# Optimal Crashing of an Activity Network with Disruptions

**Abstract**

In this paper, we consider an optimization problem involving crashing an activity network under a single disruption. A disruption is an event whose magnitude and timing are random. When a disruption occurs, the duration of an activity, which has not yet started, can change. We formulate a two-stage stochastic mixed-integer program, in which the timing of the stage is random. In our model the recourse problem is a mixed-integer program. We prove the problem is NP-hard, and using simple examples, we illustrate properties that differ from the problem's deterministic counterpart. Obtaining a reasonably tight optimality gap can require a large number of samples in a sample average approximation, leading to large-scale instances that are computationally expensive to solve. Therefore, we develop a branch-and-cut decomposition algorithm, in which spatial branching of the first stage continuous variables, and linear programming approximations for the recourse problem, are sequentially tightened. We test our decomposition algorithm with multiple improvements and show it can significantly reduce solution time over solving the problem directly.

## 1 Introduction

The management of complex projects through optimization has a rich history in operations research, beginning with the critical path method of Kelly Kelly (1961); see Söderlund Söderlund (2004) for an overview. A project is a collection of activities, between which there are precedence relationships due to logical or technological considerations. A precedence relationship is usually reflected as the start time of one activity following the completion of another. Typically, multiple activities can be processed at the same time, as long as the precedence requirements are satisfied. See Elmaghraby Elmaghraby (1977) for a detailed treatment of activity networks. In this setting, "crashing" is an action that consumes a certain amount of one or more resources and shortens the duration of an activity accordingly Kuhl and Tolentino-Peña (2008). A deterministic optimization model for crashing an activity network was proposed in the 1960s Fulkerson (1961), Kelly (1961), in which the goal is to complete the project in minimum time by allocating resources under one or more budget constraints. Project crashing problems see pervasive application, for example, in optimizing project management Demeulemeester and Herroelen (2006), Jaselskis and Ashley (1991), Tonchia (2018), machine scheduling Hall and Sriskandarajah (1996), Naderi et al. (2009), health services scheduling Cardoen et al. (2010), chemical processes Li and Ierapetritou (2008), and digital circuit sizing Kim et al. (2007).

When the program evaluation and review technique (PERT) was introduced Malcolm et al. (1959), activity durations were modeled as independent beta random variables, with the project duration approximated by a normal distribution. Extensions that allow for more general assumptions were then developed Elmaghraby (1977), and Monte Carlo simulation began to play a role in estimating the expected project span, which is difficult to express analytically Burt and Garman (1971), van Slyke (1963). In stochastic versions of the project crashing problem, a crashing plan is made at the beginning of the horizon. The start time of each activity is then determined by a

recourse decision variable after random activity durations are realized. Heuristics and simulation-based algorithms have been developed to approximately solve the stochastic project crashing problem Aghaie and Mokhtari (2009), Bowman (1994), Ke (2014), Kim et al. (2007). Another approach to handle uncertainty in activity duration is robust optimization, in which the objective is to minimize the worst-case project span over a specified uncertainty set. While affinely adaptive recourse decisions are computationally tractable as linear, or second-order cone, programs, this restriction may lead to suboptimal or infeasible solutions Chen et al. (2008), Cohen et al. (2007). However, once recourse decisions can take general form, the robust model is only tractable with rectangular uncertainty sets Wiesemann et al. (2012). Ahipasaoglu et al. Ahipasaoglu et al. (2016) propose a distributionally robust optimization scheme applied to a PERT network, which reformulates the problem as a semidefinite program or a copositive program, depending on the description of uncertainty.

Another major stream of work relevant to project crashing involves the resource-constrained project scheduling problem (RCPSP). In an RCPSP, makespan is minimized while satisfying precedence constraints. Resources, such as equipment and labor, are required to execute each activity in an RCPSP, and constraints limit their use in each time period. In contrast in our work, resources only constrain crashing decisions, which reduce an activity's duration. Even a deterministic RCPSP is NP-hard Blazewicz et al. (1983). Solving an RCPSP with uncertain activity durations is challenging, with most solution methods focusing on restrictive classes of policies or heuristics Hartmann and Kolisch (2000), Lamas and Demeulemeester (2016), Möhring and Stork (2000).

In this paper, we propose the concept of stochastic disruptions to model uncertainty in the duration of activities, which differs from existing approaches in stochastic programming and robust optimization. A stochastic disruption is an event that may occur at any time in the problem's horizon and results in a change—typically a significant change—in the system's parameters. A few authors apply this idea in models with discrete time periods, in which the disruption occurs in a set of prespecified time periods. Yu and Qi Yu and Qi (2004) introduce scenario-based optimization models for airline scheduling. Salmerón et al. Salmerón et al. (2009) introduce a sealift scheduling problem under a finite number of stochastic disruptions within a stochastic programming structure; this model structure "falls between standard two-stage and multi-stage stochastic programs for a multi-period problem" and reduces the size of the problem (scenario tree) to quadratic, rather than exponential, growth in the number of time periods. Our setting inherits the philosophy of Salmerón et al. Salmerón et al. (2009), but enhances the model by allowing the random disruption time to be continuous in the context of an activity network, where time is naturally modeled as being continuous.

Given a limited number of disruption scenarios, the problem of optimizing crashing decisions to minimize expected completion time can be formulated as a stochastic mixed-integer program,

and we present the model in Section 2. If we assume a continuous distribution for the disruption time and magnitude, a sample average approximation (SAA) can be used to create a finite set of scenarios and approximate the original problem by a finite-sized optimization problem. In Section 3 we show that the problem is NP-hard even with a continuous allocation of crashing effort and just two scenarios. Section 4 presents properties of the problem using a serial activity network as a special case. The potentially large scale and the discrete, non-convex nature of the SAA problem's formulation suggest that it may be computationally challenging to solve. In Section 5, a method based on Benders' decomposition is developed to solve our problem of optimizing crashing decisions under stochastic disruptions. We show that the decomposition method can significantly reduce the time required to solve the integer program compared to solving the extensive formulation using a commercial off-the-shelf solver. Experimental results are presented in Section 6, including the empirical relationship between solution quality and sample size, a comparison between the quality of our solution and solutions of alternative models, and the computational performance of the decomposition method of Section 5. We conclude with remarks on potential extensions of our model in Section 7.

## 2 Problem Formulation

Nomenclature:

*Indices and index sets*

| | |
|---|---|
| $I$ | set of activities; |
| $J_i$ | set of crashing options for activity $i \in I$; |
| $\Omega$ | index set for disruption scenarios (sample space); |
| $\mathcal{A}$ | set of arcs, which represents precedence relationships; |

*Parameters*

| | |
|---|---|
| $D_{ik}$ | nominal duration between possible start times of activities $i$ and $k$, $(i,k) \in \mathcal{A}$; |
| $e_{ij}$ | effectiveness of crashing option $j \in J_i$ for activity $i \in I$; |
| $B$ | total budget for crashing options; |
| $b_{ij}$ | cost of crashing option $j \in J_i$ for activity $i \in I$; |
| $\bar{x}_{ij}$ | upper bound on option $j \in J_i$ for activity $i \in I$ with $\bar{x}_{ij} \leq 1$; |
| $H^\omega$ | disruption time under scenario $\omega \in \Omega$; |
| $d_{ik}^\omega$ | increase in duration of $(i,k) \in \mathcal{A}$ under $\omega \in \Omega$, if started after the disruption; |
| $p^\omega$ | the probability of scenario $\omega \in \Omega$; |
| $p^0$ | the probability of no disruption; |

*Decision variables*

| | |
|---|---|
| $t_i$ | continuous nominal start time of activity $i \in I$; |
| $x_{ij}$ | continuous nominal crashing of activity $i \in I$ by option $j \in J_i$; |
| $t_i^\omega$ | continuous start time of activity $i \in I$ under scenario $\omega \in \Omega$; |
| $x_{ij}^\omega$ | continuous crashing of activity $i \in I$ by option $j \in J_i$ under |

| | |
|---|---|
| | scenario $\omega \in \Omega$; |
| $G_i^\omega$ | binary indicator whether activity $i \in I$ starts after disruption under $\omega \in \Omega$; |
| $z_{ij}^\omega$ | continuous term to linearize bilinear term, $G_i^\omega x_{ij}^\omega$, $i \in I$, $j \in J_i, \omega \in \Omega$. |

We first review an optimization model for a deterministic crashing problem; see Fulkerson Fulkerson (1961) and Kelley Kelly (1961). A set of activities, $I$, together with precedence relationships, $\mathcal{A} \subseteq I \times I$, form an acyclic activity network $\mathcal{G} = (I, \mathcal{A})$, which represents the project. An arc $(i,k) \in \mathcal{A}$ indicates that activity $i$ has to finish before activity $k$ starts, and its length, $D_{ik}$, shows that the start time of activity $i$ has to be at least $D_{ik} \geq 0$ before the start time of activity $k$. This corresponds to the activity-on-node representation in Elmaghraby (1977). We allow for precedence-dependent durations to model finish-to-start lags between specified pairs of activities. For example, activity $k_1$ can start immediately after activity $i$ ends but additional set-up time is required to commence activity $k_2$, as is common practice in project management Project Management Institute (2017). We create two dummy activities $S, T \in I$ to represent the start and the termination of the entire project. Activity $S$ should precede every activity $i \in I \backslash \{S\}$ and $T$ should succeed every activity $i \in I \backslash \{T\}$, either directly or by implication, and they both have zero duration.

We can apply a finite set of crashing options, $j \in J_i$, to activity $i \in I$. One unit application of each option incurs a cost of $b_{ij}$, and decreases the corresponding durations by $D_{ik}e_{ij}$, $\forall (i,k) \in \mathcal{A}$, where $e_{ij} \in [0,1]$ denotes the unit effectiveness of crashing option $j$. This construct, coupled with upper bounds $\bar{x}_{ij}$, can model a piecewise linear concave relationship between resources devoted to crashing and the reduction in duration, representing diminishing returns on investment for each activity. For example, suppose the duration between the start time of activities 1 and 2 is $D_{12} = 10$, applying one unit of crashing option 1 decreases the duration by half ($e_{11} = 0.5$) but we can only apply $\bar{x}_{11} = 0.4$ units, and option 2 for activity 1 has unit effectiveness $e_{12} = 0.3$. The maximum reduction in $D_{12}$ is then $10(1 - 0.4 \cdot 0.5 - 0.6 \cdot 0.3) = 6.2$. Even if $\bar{x}_{11} = 1$ we might limit use of option 1 and/or choose $x_{12} > 0$ due to $b_{11}$, $b_{12}$, and the overall budget $B$, which the total cost of crashing cannot exceed. The objective is then to minimize the start time of activity $T$, and thus, we formulate the deterministic project crashing problem as:

$$\min \quad t_T \tag{1a}$$

$$\text{s.t.} \quad t_k - t_i \geq D_{ik}\left(1 - \sum_{j \in J_i} e_{ij} x_{ij}\right) \qquad \forall\, (i,k) \in \mathcal{A} \tag{1b}$$

$$\sum_{i \in I} \sum_{j \in J_i} b_{ij} x_{ij} \leq B \tag{1c}$$

$$\sum_{j \in J_i} x_{ij} \leq 1 \qquad \forall\, i \in I \tag{1d}$$

$$0 \le x_{ij} \le \bar{x}_{ij} \qquad \forall\, i \in I, j \in J_i \tag{1e}$$

$$t_i \ge 0 \qquad \forall\, i \in I. \tag{1f}$$

In this formulation, $t_i$ represents the start time of activity $i \in I$. We aim to minimize the project span, which is the start time of the terminal activity, $t_T$. Constraint (1b) guarantees the precedence relationship: if activity $i$ precedes activity $k$, activity $k$ cannot start until time $t_i + D_{ik}(1 - \sum_{j \in J_i} e_{ij} x_{ij})$. Constraint (1c) is the budget constraint, and constraint (1d) ensures that no more than one unit of total crashing effort can be applied to an activity. Constraints (1e) and (1f) enforce simple bounds.

A number of variants of model (1) are possible. The budget constraint (1c) is for a single resource, but an extension to multiple resources is straightforward. We model continuous crashing decisions, and allow multiple options to be used in concert. Instead, discrete crashing decisions, or a hybrid, can be used in which a fixed cost, and subsequent per unit costs are incurred. The crashing options for each alternative can instead be mutually exclusive so that at most one option can be selected for each activity. All such variants can be employed in a straightforward way throughout the paper's development.

For a project crashing problem under stochastic disruptions, we assume at most one stochastic disruption can occur at a random time in the project span. While this assumption may be limiting in some settings, it is appropriate when it is unlikely for two or more disruptions to occur during the time horizon, and can apply, e.g., for natural disasters, major market crashes, cyber attacks, and work stoppages. For example, suppose we manage a construction project, and we aim to plan against the potential hazard caused by an earthquake or an employee strike. It may be unlikely for two major earthquakes or strikes to affect the same project within the relevant time period. We further assume that for each activity $i \in I$, the crashing decision needs to be made prior to the start of that activity, which is reasonable, e.g., when contracts are involved in commitment of resources Oberlender (1993). We assume a disruption does not affect activities that have already started—including those already finished—at the time of the disruption, but the disruption changes the length of activities that have not yet started according to a known probability distribution. (This assumption is revisited at the end of this section.) It is usually hard to compute the recourse function directly when random parameters have a continuous distribution, and therefore we use sample average approximation (SAA) Kim et al. (2015), Shapiro et al. (2009). In this paper, we assume there is a finite set of scenarios indexed by $\omega \in \Omega$. For each scenario $\omega$, the random realization of parameters, which we denote $\xi^\omega$, consists of the timing of the disruption, $H^\omega$, and the magnitude of the disruption via increases in the duration parameters, $d^\omega_{ik}, \forall (i,k) \in \mathcal{A}$.

Because we assume at most one disruption, we can model the problem as a two-stage stochastic mixed-integer program, in which the timing of the second stage is random. That is, the definition of our stages differs from the usual stochastic programming setting. Here the first stage contains

decisions through completion of the project, and we follow this policy if no disruption occurs. And, the second stage characterizes the decisions for each realization of the disruption, which commences at the random time, $H^\omega$. The first stage decision variables are carried out until the disruption if it ever occurs, and after the disruption, the scenario-specific recourse decisions are executed.

The extensive formulation of the two-stage stochastic program is shown as formulation (2):

$$
\begin{aligned}
z^* = \min \quad & p^0 t_T + \sum_{\omega\in\Omega} p^\omega t_T^\omega && && \text{(2a)}\\
\text{s.t.} \quad & t_k - t_i \geq D_{ik}\left(1 - \sum_{j\in J_i} e_{ij}x_{ij}\right) && \forall\,(i,k)\in\mathcal{A} && \text{(2b)}\\
& \sum_{i\in I}\sum_{j\in J_i} b_{ij}x_{ij} \leq B && && \text{(2c)}\\
& \sum_{j\in J_i} x_{ij} \leq 1 && \forall\, i\in I && \text{(2d)}\\
& H^\omega + MG_i^\omega \geq t_i && \forall\, i\in I, \omega\in\Omega && \text{(2e)}\\
& H^\omega - M(1-G_i^\omega) \leq t_i && \forall\, i\in I, \omega\in\Omega && \text{(2f)}\\
& t_i^\omega + M'G_i^\omega \geq t_i && \forall\, i\in I, \omega\in\Omega && \text{(2g)}\\
& t_i^\omega - M'G_i^\omega \leq t_i && \forall\, i\in I, \omega\in\Omega && \text{(2h)}\\
& x_{ij}^\omega + \bar{x}_{ij}G_i^\omega \geq x_{ij} && \forall\, i\in I, j\in J_i, \omega\in\Omega && \text{(2i)}\\
& x_{ij}^\omega - \bar{x}_{ij}G_i^\omega \leq x_{ij} && \forall\, i\in I, j\in J_i, \omega\in\Omega && \text{(2j)}\\
& t_k^\omega - t_i^\omega \geq D_{ik} + d_{ik}^\omega G_i^\omega - \\
& \quad \sum_{j\in J_i} D_{ik}e_{ij}x_{ij}^\omega - \sum_{j\in J_i} d_{ik}^\omega e_{ij}z_{ij}^\omega && \forall\,(i,k)\in\mathcal{A}, \omega\in\Omega && \text{(2k)}\\
& \sum_{i\in I}\sum_{j\in J_i} b_{ij}x_{ij}^\omega \leq B && \forall\,\omega\in\Omega && \text{(2l)}\\
& \sum_{j\in J_i} x_{ij}^\omega \leq 1 && \forall\, i\in I, \omega\in\Omega && \text{(2m)}\\
& z_{ij}^\omega \leq \bar{x}_{ij}G_i^\omega && \forall\, i\in I, j\in J_i, \omega\in\Omega && \text{(2n)}\\
& z_{ij}^\omega \leq x_{ij}^\omega && \forall\, i\in I, j\in J_i, \omega\in\Omega && \text{(2o)}\\
& z_{ij}^\omega \geq x_{ij}^\omega + \bar{x}_{ij}(G_i^\omega - 1) && \forall\, i\in I, j\in J_i, \omega\in\Omega && \text{(2p)}\\
& t_i \geq 0 && \forall\, i\in I && \text{(2q)}\\
& t_i^\omega \geq H^\omega G_i^\omega && \forall\, i\in I, \omega\in\Omega && \text{(2r)}\\
& 0 \leq x_{ij} \leq \bar{x}_{ij} && \forall\, i\in I, j\in J_i && \text{(2s)}\\
& 0 \leq x_{ij}^\omega \leq \bar{x}_{ij} && \forall\, i\in I, j\in J_i, \omega\in\Omega && \text{(2t)}\\
& 0 \leq z_{ij}^\omega \leq 1 && \forall\, i\in I, j\in J_i, \omega\in\Omega && \text{(2u)}\\
& G_i^\omega \in \{0,1\}. && \forall\, i\in I, \omega\in\Omega. && \text{(2v)}
\end{aligned}
$$

In model (2), we minimize the expected project span, weighing the span under each scenario by its probability. We replicate constraints (1b)-(1d) for the nominal scenario as (2b)-(2d). In constraints (2e)-(2f), variable $G_i^\omega$ takes value 1 if activity $i$ starts after the disruption time; otherwise it takes value 0. The value of $M$ must be sufficiently large to enforce this logic, and can be the largest possible start time of any activity in an optimal solution. We derive such value, $T_{\max}$, in Section 5. The logic constraints are important because the duration of each activity depends on its temporal relationship to the disruption time, which is reflected in constraint (2k). Also, we must ensure that decisions made before the disruption time in each scenario match the nominal decisions, and constraints (2g)-(2j) capture these non-anticipativity conditions. For each scenario, the duration between activity $i$ and $k$ becomes $(D_{ik} + d_{ik}^\omega G_i^\omega)(1 - \sum_{j \in J_i} e_{ij} x_{ij}^\omega)$, which expands to the form of constraint (2k). If $G_i^\omega = 0$, which means activity $i$ starts before the disruption time of scenario $\omega$, this expression is the same as $D_{ik}(1 - \sum_{j \in J_i} e_{ij} x_{ij})$ because $x_{ij} = x_{ij}^\omega$ is enforced by constraints (2i) and (2j). If $G_i^\omega = 1$, then the duration between activity $i$ and $k$ is changed to $D_{ik} + d_{ik}^\omega \geq 0$. We allow a negative "increase" in duration $d_{ik}^\omega$, but choose the parameters so that the overall duration is nonnegative. The expression $(D_{ik} + d_{ik}^\omega G_i^\omega)(1 - \sum_{j \in J_i} e_{ij} x_{ij}^\omega)$ contains a bilinear term $G_i^\omega x_{ij}^\omega$, which we linearize using variable $z_{ij}^\omega$, constraints (2n)-(2p), and a big-$M$ value of $\bar{x}_{ij}$ via constraint (2t).

Model (2) assumes that disruptions can only affect (typically delay) activities that have not yet started. Appendix A details an alternative in which disruptions affect ongoing activities that have yet to end.

## 3 NP-Hardness

We show that the optimal project crashing problem under a stochastic disruption is NP-hard even with a single disruption scenario, which occurs with probability one at time zero. Our proof relies on a transformation from the exactly-one-in-three `3SAT` (`EOIT_3SAT`) problem: Let $U = \{u_1, u_2, \ldots, u_n\}$ be a set of variables. A literal can be either $u$ or $\bar{u} = \neg u$ for $u \in U$. Let $C = \{c_1, c_2, \ldots, c_m\}$ be a set of clauses, each of which is formed by a disjunction of three literals, e.g., $c_i = u_j \vee u_k \vee \bar{u}_\ell$. The `EOIT_3SAT` problem asks whether there is a truth assignment for each $u \in U$ such that each clause in $C$ has exactly one true literal. De et al. De et al. (1997) use `EOIT_3SAT` to prove that an activity network problem, in which there are a finite set of alternatives for each activity with different duration and cost, is NP-hard, and we use similar proof constructs.

