to improve their capacity utilization and profitability. Second, our sensitivity analysis results indicate for which types of projects the ADRO planning approach delivers greater benefits. Third, our sensitivity analysis results also indicate how network depth modulates the correct amount of capacity reservation. Fourth, our robustness study reveals the value of obtaining a more precise definition of the probability distributions of task time durations. Finally, it is possible to adjust the information set constraints parametrically. This provides an understanding of the effect of different levels of uncertainty on capacity reservations decisions, which can inform companies about the value of reducing uncertainty.

Extension

While our model is based on the assumption that all project uncertainties realize before the execution of any task, it can be extended to a more dynamic setting where some uncertainties realize during project execution. In that case, when making crashing decision for task $n$, $n \in [N]$, only $\tilde{\delta}_k$, $k \in \mathcal{T}_n$ has realized. Typically, $\mathcal{T}_k \subseteq [K]$ is a given index set that includes the indices of all uncertain
factors which would affect the uncertain duration of both task $n$ and previous tasks. Therefore, when making crashing decisions for task $n$, the task durations of that task and all preceding tasks have realized. Similarly, when making fast tracking decisions for tasks $n_1$ and $n_2$, $n_1, n_2 \in [N]$ and $n_1 \rightarrow n_2$, only $\tilde{\delta}_k$, $k \in \mathcal{T}_n$ has realized. After the capacity reservation decision has been made and as the project processes, the crashing decision $\mathbf{w}^C$ and fast tracking decision $\mathbf{w}^F$ have to satisfy the following non-anticipativity requirement,

$$w_n^C \in \Xi(\mathcal{T}_n), \forall n \in [N],$$

$$w_{n_1,n_2}^F \in \Xi(\mathcal{T}_{n_2}), \forall n_1, n_2 \in [N] \text{ and } n_1 \rightarrow n_2. \tag{36}$$

Here for any given index set $\mathcal{T} \subseteq [K]$, $\Xi(\mathcal{T})$ is a set of functions:

$$\Xi(\mathcal{T}) = \left\{ f : \mathbb{R}^N \rightarrow \mathbb{R} \ \bigg| \ f\left( \delta + \sum_{k \in [K] \setminus \mathcal{T}} \lambda_k e_k \right) = f(\delta), \forall \delta, \lambda \in \mathbb{R}^N \right\}.$$  

By definition, any function in $\Xi(\mathcal{T})$ is independent of argument $k$, for any $k \not\in \mathcal{T}$. Our model can then be modified to consider the dynamics between the uncertainty realization and operational decisions (i.e., fast tracking and crashing). Thus, all components in $\mathbf{w}^F$ and $\mathbf{w}^C$ are solved with the dynamics and the additional constraint of (36). This greatly complicates the solution procedure, hence only approximate solutions can be obtained efficiently. To achieve this, we may assume the decisions, $\mathbf{w}^F$, $\mathbf{w}^C$ are affine functions in $\delta$ (Ben-Tal et al. 2004), which enables a standard solution procedure from distributionally robust optimization (Bertsimas et al. 2019) to solve the problem approximately.

**Future research**

Our work suggests several directions for future research. First, whereas we assume a piecewise linear time cost tradeoff function, there are project management applications where this tradeoff is discrete. However, this alternative is complicated by the intractability of project scheduling with discrete crashing options (De et al. 1997). Second, our work can be generalized to consider scope changes requested by a project owner. Third, it would be valuable to study specific applications where unique characteristics such as high correlation between the task durations may provide opportunities for delivering greater value from our work. Finally, for companies with multiple projects, our work can inform capacity reservation decisions across multiple projects. In conclusion, we hope that our work will encourage research on these topics of substantial practical value to project companies.
References


Proofs of Statements

EC.1. Proof of Theorem 2.

Proof. We prove that this problem is formally intractable. To do so, we first define the following problem:

Discrete Time-Cost Tradeoff (DTCT): given a project network where each task \( n \) has at most two discrete (time, cost) processing alternatives \( (t_{1n}, c_{1n}) \) and if there is a second alternative \( (t_{2n}, c_{2n}) \), where \( t_{1n} < t_{2n} \) and \( c_{1n} > c_{2n} \), and a given deadline \( D \), does there exist a feasible project schedule that meets the deadline \( D \) and has total cost less than or equal to a given value \( C \)?

De et al. (1997) prove that the recognition version of problem DTCT is unary \( NP \)-complete. We now establish a similar result for problem (10).

