A compact reformulation of the two-stage robust resource-constrained project scheduling problem

Matthew Bold[1] and Marc Goerigk[2]

[1]STOR-i Centre for Doctoral Training, Lancaster University, United Kingdom
[2]Network and Data Science Management, University of Siegen, Germany

Abstract

This paper considers the resource-constrained project scheduling problem with uncertain activity durations. We assume that activity durations lie in a budgeted uncertainty set, and follow a robust two-stage approach, where a decision maker must resolve resource conflicts subject to the problem uncertainty, but can determine activity start times after the uncertain activity durations become known.

We introduce a new reformulation of the second-stage problem, which enables us to derive a compact robust counterpart to the full two-stage adjustable robust optimisation problem. Computational experiments show that this compact robust counterpart can be solved using standard optimisation software significantly faster than the current state-of-the-art algorithm for solving this problem, reaching optimality for almost 50% more instances on the same benchmark set.

Keywords: project scheduling; robust optimisation; resource constraints; budgeted uncertainty

1 Introduction

The resource-constrained project scheduling problem (RCPSP) consists of scheduling a set of activities, subject to precedence constraints and limited resource availability, with the objective of minimising the overall project duration, known as the makespan. Given its practical relevance to a number of industries, including construction (Kim, 2013), manufacturing (Gourgand et al., 2008), R&D (Vanhoucke, 2006), and personnel scheduling (Drezet and Billaut, 2008), the RCPSP and many of its variants have been widely studied since a first model was introduced by Pritsker et al. (1969). The vast majority of this research, however, has examined the RCPSP under the assumption that the model parameters are known deterministically (for a survey of the deterministic RCPSP, see Artigues et al. (2008)), but clearly, in practice, large projects are subject to non-trivial uncertainties. For instance, poor weather might delay construction times, uncertain delivery times of parts may delay manufacturing activities, and the duration of research activities are inherently uncertain.

As a result, in recent years, increasing attention has been given to the uncertain RCPSP, where scheduling decisions must be made whilst activity durations are unknown.

There exist two main approaches for solving the uncertain RCPSP. The first is to view the problem as a dynamic optimisation problem where scheduling decisions are made each time new information becomes available according to a scheduling policy (Igelmund and Radermacher, 1983a,b; Möhring and Stork, 2000). Most recently, Li and Womer (2015) use approximate dynamic programming to find an adaptive closed-loop scheduling policy for the uncertain RCPSP.

The second approach aims to proactively develop a robust baseline schedule that protects against delays in the activity durations. Zhu et al. (2007) present a two-stage stochastic programming formulation for building baseline schedules for projects with a single resource. Bruni et al. (2015) present a chance-constraint-based heuristic for constructing robust baseline schedules and Lamas and Demeulemeester (2016) introduce a procedure for generating robust baseline schedules that is independent of later reactive scheduling procedures. For a review of both dynamic and proactive project scheduling, see Herroelen and Leus (2005).

Although frequently referred to as robust, none of the scheduling methods described above make use of robust optimisation in the sense of Ben-Tal and Nemirovski (1998, 1999, 2000). Over the last 20 years, robust optimisation has emerged as an effective framework for modelling uncertain optimisation problems. Unlike stochastic programming, robust optimisation does not require probabilistic knowledge of the uncertain data. Instead, the robust optimisation approach only assumes that the uncertain data lie somewhere in a given uncertainty set, and then aims to find solutions that are robust for all scenarios that can arise from that uncertainty set.

The applicability of robust optimisation as a method for solving uncertain optimisation problems has increased following the introduction of adjustable robust optimisation (Ben-Tal et al., 2004; Yanıkoğlu et al., 2019). Adjustable robust optimisation extends static robust optimisation into a dynamic setting, where a subset of the decision variables must be determined under uncertainty, whilst other variables can be adjusted following observations of the uncertain data. As well as accurately modelling the decision process undertaken by many real-world decision-makers, adjustable robust optimisation overcomes the over-conservativeness that restricts the applicability of static robust optimisation models. For extensive surveys on robust optimisation, see Ben-Tal et al. (2009); Bertsimas et al. (2011); Gorissen et al. (2015); Goerigk and Schöbel (2016).

Despite the successful application of robust optimisation in many different fields (see Bertsimas et al. (2011)), few papers have directly applied robust optimisation in the construction of robust baseline project schedules. Balouka and Cohen (2019) consider the multi-mode RCPSP under the framework of robust optimisation, and present a solution approach based on Benders' decomposition (Benders, 1962). Artigues et al. (2013) consider the RCPSP under uncertain activity durations and present an iterative scenario-relaxation algorithm, with the objective of minimising the worst-case absolute regret (Kouvelis and Yu, 1997).

Bruni et al. (2017) introduce the two-stage adjustable robust RCPSP that we consider in this paper. For the case of budgeted uncertainty, they solve this problem using a Benders'-style decomposition approach. Bruni et al. (2018) extend this work and present a computational study of solution methods for solving the two-stage adjustable RCPSP. An additional Benders'-style algorithm is compared against a primal decomposition algorithm, as well as the algorithm presented in Bruni et al. (2017). The primal decomposition algorithm is shown to be the best performing algorithm for solving the two-stage adjustable RCPSP. At the same time, only 767 of 1440 instances could be solved to optimality, which means that a large gap of unsolved instances remains.

Simultaneously and independently of this work, Pass-Lanneau et al. (2020) have studied a related problem known as the anchor-robust RCPSP, where baseline schedules are developed such that the start times of the activities in a given subset are guaranteed to remain unchanged upon the realisation of the uncertain data. Examining their problem, they independently obtain a similar formulation to the one we present in Section 3.2.

The contributions of this paper are as follows. We present a new compact reformulation of the two-stage adjustable robust RCPSP with budgeted uncertainty. This is the first compact formulation for this problem, allowing us to solve it directly using standard optimisation software. As a result, and as computational experiments confirm, our compact reformulation can be solved significantly faster, and for a much greater number of instances than the current best algorithm for solving this problem.

The remainder of this paper is organised as follows: Section 2 introduces the two-stage adjustable robust RCPSP in detail, before Section 3 derives a compact reformulation of this problem and computational experiments are presented in Section 4. Concluding remarks are made in Section 5.

2 The two-stage robust RCPSP

A project consists of a set $V = \{0, 1, \ldots, n, n+1\}$ of non-preemptive activities, where 0 and $n+1$ are dummy source and sink activities with duration 0. Each activity $i \in V$ requires an amount $r_{ik} \geq 0$ of resource $k \in K$, where K is the set of project resource types. Each resource $k \in K$ has a finite availability R_k in each time period. Each activity $i \in V$ has a nominal duration given by $\bar{\theta}_i$, and a worst-case duration given by $\bar{\theta}_i + \hat{\theta}_i$, where $\hat{\theta}_i$ is its maximum deviation. In addition to resource constraints, the project activities must be scheduled in a manner that respects a set E of strict finish-to-start precedence constraints, where $(i, j) \in E$ enforces that activity i must have finished before activity j can begin. A project can be represented on a directed graph $G(V, E)$. An example project involving seven non-dummy activities and a single resource is shown in Figure 1.