Starting with an instance of `EOIT_3SAT`, we formulate an activity network using three layers of nodes. The first layer contains $3n$ nodes and represents the truth assignment of each variable in `EOIT_3SAT`. The second layer contains $3m$ nodes and represents the value of `EOIT_3SAT`'s clauses. And, the third layer consists of terminal node $T$, the end of the project. Each of the first layer's $n$ components corresponds to a variable and contains three nodes, denoted $u_{j1}$, $u_{j2}$, and $u_{j3}$, which

are connected as shown in Figure 1. The figure also shows how the first layer connects to the third layer. We let $\Omega = \{1\}$, and $H^1 = 0$ with probability $p^1 = 1$. We define the parameter values associated with the arcs in Figure 1 as:

$$D_{u_{j1},u_{j2}} = 1 \qquad d^1_{u_{j1},u_{j2}} = 1 \tag{3a}$$

$$D_{u_{j1},u_{j3}} = 2 \qquad d^1_{u_{j1},u_{j3}} = -1 \tag{3b}$$

$$D_{u_{j3},T} = 0 \qquad d^1_{u_{j3},T} = 0. \tag{3c}$$

No activities in the first layer can be crashed, i.e.,

$$J_{u_{jk}} = \emptyset, \text{ for all } j = 1, 2, \ldots, n, \ k = 1, 2, 3. \tag{4}$$

The start node, $S$, connects to each $u_{j1}$ with zero duration. It is optimal to start each activity $u_{j1}$ at time 0 because the inclusive inequalities in constraints (2e)-(2f) still allow us to choose $G^1_{u_{j1}} \in \{0, 1\}$ for each variable in $U$. With this setup, the length of the longer path through the $j$-th component is always 2, $j = 1, 2, \ldots, n$. Whether the longer path traverses activity $u_{j2}$ (top path in Figure 1) or activity $u_{j3}$ (bottom path) depends on the value of $G^1_{u_{j1}}$. If $G^1_{u_{j1}} = 1$ then the top path yields a length of $D_{u_{j1},u_{j2}} + d^1_{u_{j1},u_{j2}} = 1 + 1 = 2$, while the bottom path yields a length of $D_{u_{j1},u_{j3}} + d^1_{u_{j1},u_{j3}} = 2 - 1 = 1$. If $G^1_{u_{j1}} = 0$ then the top path yields a length of $D_{u_{j1},u_{j2}} = 1$, while the bottom path yields a length of $D_{u_{j1},u_{j3}} = 2$. We can consider the value of $G^1_{u_{j1}}$ as the truth assignment of variable $u_j$. If variable $u_j$ is TRUE, the top path is longer; if it is FALSE the bottom path is longer. The arcs from activities $u_{j2}$ and $u_{j3}$ in Figure 1 point to activities in the

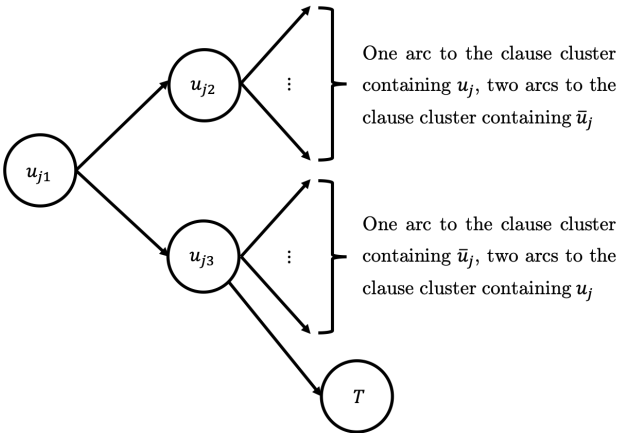



Figure 1: A component corresponding to variable $u_j$, $j = 1, 2, \ldots, n$, in the first layer of the activity network for `EOIT_3SAT`.

second layer, which we now construct, again following ideas in De et al. De et al. (1997). Consider the clause $c_i = u_j \vee u_k \vee \bar{u}_\ell$, with literals consisting of two original variables, $u_j$ and $u_k$, and one complement, $\bar{u}_\ell$. We consider an activity, $c_{ip}$, corresponding to the truth assignment of the variable $u_p$, for $p \in \{j, k, \ell\}$. For the original variables $u_p$, $p \in \{j, k\}$, we connect activity $u_{p3}$ to activity

$c_{ip}$, and we connect activity $u_{p2}$ to the other two activities $c_{iq}$, where $q \in \{j,k,\ell\}, q \neq p$. For the complemented variable $\bar{u}_\ell$, we do the opposite, connecting activity $u_{\ell 2}$ to activity $c_{i\ell}$, and activity $u_{\ell 3}$ to activities $c_{iq},\ q \in \{j,k\}$. This illustrates the general rule by which a clause with three literals (typically a mix of original and complemented variables) yields the network topology:

- *original variables* result in a connection from $u_{j3}$ to $c_{ij}$ via a single arc and a connection from $u_{j2}$ to the other two $c_j$-activity nodes; and, (5a)
- *complemented variables* result in the opposite. (5b)

From now on we refer to the activities representing variables as $u$-activities and those representing clauses as $c$-activities.

We make the following assignments:

$$D_{u_{jp},c_{ij}} = d^1_{u_{jp},c_{ij}} = 0 \qquad \forall i = 1,\ldots,m, j = 1,\ldots,n, p = 2,3 : u_j \text{ is in clause } c_i \tag{6a}$$

$$D_{c_{ij},T} = 1 \text{ and } d^1_{c_{ij},T} = 0 \qquad \forall i = 1,\ldots,m, j = 1,\ldots,n : u_j \text{ is in clause } c_i \tag{6b}$$

$$e_{c_{ij},1} = 1 \qquad \forall i = 1,\ldots,m, j = 1,\ldots,n : u_j \text{ is in clause } c_i \tag{6c}$$

$$b_{ij} = 1 \qquad \forall i = 1,\ldots,m, j = 1,\ldots,n : u_j \text{ is in clause } c_i \tag{6d}$$

$$B = 2m \tag{6e}$$

The nominal duration and the disrupted duration for the arcs between $u$-activities and $c$-activities are 0 per equation (6a). The nominal and disrupted durations for the arcs between the $c$-activities and $T$ are specified in equation (6b). Unlike the $u$-activities, each $c$-activity can be crashed with a single option with unit effectiveness as given in equation (6c). We assign the budget in equation (6e), where $m$ is the total number of clauses, and assign unit $b_{ij}$ values in equation (6d). We illustrate the logic behind this construction using Figure 2. For variable $u_j$, if $G^1_{u_{j1}} = 1$ then the earliest time activity $u_{j2}$ can start is 2, and activity $u_{j3}$ can start at time 1. If $G^1_{u_{j1}} = 0$ then the earliest time activity $u_{j3}$ can start is 2, and activity $u_{j2}$ can start at time 1. The same holds for variable $u_k$, and the opposite for variable $u_\ell$. The truth assignments indicate which path is longer.

Next, we establish two lemmas, which relate start times at certain nodes in the activity network corresponding to an instance of `EOIT_3SAT`.

**Lemma 1.** *Consider an instance of* `EOIT_3SAT`*, and the corresponding activity network for this instance. For each clause, $c_i$, $i = 1,2,\ldots,m$, there is at most one activity $c_{iq^*},\ q^* \in \{j,k,\ell\}$, that has 1 as its earliest start time, and the start time for $c_{iq}$ is 2, for $q \in \{j,k,\ell\}, q \neq q^*$.*

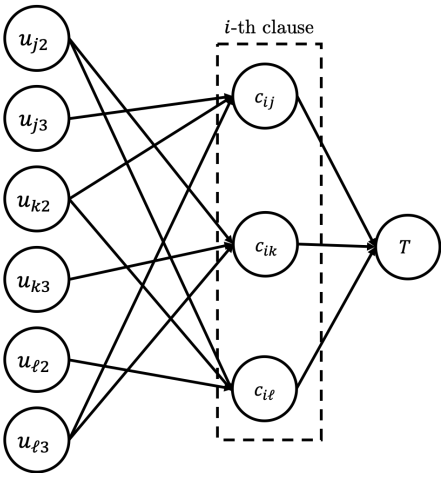



Figure 2: The $i$-th clause, $u_j \vee u_k \vee \bar{u}_\ell$, in the second layer of the constructed activity network for `EOIT_3SAT` with the arcs connecting it with the first and the third layer.

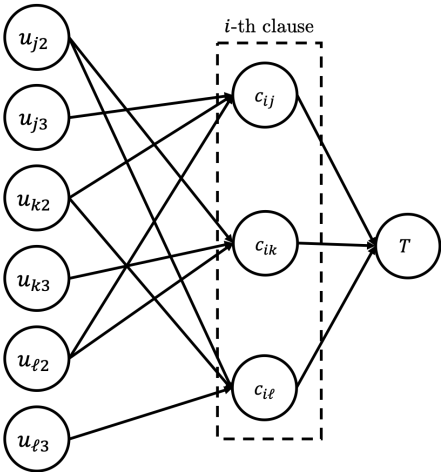



Figure 3: The $i$-th clause, $u_j \vee u_k \vee u_\ell$, in the second layer of the constructed activity network for `EOIT_3SAT` with the arcs connecting it with the first and the third layer.

*Proof of Lemma 1.* The proof enumerates eight cases, and we begin with $c_i = u_j \vee u_k \vee u_\ell$, which has a component of the activity network illustrated in Figure 3. Without loss of generality, suppose both $c_{ij}$ and $c_{ik}$ can start at time 1. This means that both $u_{j2}$ and $u_{j3}$ need to start at time 1, which is impossible because regardless of $G^1_{u_{j1}}$'s value, at least one activity in $\{u_{j2}, u_{j3}\}$ can start no earlier than time 2. The proof is completed by enumerating the remaining seven cases—with variables $u_j, u_k, u_\ell$ in all combinations of original or complemented form—in analogous fashion. □

□

**Lemma 2.** *Consider an instance of* `EOIT_3SAT`*, and the corresponding activity network for this instance. For any clause, $c_i$, activity $c_{iq^*}$, $q^* \in \{j, k, \ell\}$, has an earliest start time of* 1 *if and only if the corresponding literal, $u_{q^*}$ or $\bar{u}_{q^*}$, is the only literal in the clause to which the truth assignment*

*is TRUE.*

*Proof of Lemma 2.* The proof again enumerates eight cases, and we begin with $c_i = u_j \vee u_k \vee u_\ell$; see Figure 3. Without loss of generality, we assume $q^* = j$.
( $\Longrightarrow$ ): In turn we suppose $u_j$ is FALSE or $u_k$ is TRUE or $u_\ell$ is TRUE. First, suppose $u_j$ is FALSE, i.e., $G^1_{u_{j1}} = 0$. Then activity $u_{j3}$ can start no earlier than time 2, which leads to the contradiction that $c_{ij}$ can start as early as time 1; see Figure 3. Suppose $u_k$ is TRUE. Since there is an arc from $u_{k2}$ to activity $c_{ij}$, and since $u_{k2}$ cannot start before time 2 this again contradicts that $c_{ij}$ can start as early as time 1. The argument for $u_\ell$ being TRUE is identical. Therefore, if activity $c_{ij}$ can start at time 1 then $u_j$ is the only literal which is TRUE.
( $\Longleftarrow$ ): if only $u_j$ is TRUE, then $u_{j3}, u_{k2}$ and $u_{\ell 2}$ can all start as early as time 1; again, see Figure 3. This means that the earliest start time for $c_{ij}$ is 1.
We again complete the proof by enumerating the remaining seven cases. □ □

As a result of Lemmas 1 and 2, we can transform an `EOIT_3SAT` instance to an instance of model (2) using the activity network construction process just described. In particular, we know that if there exists a truth assignment to the variables of $U$ that meets the requirement of `EOIT_3SAT`, there are exactly $2m$ $c$-activities (two per clause) that can start no earlier than time 2. Since we have budget $B = 2m$, we can crash all of those $c$-activities to achieve a project length of 2. If there is no truth assignment that meets the requirement of `EOIT_3SAT` then model (2)'s optimal value is 3. We formalize this in what follows.

**Definition 1.** Stochastic Crashing Decision Problem: *Is there a feasible solution,* $(t, x, G)$*, to model* (2) *with objective function value of at most* $\tau$*?*

**Theorem 3.** *Consider an instance of* `EOIT_3SAT`*, and the corresponding activity network for this instance. In particular, let* $\Omega = \{1\}$*,* $H^1 = 0$*,* $p^1 = 1$*, and let the network topology and model parameters be given by Figure 1, rule* (5) *and equations* (3)*,* (4)*, and* (6)*. Let* $\tau = 2$*. The answer to the* Stochastic Crashing Decision Problem *is yes if and only if the given instance of* `EOIT_3SAT` *problem has a solution, i.e., a truth assignment to the variables so that each clause has exactly one true literal.*

*Proof of Theorem 3.* The `EOIT_3SAT` problem has $n$ variables and $m$ clauses, and the constructed activity network for the project crashing problem has $3n + 3m + 2$ activities and $4n + 12m$ arcs. Thus the size of the activity network and the time required to construct the network are both polynomial in the size of the original `EOIT_3SAT` instance.
( $\Longleftarrow$ ) Suppose the `EOIT_3SAT` instance has a solution. A feasible solution to the instance of model (2) starts every activity as early as possible. Under the `EOIT_3SAT` hypothesis, by Lemmas 1 and 2 exactly $2m$ $c$-activities have earliest start times of 2 and the remaining $m$ have an earliest

start time of 1. Spending the budget, $B = 2m$, to crash all $2m$ $c$-activities that correspond to the literals with FALSE assignment, yields an objective function value of 2.
($\Longrightarrow$) Suppose the instance of model (2) has a solution, $(\hat{t}, \hat{x}, \hat{G})$, with an objective function value of 2. By Lemma 1 we know that for each clause there is at most one $c$-activity that can start at time 1, which means there must be at least $2m$ $c$-activities with a start time of at least 2. Since $B = 2m$, if there are more than $2m$ $c$-activities that start at time 2 or after, the objective function value of model (2) must exceed 2. Hence, there are exactly $2m$ $c$-activities starting at time 2. Lemmas 1 and 2 then imply that exactly one $c$-activity in each of the $m$ clauses starts at time 1; i.e,. for each clause there is exactly one variable to which the truth assignment is TRUE.

`EOIT_3SAT` is NP-complete Garey and Johnson (1979). The Stochastic Crashing Decision Problem is in NP because we can check in $O(n+m)$ time whether a given solution is feasible and has an objective function value of at most $\tau$. □ □

As a result of Theorem 3 we immediately obtain the following result.

**Corollary 4.** *Model* (2) *is NP-hard, even under a single disruption scenario, which occurs with probability one at time zero.*

## 4 Illustration of Problem Properties via Examples

We show two examples of serial activity networks to give insight regarding the nature of the project crashing problem under a stochastic disruption, and to draw distinctions relative to its deterministic counterpart. In the deterministic project crashing problem, all activities on the critical path should start as soon as possible. However, with a stochastic disruption, it is sometimes optimal to delay the start of one or more activities. In addition, under a stochastic disruption, it is possible that on a critical path, an activity with a shorter expected duration is crashed with a larger amount of resource, while in the deterministic case, it is always optimal to crash the activity with the longest duration on the critical path, under equal $b_{ij}$ and $e_{ij}$ values. We use two examples to show that the deterministic optimal solution can be significantly inferior in the stochastic setting because of these two properties. Here, we assume that the required duration that separates the start of activity $i$ and the start of a successor, $k$, only depends on $i$; i.e., for each activity $i$ we use $D_i$ to denote the duration of activity $i$ and $d_i^\omega$ to denote the change in duration under scenario $\omega$:

$$D_{ik} = D_i \qquad \forall (i,k) \in \mathcal{A}$$
$$d_{ik}^\omega = d_i^\omega \qquad \forall (i,k) \in \mathcal{A}, \omega \in \Omega.$$

Clearly delaying the start of an activity may be beneficial when $d_i^\omega < 0$ for some $i \in I, \omega \in \Omega$ because the expected decrease in duration may exceed the delay required to move the start of

activity $i$ after a potential disruption. In the following example, we show the value of a delay, even if all activities are lengthened by the disruption; i.e., $d_i^\omega > 0, \forall i \in I, \omega \in \Omega$.

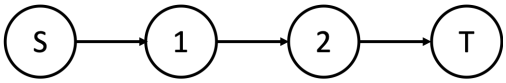



Figure 4: Example of a 2-activity serial network project.

**Example 1.** *Consider a network with two activities in series, as shown in Figure 4 with* $I = \{S, 1, 2, T\}$*, and let parameter* $\kappa > 4$*. Let the nominal durations be* $D_1 = \kappa$ *and* $D_2 = 1$*. We assume only one crashing option for each activity, and so we omit index* $j$*. We let* $e_1 = e_2 = 1 - \frac{1}{2\kappa}$*, assume* $b_i = 1$ *for all* $i \in I$*, and we let* $B = 1$*. Let* $\Omega = \{1\}$ *so that either we have no disruption with probability* $p^0 = 1 - \frac{1}{\kappa}$*, or we have a disruption that occurs at time* $\varepsilon < \frac{1}{2}$ *with probability* $p^1 = \frac{1}{\kappa}$*. If a disruption occurs, the nominal activity durations are lengthened by* $d_1 = \kappa$ *and* $d_2 = (\kappa - 1)^2$*.*

*If we start each activity without delay, then* $t_1 = 0$*,* $x_1 = x_1^1$*, and for any* $x_1 \leq 1$*,* $t_2 \geq \frac{1}{2}$ *because* $D_1(1 - e_1 x_1) \geq \kappa(1 - 1 + \frac{1}{2\kappa}) = \frac{1}{2}$*, which means activity 2 will start after the disruption. Since* $\kappa > 1$*, the duration of activity 1,* $D_1 = \kappa$*, exceeds the expected duration of activity 2,* $D_2 + p^1 d_2 = 1 + \frac{1}{\kappa}(\kappa - 1)^2 = \kappa - 1 + \frac{1}{\kappa}$*. As a result, it is optimal to spend the entire budget on activity 1:* $x_1 = x_1^1 = 1$ *and* $x_2 = x_2^1 = 0$*, and the expected project duration is:*

$$\begin{aligned}
&D_1(1 - e_1 x_1) + p^0 D_2 + p^1(D_2 + d_2) \\
=&\kappa\left(1 - \left(1 - \frac{1}{2\kappa}\right) \cdot 1\right) + \left(1 - \frac{1}{\kappa}\right) \cdot 1 + \frac{1}{\kappa}\left(1 + (\kappa - 1)^2\right) \\
=&\kappa - \frac{1}{2} + \frac{1}{\kappa}.
\end{aligned}$$

*On the other hand, if we delay the start of activity 1 until* $t_1 = \varepsilon$ *then* $x_1$ *and* $x_1^1$ *need not be equal. Since for* $\kappa > 2 + \sqrt{2}$*,* $\kappa = D_1 > D_2 = 1$ *and* $(\kappa - 1)^2 + 1 = D_2 + d_2 > D_1 + d_1 = \kappa + \kappa$*, we have* $x_1 = 1$*,* $x_1^1 = 0$*,* $x_2^1 = 1$ *in an optimal solution, and the expected duration is:*

$$\begin{aligned}
&\varepsilon + p^0\left[D_1 \cdot \left(1 - \left(1 - \frac{1}{2\kappa}\right) \cdot 1\right) + D_2\right] + p^1\Bigg[(D_1 + d_1) + \\
&\qquad \left(1 - \left(1 - \frac{1}{2\kappa}\right) \cdot 1\right)(D_2 + d_2)\Bigg] \\
=&\varepsilon + \frac{\kappa - 1}{\kappa}\left(\kappa \cdot \frac{1}{2\kappa} + 1\right) + \frac{1}{\kappa}\left[(\kappa + \kappa) + \frac{1}{2\kappa}\left((\kappa - 1)^2 + 1\right)\right] \\
=&\varepsilon + 4 - \frac{5}{2\kappa} + \frac{1}{\kappa^2}.
\end{aligned}$$

□

In Example 1, if we require that activity 1 start without delay then the objective function grows to infinity with $\kappa$, but the optimal project span by delaying the start of activity 1 by $\varepsilon$ has

a constant limit of $\varepsilon + 4$. This example shows that the gap between the optimal solution under a no-delay policy and an optimal solution that allows for delay—as we do in model (2)—can be arbitrarily large. Intentionally delaying the start of certain activities can provide flexibility for crashing decisions. We pay the cost of a delay but save more time in expectation by being able to use adaptive second-stage crashing decisions. Because it is possible for an optimal crashing plan to contain a delay for some activities, model (2) uses decision variables $t_i$, $\forall i \in I$, as the start time of each activity, rather than assuming that each activity starts as soon as all of its predecessors are finished.

**Example 2.** *We again consider the network with two activities from Figure 4. Let* $D_2 > D_1$, $d_1 = 0$, $d_2 > 0$, $e_1 = e_2 = \frac{1}{2}$, *and* $B = 1$. *We again consider a single disruption scenario,* $\Omega = \{1\}$, *so that either we have no disruption with probability* $p^0 = \frac{1}{2}$, *or we have a disruption that occurs at time* $H^1 = \frac{1}{2}D_1$ *with probability* $p^1 = \frac{1}{2}$. *Here, the optimal solution is to crash the shorter activity, i.e.,* $x_1 = 1$, *which yields an expected project span of* $\frac{1}{2}D_1 + D_2$ *with start times* $t_1 = 0$ *and* $t_2 = \frac{1}{2}D_1$. *In contrast, if* $0 \le x_1 < 1$ *then the expected duration is* $D_1(1 - \frac{1}{2}x_1) + (D_2 + \frac{1}{2}d_2)(1 - \frac{1}{2}x_2)$, *so that the ratio of the objective functions grows arbitrarily large as* $d_2$ *grows.* □

In Example 2 the intuition behind crashing the shorter activity is that it allows us to initiate activity 2 in time to avoid incurring delay $d_2$. Both examples in this section suggest that the intuition associated with the deterministic version of the optimal crashing problem does not always apply in the stochastic setting, and provides further motivation for employing a model like that in formulation (2).