By reduction from problem DTCT. Given an arbitrary instance of problem DTCT with \( N \) tasks, we construct an instance of problem (10) as follows. Define \( \gamma_{ij} = 0 \), for \( i, j = 1, \ldots, i \rightarrow j \); hence, no fast tracking is possible. We let \( \theta = 0 \) such that there is no need for any general capacity, hence without loss of generality the optimal value of \( x \) in problem (10) is \( 0 \). Define \( \eta = 1 \); hence, customized capacity is needed for crashing any task. We also define \( g_C = 0 \); hence, the crashing of any task is free, once capacity has been reserved for it. For any task \( j \) with only one processing alternative, we let \( b_j \) denote its corresponding processing time in DTCT, and let \( u_{yj} = \infty \); hence, we do not reserve customized capacity to crash task \( j \) since doing so is too expensive. Observe that the tasks with two processing alternatives in problem DTCT have a “faster” alternative 1, and a “cheaper” alternative 2. For any task \( j \) with two processing alternatives, we let \( b_j = t_{2j} \), i.e., the processing time of the “cheaper” alternative; let \( u_{yj} = c_{1j} - c_{2j} \), i.e., the cost difference between the two alternatives in problem DTCT; in addition, we define \( q_j = 1 \) and \( e_{ij}^C = t_{2j} - t_{1j} \), i.e., the largest possible crashing of task \( j \) reduces its processing time to \( b_j - e_{ij}^C = t_{1j} \). Therefore, since \( g_C = 0 \), any task \( j \) with two processing alternatives in DTCT without loss of generality also has two (time, cost) pairs in problem (10): \( (t_{1j}, c_{1j} - c_{2j}) \) if customized capacity is reserved, and \( (t_{2j}, 0) \) otherwise. Finally, we define \( l_1 = D \), \( l_2 = D + \sigma \), \( d_1 = 0 \), and \( d_2 = +\infty \), where \( \sigma > 0 \) is an arbitrarily small constant; hence, the completion time penalty is zero up to time \( D \), and infinite thereafter. We let \( C' = C - \sum_{j=1}^{N} c_{2j} \) denote a threshold cost for the constructed instance of problem (10).
We prove that there exists a feasible schedule for the constructed instance of problem (10) with cost less than or equal to $C'$, if and only if there exists a “yes” answer to problem DTCT.

($\Rightarrow$) Consider any feasible schedule that provides a “yes” answer to problem DTCT. We construct a solution to problem (10) as follows. For any task with only one processing alternative, we do not reserve the customized capacity. For any task with two processing alternatives, we reserve (respectively, do not reserve) capacity in problem (10) if the faster (resp., cheaper) alternative is used for the corresponding task in problem DTCT. By definition of the instance of problem (10), this solution to problem (10) completes the project at the same time as the feasible schedule to problem DTCT, i.e., by time $D$, and thus has zero completion time penalty. Now, each task $j$ with reserved capacity in problem (10) has capacity reservation cost of $(c_{1j} - c_{2j})$ and linear crashing cost of $g_{ij}^c = 0$. Hence, task $j$ has total cost $(c_{1j} - c_{2j})$ in problem (10), compared to a cost of $c_{1j}$ in problem DTCT. Further, each task $j$ without reserved capacity in problem (10) has zero capacity reservation and linear crashing cost, hence task $j$ has total cost 0, compared to total cost $c_{2j}$ in problem DTCT. Combining these two observations, each task $j$ costs exactly $c_{2j}$ less in problem (10) than in problem DCTC, for $j = 1, \ldots, N$. Therefore, the total cost of the schedule in problem (10) is $C - \sum_{j=1}^{N} c_{2j} = C'$.

($\Leftarrow$) Given a feasible schedule for problem (10) with cost less than or equal to $C'$, we prove that the answer to problem DTCT is “yes”. Since $l_2 = D + \sigma$ and $d_2 = +\infty$, any schedule for problem (10) with cost less than or equal to $C'$ must complete by time $D$. From the discussion in the previous paragraph, the corresponding schedule for problem DTCT also completes by time $D$, and hence is feasible. By assumption, the total cost in problem (10) is less than or equal to $C'$. Moreover, from the discussion in the previous paragraph, each task has a total cost in problem DTCT that is greater by exactly $t_{2j}$ than it is in problem (10). Hence, the cost of the solution in problem DTCT is less than or equal to $C' + \sum_{j=1}^{N} c_{2j} = C$, and the schedule provides a “yes” answer. Q.E.D.