We assume that the duration of each activity $i \in V$ lies somewhere between its nominal value $\bar{\theta}_i$ and its worst-case value $\bar{\theta}_i + \hat{\theta}_i$. Additionally, we follow Bertsimas and Sim (2004) and assume that only a subset of all activities can simultaneously attain their worst-case values. Hence, the set in which we assume durations can lie, known as the uncertainty set,

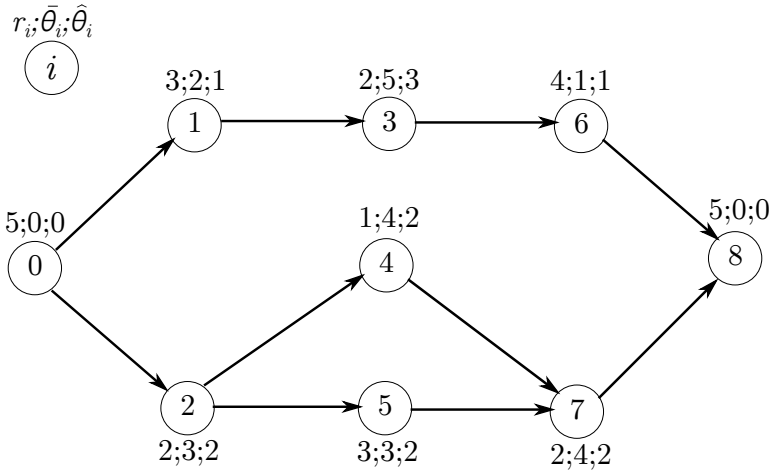


Figure 1: Example project involving seven non-dummy activities and a single resource with $R_1 = 5$.

is given by

$$\mathcal{U}(\Gamma) = \left\{ \theta \in \mathbb{R}_+^{|V|} \,:\, \theta_i = \bar{\theta}_i + \delta_i \hat{\theta}_i,\, 0 \leq \delta_i \leq 1 \,\forall i \in V,\, \sum_{i \in V} \delta_i \leq \Gamma \right\},$$

where Γ determines the robustness of the solution by controlling the number of activities that are allowed to reach their worst-case duration simultaneously. For $\Gamma = 0$, each activity takes its nominal duration and the problem reduces to the deterministic RCPSP. At the other extreme, when $\Gamma = n$, every activity can take its worst-case duration, and this uncertainty set becomes equivalent to interval uncertainty.

The robust RCPSP lends itself naturally to a two-stage decision process, where resource allocation decisions need to be made at the start of the project before the uncertain activity durations become known, but the activity start times can be decided following the realisation of the activity durations. Hence, resource allocation decisions constitute the set of first-stage decisions, whilst the activity start times constitute the set of second-stage decisions.

More specifically, the first-stage resource allocation decisions consist of determining a feasible extension of the project precedence relationships E so that all resource conflicts are resolved. A forbidden set (Igelmund and Radermacher, 1983a) is any subset $F \subseteq V$ of non-precedence-related activities such that $\sum_{i \in F} r_{ik} > R_k$ for at least one $k \in K$, i.e. the activities of F cannot be executed simultaneously without violating a resource constraint. A minimal forbidden set is a forbidden set that does not contain any other forbidden set as a subset. We denote the set of minimal forbidden sets by $\mathcal{F}$. For the example project in Figure 1, $\mathcal{F} = \big\{\{1,5\},\{2,6\},\{5,6\},\{6,7\},\{3,4,5\}\big\}$. The resource conflict represented by each minimal forbidden set can be resolved by adding an additional precedence relationship to the project network. Bartusch et al. (1988) show that solving the RCPSP is equivalent to finding an optimal choice of additional precedence relationships $X \subseteq V^2 \setminus E$, such that

the extended project network $G'(V, E \cup X)$ is acyclic and contains no forbidden sets. Such an extension X to the project precedence network is referred to as a sufficient selection. Hence, a solution to the first-stage problem corresponds to the choice of a sufficient selection X. Figure 2 shows the extended project network for a sufficient selection to the example project shown in Figure 1 (arcs in X are dashed).

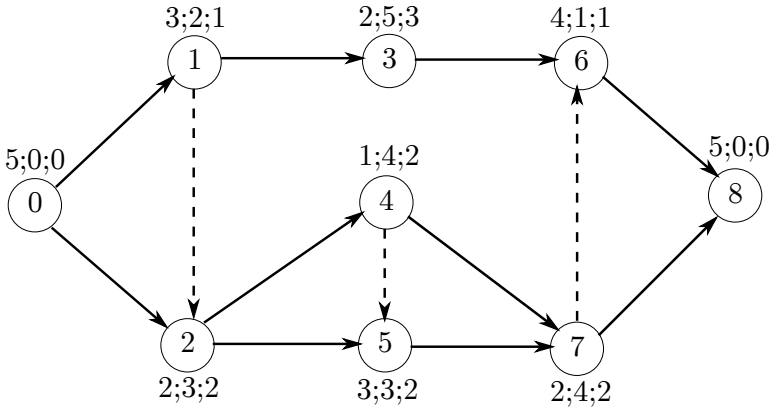


Figure 2: An extension of the example project shown in Figure 1, corresponding to the sufficient selection given by the dashed arcs.

Given the extended project network resulting from the choice of sufficient selection made in the first stage, the second stage problem consists of determining activity start times in order to minimise the worst-case makespan in this extended network. Since all resource conflicts have been resolved in the first-stage problem, the second stage problem contains no resource constraints.

Hence, the two-stage robust RCPSP under budgeted uncertainty is given by:

$$\min_{X \in \mathcal{X}} \max_{\theta \in \mathcal{U}(\Gamma)} \min_{S \in \mathcal{S}(X,\theta)} S_{n+1} \tag{1}$$

where $\mathcal{X}$ is the set of sufficient selections, and $\mathcal{S}(X, \theta)$ is the set of feasible activity start times given the activity durations $\theta \in \mathcal{U}(\Gamma)$ and choice of sufficient selection X. That is,

$$\mathcal{S}(X, \theta) = \left\{ S \in \mathbb{R}_+^{|V|} \,:\, S_0 = 0,\ S_j - S_i \geq \theta_i \, \forall (i,j) \in E \cup X \right\}.$$

To solve this problem we propose a mixed-integer programming formulation, outlined in the following section.

3 A compact reformulation

In this section, we present a reformulation of the two-stage robust RCPSP. Unlike existing formulations for the two-stage adjustable RCPSP, the formulation we propose is *compact*, i.e. it contains polynomially many constraints and variables. We begin by first examining

the adversarial sub-problem of maximising the worst-case makespan for a given sufficient selection.