## 5 A Decomposition Algorithm

Model (2) is a two-stage stochastic mixed-integer program:

$$z^* = \min \quad p^0 t_T + \sum_{\omega \in \Omega} p^\omega f^\omega(t, x) \tag{7a}$$

$$\text{s.t.} \quad t_k - t_i \ge D_{ik}\left(1 - \sum_{j \in J_i} e_{ij} x_{ij}\right) \qquad \forall (i,k) \in \mathcal{A} \tag{7b}$$

$$\sum_{i \in I} \sum_{j \in J_i} b_{ij} x_{ij} \le B \tag{7c}$$

$$\sum_{j \in J_i} x_{ij} \le 1 \qquad \forall\, i \in I \tag{7d}$$

$$t_i \ge 0 \qquad \forall\, i \in I \tag{7e}$$

$$0 \le x_{ij} \le \bar{x}_{ij} \qquad \forall\, i \in I, j \in J_i. \tag{7f}$$

With the first stage solution $(\hat{t}, \hat{x})$ as input, the second stage problem is:

$$(S^\omega) \quad f^\omega(\hat{t}, \hat{x}) = \min \quad t_T \tag{8a}$$

$$
\begin{aligned}
\text{s.t.} \quad & H^{\omega} + MG_i \geq \hat{t}_i && \forall\, i \in I && \text{(8b)}\\
& H^{\omega} - M(1 - G_i) \leq \hat{t}_i && \forall\, i \in I && \text{(8c)}\\
& t_i + M' G_i \geq \hat{t}_i && \forall\, i \in I && \text{(8d)}\\
& t_i - M' G_i \leq \hat{t}_i && \forall\, i \in I && \text{(8e)}\\
& x_{ij} + \bar{x}_{ij} G_i \geq \hat{x}_{ij} && \forall\, i \in I, j \in J_i && \text{(8f)}\\
& x_{ij} - \bar{x}_{ij} G_i \leq \hat{x}_{ij} && \forall\, i \in I, j \in J_i && \text{(8g)}\\
& t_k - t_i \geq D_{ik} + d^{\omega}_{ik} G_i - \\
& \quad \sum_{j \in J_i} D_{ik} e_{ij} x_{ij} - \sum_{j \in J_i} d^{\omega}_{ik} e_{ij} z_{ij} && \forall\, (i,k) \in \mathcal{A} && \text{(8h)}\\
& \sum_{i \in I} \sum_{j \in J_i} b_{ij} x_{ij} \leq B && && \text{(8i)}\\
& \sum_{j \in J_i} x_{ij} \leq 1 && \forall\, i \in I && \text{(8j)}\\
& z_{ij} \leq \bar{x}_{ij} G_i && \forall\, i \in I, j \in J_i && \text{(8k)}\\
& z_{ij} \leq x_{ij} && \forall\, i \in I, j \in J_i && \text{(8l)}\\
& z_{ij} \geq x_{ij} + \bar{x}_{ij} (G_i - 1) && \forall\, i \in I, j \in J_i && \text{(8m)}\\
& t_i \geq 0 && \forall i \in I && \text{(8n)}\\
& t_i \geq H^{\omega} G_i && \forall\, i \in I && \text{(8o)}\\
& 0 \leq x_{ij} \leq \bar{x}_{ij} && \forall\, i \in I, j \in J_i && \text{(8p)}\\
& 0 \leq z_{ij} \leq 1 && \forall\, i \in I, j \in J_i && \text{(8q)}\\
& G_i \in \{0, 1\} && \forall\, i \in I. && \text{(8r)}
\end{aligned}
$$

A number of existing approaches for stochastic mixed-integer programming assume a special structure not satisfied by our model. For example, Gade et al. Gade et al. (2014) solve two-stage stochastic programs with pure binary first stage variables, and general integer second stage variables; they derive a finitely convergent sequential convex approximation and a branch-and-cut framework involving Gomory cuts that are parameterized by the first-stage decision variables. Zou et al. Zou et al. (2019) assume state variables are binary (or general integer via binary expansion) in a multi-stage setting so that the Lagrangian cuts are a tight approximation of the recourse function; see also Philpott et al. Philpott et al. (2020). Carøe and Tind Carøe and Tind (1998) solve a more general case of two-stage models by using integer programming duality, but there is limited computational work investigating their approach. Qi and Sen Qi and Sen (2017) allow mixed-integer variables in both the first stage and the recourse problem, and parametric disjunctive cuts convexify recourse problems while Benders' cuts approximate recourse functions Chen et al. (2012). Although this method suits our problem setting, preliminary computational results (discussed below) found that it was not competitive with the scheme we describe here, which makes use of the special

structure of our problem.

A simple approach is to relax the integrality constraints (8r) of *subproblem* ($S^\omega$), and execute a multi-cut L-shaped decomposition algorithm on this linear programming (LP) relaxation. The resulting optimality cuts provide a valid lower approximation of the recourse functions, $f^\omega(t,x), \forall \omega \in \Omega$, but may not be tight. In each iteration of the decomposition, an upper bound can be obtained by solving subproblems (8) with the first stage solution. The main challenge is how to iteratively tighten the lower bound while quickly locating a good upper bound. The topics in this section aim to tackle these two issues.

The combinatorial decision, $G_i \in \{0,1\}$, $i \in I$, for each ($S^\omega$) is (almost fully) decided by the first-stage continuous variables, $t_i$. If $\hat{t}_i > H^\omega$ then $G_i = 1$, if $\hat{t}_i < H^\omega$ then $G_i = 0$, and only if they are equal is there a combinatorial choice. This observation motivates the decomposition algorithm that we develop. We could pull these binary variables to the first stage, but doing so involves $|I||\Omega|$ variables and does not scale well. Instead, we partition an interval, which we denote $[0, T_{\max}]$, containing each $t_i$, and we adaptively refine that partition. This helps control the number of binary first-stage variables, and has further benefits in terms of tightening lower bounds, as we describe in what follows. We will be specific later regarding the value of $T_{\max}$, but for now we simply assume we have a value such that $t_i \in [0, T_{\max}]$ is a redundant constraint in model (2).

We assume that $\Omega = \{1, 2, \ldots, |\Omega|\}$ is such that $\omega < \omega'$ implies $H^\omega \leq H^{\omega'}$ with strict inequality if the realizations of $H$ are distinct, and we let $H^0 \equiv 0 \leq H^1$ and $H^{|\Omega|+1} \equiv T_{\max} \geq H^{|\Omega|}$. We define a partition of $[0, T_{\max}]$ for each $i \in I$ as follows.

**Definition 2.** *For each activity $i \in I$, we define a partition of interval $[0, T_{\max}]$ as an ordered set of two-element tuples $\mathcal{P}_i = \{[\underline{H}^q, \bar{H}^q],\ q \in \mathcal{Q}_i\}$ with an index set $\mathcal{Q}_i = \{1, 2, \ldots, |\mathcal{Q}_i|\}$ and the following properties:*

- $\underline{H}^1 = H^0 \equiv 0$
- $\bar{H}^{|\mathcal{Q}_i|} = H^{|\Omega|+1} \equiv T_{\max}$
- $\underline{H}^q < \bar{H}^q \quad \forall q \in \mathcal{Q}_i$
- $\bar{H}^q = \underline{H}^{q+1} \quad \forall q \in \mathcal{Q}_i$.

With the possible exceptions of $\underline{H}^1$ and $\bar{H}^{|\mathcal{Q}_i|}$, each $\underline{H}^q$ and $\bar{H}^q$ corresponds to a disruption time of some scenario, and a simple example is illustrated in Figure 5 in which we have five scenarios, $\Omega = \{1, 2, 3, 4, 5\}$. The partition has three intervals as illustrated. The second interval has lower bound $H^2$ and upper bound $H^5$.

With this setup, we make sure the start time of each activity falls in some element of the partition. In particular, for each activity $i \in I$ the first-stage start time $t_i$ lies an interval of $\mathcal{P}_i$ as

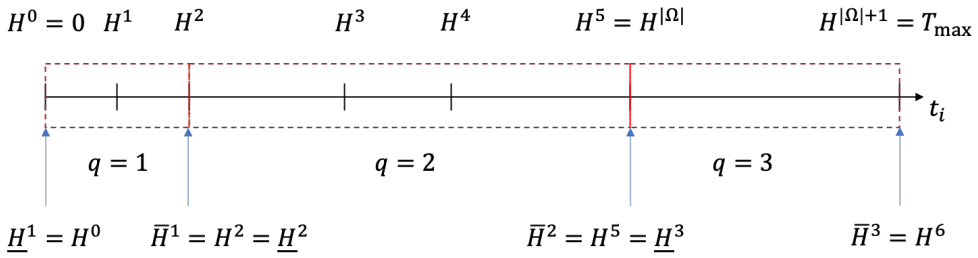



Figure 5: An illustration of a partition of interval $[0, T_{\max}]$.

specified by a first-stage indicator variable:

$$\sum_{q \in \mathcal{Q}_i} \underline{H}^q y_i^q \leq t_i \leq \sum_{q \in \mathcal{Q}_i} \bar{H}^q y_i^q \qquad \forall i \in I \tag{9a}$$

$$\sum_{q \in \mathcal{Q}_i} y_i^q = 1 \qquad \forall i \in I \tag{9b}$$

$$y_i^q \in \{0, 1\}, \qquad \forall i \in I, q \in \mathcal{Q}_i. \tag{9c}$$

Constraints (9) require that $t_i$ be associated with one of the intervals of the partition. In model (2) if $t_i = H^\omega$ then $G_i^\omega$ can either be 0 (activity $i$ is said to start before $\omega$'s disruption) or 1 ($i$ starts after the disruption). If $t_i = \bar{H}^q = \underline{H}^{q+1} = H^\omega$ for some $\omega$ then the $y$-variables have a similar choice, and our convention is that if the $y$-variables choose $t_i \in [\underline{H}^q, \bar{H}^q]$ then activity $i$ is said to start before the disruption and if $t_i \in [\underline{H}^{q+1}, \bar{H}^{q+1}]$ then $i$ starts after the disruption.

### 5.1 Tightening Big-$M$ with Partitions

The tightness of $(S^\omega)$'s LP relaxation relies, in part, on the big-$M$ value used in constraints to represent the logical condition of whether activity $i$ starts before or after a disruption. A smaller, but still valid, big-$M$ value yields a tighter relaxation, and can further help prevent numerical issues, e.g., Camm et al. (1990), Klotz and Newman (2013). We can rewrite constraints (8b) and (8c) as:

$$(\hat{t}_i - H^\omega)/M \leq G_i \leq (\hat{t}_i - H^\omega)/M + 1. \tag{10}$$

Variable $G_i$ can take a wider range of values when $M$ is large. Tightening $M$ hinges on specifying valid ranges for $\hat{t}_i$. Furthermore, we know that if $\hat{t}_i > H^{\omega'}$ for some $\omega'$ then $G_i$ has to take value 1 in all subproblems $(S^\omega)$ with $\omega < \omega'$. On the other hand, if $\hat{t}_i < H^{\omega'}$ for some $\omega'$ then $G_i$ must be 0 for all subproblems $(S^\omega)$ in which $\omega > \omega'$. If we can fix the $G_i$ for all $i \in I$ to either 0 or 1 the resulting optimality cuts will be tight.

**Proposition 5.** *Let $t_i^0$ be the length of the longest $S$-$i$ path in the activity network $\mathcal{G} = (I, \mathcal{A})$ in which the arc length of $(i,k) \in \mathcal{A}$ is $D_{ik}$ and in which no crashing is allowed. Then there exists (part of) an optimal solution to model (2), $t^*$, such that $t_i^* \in [0, H^{|\Omega|} + t_i^0]$, $\forall i \in I$, provided $M$ and $M'$ are sufficiently large.*

*Proof of Proposition 5.* Constraint (2q) enforces the lower bound of 0. By hypothesis $t_k^0 - t_i^0 \geq D_{ik}, \forall (i,k) \in \mathcal{A}$ since $t_i^0$ is the length of the longest $S$-$i$ path of $\mathcal{G}$ in which the arc length of $(i,k) \in \mathcal{A}$ is $D_{ik}$ and in which no crashing is allowed. We prove the upper bound on $t_i^*$ by contradiction.

Suppose there does not exist a $t^*$ such that $t_i^* \in [0, H^{|\Omega|} + t_i^0],\ \forall i \in I$. Then for every $t^*$, there must be a set $I^* \subseteq I$ such that $t_i^* > H^{|\Omega|} + t_i^0$ for $i \in I^*$. Let the corresponding optimal values of variables $x_{ij}, G_i^\omega, t_i^\omega, x_{ij}^\omega, z_{ij}^\omega$ be denoted $x_{ij}^*, G_i^{\omega,*}, t_i^{\omega,*}, x_{ij}^{\omega,*}, z_{ij}^{\omega,*}$, respectively. We can establish a feasible solution to model (2) as follows:

$$\tilde{t}_i = H^{|\Omega|} + t_i^0 \qquad \forall i \in I^* \tag{11a}$$

$$\tilde{t}_i = t_i^* \qquad \forall i \in I \backslash I^* \tag{11b}$$

$$\tilde{x}_{ij} = x_{ij}^* \qquad \forall i \in I, j \in J_i \tag{11c}$$

$$\tilde{G}_i^\omega = G_i^{\omega,*} \qquad \forall i \in I, \omega \in \Omega \tag{11d}$$

$$\tilde{t}_i^\omega = t_i^{\omega,*} \qquad \forall i \in I, \omega \in \Omega \tag{11e}$$

$$\tilde{x}_{ij}^\omega = x_{ij}^{\omega,*} \qquad \forall i \in I, j \in J_i, \omega \in \Omega \tag{11f}$$

$$\tilde{z}_{ij}^\omega = z_{ij}^{\omega,*} \qquad \forall i \in I, j \in J_i, \omega \in \Omega. \tag{11g}$$

We see this solution is feasible by examining the constraints of model (2):

- For constraint (2b), we examine the following four possible cases:
  - $i \in I^*, k \in I^*$: since $\tilde{x}_{ij} \geq 0$ and $t_k^0 - t_i^0 \geq D_{ik}$, we have
  $$\tilde{t}_k - \tilde{t}_i = t_k^0 - t_i^0 \geq D_{ik} \geq D_{ik} \left(1 - \sum_{j \in J_i} e_{ij} \tilde{x}_{ij}\right);$$
  - $i \in I^*, k \notin I^*$: we have $\tilde{t}_i < t_i^*$ since $i \in I^*$, and then
  $$\tilde{t}_k - \tilde{t}_i > t_k^* - t_i^* \geq D_{ik} \left(1 - \sum_{j \in J_i} e_{ij} x_{ij}^*\right) = D_{ik} \left(1 - \sum_{j \in J_i} e_{ij} \tilde{x}_{ij}\right);$$
  - $i \notin I^*, k \in I^*$: since $t_i^* \leq H^{|\Omega|} + t_i^0$, $\tilde{x}_{ij} \geq 0$ and $t_k^0 - t_i^0 \geq D_{ik}$, we have
  $$\begin{aligned} \tilde{t}_k - \tilde{t}_i = H^{|\Omega|} + t_k^0 - t_i^* \geq & H^{|\Omega|} + t_k^0 - \left(H^{|\Omega|} + t_i^0\right) \\ = & t_k^0 - t_i^0 \geq D_{ik} \geq D_{ik} \left(1 - \sum_{j \in J_i} e_{ij} \tilde{x}_{ij}\right); \end{aligned}$$
  - $i, k \notin I^*$: the constraint is unchanged and feasible;
- For variable $G_i^\omega$:
  - if $i \notin I^*$: we have $\tilde{t}_i = t_i^*$. Therefore, constraints (2e)-(2h) is unchanged and feasible;

– if $i \in I^*$: variable $G_i^{*,\omega}$ is forced to take value 1 for all $\omega \in \Omega$. Since $\tilde{t}_i = H^{|\Omega|} + t_i^0 \geq H^{|\Omega|}$, for any $\omega \in \Omega$, $\tilde{G}_i^\omega = G_i^{*,\omega} = 1$ remains feasible. Therefore, constraints (2e)-(2h) hold for $M$ and $M'$ sufficiently large.

- Since the values for $\tilde{t}_i^\omega, \tilde{x}_{ij}, \tilde{x}_{ij}^\omega, G_i^\omega, z_{ij}^\omega$ all remain the same, constraints (2c), (2d), (2i)-(2v) are all satisfied by the solution in (11).

We can also see that $\tilde{t}_i \leq t_i^*,\ \forall i \in I$ because of equations (11a) and (11b). Thus, the feasible solution in (11) yields an objective function value $p^0\tilde{t}_T + \sum_{\omega\in\Omega} p^\omega \tilde{t}_T^\omega \leq p^0 t_T^* + \sum_{\omega\in\Omega} p^\omega t_T^{*,\omega}$. This means that the feasible solution in (11) is also optimal. As $\tilde{t}_i \leq H^{|\Omega|} + t_i^0,\ \forall i \in I$, it contradicts the assumption that there does not exist an optimal solution with $t_i^* \in [0, H^{|\Omega|} + t_i^0],\ \forall i \in I$. □□

For simplicity of exposition, we use an identical upper bound on every $t_i$ for $i \in I$ as $T_{\max} = H^{|\Omega|} + t_T^0$. From a computational perspective, there is an efficient algorithm to obtain the longest path length $t_T^0$ as the graph is acyclic. While Proposition 5 bounds the start-time variables, $t_i,\ \forall i \in I$, to the interval $[0, T_{\max}]$, the $y$-variables of (9) allow for tighter bounds. Given a partition for each activity and given a first-stage solution with $\hat{y}_i^q, \forall i \in I, q \in \mathcal{Q}_i$, we can replace constraints (8b) and (8c) with:

$$H^\omega + G_i \left( \sum_{q\in\mathcal{Q}_i} \bar{H}^q \hat{y}_i^q - H^\omega \right) \geq \hat{t}_i \qquad \forall i \in I \tag{12a}$$

$$H^\omega - (1 - G_i) \left( H^\omega - \sum_{q\in\mathcal{Q}_i} \underline{H}^q \hat{y}_i^q \right) \leq \hat{t}_i \qquad \forall i \in I. \tag{12b}$$

In a first-stage solution, $\hat{y}_i^q = 1$ for a specific $q$ satisfying $\underline{H}^q \leq \hat{t}_i \leq \bar{H}^q$, and verifying the validity of constraints (12) in replacing (8b)-(8c) is straightforward by enumerating the cases $H^\omega \in [\underline{H}^q, \bar{H}^q]$, $H^\omega < \underline{H}^q$, and $H^\omega > \bar{H}^q$. In the degenerate case in which $H^\omega$ coincides with $\bar{H}^q$ or $\underline{H}^q$, the corresponding constraint from (12) is as tight as possible, i.e., a simple bound on $\hat{t}_i$ involving $H^\omega$. Otherwise constraints (12) reduce to the following analog of (10):

$$(\hat{t}_i - H^\omega)/(\bar{H}^q - H^\omega) \leq G_i \leq (\hat{t}_i - H^\omega)/(H^\omega - \underline{H}^q) + 1, \tag{13}$$

and we see that smaller values of $\bar{H}^q$ and larger values of $\underline{H}^q$ have the effect of tightening the big-$M$ value in constraints (10).

We introduce variables $F_i^q$ to linearize the bilinear terms, and rewrite constraints (12) as follows:

$$\sum_{q\in\mathcal{Q}_i} \bar{H}^q F_i^q - H^\omega G_i \geq \hat{t}_i - H^\omega \qquad \forall i \in I \tag{14a}$$

$$H^\omega G_i - \sum_{q\in\mathcal{Q}_i} \underline{H}^q F_i^q \le \hat{t}_i - \sum_{q\in\mathcal{Q}_i} \underline{H}^q \hat{y}_i^q \qquad \forall i \in I \tag{14b}$$

$$F_i^q \le G_i \qquad \forall i \in I, q \in \mathcal{Q}_i \tag{14c}$$

$$F_i^q \le \hat{y}_i^q \qquad \forall i \in I, q \in \mathcal{Q}_i \tag{14d}$$

$$F_i^q \ge G_i + \hat{y}_i^q - 1 \qquad \forall i \in I, q \in \mathcal{Q}_i. \tag{14e}$$

We can further tighten the formulation by adding two constraints involving $y$ that cover cases when $G_i$ can be fixed to 0 or 1. Again given a partition of each activity and a first-stage solution $\hat{y}_i^q, \forall i \in I, q \in \mathcal{Q}_i$, we have:

$$\sum_{q\in\mathcal{Q}_i, H^\omega \le \underline{H}^q} \hat{y}_i^q \le G_i \le 1 - \sum_{q\in\mathcal{Q}_i, H^\omega \ge \bar{H}^q} \hat{y}_i^q \qquad \forall i \in I. \tag{15}$$

For the case in which $H^\omega \in (\underline{H}^q, \bar{H}^q)$ for the $q$ with $\hat{y}_i^q = 1$, constraint (15) adds no restriction, but for the cases in which $H^\omega \le \underline{H}^q$ and $H^\omega \ge \bar{H}^q$, $G_i$ is forced to 1 and 0, respectively.