**EC.2. Proof of Proposition 1**

**Proof.** Note that $\alpha^* \leq \alpha^*_e$ follows from the feasible set of Problem (20) being a subset of that from Problem (19). To prove $\alpha^*_e \leq \alpha^* + \epsilon$, consider any feasible $(x, y, \alpha)$ in Problem (19), we show the
feasibility of \((x, y, \alpha + \epsilon)\) in Problem (20). For any \(\mathbb{P} \in \mathcal{P}\),

\[
\mathbb{E}_\mathbb{P} \left[ \max_{m \in [M]} \left\{ a_m h(x, y, \tilde{\delta}) - a_m \tau + b_m (\alpha + \epsilon) \right\} \right] \\
= (\alpha + \epsilon) \mathbb{E}_\mathbb{P} \left[ \max_{m \in [M]} \left\{ a_m \frac{h(x, y, \tilde{\delta}) - \tau}{\alpha + \epsilon} + b_m \right\} \right] \\
\leq (\alpha + \epsilon) \mathbb{E}_\mathbb{P} \left[ \frac{\alpha}{\alpha + \epsilon} \max_{m \in [M]} \left\{ a_m \frac{h(x, y, \tilde{\delta}) - \tau}{\alpha} + b_m \right\} + \frac{\epsilon}{\alpha + \epsilon} \max_{m \in [M]} \{ a_m \cdot 0 + b_m \} \right] \\
= \alpha \mathbb{E}_\mathbb{P} \left[ \max_{m \in [M]} \left\{ a_m \frac{h(x, y, \tilde{\delta}) - \tau}{\alpha} + b_m \right\} \right] \\
\leq \alpha \sup_{\mathbb{P} \in \mathcal{P}} \mathbb{E}_\mathbb{P} \left[ \max_{m \in [M]} \left\{ a_m \left( \frac{h(x, y, \tilde{\delta}) - \tau}{\alpha} \right) + b_m \right\} \right] \\
\leq 0,
\]

where the first inequality is due to the convexity of the piecewise-linear utility, the last equality holds since the utility function is normalized by \(u(0) = 0\), and the last inequality follows from the feasibility of \((x, y, \alpha)\) in Problem (19). Taking the supremum over all \(\mathbb{P} \in \mathcal{P}\) for the left hand side of Inequality (EC.1), we conclude that \((x, y, \alpha + \epsilon)\) is feasible in Problem (20). Q.E.D.

**EC.3. Proof of Lemma 1.**

**Proof.** Consider \(\sup_{\mathbb{P} \in \mathcal{P}} \mathbb{E}_\mathbb{P} \left[ \max_{m \in [M]} \left\{ a_m h(x, y, \tilde{\delta}) - a_m \tau + b_m \alpha \right\} \right] \) as a primal optimization problem where the decision variables are \(\mathbb{P}\). Observe that since we assume \(\tilde{\delta}\) follows a discrete distribution, the decision variables \(\mathbb{P}\) is actually a finite dimensional vector. Hence, we have a finite dimensional linear programming problem. Moreover, let \(\delta^i, \omega^S_{j,k,i}, \omega^C_{j,i}\) be defined as in the lemma.
statement. Then
\[
\sup_{\mathcal{P} \in \mathcal{P}} \mathbb{E}_\mathcal{P} \left[ \max_{m \in [M]} \left\{ a_m h \left( \mathbf{x}, \delta \right) - a_m \tau + b_m \alpha \right\} \right] = \sup_{\mathcal{P}} \mathbb{E}_\mathcal{P} \left[ \max_{m \in [M]} \left\{ a_m h \left( \mathbf{x}, \delta \right) - a_m \tau + b_m \alpha \right\} \right]
\]
\[
\text{s.t. } \mathbb{P} \left( \delta \in \mathcal{Q} \right) = 1,
\mathbb{E}_\mathcal{P} \left[ x_k \left( \delta \right) \right] \leq a_{j,k}, \; k \in [K], \; j \in [J_{S,k}]
\mathbb{E}_\mathcal{P} \left[ x_j \left( \xi^T \delta + \epsilon_j \right) \right] \leq \sigma_j, \; j \in [J_C]
\]
\[
= \min_{\mathcal{P}} \tau_0 + \sum_{k=1}^K \sigma_S^{S^T} \tau^{S^k} + \sigma^T \tau^C
\]
\[
\text{s.t. } \tau_0 + \sum_{k=1}^K \sum_{j=1}^{J_{S,k}} \omega_{S,k,i}^{S,k} \tau_{S,k} + \sum_{j=1}^{J_C} \omega_{C,i} \tau^C \geq \max_{m \in [M]} \left\{ a_m h \left( \mathbf{x}, \delta \right) - a_m \tau + b_m \alpha \right\}, \; \forall i \in [I]
\tau^{S^k}, \tau^C \geq 0, \; k \in [K]
\]
\[
= \min_{\mathcal{P}} \tau_0 + \sum_{k=1}^K \sum_{j=1}^{J_{S,k}} \omega_{S,k,i}^{S,k} \tau_{S,k} + \sum_{j=1}^{J_C} \omega_{C,i} \tau^C \geq a_m h \left( \mathbf{x}, \delta \right) - a_m \tau + b_m \alpha, \; \forall i \in [I], \; m \in [M]
\tau^C, \tau^{S^k} \geq 0, \; k \in [K].
\]
(E.C.2)