3.1 The adversarial sub-problem

Suppose the solution to the first-stage problem provides a sufficient selection $X \in \mathcal{X}$, and is given by a vector $y \in \{0,1\}^{V \times V}$ where

$$y_{ij} = \begin{cases} 1 & \text{if } (i,j) \in E \cup X \\ 0 & \text{otherwise.} \end{cases}$$

The second-stage sub-problem that arises can be considered from the point of view of an adversary who wishes to choose the worst-case scenario of delays for the given first-stage solution. Following the adversary's choice of delays, we can determine the start time of each activity in order to minimise this worst-case makespan.

Let us assume a fixed scenario $\theta \in \mathcal{U}(\Gamma)$ given by the vector $\delta \in [0,1]^{|V|}$. In this case, the inner minimisation problem becomes

$$\min \; S_{n+1} \tag{2}$$
$$\text{s.t. } S_0 = 0 \tag{3}$$
$$S_j - S_i \geq \bar{\theta}_i + \delta_i \hat{\theta}_i - M(1 - y_{ij}) \qquad \forall (i,j) \in V^2 \tag{4}$$
$$S_i \geq 0 \qquad \forall i \in V, \tag{5}$$

where M is some number greater than or equal to the maximum possible minimum makespan. By taking the dual of (2)-(5), and then introducing the adversarial delay variables δ_i, $i \in V$, we obtain the following non-linear mixed-integer programming formulation for the adversarial sub-problem, first introduced in Bruni et al. (2017):

$$\max \sum_{(i,j) \in V^2} \left(\bar{\theta}_i + \delta_i \hat{\theta}_i - M(1 - y_{ij}) \right) \alpha_{ij} \tag{6}$$
$$\text{s.t. } \sum_{(i,j) \in V^2} \alpha_{ij} - \sum_{(j,i) \in V^2} \alpha_{ji} = 0 \qquad \forall j \in V \tag{7}$$
$$\sum_{(0,i) \in V^2} \alpha_{0i} = 1 \tag{8}$$
$$\sum_{(i,n+1) \in V^2} \alpha_{i,n+1} = 1 \tag{9}$$
$$\sum_{i \in V} \delta_i \leq \Gamma \tag{10}$$
$$0 \leq \delta_i \leq 1 \qquad \forall i \in V \tag{11}$$
$$\alpha_{ij} \in \{0,1\} \qquad \forall (i,j) \in V^2. \tag{12}$$

Observe that (6)-(9) correspond to the dual of (2)-(5), where the dual variables α_{ij} are continuous. Note also how the delay variables δ_i, $i \in V$, are related to the dual variables

α_{ij} through the objective (6). The dual variables α_{ij} determine a longest path through the network defined by the first-stage variables y_{ij}. Hence, when $\alpha_{ij} = 1$, edge (i, j) is included in this longest path, and the duration of activity i, which determined by its delay variable δ_i, contributes to its length. Therefore, with the addition of the delay variables δ_i, $i \in V$, this adversarial sub-problem can be thought of as a non-linear longest-path problem through the network determined in the first-stage problem, where up to Γ units of delay can be distributed among activities in order to further maximise this longest path. For fixed choice of δ, it is possible to find an optimal solution to this problem where each α_{ij} is binary. The advantage of binary variables α_{ij} is that products $\delta_i \alpha_{ij}$ can be easily linearised with the introduction of additional variables. As shown by Bruni et al. (2017), this linearised model is given as follows:

$$\max \sum_{(i,j)\in V^2} \left(\bar{\theta}_i \alpha_{ij} + \hat{\theta}_i w_{ij} - M(1 - y_{ij})\alpha_{ij}\right) \tag{13}$$

$$\text{s.t.} \sum_{(i,j)\in V^2} \alpha_{ij} - \sum_{(j,i)\in V^2} \alpha_{ji} = 0 \qquad \forall j \in V \tag{14}$$

$$\sum_{(0,i)\in V^2} \alpha_{0i} = 1 \tag{15}$$

$$\sum_{(i,n+1)\in V^2} \alpha_{i,n+1} = 1 \tag{16}$$

$$w_{ij} \leq \delta_i \qquad \forall (i,j) \in V^2 \tag{17}$$

$$w_{ij} \leq \alpha_{ij} \qquad \forall (i,j) \in V^2 \tag{18}$$

$$\sum_{i\in V} \delta_i \leq \Gamma \tag{19}$$

$$0 \leq \delta_i \leq 1 \qquad \forall i \in V \tag{20}$$

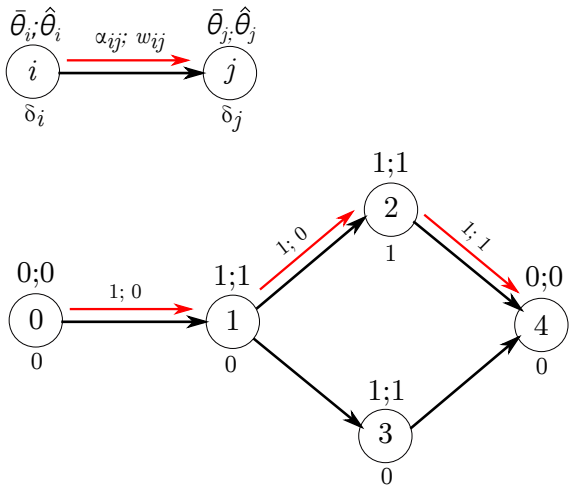
$$\alpha_{ij} \in \{0, 1\} \qquad \forall (i,j) \in V^2 \tag{21}$$

$$w_{ij} \geq 0 \qquad \forall (i,j) \in V^2. \tag{22}$$

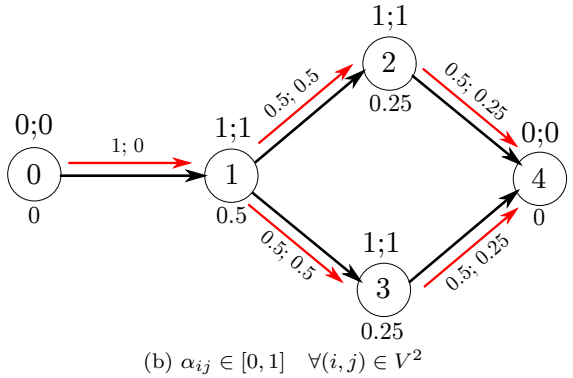
It is claimed in Proposition 4 of Bruni et al. (2017) that this problem is equivalent to its linear relaxation, where $\alpha_{ij} \in [0, 1]$ for all $(i, j) \in V^2$. This, however, is not the case, as the following counter-example demonstrates.

Figure 3 shows a project with three non-dummy activities, each with a nominal duration of $\bar{\theta}_i = 1$, and a maximum deviation of $\hat{\theta}_i = 1$, $i = 1, 2, 3$. Suppose a feasible first-stage solution has been found, resulting in the network shown in Figure 3. We consider this problem from the point of view of the adversary, who wishes to distribute up to $\Gamma = 1$ units of delay, in order to maximise the minimum makespan. If (13)-(22) is equivalent to its linear relaxation, then the adversary gains no advantage by choosing $\alpha \in (0, 1)$ and splitting the unit flow on its route from the source-node 0 to the sink-node 4. However, as can be seen with this example, the adversary does in fact obtain an advantage.