### 5.2 Partition-based Decomposition Method

Our decomposition algorithm to solve model (2) iteratively partitions the continuous feasible region of the first-stage $t$-variables by introducing binary variables that facilitate tighter optimality cuts. With the addition of constraints (14) and (15), the tightened subproblem is:

$$(S_{\mathcal{P}}^\omega) \quad f_{\mathcal{P}}^\omega(\hat{t}, \hat{x}, \hat{y}) = \min \quad t_T \tag{16a}$$

$$\text{s.t.} \quad \sum_{q\in\mathcal{Q}_i} \bar{H}^q F_i^q - H^\omega G_i \ge \hat{t}_i - H^\omega \qquad \forall i \in I \tag{16b}$$

$$H^\omega G_i - \sum_{q\in\mathcal{Q}_i} \underline{H}^q F_i^q \le \hat{t}_i - \sum_{q\in\mathcal{Q}_i} \underline{H}^q \hat{y}_i^q \qquad \forall i \in I \tag{16c}$$

$$F_i^q \le G_i \qquad \forall i \in I, q \in \mathcal{Q}_i \tag{16d}$$

$$F_i^q \le \hat{y}_i^q \qquad \forall i \in I, q \in \mathcal{Q}_i \tag{16e}$$

$$F_i^q \ge G_i + \hat{y}_i^q - 1 \qquad \forall i \in I, q \in \mathcal{Q}_i. \tag{16f}$$

$$G_i \ge \sum_{q\in\mathcal{Q}_i, H^\omega \le \underline{H}^q} \hat{y}_i^q \qquad \forall i \in I \tag{16g}$$

$$G_i \le 1 - \sum_{q\in\mathcal{Q}_i, H^\omega \ge \bar{H}^q} \hat{y}_i^q \qquad \forall i \in I \tag{16h}$$

$$\text{constraints (8d)-(8q)} \tag{16i}$$

$$0 \le F_i^q \le 1 \qquad \forall\, i \in I, q \in \mathcal{Q}_i \tag{16j}$$

$$0 \le G_i \le 1. \qquad \forall\, i \in I. \tag{16k}$$

We express $(S_{\mathcal{P}}^\omega)$ in LP relaxation form, excluding constraints (8r). Let $\ell$ denote the iteration of the decomposition algorithm, and $(\hat{t}^\ell, \hat{x}^\ell, \hat{y}^\ell)$ denote a given first-stage decision. We solve $(S_{\mathcal{P}}^\omega)$ for

each $\omega \in \Omega$ and construct an optimality cut of the form:

$$\theta^\omega \geq v^{\omega,\ell} + \sum_{i\in I} \pi_i^{\omega,\ell}(t_i - \hat{t}_i^\ell) + \sum_{i\in I}\sum_{j\in J_i} \lambda_{ij}^{\omega,\ell}(x_{ij} - \hat{x}_{ij}^\ell) + \sum_{i\in I}\sum_{q\in\mathcal{Q}_i^\ell} \gamma_i^{\omega,\ell,q}\left(y_i^q - \hat{y}_i^{q,\ell}\right). \tag{17}$$

Here, $\theta^\omega$ is a continuous decision variable that forms an outer-linearization of $f_{\mathcal{P}}^\omega(t,x,y)$; parameter $v^{\omega,\ell}$ is the optimal value of $(S_{\mathcal{P}}^\omega)$ at $(\hat{t}^\ell, \hat{x}^\ell, \hat{y}^\ell)$; and, coefficients $\pi, \lambda$ and $\gamma$ are appropriate sums of dual variables from the LP relaxation—e.g., $\pi_i^{\omega,\ell}$ involves dual variables from constraints (8d)-(8e) and (16b)-(16c). Since we solve a linear relaxation, $\theta^\omega$ is a lower bound on $f^\omega(\hat{t}^\ell, \hat{x}^\ell, \hat{y}^\ell)$. However, the cut needs to be modified once the partition is updated to maintain validity, and we assume that the update refines the partition for each $i \in I$ by subdividing one or more intervals as indicated in the following definition.

**Definition 3.** *For two partitions $\mathcal{P}_i^1$ and $\mathcal{P}_i^2$, indexed by $\mathcal{Q}_i^1$ and $\mathcal{Q}_i^2$, respectively, we say $\mathcal{P}_i^2$ is a* refinement *of $\mathcal{P}_i^1$ provided:*

$$\forall q^2 \in \mathcal{Q}_i^2, \exists q^1 \in \mathcal{Q}_i^1 \text{ s.t. } \bar{H}^{q^1} \geq \bar{H}^{q^2} \text{ and } \underline{H}^{q^1} \leq \underline{H}^{q^2}.$$

At the current iteration for each $i \in I$, let the partition $\mathcal{P}_i$ be indexed by $\mathcal{Q}_i$, and assume this partition is formed from earlier partitions by a sequence of refinements satisfying Definition 3. We can then find a set of intervals in the current partition, $\mathcal{P}_i$, whose union is the $q$-th interval in partition $\mathcal{P}_i^\ell$ from previous iteration $\ell$. We index such a *descendant set* by $\Delta_i(\ell, q)$. Cut (17) can then be updated to the following form:

$$\theta^\omega \geq v^{\omega,\ell} + \sum_{i\in I} \pi_i^{\omega,\ell}(t_i - \hat{t}_i^\ell) + \sum_{i\in I}\sum_{j\in J_i} \lambda_{ij}^{\omega,\ell}(x_{ij} - \hat{x}_{ij}^\ell) + \sum_{i\in I}\sum_{q\in\mathcal{Q}_i^\ell} \gamma_i^{\omega,\ell,q}\left(\sum_{q'\in\Delta_i(\ell,q)} y_i^{q'} - \hat{y}_i^{q,\ell}\right). \tag{18}$$

We show that given a partition, $\mathcal{P} = \bigtimes_{i\in I} \mathcal{P}_i$, which is updated by sequential refinement from a previous partition, $\mathcal{P}^\ell = \bigtimes_{i\in I} \mathcal{P}_i^\ell$, the optimality cut (18) is a valid lower approximation for $f_{\mathcal{P}}^\omega(t,x,y)$.

**Proposition 6.** *For each $i \in I$, suppose we have a partition, $\mathcal{P}_i$, indexed by $\mathcal{Q}_i$, which is a refinement of $\mathcal{P}_i^\ell$, indexed by $\mathcal{Q}_i^\ell$. Then at any given feasible $(t,x,y)$ we have*

$$f_{\mathcal{P}}^\omega(t,x,y) \geq v^{\omega,\ell} + \sum_{i\in I} \pi_i^{\omega,\ell}(t_i - \hat{t}_i^\ell) + \sum_{i\in I}\sum_{j\in J_i} \lambda_{ij}^{\omega,\ell}(x_{ij} - \hat{x}_{ij}^\ell) + \sum_{i\in I}\sum_{q\in\mathcal{Q}_i^\ell} \gamma_i^{\omega,\ell,q}\left(\sum_{q'\in\Delta_i(\ell,q)} y_i^{q'} - \hat{y}_i^{q,\ell}\right). \tag{19}$$

*Proof of Proposition 6.* We denote the recourse function corresponding to partition $\mathcal{P}^\ell$ by $f^\omega_{\mathcal{P}^\ell}(t,x,y)$, where $y$ has the correct dimension according to $\mathcal{P}^\ell$. We first show that

$$f^\omega_{\mathcal{P}}(t,x,y) \geq f^\omega_{\mathcal{P}^\ell}(t,x,\tilde{y}), \tag{20}$$

where

$$\tilde{y}_i^q = \sum_{q' \in \Delta_i(\ell,q)} y_i^{q'} \qquad \forall i \in I, q \in \mathcal{Q}_i^\ell. \tag{21}$$

Suppose for given $(t,x,y)$, we solve $(S^\omega_{\mathcal{P}})$ and obtain an optimal solution $(t^\omega, x^\omega, G^\omega, F^\omega)$. We then form

$$\tilde{F}_i^{\omega,q} = \sum_{q' \in \Delta_i(\ell,q)} F_i^{\omega,q'} \qquad \forall i \in I, q \in \mathcal{Q}_i^\ell,$$

and we obtain a feasible solution $(t^\omega, x^\omega, G^\omega, \tilde{F}^\omega)$ to subproblem $(S^\omega_{\mathcal{P}^\ell})$. Therefore, inequality (20) holds. Furthermore, the cut generated under partition $\mathcal{P}^\ell$ is

$$\theta^\omega \geq v^{\omega,\ell} + \sum_{i\in I} \pi_i^{\omega,\ell}(t_i - \hat{t}_i^\ell) + \sum_{i\in I}\sum_{j\in J_i} \lambda_{ij}^{\omega,\ell}(x_{ij} - \hat{x}_{ij}^\ell) + \sum_{i\in I}\sum_{q\in\mathcal{Q}_i^\ell} \gamma_i^{\omega,\ell,q}\left(y_i^q - \hat{y}_i^{q,\ell}\right),$$

which means that for any feasible $(t,x,\tilde{y})$, we have

$$\begin{aligned} f^\omega_{\mathcal{P}^\ell}(t,x,\tilde{y}) \geq & v^{\omega,\ell} + \sum_{i\in I} \pi_i^{\omega,\ell}(t_i - \hat{t}_i^\ell) + \sum_{i\in I}\sum_{j\in J_i} \lambda_{ij}^{\omega,\ell}(x_{ij} - \hat{x}_{ij}^\ell) + \\ & \sum_{i\in I}\sum_{q\in\mathcal{Q}_i^\ell} \gamma_i^{\omega,\ell,q}\left(\tilde{y}_i^q - \hat{y}_i^{q,\ell}\right). \end{aligned} \tag{22}$$

Using equation (21), we replace $\tilde{y}$ by $y$, and combine inequalities (20) and (22) to obtain (19). □□

Proposition 6 states that by properly modifying the $y$-variables in cuts generated under earlier partitions, the resulting cuts (18) are valid in the sense of providing a lower approximation on the LP relaxation of the recourse function. Therefore, we incorporate the modified cuts (18) in the following master problem, given a partition $\mathcal{P}$, which is indexed by $\mathcal{Q}$:

$$(M_{\mathcal{P}}) \quad z^*_{\mathcal{P}} = \min \quad p^0 t_T + \sum_{\omega\in\Omega} p^\omega \theta^\omega \tag{23a}$$

$$\text{s.t.} \quad t_k - t_i \geq D_{ik}\left(1 - \sum_{j\in J_i} e_{ij}x_{ij}\right) \qquad \forall (i,k) \in \mathcal{A} \tag{23b}$$

$$\sum_{i\in I}\sum_{j\in J_i} b_{ij}x_{ij} \leq B \tag{23c}$$

$$\sum_{j\in J_i} x_{ij} \leq 1 \qquad \forall\, i \in I \tag{23d}$$

$$\sum_{q\in\mathcal{Q}_i} \underline{H}^q y_i^q \leq t_i \leq \sum_{q\in\mathcal{Q}_i} \bar{H}^q y_i^q \qquad \forall i \in I \tag{23e}$$

$$\sum_{q\in\mathcal{Q}_i} y_i^q = 1 \qquad \forall i \in I \tag{23f}$$

$$\theta^\omega \geq v^{\omega,\ell} + \sum_{i\in I} \pi_i^{\omega,\ell}(t_i - \hat{t}_i^\ell) + \sum_{i\in I}\sum_{j\in J_i} \lambda_{ij}^{\omega,\ell}(x_{ij} - \hat{x}_{ij}^\ell) + \sum_{i\in I}\sum_{q\in\mathcal{Q}_i^\ell} \gamma_i^{\omega,\ell,q}\left(\sum_{q'\in\Delta_i(\ell,q)} y_i^{q'} - \hat{y}_i^{q,\ell}\right) \qquad \forall \omega\in\Omega, \ell = 1,2,\ldots,L \tag{23g}$$

$$y_i^q \in \{0,1\} \qquad \forall i \in I, q \in \mathcal{Q}_i \tag{23h}$$

$$t_i \geq 0 \qquad \forall\, i \in I \tag{23i}$$

$$0 \leq x_{ij} \leq \bar{x}_{ij} \qquad \forall\, i \in I, j \in J_i. \tag{23j}$$

As we refine the partitions, the generated cuts become tighter and we provide tighter lower bounds on model (2)'s optimal value. We refine the partition by selecting the interval of $\mathcal{P}_i$ for each activity $i \in I$, where for some scenarios $\omega \in \Omega$ with $H^\omega \in [\underline{H}^q, \bar{H}^q]$, $G_i^\omega$ has a fractional value, and partition the interval as we describe in further detail in Section 5.7.

The development so far in this section is summarized in the decomposition procedure of Algorithm 1, which makes use of the tightened master problem ($M_\mathcal{P}$) in (23), the original subproblem ($S^\omega$) in (8), and the tightened subproblem ($S^\omega_\mathcal{P}$) in (16). The algorithm couples Benders' decomposition and iterative refinement of the partition defining the $y$-variables. Cuts of the form (17) are added in step 12. These optimality cuts, along with those from previous coarser partitions, are updated to the form of (18) based on the current partition in step 17. Algorithm 1 serves as a precursor to our more effective algorithm developed later in this section, but its partition-based construct suffices to prove convergence.

We prove Algorithm 1 converges in finite number of iterations. Since every partition update is a refinement and we have a finite set of scenario $\Omega$, we can prove the finite convergence of Algorithm 1 as long as with the finest partition we reach the optimum of problem (2). These results use the following hypothesis:

$$G_i^\omega = G_i^{\omega'} \quad \forall\, i \in I, \omega, \omega' \in \Omega : H^\omega = H^{\omega'}. \tag{24}$$

**Proposition 7.** *Assume that for each partition, $\mathcal{P}_i$ indexed by $\mathcal{Q}_i$, and for each $H^\omega$ we have $H^\omega = \bar{H}^q$ for some $q \in \mathcal{Q}_i$, i.e., $|\mathcal{Q}_i| = |\Omega| + 1$ so that each partition is as fine as possible. Further assume hypothesis* (24)*. Then*

$$z^* = \min_{(t,x,y)\in\mathbb{X}} p^0 t_T + \sum_{\omega\in\Omega} p^\omega f_\mathcal{P}^\omega(t,x,y), \tag{25}$$

**Algorithm 1** Partition-based decomposition algorithm to solve model (2).

Initialize cut iteration number $\ell = 0$, lower bound $LB = 0$, upper bound $UB = +\infty$, initial partition $\mathcal{P}^\ell$ with its indexed set $\mathcal{Q}^\ell$, and tolerance parameters $\epsilon \geq 0$;
**while** $\frac{UB-LB}{UB} > \epsilon$ **do**
  Solve $(M_{\mathcal{P}})$ and obtain solution $\hat{t}^\ell, \hat{x}^\ell, \hat{y}^\ell, \hat{\theta}^\ell$ and optimal value $z^*_{\mathcal{P}}$;
  **if** $z^*_{\mathcal{P}} > LB$ **then**
    Update $LB = z^*_{\mathcal{P}}$;
  For each $\omega \in \Omega$, solve $(S^\omega)$ and obtain $f^\omega(\hat{t}^\ell, \hat{x}^\ell)$ and $\hat{G}^\ell$;
  Calculate $\bar{z} = p^0 \hat{t}^\ell_T + \sum_{\omega\in\Omega} p^\omega f^\omega(\hat{t}^\ell, \hat{x}^\ell)$;
  **if** $\bar{z} < UB$ **then**
    Update $UB = \bar{z}$ and incumbent solution as $t^* = \hat{t}^\ell, x^* = \hat{x}^\ell$ and $G^* = \hat{G}^\ell$;
  **for** each $\omega \in \Omega$ **do**
    solve $(S^\omega_{\mathcal{P}})$ given $\hat{t}^\ell, \hat{x}^\ell, \hat{y}^\ell$ and obtain optimal value $v^{\omega,\ell}$ and $\pi^{\omega,\ell}, \lambda^{\omega,\ell}, \gamma^{\omega,\ell}$;
    **if** $\hat{\theta}^{\omega,\ell} < v^{\omega,\ell}$ **then** add cut of form (17) to $(M_{\mathcal{P}})$;
  **if** there are cuts added **then**
    Let $\mathcal{P}^{\ell+1} = \mathcal{P}^\ell$ and $\mathcal{Q}^{\ell+1} = \mathcal{Q}^\ell$;
  **else**
    Refine the partition and obtain the new partition $\mathcal{P}^{\ell+1}$ and its indexed sets $\mathcal{Q}^{\ell+1}$;
    Update previously generated cuts to the form of (18);
  Let $\ell = \ell + 1$;
**end while**
Output $UB$ as the $\epsilon$-optimal value of model (2), and $(t^*, x^*, G^*)$ as the $\epsilon$-optimal solution.

*where*

$$\mathbb{X} = \left\{ (t,x,y) \middle| \begin{array}{ll} \textit{constraints (23b)-(23f)} & \\ y_i^q \in \{0,1\} & \forall i \in I, q \in \mathcal{Q}_i \\ t_i \geq 0 & \forall i \in I \\ 0 \leq x_{ij} \leq \bar{x}_{ij} & \forall i \in I, j \in J_i \end{array} \right\}$$

*and where $z^*$ is the optimal value of model (2).*

*Proof of Proposition 7.* We first consider the case in which all realizations of $H^\omega$, $\omega \in \Omega$, are distinct so that hypothesis (24) immediately holds. We can formulate an extensive form for model (25), i.e., an analog of model (2), using $\mathbb{X}$ and model (16), where the decision variables of model (16) are now also indexed by $\omega$. Let $\Omega = \{1, 2, \ldots, |\Omega|\}$ so that the realizations of the disruption times are given by $H^1 < H^2 < \cdots < H^{|\Omega|}$. By Definition 2, under the first hypothesis of the proposition, the intervals of each partition are given by the finest partition, $[H^0, H^1]$, $[H^1, H^2]$, $\ldots$, $[H^{|\Omega|-1}, H^{|\Omega|}]$, $[H^{|\Omega|}, H^{|\Omega|+1}]$, which we can index by $q = 1, 2, \ldots, |\Omega|$, $|\Omega|+1$, where $H^0 = 0$ and $H^{|\Omega|+1} = T_{\max}$. Under the assumed partition, constraints (16g)-(16h) in the extensive form of (25) reduce to

$$\sum_{q=\omega+1}^{|\Omega|+1} y_i^q \leq G_i^\omega \leq 1 - \left( \sum_{q=1}^{\omega} y_i^q \right), \forall i \in I, \omega \in \Omega. \tag{26}$$

Under this finest partition, there is a one-to-one mapping between the $G$- and $y$-variables under the binary restrictions imposed by models (2) and (25). In particular,

$$y_i^q = 1 \text{ if and only if } G_i^\omega = 1 \ \forall \omega \geq q+1 \text{ and } G_i^\omega = 0 \ \forall \omega \leq q. \tag{27}$$

Any feasible solution to model (2) necessarily satisfies the condition $G_i^\omega \leq G_i^{\omega+1}$ required by (27) via constraint (2f). And, constraints (23f) and (23h) ensures the 0-0 or 1-1 nature of the left- and right-hand side of (26). By Proposition 5 we know $t_i \in [0, T_{\max}]$, and hence this restriction imposed by model (25) is nonbinding. As a result, model (2) and the extensive form of model (25) are equivalent and yield the same optimal value.

Now suppose $\hat{\Omega} \subseteq \Omega$ is such that realizations of $H^\omega$ are identical for $\omega \in \hat{\Omega}$, where $|\hat{\Omega}| > 1$; i.e., scenarios in $\hat{\Omega}$ are distinguished only by $d_{ik}^\omega$. The proposition's second hypothesis ensures that the value of $G_i^\omega$ is consistent for each activity $i$ across $\hat{\Omega}$, and so all scenarios in $\hat{\Omega}$ can be treated identically when executing the branching process. Because the disruption magnitude $d_{ik}^\omega$ does not play a role in the branching process, the finest partition argument above still holds. □ □

As discussed after formulating models (7) and (8), the first-stage continuous variables, $t_i$, determine $G_i^\omega$ except when $t_i = H^\omega$, in which case the $G$-variables can either say that activity $i$ commences before or after the disruption. Hypothesis (24), which is repeated in the next theorem, assumes that these choices are made consistently across all scenarios with the same disruption time. This hypothesis is clearly satisfied if the realizations $H^\omega$, $\omega \in \Omega$, are distinct. Moreover, the hypothesis is automatically satisfied when $t_i \neq H^\omega$.

**Theorem 8.** *Assume that we obtain a dual extreme-point solution to subproblem* $(S_{\mathcal{P}}^\omega)$ *in step 11 of Algorithm 1. Further assume hypothesis* (24)*. Then, the algorithm terminates in finite number of iterations to an* $\epsilon$*-optimal solution to model* (2) *for any* $\epsilon \geq 0$*.*

*Proof of Theorem 8.* Let $z^*$ denote the optimal value of model (2) or equivalently model (7). The value of $UB$ in the algorithm is an upper bound on $z^*$ because $(\hat{t}^\ell, \hat{x}^\ell)$ is a feasible solution, and its objective function value in model (7) is evaluated in step 7 of the algorithm. The value of $LB$ in the algorithm is a lower bound on $z^*$ because: (i) Proposition 6 ensures that the cuts (23g) are an outer linearizations of $f_{\mathcal{P}}(t, x, y)$; inequality (20) in the proof of Proposition 6 shows that $f_{\mathcal{P}}$ becomes tighter as the partition is refined; and, Proposition 7 shows that the finest partition yields a model equivalent to model (2). Thus, if Algorithm 1 terminates according to step 2 then the current incumbent is an $\epsilon$-optimal solution.