The second equality follows from strong duality, which is due to the finite dimension of the primal problem.

Following the definition of the function \( h \) in (13), given any \( i \in [I], \; m \in [M] \), the constraint
\[
\tau_0 + \sum_{k=1}^K \sum_{j=1}^{J_{S,k}} \omega_{S,k,i}^{S,k} \tau_{S,k} + \sum_{j=1}^{J_C} \omega_{C,i} \tau^C \geq a_m h \left( \mathbf{x}, \delta \right) - a_m \tau + b_m \alpha
\]
can be rewritten as
\[
\tau_0 + \sum_{k=1}^K \sum_{j=1}^{J_{S,k}} \omega_{S,k,i}^{S,k} \tau_{S,k} + \sum_{j=1}^{J_C} \omega_{C,i} \tau^C \geq a_m \left( u^T_x x + u^T_y y + g^T_F \mathbf{w}_F + g^T_C \mathbf{w}_C + d^T v \right) - a_m \tau + b_m \alpha,
\]
\[
\left( \mathbf{w}_F^i, \mathbf{w}_C^i, \mathbf{v}^i, \delta^i \right) \in \mathcal{S}(z^0 + Z \delta, \mathbf{x}, \mathbf{y}).
\]
Q.E.D.

EC.4. Proof of Theorem 3.

Proof. The equivalence between problems (20) and (23)–(29) follows immediately from Lemma 1. Moreover, given any \( \delta \), the set \( (\mathbf{w}_F^i, \mathbf{w}_C^i, \mathbf{v}^i, \delta^i) \) defines a polyhedron for \( (\mathbf{w}_F^i, \mathbf{w}_C^i, \mathbf{v}^i, \delta^i, \mathbf{x}, \mathbf{y}) \), and hence problem (23)–(29) is a mixed-integer linear programming problem. Q.E.D.
EC.5. Proof of Lemma 2.

Proof. Given $m \in [M]$, the constraints are feasible if and only if

$$\max_{\delta \in \mathcal{Q}} \left\{ - \sum_{k=1}^{K} \sum_{j=1}^{J} r_j^{S,k} \chi_j^k (\delta_k) - \sum_{j=1}^{J} r_j^{C} \chi_j^C (\xi_j^T \delta + \epsilon_j) + a_m c_2^*(\delta, x, y) \right\} \leq \tau_0 + a_m \tau - b_m \alpha - a_m u_x^T x - a_m u_y^T y, \quad (EC.3)$$

where $c_2^* (\cdot, \cdot, \cdot)$ is a mapping from the uncertain realization and first stage decisions to the lowest cost at the operational stage. In particular, $c_2^*$ is defined and can be reformulated as follows.

$$c_2^*(\delta, x, y) = \min_{g_F^T w_F + g_C^T w_C + d^T v} \quad \begin{array}{l}
\text{s.t.} \quad (w_F, w_C, v, c) \in S(\mathcal{Z}, x, y) \\
\quad g_F^T w_F + g_C^T w_C + d^T v = \min \quad (A_x x + A_y y + A_z (z^0 + Z \delta) + b^o) \\
\quad w_F \geq 0, w_C \geq 0, v \geq 0, c \geq 0 \\
\quad \text{s.t.} \quad p^T (A_F, A_C, A_v, A_z) \leq (g_F^T, g_C^T, d^T, 0^T) \\
\quad p \geq 0
\end{array} \quad (EC.4)$$

where the third equality follows from strong duality. Hence, we can reformulate the LHS of constraint (EC.3) to $q_m^*(x, y, \tau^C, \tau^{S,1}, \ldots, \tau^{S,K})$, as defined in problem (32). Further, constraint (EC.3) is feasible if and only if $q_m^*(x, y, \tau^C, \tau^{S,1}, \ldots, \tau^{S,K}) \leq \tau_0 + a_m \tau - b_m \alpha - a_m u_x^T x - a_m y^T y$.