In Figure 3a, $\alpha_{ij} \in \{0, 1\}$ for each $(i, j) \in V^2$, and hence the adversary is limited to routing the unit flow through the network via a single path. A worst-case delay in this scenario is that the unit of available delay is entirely assigned to activity 2. Hence,



(a) $\alpha_{ij} \in \{0, 1\} \quad \forall (i, j) \in V^2$



(b) $\alpha_{ij} \in [0, 1] \quad \forall (i, j) \in V^2$

Figure 3: Counter-example showing that model (13)-(22) is not equivalent to its linear relaxation.

$\delta_2 = 1$, whilst $\delta_1 = \delta_3 = 0$. Minimising the worst-case makespan in this scenario, we get $(\bar{\theta}_1\alpha_{12} + \hat{\theta}_1 w_{12}) + (\bar{\theta}_2\alpha_{24} + \hat{\theta}_2 w_{24}) = (1 + 0) + (1 + 1) = 3$.

In Figure 3b, $\alpha_{ij} \in [0, 1]$ for each $(i, j) \in V^2$, and the adversary is able to split the unit flow into multiple fractional paths on its route through the network. In this case, the adversary can distribute the unit of delay so that $\delta_1 = 0.5$, $\delta_2 = 0.25$, and $\delta_3 = 0.25$. In this scenario, the minimum makespan is $(\bar{\theta}_1\alpha_{12} + \hat{\theta}_1 w_{12}) + (\bar{\theta}_1\alpha_{13} + \hat{\theta}_1 w_{13}) + (\bar{\theta}_2\alpha_{24} + \hat{\theta}_2 w_{24}) + (\bar{\theta}_3\alpha_{34} + \hat{\theta}_3 w_{34}) = (0.5 + 0.5) + (0.5 + 0.5) + (0.5 + 0.25) + (0.5 + 0.25) = 3.5$, showing that problem (13)-(22) is not equivalent to its linear relaxation.

Note that Bruni et al. (2017) attempt to prove that model (13)-(22) is equivalent to its linear relaxation, and therefore polynomially solvable, by showing that the corresponding constraint matrix is totally unimodular. In Appendix A, we identify an error with this proof and show that the constraint matrix is not totally unimodular. This result is consistent with the above counter-example.

Since problem (13)-(22) is not equivalent to its linear relaxation, we cannot apply strong-duality to get an equivalent minimisation problem. Therefore, in order to obtain a compact

reformulation of the two-stage robust RCPSP, an alternative reformulation of the adversarial sub-problem is required.

A dynamic programming procedure for solving problem (13)-(22) when $\Gamma \in \mathbb{Z}$ is presented in Bruni et al. (2017). This procedure works by considering $\Gamma + 1$ paths from the source node 0 to node i, for each $i \in V$, where each path π_i^γ, $\gamma = 0, \ldots, \Gamma$, is characterised by the inclusion of exactly γ delayed activities. Given a path π_i^γ, its extension to each successor node $j \in Succ_i$ is evaluated by considering two possibilities: either the successor activity j is delayed, resulting in the path $\pi_j^{\gamma+1}$, or it is not delayed, resulting in the path π_j^γ. Hence, the dynamic programming algorithm has a state $ST(j, \gamma)$ for each node j at level γ, and the value of each state $V(ST(j, \gamma))$ is computed through the following recursion:

$$V(ST(0,0)) = 0, \tag{23}$$

$$V(ST(j,\gamma)) = \max_{i:(i,j)\in E\cup X} \Big\{ \max\Big(V(ST(i,\gamma)),\, V(ST(i,\gamma-1)) + \bar{\theta}_i + \hat{\theta}_i\Big)\Big\}, \quad \forall j \in V \setminus \{0\},\ \gamma = 1, \ldots, \Gamma \tag{24}$$

$$V(ST(j,0)) = \max_{i:(i,j)\in E\cup X} \Big\{V(ST(i,0)) + \bar{\theta}_i\Big\}. \tag{25}$$

This dynamic programming algorithm can be viewed as finding the critical path through the augmented project network built from $\Gamma + 1$ copies of the original project network (an example of such a network is shown in Figure 4). The inclusion of an inter-level arc, e.g. a dashed arc in Figure 4, in the critical path corresponds to the delay of the activity at the origin of that arc.

Since the second stage problem is simply a longest-path problem on this augmented network, it can be recast into the following mixed-integer linear program:

$$\max \sum_{(i,j)\in V^2} \sum_{\gamma=0}^{\Gamma} (\bar{\theta}_i - M(1 - y_{ij}))\alpha_{ij\gamma} + \sum_{(i,j)\in V^2} \sum_{\gamma=1}^{\Gamma} (\bar{\theta}_i + \hat{\theta}_i - M(1 - y_{ij}))\beta_{ij\gamma} \tag{26}$$

$$\text{s.t.} \sum_{(j,i)\in V^2} \alpha_{ji\gamma} + \sum_{(j,i)\in V^2} \beta_{ji,\gamma+1} - \sum_{(i,j)\in V^2} \alpha_{ij\gamma} - \sum_{(i,j)\in V^2} \beta_{ij\gamma} = 0 \quad \forall j \in V,\ \gamma = 1, \ldots, \Gamma - 1 \tag{27}$$

$$\sum_{(j,i)\in V^2} \alpha_{ji0} + \sum_{(j,i)\in V^2} \beta_{ji1} - \sum_{(i,j)\in V^2} \alpha_{ij0} = 0 \quad \forall j \in V \tag{28}$$

$$\sum_{(j,i)\in V^2} \alpha_{ji\Gamma} - \sum_{(i,j)\in V^2} \alpha_{ij\Gamma} - \sum_{(i,j)\in V^2} \beta_{ij\Gamma} = 0 \quad \forall j \in V \tag{29}$$

$$\sum_{(0,i)\in V^2} \alpha_{0i0} + \sum_{(0,i)\in V^2} \beta_{0i1} = 1 \tag{30}$$

$$\sum_{(i,n+1)\in V^2} \alpha_{i,n+1,\Gamma} + \sum_{(i,n+1)\in V^2} \beta_{i,n+1,\Gamma} = 1 \tag{31}$$

$$\alpha_{ij\gamma} \in \{0,1\} \quad \forall (i,j) \in V^2,\ \gamma = 0, \ldots, \Gamma \tag{32}$$

$$\beta_{ij\gamma} \in \{0,1\} \quad \forall (i,j) \in V^2,\ \gamma = 1, \ldots, \Gamma \tag{33}$$