For a fixed partition, $\mathcal{P}$, Algorithm 1 can only add a finite number of new cuts because each linear program $(S_{\mathcal{P}}^\omega)$ has a finite number of dual extreme points. After the final iteration in which new cuts are added, partition $\mathcal{P}$ is refined in step 16 of the algorithm. Because there are a finite

number of scenarios the finest possible partition, if necessary, will be obtained in a finite number of iterations. From Proposition 7 we know that with the finest partition the solution to model (25) yields an optimal solution, and we will obtain the requisite cuts (23g) so that models (23) and (25) are equivalent in a finite number of iterations. □ □

### 5.3 Pruning Partitions Using Bound Tightening

Feasibility-based and optimality-based bound tightening schemes have proved powerful in mixed-integer nonlinear programming to improve computational performance, e.g., Belotti et al. (2012), Coffrin et al. (2015), Sundar et al. (2018). A feasibility-based bound tightening (FBBT) process is suitable for our problem because the precedence relationships limit the start times of activities. We solve the following linear programs to identify lower and upper bounds on the first-stage start time of each activity:

$$\min / \max \quad t_i \tag{28a}$$

$$\text{s.t.} \quad \text{constraints (7b)-(7f)} \tag{28b}$$

$$\underline{t}_i \leq t_i \leq \bar{t}_i \qquad \forall i \in I. \tag{28c}$$

The bound tightening process starts with a set of initial bounds $\underline{t}_i = 0$ and $\bar{t}_i = T_{\max}$ for each $i \in I$. Looping over all the activities, we repeatedly solve model (28), iteratively updating the bounds $\bar{t}_i$ and $\underline{t}_i$ that appear in (28c) with the obtained solutions until the bounds converge for every $i \in I$.

We run FBBT at the beginning of our decomposition method to provide the initial partition $\mathcal{P}^0$ to start Algorithm 1. If we can tighten the bounds of some activities, for example, by branch-and-bound or heuristics, we can run FBBT to tighten the bounds for all activities, which leads to a tighter formulation of the subproblems, $(S^{\omega}_{\mathcal{P}})$, as further detailed in Section 5.8.

### 5.4 Obtaining a Heuristic Upper Bound

In our computation, we observe that it can take many iterations of Algorithm 1 before we find a good feasible solution. When there is not a good upper bound to use as a "cutoff value," it takes an integer programming solver longer to solve model $(M_{\mathcal{P}})$. On the other hand, we can quickly solve the extensive formulation (2) when the number of scenarios is small; e.g., $|\Omega| = 20$. Solving such a model provides a feasible first-stage solution $(\hat{t}, \hat{x})$, which can be used to generate an upper bound for the problem with the larger, original set of scenarios. Therefore, we can generate $N$ small subsets of scenarios from the original scenario set, i.e., $\Omega_n \subset \Omega,\ n = 1, 2, \ldots, N$, and solve model (2) with each of those subsets to generate candidate upper bounds, and select the one with the smallest expected project span under $\Omega$.

In generating $\Omega_n$, we observe that it is beneficial to have diverse scenarios within each subset. As above, we sort the scenarios by disruption time so $H^{\omega} \leq H^{\omega+1}$. For simplicity, suppose each

subset has equal size so that $N \cdot |\Omega_n| = |\Omega|,\ \forall n$. The $n$-th subset is then:

$$\Omega_n = \{\omega \,|\, \omega = n + (j-1) \cdot |\Omega_n|,\ j = 1, 2, \ldots, N\}.$$

In Section 6 we compare the computational time of the decomposition method with and without initial upper bounds, and show the significant performance improvement by including the heuristic upper bound.

### 5.5 Magnanti-Wong Cut Generation

In a Benders' decomposition algorithm, linear programming subproblems can have multiple optimal dual solutions, which means that at a specific incumbent solution, there are multiple valid cuts that could be generated. Magnanti and Wong Magnanti and Wong (1981) provide a method to select Pareto-optimal cuts, which cannot be dominated, and help tighten the LP relaxation of the master program.

We apply a technique that pursues the same goal as Magnanti and Wong Magnanti and Wong (1981) in order to tighten cuts at interior points of $(M_{\mathcal{P}})$. We record $(\hat{t}, \hat{x})$ solutions from previous iterations, compute the corresponding $\hat{y}$ for the current partition, and obtain the average, which we denote $(t^0, x^0, y^0)$. We then solve the subproblems following the procedure of Magnanti and Wong Magnanti and Wong (1981), using this average point as a proxy for the core point, i.e., a point on the relative interior of the LP relaxation of $(M_{\mathcal{P}})$'s feasible region. For each scenario $\omega \in \Omega$, first we solve model (16) with the current master solution $(\hat{t}, \hat{x}, \hat{y})$ to obtain the optimal value, $f^{\omega}_{\mathcal{P}}(\hat{t}, \hat{x}, \hat{y})$. Next we need to find dual variables that ensure the dual objective value at $(\hat{t}, \hat{x}, \hat{y})$ is within a small tolerance of $f^{\omega}_{\mathcal{P}}(\hat{t}, \hat{x}, \hat{y})$ (e.g., $\epsilon_{MW} = 10^{-5}$), while maximizing the dual objective value at $(t^0, x^0, y^0)$. Let the following denote the dual of $(S^{\omega}_{\mathcal{P}})$ in compact form, suppressing dependence on $\omega$:

$$f^{\omega}_{\mathcal{P}}(\hat{t}, \hat{x}, \hat{y}) = \max_{\pi, \lambda, \gamma, \eta} \quad \pi^\top(\hat{t} + b_t) + \lambda^\top(\hat{x} + b_x) + \gamma^\top(\hat{y} + b_y) + \eta^\top b \tag{29a}$$

$$\text{s.t.} \quad A_\pi^\top \pi + A_\lambda^\top \lambda + A_\gamma^\top \gamma + A_\eta^\top \eta \leq c. \tag{29b}$$

Here the $b$-parameters capture constant right-hand side terms in $(S^{\omega}_{\mathcal{P}})$ involving $H$, $\bar{x}$, $D$, "1", $B$, etc. We solve the following to obtain Magnanti-Wong cut parameters, $v$, $\pi$, $\lambda$, and $\gamma$:

$$v = \max_{\pi, \lambda, \gamma, \eta} \quad \pi^\top(t^0 + b_t) + \lambda^\top(x^0 + b_x) + \gamma^\top(y^0 + b_y) + \eta^\top b \tag{30a}$$

$$\text{s.t.} \quad \pi^\top(\hat{t} + b_t) + \lambda^\top(\hat{x} + b_x) + \gamma^\top(\hat{y} + b_y) + \eta^\top b \geq (1 - \epsilon_{MW}) f^{\omega}_{\mathcal{P}}(\hat{t}, \hat{x}, \hat{y}) \tag{30b}$$

$$A_\pi^\top \pi + A_\lambda^\top \lambda + A_\gamma^\top \gamma + A_\eta^\top \eta \leq c. \tag{30c}$$

Our average point, $(t^0, x^0, y^0)$, may not be in the relative interior of the master problem's feasible region, e.g., if all solutions contributing to the average have a component of $y$ taking value zero or one. Although this means the resulting cuts may not be Pareto-optimal, they may still improve computational performance, and we investigate this in Section 6.

### 5.6 Cut Selection

Algorithm 1 is a multi-cut version of Benders' decomposition Birge and Louveaux (1988). A multi-cut scheme can converge in fewer iterations, but each iteration is typically more computationally expensive. The latter issue tends to exacerbate as the algorithm proceeds and cuts accumulate, particularly when the master problem is a mixed-integer program.

As we discuss in the next section, when naively keeping all cuts, we see that the time required to solve the master problem can grow quickly as the algorithm proceeds. Therefore, we limit the number of cuts and only keep those that have been tight most recently at the end of each Benders' iteration. Refining a partition can yield many loose cuts. Therefore, keeping only a limited number of cuts eliminates unnecessary constraints, while still providing a valuable lower bound and reducing computational effort.

### 5.7 Refining Partitions

Here, we indicate how a partition is refined in step 16 of Algorithm 1. The most significant contributor to the optimality gap is that variable $G$ can take fractional values in the relaxed problem $(S^{\omega}_{\mathcal{P}})$. For example, if $d^{\omega}_{ik} \gg D_{ik}$ then $G_i$ can take a fractional value far from optimal. This can significantly alter the duration between activity $i$ and $k$ (see constraint (8h)) in the LP relaxation, and thus create a large gap between the relaxation and the original mixed-integer subproblem. Let $N_p$ be a parameter that limits the number of new partitions for each activity $i \in I$; we use $N_p = 5$ in our subsequent computation, unless stated otherwise. Suppose subproblem $(S^{\omega}_{\mathcal{P}})$ has as part of its solution $G^{\omega}_i, i \in I$. Then separately for each $i \in I$ our rule selects for partitioning up to $N_p$ scenarios with the largest values of:

$$\rho^{\omega}_i = \begin{cases} \max_{(i,k)\in\mathcal{A}} d^{\omega}_{ik} G^{\omega}_i & \text{if } t_i < H^{\omega} \\ \max_{(i,k)\in\mathcal{A}} d^{\omega}_{ik}(1 - G^{\omega}_i) & \text{if } t_i > H^{\omega} \\ \min\{\max_{(i,k)\in\mathcal{A}} d^{\omega}_{ik} G^{\omega}_i, \max_{(i,k)\in\mathcal{A}} d^{\omega}_{ik}(1 - G^{\omega}_i)\} & \text{if } t_i = H^{\omega}. \end{cases} \tag{31}$$

### 5.8 Branch-and-Cut Algorithm

In Algorithm 1 we iteratively refine the partition defining the $y$-variables, and each time we solve the mixed-integer linear program $(M_{\mathcal{P}})$, we do so with a commercial solver. Algorithm 1 is not a branch-and-cut (B&C) algorithm in that it does not adaptively generate different cuts at different parts of a branch-and-bound tree. As a potential improvement to Algorithm 1, we propose here a B&C algorithm, which involves a branch-and-bound (B&B) tree with nodes that we manage. The root node in our B&B tree involves the initial partition obtained by the FBBT procedure of Section 5.3, and we solve that node using Benders' decomposition, which iteratively adds cuts to $(M_{\mathcal{P}})$ until the problem is solved for the fixed partition, as in Algorithm 1. In the B&C algorithm we recursively branch on a continuous variable $t_i$ via $t_i \leq H^{\omega}$ and $t_i \geq H^{\omega}$ for some $i$-$\omega$ pair.

Rather than actually branching on the continuous $t$-variables, this branching is carried out using the partition $(\mathcal{P}_i, \mathcal{Q}_i)$ by fixing the corresponding subset of $y_i^q$-variables to zero. This helps manage both the number of binary variables and the number of optimality cuts in a master problem. In our implementation we solve the nodes in our B&B tree in parallel.

The optimal value of a B&B node provides a lower bound on the optimal value of (2). We also continually update a global upper bound each time we obtain a feasible solution in a Benders' decomposition iteration. If the gap between a node's lower bound and the global upper bound is smaller than the tolerance, we mark the node as fathomed. If not, we branch as follows:

$$\text{select } \omega \in \text{argmax}_{\omega \in \Omega} \left[ f^\omega(\hat{t}, \hat{x}) - f^\omega_{\mathcal{P}}(\hat{t}, \hat{x}, \hat{y}) \right] \tag{32a}$$

$$\text{select } i \in \text{argmax}_{i \in I} \left[ \rho_i^\omega \right]. \tag{32b}$$

In (32a) we select the scenario $\omega$ with the largest relaxation gap. Then in (32b) we select the activity with the largest $\rho_i^\omega$ from equation (31). This defines the $i$-$\omega$ pair for branching. Suppose $H^\omega \in [\underline{H}^q, \bar{H}^q]$ for some $q \in \mathcal{Q}_i$. The new partition $\mathcal{P}_i$ will have two new elements indexed by $q_1$ and $q_2$ defined as $[\underline{H}^{q_1}, \bar{H}^{q_1}] = [\underline{H}^q, H^\omega]$ and $[\underline{H}^{q_2}, \bar{H}^{q_2}] = [H^\omega, \bar{H}^q]$.

After a branch, for each child node we refine the partition on all activities $i \in I$ according to Section 5.7. The children inherit the parent node's cuts, updated in a similar fashion as inequality (18). To set up notation for Algorithm 2, suppose that for activity $i \in I$ the current B&B node, say node $n$, has a partition $\mathcal{P}_i^n$ indexed by set $\mathcal{Q}_i^n$, and its parent node $m$ has a partition $\mathcal{P}_i^m$ indexed by set $\mathcal{Q}_i^m$. Then, the cuts inherited from node $m$ are updated for node $n$ as:

$$\theta^\omega \geq v^{\omega,m,\ell} + \sum_{i \in I} \pi_i^{\omega,m,\ell}(t_i - \hat{t}_i^{m,\ell}) + \sum_{i \in I} \sum_{j \in J_i} \lambda_{ij}^{\omega,m,\ell}(x_{ij} - \hat{x}_{ij}^{m,\ell}) + \\ \sum_{i \in I} \sum_{q \in \mathcal{Q}_i^m} \gamma_i^{\omega,m,q} \left( \sum_{q' \in \Delta_i^n(m,q)} y_i^{q'} - \hat{y}_i^{q,m,\ell} \right) \qquad \forall \ell = 1, 2, \ldots, L. \tag{33}$$

Here, $\Delta_i^n(m, q)$ represents the descendant set of the partition $\mathcal{P}_i^n$ refined from the $q$-th element in the partition $\mathcal{P}_i^m$. The cuts remain valid by Proposition 6 because the partitions in the child node refine those of the parent node. The parent node is marked as fathomed after a refinement, and we select the next available node with the smallest lower bound.

Putting all these pieces together, we summarize our partition-based branch-and-cut decomposition method in Algorithm 2. As indicated above, in implementation we execute the algorithm's steps on each available node in parallel. As also indicated above, the algorithm inherits the convergence result of Theorem 8 due to its use of Algorithm 1's partitioning procedure for the $y$-variables.

### 5.9 Algorithm 2: Numerical Example

We illustrate Algorithm 2 on an example with $\epsilon = 0$. We consider the network with five activities that accompany the source and terminal, as shown in Figure 6. Each of the five activities has unit

**Algorithm 2** Partition-based branch-and-cut algorithm to solve model (2).

Initialize tolerance parameters $\epsilon \geq 0$, and a global upper bound $UB$ (Section 5.4);
Initialize the B&B tree with node 1, with the following properties: cut iteration number $\ell_1 = 1$, lower bound $LB^1$, initial partition $\mathcal{P}^1$ with its indexed set $\mathcal{Q}^1$ (Section 5.3);
**while** there exists an available node such that $\frac{UB - LB^n}{UB} > \epsilon$ **do**
    Select available node $n$ with smallest $LB^n$;
    Append inherited cuts from parent of node $n$ to master $(M^n_{\mathcal{P}})$ using (33);
    **repeat**
        Solve $(M^n_{\mathcal{P}})$ and obtain solution $\hat{t}^{\,\ell^n}, \hat{x}^{\ell^n}, \hat{y}^{\ell^n}, \hat{\theta}^{\ell^n}$ and optimal value $z^*_{\mathcal{P}}$;
        **if** $z^*_{\mathcal{P}} > LB^n$ **then**
            Update $LB^n = z^*_{\mathcal{P}}$;
        For each $\omega \in \Omega$, solve problem $(S^\omega)$ and obtain $f^\omega(\hat{t}^{\ell^n}, \hat{x}^{\ell^n})$ and $\hat{G}^{\ell^n}$;
        Calculate $\bar{z} = p^0 \hat{t}^{\,\ell^n}_T + \sum_{\omega \in \Omega} p^\omega f^\omega(\hat{t}^{\ell^n}, \hat{x}^{\ell^n})$;
        **if** $\bar{z} < UB$ **then**
            Update $UB = \bar{z}$ and incumbent solution as $t^* = \hat{t}^{\,\ell^n}, x^* = \hat{x}^{\ell^n}$ and $G^* = \hat{G}^{\ell^n}$;
        **for** each $\omega \in \Omega$ **do**
            solve $(S^\omega_{\mathcal{P}})$ given $\hat{t}^{\,\ell^n}, \hat{x}^{\ell^n}, \hat{y}^{\ell^n}$ and obtain optimal value $v^{\omega,\ell^n}$ and $\pi^{\omega,\ell^n}, \lambda^{\omega,\ell^n}, \gamma^{\omega,\ell^n}$ (i.e., Magnanti-Wong cut coefficients in Section 5.5);
            **if** $\hat{\theta}^{\omega,\ell} < v^{\omega,\ell}$ **then** add cut of form (17) to $(M^n_{\mathcal{P}})$;
        Let $\ell^n = \ell^n + 1$;
    **until** no cut is added;
    **if** $\frac{UB - LB^n}{UB} > \epsilon$ **then**
        Branch via (32), creating two available children nodes, $n_1$ and $n_2$, from node $n$;
        Refine partition for $n_1$ and $n_2$ to obtain $\mathcal{P}^{n_1}$ and $\mathcal{P}^{n_2}$, respectively (Section 5.7);
        Let $LB^{n_1} = LB^{n_2} = LB^n$;
        Select cuts to retain (Section 5.6);
    Mark node $n$ as fathomed;
**end while**
Output $UB$ as the $\epsilon$-optimal value of model (2), and $(t^*, x^*, G^*)$ as the $\epsilon$-optimal solution.

duration, and can be crashed with a single option that decreases the duration by 90% with one unit of resource consumption, i.e., $e_{i1} = 0.9,\ \forall i \in I \setminus \{S, T\}$, and we let $B = 2$. The probability of no disruption is $p^0 = 0.2$, and the disruption can occur at four discrete time points, $H^1 = 1,\ H^2 = 2,\ H^3 = 3,\ H^4 = 4$, with equal probability $p^\omega = 0.2,\ \forall \omega \in \Omega = \{1, 2, 3, 4\}$. The increase in duration under a disruption is $d^\omega_i = 10$ for each activity-scenario pair. By Proposition 5 we can bound the start time of each activity from above by $T_{\max} = 9$.

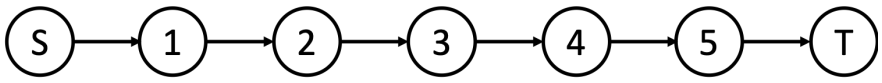



Figure 6: A five-activity serial network to illustrate Algorithm 2.

Running the FBBT procedure of Section 5.3 we obtain $\underline{t}_1 = 0$, $\underline{t}_2 = 0.1$, $\underline{t}_3 = 0.2$, $\underline{t}_4 = 1.2, \underline{t}_5 = 2.2$, and $\underline{t}_T = 3.2$. Given these values, and the realizations of $H^\omega$, we initialize node 1 of Algorithm 2

with the following partition, which precludes certain intervals for activities 4, 5, and $T$:

$$\begin{aligned}
&\mathcal{P}_1^1 = \{[0,9]\} && \mathcal{Q}_1^1 = \{1\} && \\
&\mathcal{P}_2^1 = \{[0,9]\} && \mathcal{Q}_2^1 = \{1\} && \\
&\mathcal{P}_3^1 = \{[0,9]\} && \mathcal{Q}_3^1 = \{1\} && \\
&\mathcal{P}_4^1 = \{[0,1],[1,9]\} && \mathcal{Q}_4^1 = \{1,2\} && y_4^1 = 0 \\
&\mathcal{P}_5^1 = \{[0,2],[2,9]\} && \mathcal{Q}_5^1 = \{1,2\} && y_5^1 = 0 \\
&\mathcal{P}_T^1 = \{[0,3],[3,9]\} && \mathcal{Q}_T^1 = \{1,2\} && y_T^1 = 0.
\end{aligned}$$

Executing steps 6-18 of the algorithm, we solve node 1 with Benders' decomposition, converging to an optimal value of $z_{\mathcal{P}}^* = 4.072$, and in the process, we obtain an upper bound of $UB = 6.607$. The optimal solution is:

$$\begin{aligned}
&\hat{t}_1 = 0 && \hat{x}_{11} = 0 \\
&\hat{t}_2 = 1 && \hat{x}_{21} = 0.1124 \\
&\hat{t}_3 = 1.899 && \hat{x}_{31} = 0.1124 \\
&\hat{t}_4 = 2.798 && \hat{x}_{41} = 0.8892 \\
&\hat{t}_5 = 2.997 && \hat{x}_{51} = 0.8860 \\
&\hat{t}_T = 3.2. &&
\end{aligned}$$

At this solution, by equation (32a) the largest relaxation gap is incurred at $\omega = 1$, and equation (32b) corresponds to activity 3, which has a fractional solution of $G_3 = 0.125$ in subproblem $(S_{\mathcal{P}}^1)$. We branch on $i$-$\omega$ pair 3-1, creating two children, node 2 and node 3. Node 2 has an additional constraint, $t_3 \geq H^1 = 1$ and node 3 has an additional constraint $t_3 \leq H^1 = 1$. For each activity $i$, we refine the partition by selecting the scenario with the largest nonzero $\rho_i^\omega$ for $\omega \in \Omega$, i.e., for simplicity we use $N_p = 1$; see Section 5.7. If the largest $\rho_i^\omega$ is zero for activity $i$, we do not refine that activity's partition. We again apply the ideas of Section 5.3, and solve a series of linear programs (28) to tighten the bounds of start times for nodes 2 and 3, accounting for their respective additional constraints $t_3 \geq 1$ and $t_3 \leq 1$. This refinement and bound-tightening process yields:

Node 2:

$$\begin{aligned}
&\mathcal{P}_1^2 = \{[0,9]\} && \mathcal{Q}_1^2 = \{1\} && \\
&\mathcal{P}_2^2 = \{[0,9]\} && \mathcal{Q}_2^2 = \{1\} && \\
&\mathcal{P}_3^2 = \{[0,1],[1,9]\} && \mathcal{Q}_3^2 = \{1,2\} && y_3^1 = 0 \\
&\mathcal{P}_4^2 = \{[0,1],[1,2],[2,9]\} && \mathcal{Q}_4^2 = \{1,2,3\} && y_4^1 = 0 \\
&\mathcal{P}_5^2 = \{[0,2],[2,3],[3,9]\} && \mathcal{Q}_5^2 = \{1,2,3\} && y_5^1 = 0
\end{aligned}$$

$\mathcal{P}_T^2 = \{[0,3],[3,4],[4,9]\}$ $\mathcal{Q}_T^2 = \{1,2,3\}$ $y_T^1 = 0$

Node 3:

$\mathcal{P}_1^3 = \{[0,1],[1,9]\}$ $\mathcal{Q}_1^3 = \{1,2\}$ $y_1^2 = 0$

$\mathcal{P}_2^3 = \{[0,1],[1,9]\}$ $\mathcal{Q}_2^3 = \{1,2\}$ $y_2^2 = 0$

$\mathcal{P}_3^3 = \{[0,1],[1,9]\}$ $\mathcal{Q}_3^3 = \{1,2\}$ $y_3^2 = 0$

$\mathcal{P}_4^3 = \{[0,1],[1,2],[2,9]\}$ $\mathcal{Q}_4^3 = \{1,2,3\}$ $y_4^1 = 0$

$\mathcal{P}_5^3 = \{[0,2],[2,3],[3,9]\}$ $\mathcal{Q}_5^3 = \{1,2,3\}$ $y_5^1 = 0$

$\mathcal{P}_T^3 = \{[0,3],[3,4],[4,9]\}$ $\mathcal{Q}_T^3 = \{1,2,3\}$ $y_T^1 = 0.$

We fathom node 1 and nodes 2 and 3 inherit its cuts and lower bound. Breaking the tie arbitrarily, we select and then solve node 2 with Benders' decomposition and obtain an optimal value of $z^*_{\mathcal{P}} = 6.111$. Node 2 then branches to nodes 4 and 5, which inherit the cuts and the lower bound from node 2. The upper bound value remains $UB = 6.607$.