Consider the case where $q_m^*(x, y, \tau^C, \tau^{S,1}, \ldots, \tau^{S,K}) > \tau_0 + a_m \tau - b_m \alpha - a_m u_x^T x - a_m y^T y$ for some $m \in [M]$. There must exist $m^* \in [M], i^* \in [I]$ such that (34) holds. Consider any such $m^*$ and $i^*$.

Observe that

$$\tau_0 + a_m \tau - b_m \alpha - a_m u_x^T x - a_m y^T y$$

$$< \max_{p \in P} q_{m^*} \left( p, \delta^*, \xi^C, \chi^C, \tau^{S,1}, \ldots, \tau^{S,K} \right)$$

$$= - \sum_{k=1}^{K} \sum_{j=1}^{J} r_j^{S,k} \chi_j^k (\delta_k^*) - \sum_{j=1}^{J} r_j^{C} \chi_j^C (\xi_j^T \delta^* + \epsilon) + a_m \max_{p \in P} \left\{ (A_x x + A_y y + A_z (z^0 + Z \delta^*) + b^o)^T p \right\}$$

$$= - \sum_{k=1}^{K} \sum_{j=1}^{J} r_j^{S,k} \chi_j^k (\delta_k^*) - \sum_{j=1}^{J} r_j^{C} \chi_j^C (\xi_j^T \delta^* + \epsilon) + a_m c_2^*(\delta^*, x, y)$$

$$= - \sum_{k=1}^{K} \sum_{j=1}^{J} r_j^{S,k} \chi_j^k (\delta_k^*) - \sum_{j=1}^{J} r_j^{C} \chi_j^C (\xi_j^T \delta^* + \epsilon) + a_m \min_{(w_F^*, w_C^*, v^*, \epsilon) \in S(z^0 + Z \delta^*, x, y)} \left\{ g_F^T w_F^* + g_C^T w_C^* + d^T v^* \right\}.$$
where the last two equalities follow from (EC.4). Hence, we cannot find \((w_F^*, w_C^*, v^*, c^*) \in S(z^0 + Z\delta^*, x, y)\) with \(\tau_0 + a_{m_\alpha} \tau - b_{m_\alpha} \alpha - a_{m_\alpha} u^T x - a_{m_\alpha} u^T y \geq a_{m_\alpha} g_F^T w_F^* + a_{m_\alpha} g_C^T w_C^* + a_{m_\alpha} d^T v^* - \sum_{k=1}^K \sum_{j=1}^{J_{S,k}} s_{j,k} S_k \chi_j^C \left( \delta_k^* - \sum_{j=1}^{J_{C}} \chi_j^C \left( \xi_j^T \delta^* + \epsilon_j \right) \right)\).

**EC.6. Proof of Lemma 3.**

**Proof.** Given any feasible solution \((p, \delta)\) to problem (32), we construct a feasible solution to problem (35) with the same \(p\), \(\phi_{kl} = 1\) if \(\delta_{kl} = \delta_{kl}\) and 0 otherwise, \(\zeta_{kl} = \delta_{kl} b_k p\) if \(\phi_{kl} = 1\) and 0 otherwise, \(\Omega_j = \chi_j^C (\xi_j^T \delta + \epsilon_j)\). In addition, the objective value of problem (32) for \((p, \delta)\) is identical to the objective value of problem (35) for the corresponding \((p, \phi, \zeta, \Omega)\).

On the other hand, given any feasible solution \((p, \phi, \zeta, \Omega)\) for problem (35), it follows that \((p, \delta)\) with \(\delta_k = \sum_{j=1}^{L_k} \phi_{kl} \delta_{kl}\) is also a feasible solution for problem (32). In addition, the objective value of problem (35) for \((p, \phi, \zeta, \Omega)\) is no greater than the objective value of problem (32) for \((p, \delta)\).

Q.E.D.

**EC.7. Proof of Theorem 4.**

**Proof.** The result follows directly from Lemmas 2 and 3. Q.E.D.