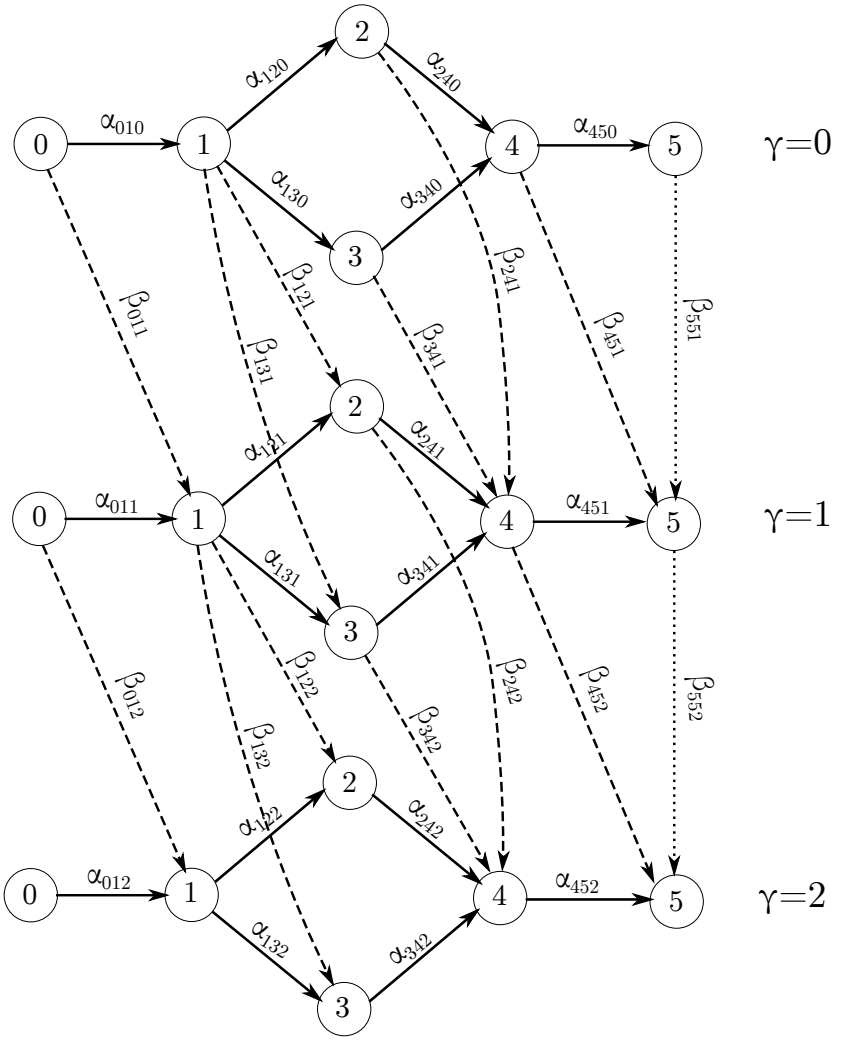


Figure 4: Example augmented graph for a project with four non-dummy activities, and where up to $\Gamma = 2$ activities can reach their worst-case durations.

where $\alpha_{ij\gamma}$ is the flow from node i to node j in level γ and $\beta_{ij\gamma}$ is the flow from node i in level $\gamma - 1$ to node j in level γ. The constraints model a unit flow through the augmented network from node 0 in level 0 (Constraint (30)) to node $n+1$ in level Γ (Constraint (31)). Constraints (27) are flow-conservation constraints that ensure that for node each in level $\gamma = 1, \ldots, \Gamma - 1$, the incoming flow from levels γ and $\gamma - 1$ must be equal to the outgoing flow to levels γ and $\gamma + 1$. Constraints (28) and (29) conserve flow over the nodes in the special cases of the first and last level, respectively.

Note that this formulation includes more $\alpha_{ij\gamma}$ and $\beta_{ij\gamma}$ variables than indicated in Figure 4, with the edges shown in Figure 4 corresponding to the edges for which $y_{ij} = 1$. The edges that are not shown are penalised by constant M in the objective (26) when $y_{ij} = 0$. To ensure that it is always possible to find a path from node 0 in level 0 to node $n + 1$ in level Γ in the augmented network (if Γ is larger than the number of activities included in the longest path from node 0 to node $n+1$ in the original project network, such a path may not be possible), the final sink nodes of each layer are connected by enforcing $y_{n+1,n+1} = 1$ (see dotted arcs in Figure 4). Since $\bar{\theta}_{n+1} + \hat{\theta}_{n+1} = 0$ these additional edges can be traversed at no extra cost to reach node $n + 1$ in level Γ.

3.2 Compact reformulation

Since the second-stage problem (26)-(33) is simply a longest-path problem over an augmented project graph, it is equivalent to its linear relaxation where $\alpha_{ij\gamma} \in [0, 1]$ for all $(i, j) \in V^2$, $\gamma = 0, \ldots, \Gamma$, and $\beta_{ij\gamma} \in [0, 1]$ for all $(i, j) \in V^2$, $\gamma = 1, \ldots, \Gamma$. Hence, we can take the dual of this problem to get an equivalent minimisation problem.

The first-stage problem aims determine a sufficient selection $X \in \mathcal{X}$ that minimises the second-stage objective value. This first-stage problem can be modelled with a flow-based formulation, as proposed by Artigues et al. (2003). This formulation makes use of continuous resource flow variables f_{ijk}, which determine the amount of resource type $k \in K$ that is transferred upon the completion of activity i to activity j. Additionally, binary variables y_{ij} capture the choice of sufficient selection by representing precedence relationships of the extended project network.

Thus, having dualised the second-stage problem (26)-(33) into a minimisation problem, the first and second-stages can be combined to obtain the following reformulation of the full two-stage robust RCPSP with budgeted uncertainty:

$$\min \; S_{n+1,\Gamma} \tag{34}$$

$$\text{s.t.} \; S_{00} = 0 \tag{35}$$

$$S_{j\gamma} - S_{i\gamma} \geq \bar{\theta}_i - M(1 - y_{ij}) \qquad \forall (i,j) \in V^2, \gamma = 0, \ldots, \Gamma \tag{36}$$

$$S_{j,\gamma+1} - S_{i\gamma} \geq \bar{\theta}_i + \hat{\theta}_i - M(1 - y_{ij}) \qquad \forall (i,j) \in V^2, \gamma = 0, \ldots, \Gamma - 1 \tag{37}$$

$$y_{ij} = 1 \qquad \forall (i,j) \in E \cup \{(n+1, n+1)\} \tag{38}$$

$$f_{ijk} \leq N_k y_{ij} \qquad \forall (i,j) \in V^2, \forall k \in K \tag{39}$$

$$\sum_{i \in V} f_{ijk} = r_{jk} \qquad \forall j \in V, \forall k \in K \tag{40}$$

$$\sum_{j \in V} f_{ijk} = r_{ik} \qquad \forall i \in V, \forall k \in K \qquad (41)$$

$$S_{i\gamma} \geq 0 \qquad \forall i \in V, \gamma \in 0, \ldots, \Gamma \qquad (42)$$

$$f_{ijk} \geq 0 \qquad \forall (i,j) \in V^2, \forall k \in K \qquad (43)$$

$$y_{ij} \in \{0,1\} \qquad \forall (i,j) \in V^2, \qquad (44)$$

where M, as before, is chosen to be greater than or equal to the maximum possible minimum makespan, and N_k is some number greater than or equal to R_k. Constraints (35)-(37) are the dual constraints of the second-stage problem (26)-(33), and correspond to makespan constraints that ensure that activity start times respect the project precedence relationships. Constraints (38) capture the original project precedences, whilst constraints (39)-(41) are resource flow constraints. Constraints (39) ensure that the resource flows respect the precedence relationships, and constraints (40) and (41) conserve flow into and out of each node, respectively. Hence problem (34)-(44) can be seen as a extended makespan minimisation problem, in which a feasible project network must be constructed with the objective of minimising the overall makespan on the corresponding augmented network.