Node 3 now has the smallest lower bound, 4.072, among unfathomed nodes, and yields an optimal value of $z^*_{\mathcal{P}} = 6$ and optimal solution:

$$\hat{t}_1 = 0 \qquad \hat{x}_{11} = 1 \tag{34a}$$

$$\hat{t}_2 = 0.1 \qquad \hat{x}_{21} = \frac{1}{9} \tag{34b}$$

$$\hat{t}_4 = 2 \qquad \hat{x}_{41} = 0 \tag{34c}$$

$$\hat{t}_5 = 3 \qquad \hat{x}_{51} = \frac{8}{9} \tag{34d}$$

$$\hat{t}_T = 3.2. \tag{34e}$$

In the process of solving node 3, we also obtain a tighter global upper bound of $UB = 6$ associated with solution (34), and hence the optimality gap at node 3 is zero. Nodes 4 and 5 can now be fathomed because their lower bound of 6.111 exceeds $UB$. The algorithm terminates with the optimal solution in equation (34), and the corresponding $G$-variables indicate that activities 1, 2, and 3 start before disruption scenario 1, activity 4 starts before disruption scenario 2, and activity 5 starts before disruption scenario 3. The optimal objective function value is $6 = 0.6 \cdot [3.2] + 0.2 \cdot [2 + 11 \cdot (1 - 0.9 \cdot 8/9) + 11] + 0.2 \cdot [3 + 11 \cdot (1 - 0.9 \cdot 8/9)]$.

## 6 Experimental Results

In this section, we address the following questions with our computational results:

1. What is the value of model (2), which takes account of randomness in both the timing and magnitude of a disruption? In other words, how does the quality of the solution to model (2) compare to those of simpler alternatives?

2. How does the solution quality improve as the number of samples grows in a sample average approximation?

3. How do Algorithms 1 and 2 perform versus solving the extensive formulation (2) using a state-of-the-art MIP solver? How effective are the computational enhancements of upper-bound generation, Magnanti-Wong cuts, and the cut-selection procedure from Sections 5.4-5.6?

Section 6.1 introduces the PERT networks and probability distributions characterizing the disruptions for our test cases. In Section 6.2 we construct deterministic and semi-deterministic alternatives to model (2), perform out-of-sample tests, and compare the quality of the resulting solutions to those of model (2). We test how the sample size affects solution quality and requisite computational effort in Section 6.3. Finally, in Section 6.4 we compare the performance of Algorithms 1 and 2 to solving extensive formulation directly.

All tests are run on a server with 30 Intel Xeon cores at 3.1 GHz and 256 GB of RAM. For Algorithm 2, each node is solved by 6 cores and we allow at most 5 nodes to be solved simultaneously so that the maximum number of cores used at any time is again 30. All models are constructed using version 0.18.0 of the JuMP package Dunning et al. (2017) on the Julia platform. All linear programs and mixed-integer programs are solved by Gurobi 8.01 Gurobi Optimization, Inc. (2016) with the integer feasibility tolerance and the primal feasibility tolerance both set to $10^{-8}$. Within each node, Benders' and Magnanti-Wong cuts are added through Gurobi callbacks. In addition, we solve all problems using an optimality-gap tolerance of $10^{-2}$, including Algorithm 2's $\epsilon = 10^{-2}$. The cuts are generated with a tolerance parameter $\delta = 10^{-4}$; i.e., a cut is generated only when $\theta^{\omega,\ell} \leq v^{\omega,\ell} - \delta$. We do so to prevent numerical issues in which (nearly) identical cuts are repeatedly generated.

### 6.1 Test Cases Construction

We construct our test cases based on two activity networks from the literature, along with one we create and one we generate randomly. In particular, we use an activity network from Plambeck et al. Plambeck et al. (1996) with 11 activities, and one from Elmaghraby Elmaghraby (1977) with 19 activities. We also manually create one activity network with 14 activities and randomly generate one network with 35 activities using the tool *RanGen* Demeulemeester et al. (2003). In the following section, we use "Case $X$" to denote the test case with $X$ activities. Data for all four test cases are detailed in Appendix B.

For each test case, the timing of the disruption is a discrete random variable sampled from a lognormal distribution, which is commonly used to model failure times, e.g., Crow and Shimizu (1987), Mullen (1998). The magnitude of the disruption for each activity follows an exponential distribution (which we again sample) whose parameter varies among the activities; the exponential distribution is widely used to model activity durations such as service times, e.g., Ross (1996).

### 6.2 Value of a Fully Stochastic Model

We compare the quality of five solutions, one from solving model (2) and four from solving simpler alternatives, to investigate the value of modeling both the random timing and random magnitude of a disruption. First, we can obtain a solution by solving a deterministic model (1), assuming no disruption occurs, i.e., $d^\omega = 0$ (denoted "DET"). Three semi-stochastic alternative models can be solved assuming: both the timing and magnitude of the disruption are deterministic at their expected values (denoted "EXP"); the timing is random but magnitude is fixed at its expected value (denoted "$H$Only"); and, the magnitude is random but timing is fixed at its expected value (denoted "$d$Only"). Finally, we construct the fully stochastic model (2) in which both the timing and magnitude are random (denoted "FULL"). We sample 500 scenarios to solve $H$Only, $d$Only, and FULL. Twenty batches of samples of size 5,000 are used to estimate an upper bound for each candidate solution. The upper bound point estimate for those five candidate solutions is shown in Figure 7, and the 95% confidence interval is shown in Table 2. Figure 7 scales the optimal value

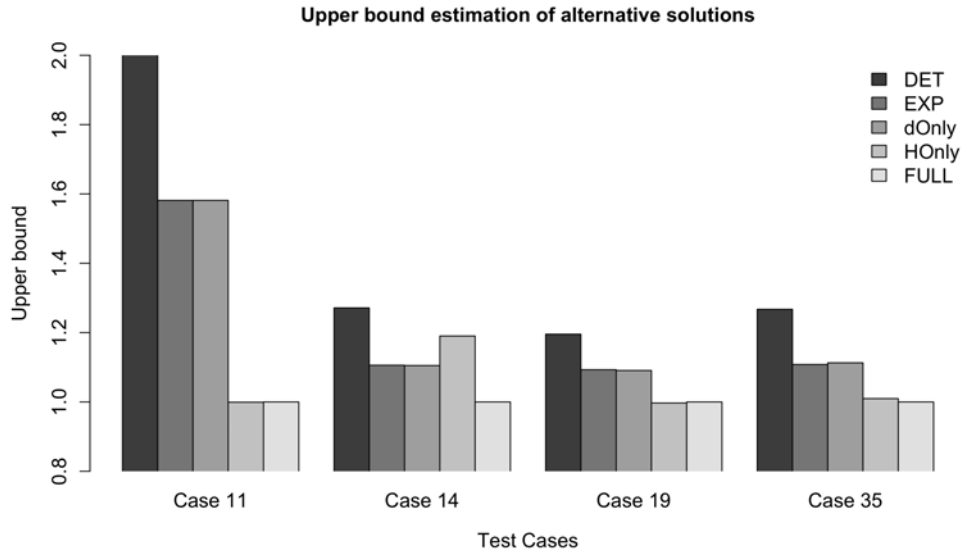



Figure 7: Comparison of quality of alternative solutions in which the optimal value of "FULL" is rescaled to 1.0.

of each test problem, dividing by that of FULL, and shows that the solution quality can be poor without considering the uncertainty in the timing of a disruption. DET's upper bound estimate is at least 20% larger than that of FULL for all test cases. The upper bound estimates of $d$Only and EXP are similar and at least 10% larger than that of FULL. For Case 14, the upper bound of $H$Only is 20% larger than that of FULL, and is closer to that of FULL for the other three cases. That said, given the sample sizes, the computational effort to solve $H$Only and FULL are comparable. These test results indicate that the fully stochastic model can outperform simpler

variants.

| | DET | EXP | $d$Only | $H$Only | FULL |
|---|---|---|---|---|---|
| Case 11 | 575.80 ± 16.88 | 454.33 ± 18.34 | 454.33 ± 18.34 | 287.05 ± 14.91 | 287.29 ± 14.92 |
| Case 14 | 3309.27 ± 177.18 | 2878.65 ± 176.70 | 2876.42 ± 176.55 | 3098.05 ± 133.06 | 2602.97 ± 110.12 |
| Case 19 | 426.09 ± 10.09 | 389.61 ± 7.88 | 388.77 ± 8.43 | 355.29 ± 7.14 | 356.42 ± 5.86 |
| Case 35 | 1353.80 ± 21.11 | 1183.74 ± 18.52 | 1188.93 ± 19.53 | 1078.57 ± 17.31 | 1068.29 ± 17.47 |

Table 2: Compare optimal values from alternatives of the disruption model.

### 6.3 Simulation Budget

We examine how the quality of a solution changes with sample size. An SAA approach can provide lower and upper bounds on $z^*$. And, these bounds converge to $z^*$ as the sample size grows to infinity, e.g., Shapiro et al. (2009). The computational effort of solving an SAA instance also grows with the sample size. We aim to determine a reasonable sample size, which yields a high-quality solution without excessive computational effort. We first examine trends of lower and upper bounds obtained from SAA as the sample size grows. For each test case, we first obtain 20 SAA solutions and corresponding lower bounds with sample sizes of 10, 20, 50, 100, 200, and 500. For each SAA solution, an upper bound estimator is evaluated using the same 5,000 samples. We present both lower- and upper-bound estimators and their 95% confidence intervals, for each sample size in Figure 8. From the figure we see the gap between the lower- and upper-bound estimators shrinks, and both estimators become less variable as the sample size grows. While upper bound estimators change little from 200 to 500 samples, lower-bound estimator still improve noticeably at these sample sizes.

We examine the gap between the lower- and upper-bound estimates further using the common random numbers procedure of Mak et al. (1999). Let $g_n(t,x)$ represent the objective function of an SAA instance of model (7), under sample size $n$, given a first-stage solution, $(t,x)$. From the tests that generate Figure 8, we have 20 candidate first-stage solutions for each sample size, among which we select the "best", denoted $(t^*,x^*)$, via the smallest value of $g_{5000}(t,x)$ using an independently generated set of 5000 samples, which are common across the 20 candidate solutions. Then we simulate another 20 instances with each sample size $n \in \{10,100,500,1000\}$. For the $i$-th instance, we obtain the lower bound estimate by solving $\min_{t,x}\ g_n^{(i)}(t,x)$—i.e., an SAA instance of model (7)—the upper bound estimate by evaluating $g_n^{(i)}(t^*,x^*)$, and their difference forms a point estimate of the optimality gap. We present the sample mean of the optimality gap and a 95% confidence interval on the gap in Table 3. Furthermore, we generate $n$ =100,000 samples, and estimate $z^*$ via a sample mean, $g_n(t^*,x^*)$, with this sample size, and the right-most column in the table reports the confidence interval as a percentage of this estimate. We observe from Table 3 that the optimality gap tends to shrink as the sample size grows, and relatively large sample sizes

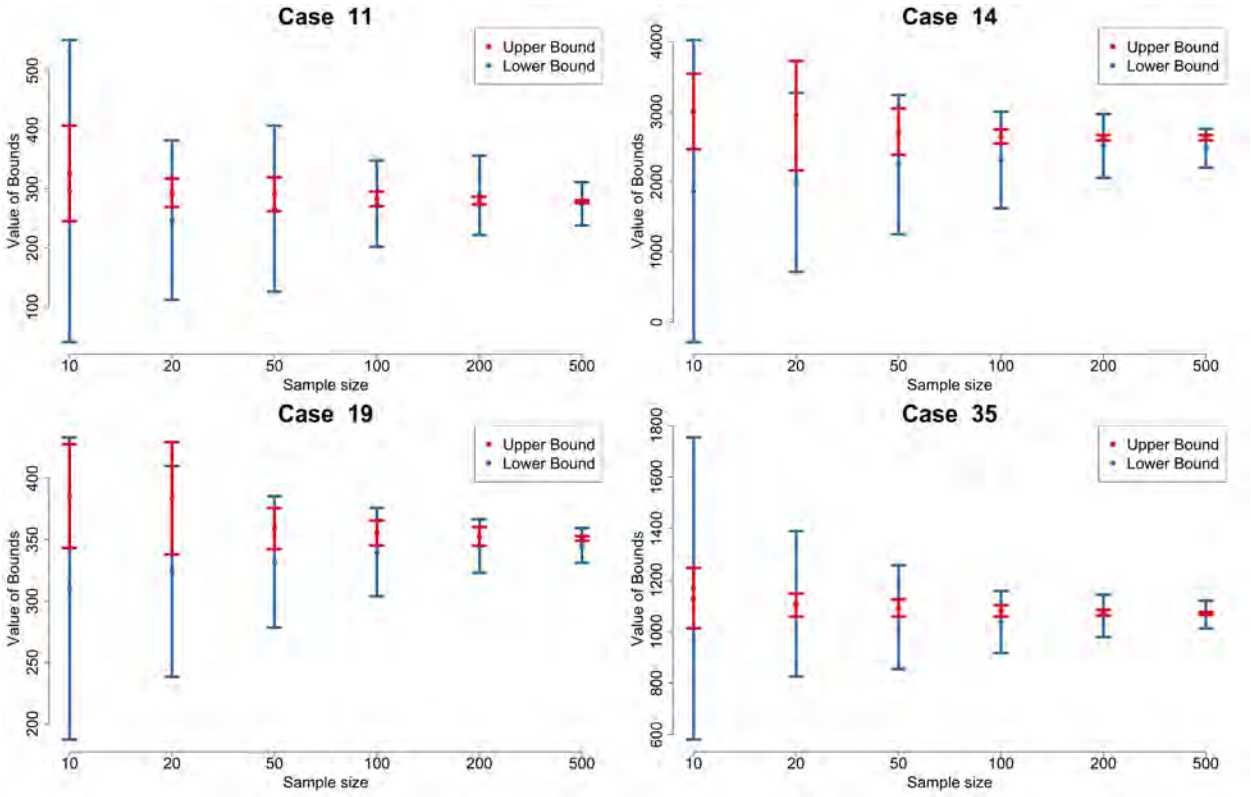



Figure 8: Confidence intervals, and point estimates, of the lower and upper bounds on $z^*$ for different sample sizes.

are needed for Cases 14 and 19.

### 6.4 Computational Performance

In this section we discuss the computational performance of our decomposition method with its various potential enhancements. As a benchmark, we compare the performance of Algorithms 1 and 2 to direct solution of the extensive formulation (2) using a commercial solver. We report the wall clock time as the running time in the results that follow.

We first briefly comment on running Algorithm 2, with and without FBBT from Section 5.3. Using a sample size of 500, we solve instances of Cases 11, 14, 19, and 35. In so doing, we use all improvements described in Sections 5.4-5.6. FBBT can significantly improve the value of the LP relaxation at the B&B tree's root node, and this sometimes leads to modest improvements in the total number of nodes that are explored. That said, the differences in overall run-times, with and without FBBT, are mixed and not particularly large, and hence we do not present these results in detail. We do employ FBBT in the remainder of this section.

Next, we show the quality of the heuristic upper bound of Section 5.4. We solve SAA instances with sample sizes of 100, 200, and 500, and in the notation of Section 5.4 we use $N = 10$ in the first two instances and $N = 20$ in the third instance. We compare the heuristic upper bound to

| | Sample size | Optimality gap mean | 95% CI on gap | Percentage of $z^*$ (%) |
|---|---|---|---|---|
| Case 11 | 10 | 3.7 | [0,4.8] | 1.7 |
| | 100 | 4.1 | [0,6.1] | 2.1 |
| | 500 | 1.1 | [0,1.5] | 0.5 |
| | 1000 | 1.3 | [0,1.7] | 0.6 |
| Case 14 | 10 | 796.5 | [0,1042.7] | 39.9 |
| | 100 | 264.3 | [0,317.1] | 12.2 |
| | 500 | 127.5 | [0,155.5] | 6.1 |
| | 1000 | 80.2 | [0,99.5] | 3.9 |
| Case 19 | 10 | 40.5 | [0,49.4] | 13.9 |
| | 100 | 15.2 | [0,17.3] | 4.9 |
| | 500 | 7.1 | [0,8.1] | 2.3 |
| | 1000 | 5.9 | [0,6.7] | 1.9 |
| Case 35 | 10 | 15.2 | [0,26.4] | 2.5 |
| | 100 | 9.0 | [0,10.9] | 1.0 |
| | 500 | 5.1 | [0,6.0] | 0.6 |
| | 1000 | 6.1 | [0,6.8] | 0.6 |

Table 3: Mean, 95% confidence interval (CI) on optimality gap, and the value of the confidence interval width relative to an estimate of $z^*$ based on evaluating the objective function at $(t^*, x^*)$ using 100,000 samples.

the optimal value of the SAA instance. Moreover, for each case/sample-size pair, we replicate the procedure on 20 independent instances. Table 4 shows the smallest gap, the average gap, and the largest gap, all as percentages, between the heuristic upper bound and the SAA optimal value across the 20 replications. The table also shows the average time required to compute the heuristic upper bound. We see that the average upper bound gap exceeds 5% only for Case 14 when the

| | Sample size | Gap (%) | | | Average |
|---|---|---|---|---|---|
| | | Smallest | Average | Largest | time (sec) |
| Case 11 | 100 | 0.01 | 0.5 | 2.3 | 5.6 |
| | 200 | 0 | 0.15 | 0.83 | 17.8 |
| | 500 | 0 | 0.07 | 0.37 | 67.5 |
| Case 14 | 100 | 1.34 | 6.02 | 16.14 | 7.7 |
| | 200 | 1.58 | 3.9 | 8.91 | 30.4 |
| | 500 | 0.75 | 2.35 | 3.96 | 104.4 |
| Case 19 | 100 | 0.56 | 2.67 | 5.28 | 9.9 |
| | 200 | 0.05 | 0.99 | 2.03 | 37.2 |
| | 500 | 0 | 0.72 | 1.59 | 127.6 |
| Case 35 | 100 | 0 | 0.64 | 2.65 | 5.8 |
| | 200 | 0.08 | 0.46 | 1.3 | 25.7 |
| | 500 | 0 | 0.32 | 1.25 | 59.0 |

Table 4: Gap information between the heuristic upper bound and optimal value for twenty random samples.

sample size is 100. Among all 240 SAA instances, there are only 15 for which the heuristic upper bound exceeds the optimal value by more than 5%. We also observe that as the sample size grows, the heuristic upper bound's quality improves.

Table 5 shows run-time results for finding a heuristic upper bound and generating Magnanti-Wong cuts while using Algorithm 2 with cut selection. Here we solve 20 replications for the sample size of 500, assuming $N = 20$ in the notation of Section 5.4, and we obtain the mean running time. In the table, "UB" means that we only use the heuristic upper bound of Section 5.4 and generate regular Benders' cuts, and "MW" means that we generate Magnanti-Wong cuts of Section 5.5 without the benefit of the upper bound heuristic. We also show the computation time of using neither and using both techniques. Table 5 shows that obtaining a heuristic upper bound can

| Running Time (sec.) | Case 11 | Case 14 | Case 19 | Case 35 |
|---|---|---|---|---|
| Neither | 73.0 | 682.3 | 1466.9 | 126.7 |
| UB | 55.4 | 677.9 | 1379.1 | 47.1 |
| MW | 122.9 | 1321.6 | 2065.1 | 207.1 |
| Both | 67.8 | 647.7 | 860.8 | 71.1 |

Table 5: Run-time results for using the heuristic upper bound and Magnanti-Wong cuts in Algorithm 2 with cut selection.

reduce computational effort; this occurs because the bound facilitates earlier fathoming of some nodes in the branch-and-bound tree when solving the master MIP. This outweighs the time required to compute the upper bound; see Table 4. Magnanti-Wong cuts are effective for the harder cases Case 14 and Case 19 when applied together with a heuristic upper bound.