With polynomially many constraints and variables (specifically, $O(|V|^2 \cdot |K| + |V| \cdot \Gamma)$ many variables and $O(|V|^2(|K| + \Gamma) + |E|)$ many constraints), this formulation can be passed directly to standard optimisation software for solving, and the results of doing so are presented in the following section. This is the first formulation of the two-stage adjustable RCPSP for which this is the case. In comparison, the formulation from Bruni et al. (2018) makes use of $O(|V|^2|K| + |V|\Delta)$ many variables and $O(|V|^3 + |V|^2(|K| + \Delta) + |E|)$ many constraints, where $\Delta = \binom{|V|}{\Gamma}$ is an exponential number in Γ.

It is important to note that this basic formulation does not enforce the transitivity of the y-variables. Instead, the formulation captures the extended project network in terms of the y-variables with constraints (38) and (39), and ensures the feasibility of activity start-times with respect to this extended network through constraints (36) and (37). In Section 4 the computational benefits of extending model (34)-(44) to include explicit transitivity constraints on the y-variables is examined.

4 Computational experiments

This section compares results obtained by solving the compact robust counterpart (34)-(44), and three slight extensions to this method, with the current state-of-the-art approach to solve the two-stage robust RCPSP proposed in Bruni et al. (2018). Before outlining the proposed extensions to the basic model detailed in the previous section, we introduce the test instances used in this computational study.

The complete sets of results from these experiments as well as Python implementations of each of the four methods we propose can be found at `https://github.com/boldm1/two-stage-robust-RCPSP`.

4.1 Instances

The test instances used in this computational study have been converted from deterministic RCPSP instances involving 30 activities, taken from the PSPLIB (Kolisch and Sprecher (1997), http://www.om-db.wi.tum.de/psplib/). The difficulty of these instances is measured and controlled by the following three parameters:

1. Network complexity $NC \in \{1.5, 1.8, 2.1\}$. This measures the average number of non-redundant (i.e. non-transitive) arcs per activity.

2. Resource factor $RF \in \{0.25, 0.5, 0.75, 1\}$. This measures the average proportion of resource types for which a non-dummy activity has a non-zero requirement.

3. Resource strength $RS \in \{0.2, 0.5, 0.7, 1\}$. This measures the restrictiveness of the availability of the resources, with a smaller RS value indicating a more constrained project instance.

The PSPLIB contains a set of 10 instances for each of the 48 possible combinations of instance parameters.

The maximum deviation of the duration of each activity is set to be $\hat{\theta} = \lceil \bar{\theta}/2 \rceil$, where $\bar{\theta}$ is the nominal duration as specified in the original instance file. For each of the 480 deterministic RCPSP instances in the PSPLIB, three robust counterparts have been generated by considering $\Gamma \in \{3, 5, 7\}$, resulting in a total of 1440 test instances. The sets of 30 robust counterparts for each combination of instance parameters are labelled J301, J302, ..., J3048. Note that the instances used in this computational study are identical to the instances used in Bruni et al. (2017) and Bruni et al. (2018).

4.2 Implementations

The following section compares the performance of model (34)-(44) with that of three slight extensions. Here, we outline these extensions and clarify details regarding the practical implementation of these models.

The first variant of the basic model (34)-(44) includes the following transitivity constraints on the y-variables:

$$y_{ij} + y_{ji} \leq 1 \qquad \forall (i,j) \in V^2 \setminus \{(n+1, n+1)\} \tag{45}$$

$$y_{ij} \geq y_{il} + y_{lj} - 1 \qquad \forall (i,l,j) \in V^3. \tag{46}$$

As explained in Section 3.2, these transitivity of the y-variables is not strictly necessary to ensure the feasibility of the activity start-times. We include them as an extension to model (34)-(44) in order to assess their impact on the computational performance.

The second extension involves the provision of a heuristic warm-start solution to the solver software. This heuristic solution is obtained with the following procedure:

1. Given an uncertain RCPSP instance, a heuristic solution is found to the corresponding deterministic instance using the latest-finish-time (LFT) priority-rule heuristic (Kolisch, 1996).

2. From this solution, a feasible set of y-variables is obtained by setting

$$y_{ij} = \begin{cases} 1 & \text{if } s_j \geq f_i \\ 0 & \text{otherwise,} \end{cases}$$

   where s_j is the start time of activity j, and f_i is the finish time of activity i.

3. These y variables are passed to the basic model (34)-(44), which is solved to provide a feasible warm-start solution.

A detailed example of this warm-start procedure is given in Appendix B. This warm-start solution can be used to tighten the big-M constraints (36) and (37), and thereby further improve the basic model. This is achieved by setting $M_{ij} = LF_i - ES_j$ for each $(i, j) \in V^2$, where ES_j is the earliest start time of activity j, and LF_i is the latest finish time of activity i, calculated relative to the makespan of the warm-start solution. As before, these values are computed recursively via a forward-pass and backward-pass of the project network, respectively.

Note that, although the S-variables of formulation (34)-(44) are in general continuous, for the purposes of this computational study, the S-variables have been set to be integer. Since $\hat{\theta} = \lceil \bar{\theta}/2 \rceil \in \mathbb{Z}$ for the instances solved in this study, the correctness of the formulation is unaffected by this specification.

In summary, the following section presents results from the following five solution approaches:

1. Basic model (34)-(44),

2. Basic model with transitivity constraints, i.e (34)-(46),

3. Basic model with warm-start,

4. Basic model with warm-start and transitivity constraints,

5. Primal method from Bruni et al. (2018). This is the strongest existing approach for solving the two-stage robust RCPSP.

All the models proposed in this paper have been solved using Gurobi 9.0.1, running on 4 cores of a 2.30GHz Intel Xeon CPU, limited to 16GB RAM. Note that the specifications of this machine have been chosen to be as similar as possible to that of the CPU used in the experiments performed in Bruni et al. (2017) and Bruni et al. (2018). A limit of 20 minutes was imposed on the solution time of each model, the same as used for the experiments performed in Bruni et al. (2017) and Bruni et al. (2018). Results for the primal method have been reproduced from Bruni et al. (2018).

4.3 Results

In this section, we first present and analyse results from solving model (34)-(44) and the three variants proposed in the previous section, before we compare these results with those from the current best iterative algorithm presented in Bruni et al. (2018).

We start by considering the performance profile (Dolan and Moré, 2002) plot shown in Figure 5. The performance profile uses the *performance ratio* as a measure by which the different solution methods can be compared. The performance ratio of method $m \in \mathcal{M}$ for problem instance $i \in \mathcal{I}$ is defined to be

$$p_{im} = \frac{t_{im}}{\min_{m \in \mathcal{M}} t_{im}},$$

where t_{im} is the time required to solve instance i using method m. If method m is unable to solve instance i to optimality within the 20 minute time-limit, then $p_{im} = P$, where $P \geq \max_{i,m} r_{im}$. The performance profile of method $m \in \mathcal{M}$ is defined to be the function

$$\rho_m(\tau) = \frac{|\{p_{im} \leq \tau \, : \, i \in \mathcal{I}\}|}{|\mathcal{I}|},$$

i.e. the probability that the performance ratio of method m is within a factor τ of the best performance ratio. The performance profile in Figure 5 has been plotted on the log scale for clarity.

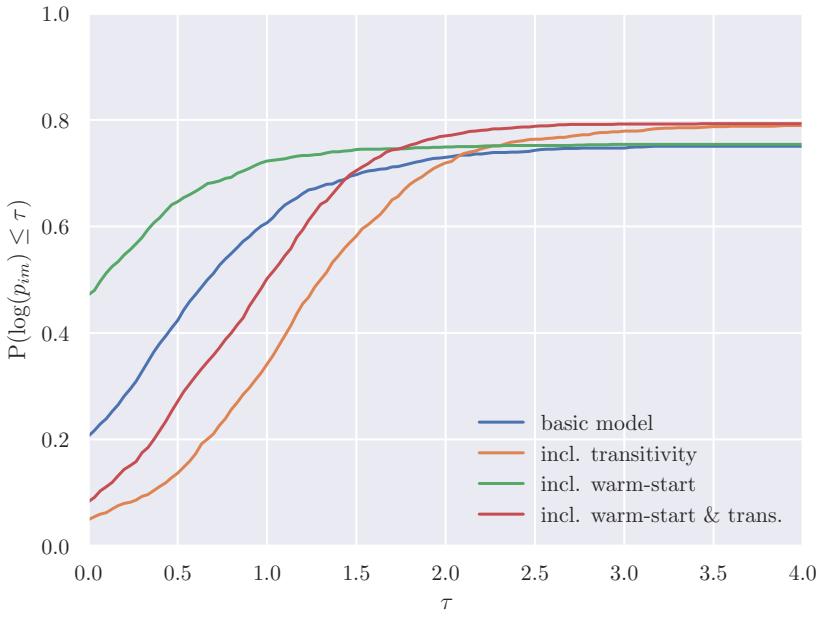


Figure 5: Performance profile of relative solution times.

It is clear from Figure 5 that the provision of a heuristic warm-start solution improves

solution time, with the model that make use of a warm-start solution being faster to solve for a greater proportion of instances that their respective models without a warm-start. It can also be seen that the models that make use of transitivity constraints are slower to solve to optimality for a greater proportion of instances than their respective models that do not use transitivity constraints. However, the inclusion of transitivity constraints does increase the proportion of instances that can be solved to optimality, by 5.3% for the basic model, and by 5.2% for the model with warm-start.

Figure 6 plots the cumulative percentage of instances solved to within a given optimality gap within the 20 minute time-limit. Note that the left-hand y-intercept of this figure gives the same information as the right-hand y-intercept in Figure 5, that is, the proportion of instances solved to optimality using each method. Looking at Figure 6, it can be seen that as well as increasing the proportion of instances that can be solved to optimality, the inclusion of transitivity constraints increases the proportion of instances that can be solved to within a given optimality gap. Of the 255 instances for which an optimal solution was unable to be found with any model, but for which a feasible solution was found using all models, the average optimality gap was 24.53% for the basic model, 22.80% with the inclusion of transitivity constraints, 24.71% with the inclusion of a warm-start solution, and 22.36% with the inclusion of both a warm-start solution and transitivity constraints. Note however that the basic model fails to find a feasible solution for only 3 instances, whilst the model that includes transitivity constraints fails to find a feasible solution for 24 instances. The other two variants find feasible solutions to all 1440 instances.

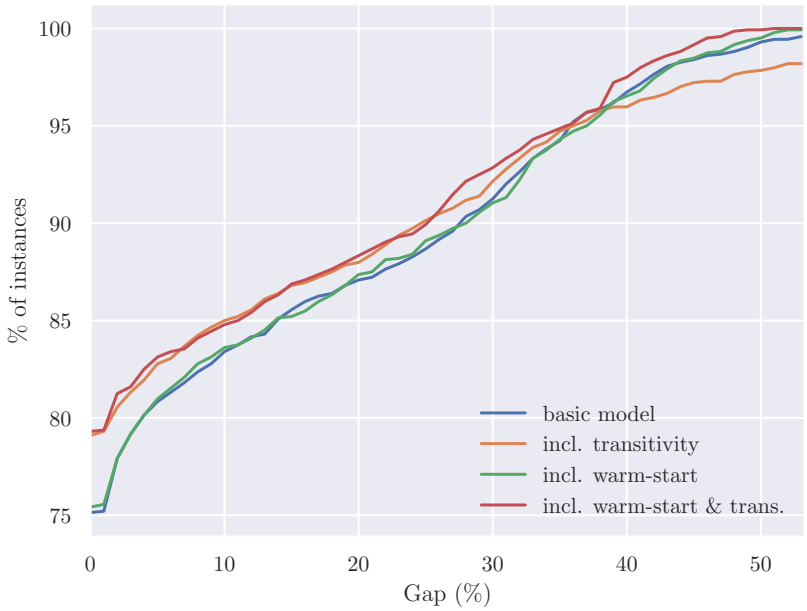


Figure 6: Cumulative percentage of instances solved to within given gap of optimality within time-limit.

Γ	basic model (34)-(44)			incl. trans		
	time	*gap*	*#solv*	*time*	*gap*	*#solv*
3	318.33	5.18	362	285.28	4.99	388
5	327.37	5.37	361	303.23	5.37	377
7	334.58	5.79	359	308.92	7.05	374
	326.76	5.45	1082	299.14	5.80	1139

Γ	incl. warm-start			incl. warm-start + trans.		
	time	*gap*	*#solv*	*time*	*gap*	*#solv*
3	312.77	5.29	366	283.74	4.49	386
5	319.83	5.22	361	292.14	4.60	383
7	329.24	5.49	359	310.07	4.87	373
	320.61	5.33	1086	295.32	4.65	1142

Table 1: Comparison of the variants of model (34)-(44) for different values of Γ.

From Figures 5 and 6, we can see that the inclusion of a warm-start solution and transitivity constraints in model (34)-(44), is the best performing variant: it solves the greatest number of instances to optimality, is the strongest performing model over the instances which no model can solve to optimality, and is significantly faster to solve than the transitive model without a warm-start.

In Table 1, we consider the impact of the uncertainty budget Γ on the performance of the basic model (34)-(44) and its three variants. For each method we report the average CPU time in seconds (*time*), the average optimality gap in percent (*gap*), and the number of instances solved to optimality (*#solv*). Note that in the case where a method was unable to find a feasible solution to given instance, an optimality gap of 100% has been reported. These results show that although the effect is limited, instances do appear to get more difficult to solve as Γ increases for all four methods.