Table 6 compares the run time of Algorithm 1 (denoted A1), Algorithm 2 (denoted A2), with and without the cut selection procedure (CS). Here we use all improvements described in Section 5.3-5.5. The table also shows the run-time for solving the extensive formulation (2) using Gurobi, again to a relative optimality tolerance of 0.01. Due to the computational effort required to execute some of the less efficient algorithms on large problem instances, we report results for a single replication. Table 6 shows that although directly solving the extensive formulation may be faster when the sample size is small, our decomposition algorithms tend to perform better as the sample size grows. The improved computational performance occurs because, while our decomposition methods must solve the master problem multiple times, the number of binary variables is significantly smaller than that of the extensive formulation. We compare the computational progress of Algorithm 2 versus directly solving the extensive formulation in Figure 9. Algorithm 2 finds a good lower bound more quickly because its form of branching can determine the value of many binary variables such that the linear relaxation of the subproblems is relatively tight. From Table 4 we know the heuristic upper bound is relatively tight, and so good lower bounds allow Algorithm 2 to terminate after branching only twice in this example (Figure 9b).

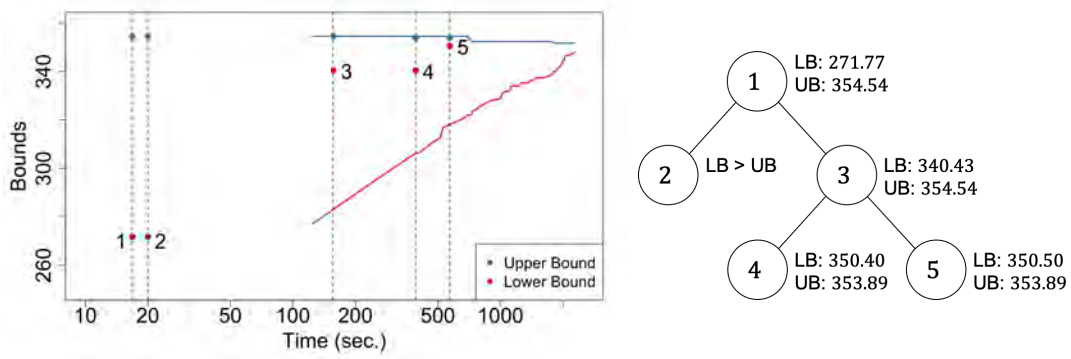
| Running Time (sec.) | Sample Size | Methods | | | | |
|---|---|---|---|---|---|---|
| | | A1 | A1+CS | A2 | A2+CS | Extensive |
| Case 11 | 100 | 8.7 | 7.4 | 20.0 | 10.0 | 5.0 |
| | 200 | 16.7 | 34.8 | 20.0 | 20.0 | 40.3 |
| | 500 | 273.3 | 240.9 | 103.4 | 114.1 | 1597.1 |
| | 1000 | 279.9 | 326.7 | 73.0 | 80.1 | 5019.0 |
| Case 14 | 100 | 45.2 | 31.0 | 43.7 | 40.0 | 13.7 |
| | 200 | 137.5 | 126.0 | 150.2 | 101.0 | 158.7 |
| | 500 | 1710.7 | 1578.0 | 1252.0 | 795.6 | 1263.1 |
| | 1000 | 6284.1 | 7416.6 | 3135.1 | 1684.2 | 9808.3 |
| Case 19 | 100 | 201.2 | 107.4 | 80.4 | 60.1 | 58.4 |
| | 200 | 239.9 | 194.2 | 400.6 | 187.1 | 279.8 |
| | 500 | 1643.7 | 1421.0 | 747.2 | 607.5 | 2792.9 |
| | 1000 | 11259.1 | 8824.9 | 9785.2 | 2562.8 | 33698.5 |
| Case 35 | 100 | 33.4 | 31.7 | 40.1 | 30.0 | 7.0 |
| | 200 | 14.1 | 13.6 | 10.0 | 10.0 | 35.9 |
| | 500 | 66.1 | 66.2 | 20.0 | 20.0 | 358.6 |
| | 1000 | 241.0 | 248.0 | 40.1 | 40.1 | 662.0 |

Table 6: Computational performance with different sample sizes for decomposition methods and directly solving the extensive formulation using Gurobi.

Algorithm 2 outperforms Algorithm 1 when the sample size exceeds 500. When the sample size is large, Algorithm 1 requires many master iterations to converge, and solving the mixed-integer master program becomes significantly harder. For Algorithm 2, the time saved by processing nodes in parallel outweighs the slightly longer solution time at each node due to using fewer cores. Moreover, because of branching, each master problem at a node in the B&B tree has a small number of binary variables in Algorithm 2, which further accelerates the B&C algorithm. The results also show that selecting cuts can significantly improve the decomposition methods. On the most difficult instances, applying the cut selection scheme yields larger improvements for Algorithm 2 than for Algorithm 1. On a different note, our implementation of the method of Qi and Sen Qi and Sen (2017) does not solve any of our cases more quickly than directly solving the extensive formulation. We also attempted to solve our instances using Benders' decomposition in which we move all binary variables to the first stage and generate cuts using Gurobi's callback feature. This method is also inferior to directly solving the extensive formulation and was unable to achieve the optimality-gap tolerance within a reasonable time when the sample size exceeds 500.

## 7 Conclusions

In this paper, we introduce the concept of a stochastic disruption in the context of a project crashing problem. We consider the case of a single disruption, and formulate the model as a two-stage stochastic mixed-integer program in which the timing of the stage, i.e., the disruption time, is ran-




(a) (b)

Figure 9: Illustration of Case 19 with 500 scenarios: (a) computational progress of Algorithm 2 versus directly solving the extensive formulation with Gurobi; note the log scale on the $x$-axis. Dots represent the upper and lower bounds after solving the correspondingly indexed node in the branch-and-bound tree of Algorithm 2 and the curves represent the upper and lower bounds of directly solving the extensive formulation using Gurobi. (b) branch-and-bound tree of Algorithm 2, LB and UB represent the lower bound and the upper bound obtained by solving the two-stage relaxed problem at each node. Nodes 2, 4 and 5 are fathomed because their local lower bounds either exceed or lie within the tolerance of the best identified upper bound. The lower bounds in (a) are the best available when each node is solved.

dom. We use examples to illustrate properties of our problem that deviate from its deterministic counterpart, including the fact that it can be optimal to delay the start of an activity or crash a shorter-duration activity, even under proportional reduction. While, conceptually, the underlying problem involves continuous decision variables, we argue that the problem is NP-hard. In our two-stage stochastic mixed-integer program, second-stage binary variables capture the logic of the start time of an activity, relative to the disruption time. The resulting model is computationally challenging, but we propose a decomposition method which exploits the logical temporal relationship just mentioned, and sequentially partitions the feasible region of continuous first-stage decision variables to generate tighter cuts in a Benders' decomposition algorithm. The proposed method can significantly improve computational performance, especially as sample sizes grow large.

The temporal aspect of a stochastic disruption is valuable in modeling contingencies and recovery plans in infrastructure systems Yang and Nagarajan (2020), Yuan et al. (2016), and our ideas may help advance future work for handling such disruptions in continuous time. Our decomposition algorithm can be easily adapted to other scheduling problems under stochastic disruptions. For example, in a spatial batch scheduling problem in manufacturing Srinivasan et al. (2015), we need to decide the location and the start time of each job in continuous space and time, which induces precedence relations for scheduling other jobs. In a model variant with a stochastic disruption, the durations of processing times change after the disruption, and a delayed job will block other jobs from being processed in its space, causing cascading effects. Our model can also be viewed

as handling a type of decision-dependent uncertainty Hellemo et al. (2018) in that investments in crashing reduce uncertainty in an activity's duration. So, our branch-and-cut algorithm may prove useful in other such models in which the value of a stochastic parameter relies on a logical condition between first-stage decision variables and other independent stochastic parameters, like the time of disruption.

The ideas in this paper can be extended in further ways. There may be opportunities to exploit network structure in tailoring the branch-and-cut algorithm that we have developed. The distribution governing the disruption time and, conditional on that time, the magnitude of the disruption may facilitate sampling strategies that reduce variance and improve solution quality. We have considered a model and algorithm that allow for at most one disruption, but handling a small number of disruptions could be attractive.

## References


A. Aghaie and H. Mokhtari. Ant colony optimization algorithm for stochastic project crashing problem in PERT networks using MC simulation. *The International Journal of Advanced Manufacturing Technology*, 45(11):1051–1067, 2009.

S. D. Ahipasaoglu, K. Natarajan, and D. Shi. Distributionally robust project crashing with partial or no correlation information. *Optimization-Online*, 2016. URL `http://www.optimization-online.org/DB_FILE/2016/11/5715.pdf`.

P. Belotti, S. Cafieri, J. Lee, and L. Liberti. On feasibility based bounds tightening. *Optimization-Online*, 2012. URL `http://www.optimization-online.org/DB_FILE/2012/01/3325.pdf`.

J. R. Birge and F. V. Louveaux. A multicut algorithm for two-stage stochastic linear programs. *European Journal of Operational Research*, 34:384–392, 1988.

J. Blazewicz, J. K. Lenstra, and A. H. G. R. Kan. Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics*, 5(1):11–24, 1983.

R. A. Bowman. Stochastic gradient-based time-cost tradeoffs in PERT networks using simulation. *Annals of Operations Research*, 53(1):533–551, 1994.

J. M. Burt and M. B. Garman. Conditional Monte Carlo: A simulation technique for stochastic network analysis. *Management Science*, 18(3):207–217, 1971.

J. D. Camm, A. S. Raturi, and S. Tsubakitani. Cutting big M down to size. *Interfaces*, 20(5): 61–66, 1990.

B. Cardoen, E. Demeulemeester, and J. Beliën. Operating room planning and scheduling: A literature review. *European Journal of Operational Research*, 201(3):921–932, 2010.

C. Carøe and J. Tind. L-shaped decomposition of two-stage stochastic programs with integer recourse. *Mathematical Programming*, 83(1-3):451–464, 1998.

B. Chen, S. Küçükyavuz, and S. Sen. A computational study of the cutting plane tree algorithm for general mixed-integer linear programs. *Operations Research Letters*, 40(1):15–19, 2012.

X. Chen, M. Sim, P. Sun, and J. Zhang. A linear decision-based approximation approach to stochastic programming. *Operations Research*, 56(2):344–357, 2008.

C. Coffrin, H. L. Hijazi, and P. Van Hentenryck. Strengthening convex relaxations with bound tightening for power network optimization. In *International Conference on Principles and Practice of Constraint Programming*, pages 39–57. Springer, 2015.

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

E. L. Crow and K. Shimizu. *Lognormal distributions: theory and applications*. CRC Press, 1987.

P. De, E. J. Dunne, J. B. Ghosh, and C. E. Wells. Complexity of the discrete time-cost tradeoff problem for project networks. *Operations Research*, 45(2):302–306, 1997.

E. Demeulemeester and W. S. Herroelen. *Project scheduling: a research handbook*, volume 49. Springer Science & Business Media, 2006.

E. Demeulemeester, M. Vanhoucke, and W. Herroelen. RanGen: A random network generator for activity-on-the-node networks. *Journal of scheduling*, 6(1):17–38, 2003.

I. Dunning, J. Huchette, and M. Lubin. JuMP: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320, 2017.

S. E. Elmaghraby. *Activity networks: project planning and control by network models*. Wiley, 1977.

D. R. Fulkerson. A network flow computation for project cost curves. *Management Science*, 7(2): 167–178, 1961.

D. Gade, S. Küçükyavuz, and S. Sen. Decomposition algorithms with parametric Gomory cuts for two-stage stochastic integer programs. *Mathematical Programming*, 144(1-2):39–64, 2014.

M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979.

Gurobi Optimization, Inc. *Gurobi Optimizer Reference Manual*, 2016. URL `http://www.gurobi.com`.

N. G. Hall and C. Sriskandarajah. A survey of machine scheduling problems with blocking and no-wait in process. *Operations Research*, 44(3):510–525, 1996.

S. Hartmann and R. Kolisch. Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 127(2): 394–407, 2000.

L. Hellemo, P. I. Barton, and A. Tomasgard. Decision-dependent probabilities in stochastic programs with recourse. *Computational Management Science*, 15(3-4):369–395, 2018.

E. J. Jaselskis and D. B. Ashley. Optimal allocation of project management resources for achieving success. *Journal of Construction Engineering and Management*, 117(2):321–340, 1991.

H. Ke. A genetic algorithm-based optimizing approach for project time-cost trade-off with uncertain measure. *Journal of Uncertainty Analysis and Applications*, 2(1):8, 2014.

J. E. Kelly. Critical-path planning and scheduling: Mathematical basis. *Operations Research*, 9(3): 296–320, 1961.

S. Kim, S. P. Boyd, S. Yun, D. D. Patil, and M. A. Horowitz. A heuristic for optimizing stochastic activity networks with applications to statistical digital circuit sizing. *Optimization and Engineering*, 8(4):397–430, 2007.

S. Kim, R. Pasupathy, and S. G. Henderson. A guide to sample average approximation. In *Handbook of Simulation Optimization*, pages 207–243. Springer, 2015.

E. Klotz and A. M. Newman. Practical guidelines for solving difficult mixed integer linear programs. *Surveys in Operations Research and Management Science*, 18(1-2):18–32, 2013.

M. E. Kuhl and R. A. Tolentino-Peña. A dynamic crashing method for project management using simulation-based optimization. In *Proceedings of the 40th Conference on Winter Simulation*, pages 2370–2376. Winter Simulation Conference, 2008.

P. Lamas and E. Demeulemeester. A purely proactive scheduling procedure for the resource-constrained project scheduling problem with stochastic activity durations. *Journal of Scheduling*, 19(4):409–428, 2016.

Z. Li and M. Ierapetritou. Process scheduling under uncertainty: Review and challenges. *Computers & Chemical Engineering*, 32(4-5):715–727, 2008.

T. L. Magnanti and R. T. Wong. Accelerating Benders decomposition: Algorithmic enhancement and model selection criteria. *Operations research*, 29(3):464–484, 1981.

W. K. Mak, D. P. Morton, and R. K. Wood. Monte Carlo bounding techniques for determining solution quality in stochastic programs. *Operations Research Letters*, 24(1):47–56, 1999.

D. G. Malcolm, J. H. Roseboom, C. E. Clark, and W. Fazar. Application of a technique for research and development program evaluation. *Operations Research*, 7(5):646–669, 1959.

R. H. Möhring and F. Stork. Linear preselective policies for stochastic project scheduling. *Mathematical Methods of Operations Research*, 52(3):501–515, 2000.

R. E. Mullen. The lognormal distribution of software failure rates: origin and evidence. In *Proceedings Ninth International Symposium on Software Reliability Engineering (Cat. No.98TB100257)*, pages 124–133, Nov 1998.

B. Naderi, M. Zandieh, and S. M. T. Fatemi Ghomi. Scheduling job shop problems with sequence-dependent setup times. *International Journal of Production Research*, 47(21):5959–5976, 2009.

G. D. Oberlender. *Project management for engineering and construction*, volume 2. McGraw-Hill New York, 1993.

A. B. Philpott, F. Wahid, and J. F. Bonnans. MIDAS: A mixed integer dynamic approximation scheme. *Mathematical Programming*, 181(1):19–50, 2020.

E. L. Plambeck, B. Fu, S. M. Robinson, and R. Suri. Sample-path optimization of convex stochastic performance functions. *Mathematical Programming*, 75(2):137–176, 1996.

Project Management Institute. *A guide to the project management body of knowledge: PMBOK guide*. Project Management Institute, Newtown Square, Pennsylvania, 2017.

Y. Qi and S. Sen. The ancestral Benders cutting plane algorithm with multi-term disjunctions for mixed-integer recourse decisions in stochastic programming. *Mathematical Programming*, 161 (1-2):193–235, 2017.

S. M. Ross. *Stochastic processes*, volume 2. Wiley New York, 1996.

J. Salmerón, R. K. Wood, and D. P. Morton. A stochastic program for optimizing military sealift subject to attack. *Military Operations Research*, 14(2):19–39, 2009.

A. Shapiro, D. Dentcheva, and A. Ruszczyński. *Lectures on stochastic programming: modeling and theory*. SIAM, 2009.

J. Söderlund. Building theories of project management: past research, questions for the future. *International Journal of Project Management*, 22(3):183–191, 2004.

S. Srinivasan, J. P. Brooks, and J. H. Wilson. *Batching-based approaches for optimized packing of jobs in the spatial scheduling problem*, pages 243–263. Springer International Publishing, Cham, 2015.

K. Sundar, H. Nagarajan, S. Misra, M. Lu, C. Coffrin, and R Bent. Optimization-based bound tightening using a strengthened QC-relaxation of the optimal power flow problem. *arXiv preprint arXiv:1809.04565*, 2018.

S. Tonchia. *Industrial project management*. Springer, 2018.

R. M. van Slyke. Letter to the editor—Monte Carlo methods and the PERT problem. *Operations Research*, 11(5):839–860, 1963.

W. Wiesemann, D. Kuhn, and B. Rustem. Robust resource allocations in temporal networks. *Mathematical Programming*, 135(1):437–471, 2012.

H. Yang and H. Nagarajan. Optimal power flow in distribution networks under stochastic N-1 disruptions. *Electric Power Systems Research*, 189:106689, 2020.

G. Yu and X. Qi. *Disruption Management: Framework, Models and Applications*. World Scientific, 2004.

W. Yuan, J. Wang, F. Qiu, C. Chen, C. Kang, and B. Zeng. Robust optimization-based resilient distribution network planning against natural disasters. *IEEE Transactions on Smart Grid*, 7(6):2817–2826, 2016.

J. Zou, S. Ahmed, and X. A. Sun. Stochastic dual dynamic integer programming. *Mathematical Programming*, 175(1-2):461–502, 2019.

## A Model when Disruption Affects Activities Yet to End

Model (2) assumes that activities that have started are not affected by the disruption, even if they have yet to end. That is, the disruption only affects activities that have yet to start. Here, we formulate a model under the complementary assumption that the disruption affects activities that have yet to end. To do so, we model the end time of activities as the decision variables denoted by $t$ and $t^\omega$. In addition variable $G_i^\omega$ indicates whether activity $i$ finishes after the disruption in scenario $\omega$. In this situation, we further assume that the crashing decisions can be adjusted before the activity ends. To obtain comparable computational results, we assume that durations $D_i$ and $d_i^\omega$ only depend on the activity, and not the precedence relationship, which is the same assumption made in the computational experiments in the main text. For simplicity we formulate the model under this assumption.

$$z^* = \min \quad p^0 t_T + \sum_{\omega\in\Omega} p^\omega t_T^\omega \tag{35a}$$

$$\text{s.t.} \quad t_k - t_i \geq D_k \left(1 - \sum_{j\in J_k} e_{kj} x_{kj}\right) \qquad \forall\,(i,k)\in\mathcal{A} \tag{35b}$$

$$\sum_{i\in I}\sum_{j\in J_i} b_{ij} x_{ij} \leq B \tag{35c}$$

$$\sum_{j\in J_i} x_{ij} \leq 1 \qquad \forall\, i\in I \tag{35d}$$

$$H^\omega + M G_i^\omega \geq t_i \qquad \forall\, i\in I, \omega\in\Omega \tag{35e}$$

$$H^\omega - M(1 - G_i^\omega) \leq t_i \qquad \forall\, i\in I, \omega\in\Omega \tag{35f}$$

$$t_i^\omega + M' G_i^\omega \geq t_i \qquad \forall\, i\in I, \omega\in\Omega \tag{35g}$$

$$t_i^\omega - M' G_i^\omega \leq t_i \qquad \forall\, i\in I, \omega\in\Omega \tag{35h}$$

$$x_{ij}^\omega + \bar{x}_{ij} G_i^\omega \geq x_{ij} \qquad \forall\, i\in I, j\in J_i, \omega\in\Omega \tag{35i}$$

$$x_{ij}^\omega - \bar{x}_{ij} G_i^\omega \leq x_{ij} \qquad \forall\, i\in I, j\in J_i, \omega\in\Omega \tag{35j}$$

$$t_k^\omega - t_i^\omega \geq D_k + d_k^\omega G_k^\omega - \sum_{j\in J_k} D_k e_{kj} x_{kj}^\omega - \sum_{j\in J_k} d_k^\omega e_{kj} z_{kj}^\omega \qquad \forall\,(i,k)\in\mathcal{A}, \omega\in\Omega \tag{35k}$$

$$\sum_{i\in I}\sum_{j\in J_i} b_{ij} x_{ij}^\omega \leq B \qquad \forall\,\omega\in\Omega \tag{35l}$$

$$\sum_{j\in J_i} x_{ij}^\omega \leq 1 \qquad \forall\, i\in I, \omega\in\Omega \tag{35m}$$

$$z_{ij}^\omega \leq \bar{x}_{ij} G_i^\omega \qquad \forall\, i\in I, j\in J_i, \omega\in\Omega \tag{35n}$$

$$z_{ij}^\omega \leq x_{ij}^\omega \qquad \forall\, i\in I, j\in J_i, \omega\in\Omega \tag{35o}$$

$$z_{ij}^{\omega} \geq x_{ij}^{\omega} + \bar{x}_{ij}(G_i^{\omega} - 1) \qquad \forall\, i \in I, j \in J_i, \omega \in \Omega \qquad (35p)$$

$$t_i \geq 0 \qquad \forall\, i \in I \qquad (35q)$$

$$t_i^{\omega} \geq H^{\omega} G_i^{\omega} \qquad \forall\, i \in I, \omega \in \Omega \qquad (35r)$$

$$0 \leq x_{ij} \leq \bar{x}_{ij} \qquad \forall\, i \in I, j \in J_i \qquad (35s)$$

$$0 \leq x_{ij}^{\omega} \leq \bar{x}_{ij} \qquad \forall\, i \in I, j \in J_i, \omega \in \Omega \qquad (35t)$$

$$0 \leq z_{ij}^{\omega} \leq 1 \qquad \forall\, i \in I, j \in J_i, \omega \in \Omega \qquad (35u)$$

$$G_i^{\omega} \in \{0, 1\}. \qquad \forall\, i \in I, \omega \in \Omega. \qquad (35v)$$

The bulk of the model mirrors that of model (2), and so we do not repeat the corresponding narrative. Constraints (35b) and (35k) differ due to the altered definition of the end-time decision variables: the duration between the end time of activity $i$ and $k$, if $i$ precedes $k$, is the duration of activity $k$, instead of that of activity $i$ in model (2). Constraint (35r) ensures that if the disruption occurs during an activity, the crashing decision cannot cause the activity to end before the disruption.

Next, we present the effect of this altered assumption on the optimal values, i.e., the expected completion times of the projects, in the first two rows of Table 7 using $|\Omega| = 500$. We also test how well the optimal solution to model (2) performs in model (35), and the optimality gap ranges from 5% to 20%; see the final row of the table. This suggests the importance of selecting the most realistic assumption when formulating the model. The decomposition algorithm can still be used to solve model (35) with necessary, but minimal, changes in the recourse problem formulation. We observe that changing the assumption does not significantly change the requisite computational effort.

| Optimal value | Case 11 | Case 14 | Case 19 | Case 35 |
|---|---|---|---|---|
| Model (2) | 229.8 | 2384.7 | 351.6 | 1043.3 |
| Model (35) | 402.0 | 12015.3 | 703.3 | 1431.1 |
| (2) $\rightarrow$ (35) | 481.6 | 13527.5 | 823.6 | 1498.2 |

Table 7: The first two numerical rows show the optimal values of models (2) and (35), with sample size 500. The final row shows the suboptimal objective function value obtained by using the optimal crashing solution of model (2) in model (35).

While we do not explicitly write the model here, it is possible to partition activities into two groups that correspond to the respective assumptions of models (2) and (35) regarding whether in-process activities are affected by a disruption. This can be done by defining both start and end time variables for each activity in the same model, and enforcing the nonanticipativity constraints differently for the two types of activities.

## B Test Cases Data

We present the data of four test cases here. For all test cases we assume there is only one possible crashing option for each activity. The option consumes 1 unit of resource and has the effectiveness parameter of $e_{i1} = 0.5$ for all $i \in I$. The nominal scenario probability is $p^0 = 0.1$ and $p^\omega = \frac{1-p^0}{|\Omega|}$. The timing of the disruption follows a lognormal distribution with parameters $\mu$ and $\sigma$ where the mode is $e^{\mu-\sigma^2}$. We also assume that the duration only depends on the predecessor, i.e., $D_{ik} = D_i$ and $d^\omega_{ik} = d^\omega_i$. All $d_i$ follow an exponential distribution with a mean of $\mu_i$. The value of $D_i$ and $\mu_i$ are shown in the following tables.

- Case 11: $B = 3, \mu = \ln 6, \sigma = 0.5$

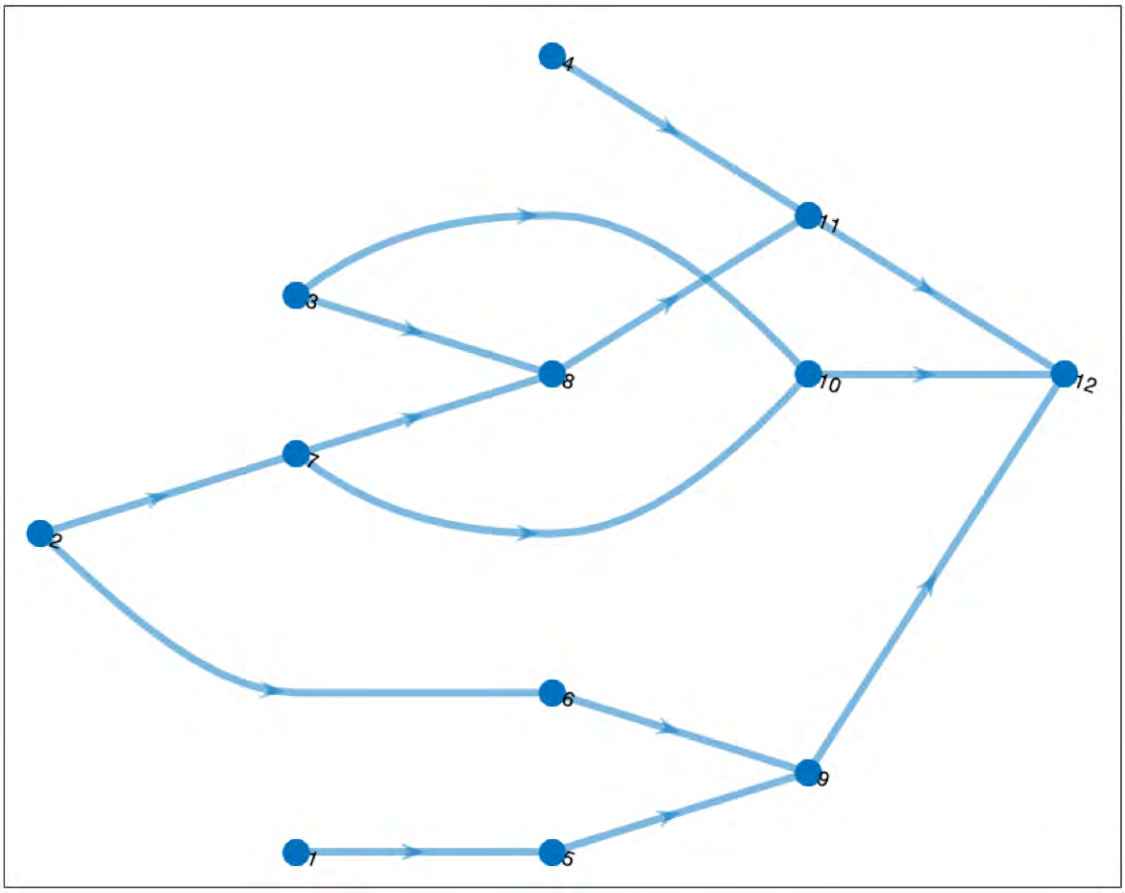



Figure 10: Activity network of Case 11

| Activity | $D_i$ | $\mu_i$ |
|---|---|---|
| 1 | 10 | $10^{-5}$ |
| 2 | 2 | 4 |
| 3 | 10 | 2 |
| 4 | 12 | 30 |
| 5 | 3 | 1500 |
| 6 | 10 | 1 |

| Activity | $D_i$ | $\mu_i$ |
|---|---|---|
| 7 | 7.3 | 1 |
| 8 | 4.9 | 50 |
| 9 | 11.1 | 40 |
| 10 | 3.5 | 40 |
| 11 | 9.9 | 5 |

Table 8: Activity duration $D_i$ and the mean of disruption magnitude $\mu_i$ for Case 11

- Case 14: $B = 4, \mu = \ln 35, \sigma = 0.5$

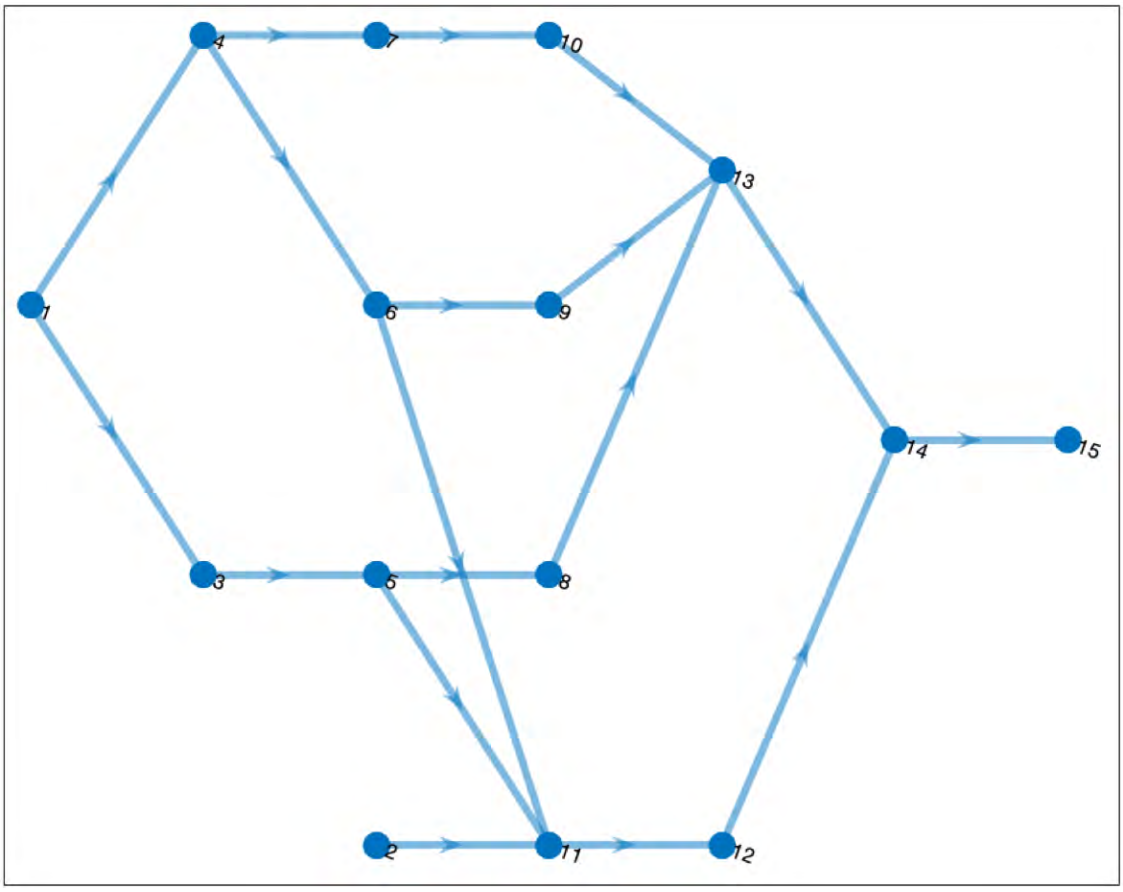



Figure 11: Activity network of Case 14

| Activity | $D_i$ | $\mu_i$ |
|---|---|---|
| 1 | 5 | $10^{-5}$ |
| 2 | 30 | 5 |
| 3 | 25 | 40000 |
| 4 | 20 | 40000 |
| 5 | 15 | 1500 |
| 6 | 24 | 20000 |
| 7 | 30 | 20000 |

| Activity | $D_i$ | $\mu_i$ |
|---|---|---|
| 8 | 49 | 4000 |
| 9 | 40 | 4000 |
| 10 | 30 | 3000 |
| 11 | 45 | 4000 |
| 12 | 25 | 5 |
| 13 | 21 | 5 |
| 14 | 5 | 5 |

Table 9: Activity duration $D_i$ and the mean of disruption magnitude $\mu_i$ for Case 14

- Case 19: $B = 4, \mu = \ln 8, \sigma = 0.5$

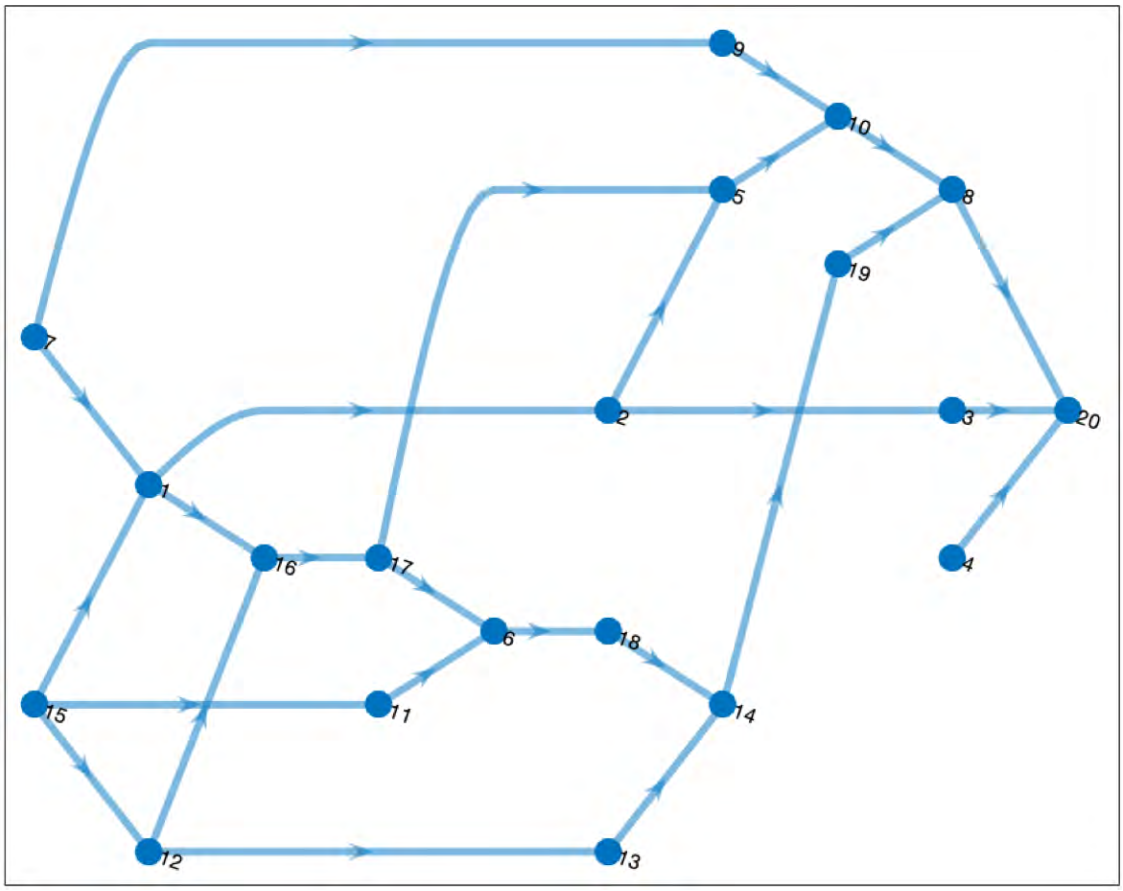



Figure 12: Activity network of Case 19

| Activity | $D_i$ | $\mu_i$ |
|---|---|---|
| 1 | 5 | 10 |
| 2 | 10 | 100 |
| 3 | 18 | 300 |
| 4 | 35 | 300 |
| 5 | 6 | 50 |
| 6 | 6 | 50 |
| 7 | 2.5 | 400 |
| 8 | 8 | 20 |
| 9 | 10 | 1000 |
| 10 | 1 | 50 |

| Activity | $D_i$ | $\mu_i$ |
|---|---|---|
| 11 | 6 | 400 |
| 12 | 4.5 | 300 |
| 13 | 12 | 1000 |
| 14 | 1.5 | 10 |
| 15 | 2.1 | 200 |
| 16 | 8.1 | 2 |
| 17 | 6.1 | 100 |
| 18 | 0.001 | 40 |
| 19 | 0.001 | 300 |

Table 10: Activity duration $D_i$ and the mean of disruption magnitude $\mu_i$ for Case 19

- Case 35: $B = 8, \mu = \ln 4, \sigma = 0.3$

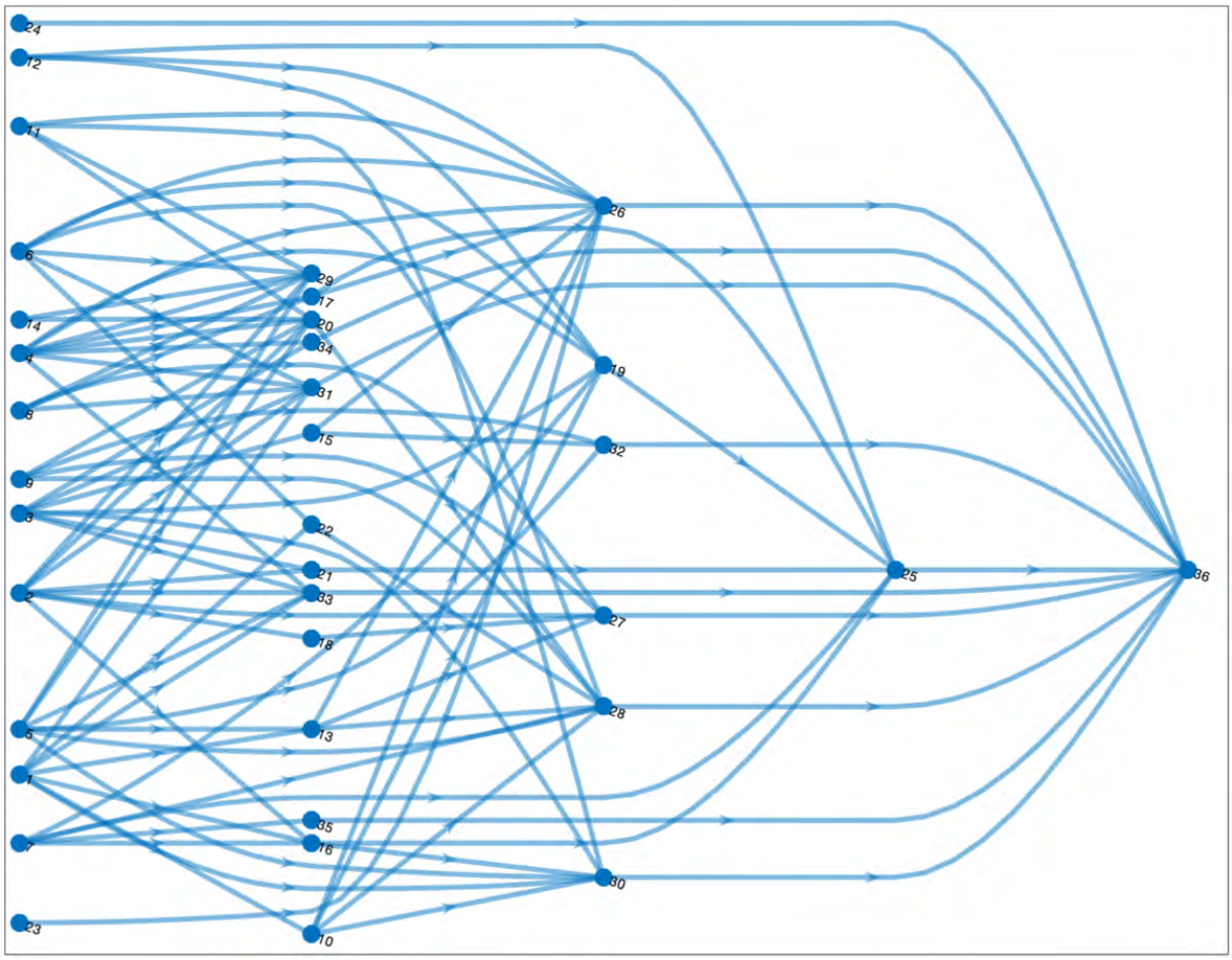



Figure 13: Activity network of Case 35

| Activity | $D_i$ | $\mu_i$ |
|---|---|---|
| 1 | 9 | 10 |
| 2 | 7 | 40 |
| 3 | 3 | 30 |
| 4 | 4 | 100 |
| 5 | 6 | 50 |
| 6 | 3 | 10 |
| 7 | 10 | 10 |
| 8 | 4 | 20 |
| 9 | 3 | 10 |
| 10 | 6 | 1000 |
| 11 | 9 | 10 |
| 12 | 8 | 500 |

| Activity | $D_i$ | $\mu_i$ |
|---|---|---|
| 13 | 5 | 200 |
| 14 | 2 | 10 |
| 15 | 5 | 400 |
| 16 | 2 | 10 |
| 17 | 10 | 10 |
| 18 | 4 | 2000 |
| 19 | 8 | 10 |
| 20 | 8 | 500 |
| 21 | 1 | 500 |
| 22 | 5 | 500 |
| 23 | 2 | 10 |
| 24 | 7 | 10 |

| Activity | $D_i$ | $\mu_i$ |
|---|---|---|
| 25 | 1 | 300 |
| 26 | 4 | 400 |
| 27 | 3 | 200 |
| 28 | 4 | 1000 |
| 29 | 10 | 300 |
| 30 | 7 | 500 |
| 31 | 2 | 200 |
| 32 | 9 | 100 |
| 33 | 7 | 100 |
| 34 | 1 | 100 |
| 35 | 7 | 200 |

Table 11: Activity duration $D_i$ and the mean of disruption magnitude $\mu_i$ for Case 35