In Table 2, we now compare the performance of the basic model (34)-(44) and its strongest extension, with the results of the strongest existing algorithm for the two-stage robust RCPSP, the *primal method* (Bruni et al., 2018). For each set of test instances, J301, ..., J3048, Table 2 reports instance parameters (NC, RF, RS), as well as the same measures that were reported in Table 1 (*time*, *gap*, *#solv*).

Of the 1440 test instances, 1160 have been solved to optimality within the time-limit by at least one of the four variants of model (34)-(44) proposed in this paper. As seen in Table 2, the primal method solves 767/1440 instance to optimality, whilst the basic method solves 1082/1440 instances to optimality (∼41% more than the primal method), and the strongest performing method, which includes the warm-start and transitivity constraints, solves 1142/1440 instances to optimality (∼49% more than the primal method). Furthermore, our methods reduce the average gap (4.66% instead of 6.30%) and result in smaller average solution times (295 seconds instead of 621 seconds).

There are only six out of 48 instance sets for which the primal method shows a slightly better performance than the methods we propose (J309, J3013, J3025, J3029, J3041, J3045). These contain some of the most difficult instances, for which all the methods

perform poorly. All three methods fail to find an optimal solution to almost all of the instances in these sets, however the primal method achieves a smaller optimality gap in the time limit. The iterative approach utilised by the primal method incrementally improves upon a feasible solution by solving a series of subproblems. For the most challenging instances, this iterative approach is more effective at reducing the optimality gap earlier in the solution process than the methods that we propose, which attempt to solve the full problem at once. It is important to note however that this does not necessarily mean that the primal method is able to solve these instances more quickly than the methods we propose, and that the overall solution times of all three methods on these instances remain unknown. Note also that the primal method solves some instances to optimality whilst simultaneously reaching the maximum time-limit of 1200 seconds. It is therefore unclear whether or not this is a numerical inaccuracy in the results presented in Bruni et al. (2018).

Overall, we find that the methods proposed in this paper considerably outperform the previous best approach, solving almost 50% more instances in a considerably shorter computation time. While the primal method has the drawback that several models have to be solved subsequently in an iterative process, our reformulation makes it possible to solve the two-stage adjustable RCPSP with a single mixed-integer program, utilising the strength of current solvers such as Gurobi.

5 Conclusion

This paper has introduced a new mixed-integer linear programming formulation for the robust counterpart to the two-stage adjustable robust RCPSP. This new compact formulation has been derived by considering a reformulation of the second-stage adversarial sub-problem of maximising the worst-case delayed makespan for a project without resource conflicts. The reformulation of this sub-problem is equivalent to a longest-path problem over an augmented project network made from multiple copies of the original project network. Hence, the dual of this longest-path problem can be inserted into the first-stage resource allocation problem to obtain a compact minimisation problem for the full two-stage robust RCPSP.

The performance of this new formulation has been examined over 1440 instances of varying characteristics and difficulty. Results show that the proposed formulation can be solved by standard optimisation software significantly faster than the current best algorithm for solving this problem. Using our approach, almost 50% more instances can be solved to optimality within the same time-limit, while also achieving a smaller average gap and a smaller average solution time.

Regarding future research on the two-stage robust RCPSP, the development of heuristic approaches for solving larger and more-challenging instances of this problem would seem to be a natural and worthwhile objective.

				primal method (Bruni et al., 2018)			basic model			incl. warm-start + trans.		
	NC	*RF*	*RS*	*time*	*gap*	*#solv*	*time*	*gap*	*#solv*	*time*	*gap*	*#solv*
J301	1.50	0.25	0.20	497.83	1.66	21	6.96	0.00	30	19.69	0.00	30
J302	1.50	0.25	0.50	192.39	0.24	28	2.68	0.00	30	6.57	0.00	30
J303	1.50	0.25	0.70	52.61	0.15	29	1.11	0.00	30	2.42	0.00	30
J304	1.50	0.25	1.00	124.97	1.18	27	0.78	0.00	30	1.62	0.00	30
J305	1.50	0.50	0.20	1200.00	15.92	0	1182.58	16.32	1	1099.26	13.29	8
J306	1.50	0.50	0.50	1115.86	11.56	3	155.59	0.05	29	102.27	0.00	30
J307	1.50	0.50	0.70	605.15	3.01	19	8.10	0.00	30	10.74	0.00	30
J308	1.50	0.50	1.00	363.31	1.85	22	1.45	0.00	30	1.95	0.00	30
J309	1.50	0.75	0.20	1200.00	10.19	0	1200.00	35.81	0	1200.00	30.71	0
J3010	1.50	0.75	0.50	1140.87	20.71	2	791.62	2.04	13	646.61	1.86	20
J3011	1.50	0.75	0.70	974.32	9.84	7	167.99	0.10	28	130.60	0.09	28
J3012	1.50	0.75	1.00	272.53	0.65	26	1.89	0.00	30	2.38	0.00	30
J3013	1.50	1.00	0.20	1200.00	50.55	1	1200.00	40.09	0	1200.00	37.27	0
J3014	1.50	1.00	0.50	1149.39	18.94	2	988.55	4.40	7	853.59	3.44	12
J3015	1.50	1.00	0.70	853.66	5.60	12	129.14	0.37	27	132.07	0.37	27
J3016	1.50	1.00	1.00	207.71	0.74	27	1.33	0.00	30	2.79	0.00	30
J3017	1.80	0.25	0.20	227.35	0.15	28	4.20	0.00	30	7.47	0.00	30
J3018	1.80	0.25	0.50	18.26	0.00	30	1.31	0.00	30	2.20	0.00	30
J3019	1.80	0.25	0.70	65.78	0.35	29	0.80	0.00	30	1.59	0.00	30
J3020	1.80	0.25	1.00	87.68	0.38	28	0.40	0.00	30	1.28	0.00	30
J3021	1.80	0.50	0.20	1200.00	9.28	2	967.73	7.77	10	757.75	5.04	18
J3022	1.80	0.50	0.50	877.51	7.11	10	45.13	0.00	30	43.52	0.00	30
J3023	1.80	0.50	0.70	356.09	0.86	24	2.71	0.00	30	4.65	0.00	30
J3024	1.80	0.50	1.00	201.76	1.13	26	0.95	0.00	30	1.75	0.00	30
J3025	1.80	0.75	0.20	1200.00	13.15	0	1200.00	31.71	0	1200.00	29.77	0
J3026	1.80	0.75	0.50	987.24	6.64	9	271.10	0.39	26	155.52	0.05	29
J3027	1.80	0.75	0.70	628.61	3.51	16	3.29	0.00	30	4.15	0.00	30
J3028	1.80	0.75	1.00	177.53	0.61	27	0.91	0.00	30	1.28	0.00	30
J3029	1.80	1.00	0.20	1200.00	10.5	1	1200.00	42.12	0	1200.00	39.23	0
J3030	1.80	1.00	0.50	1200.00	19.98	0	1158.57	4.51	3	1086.47	3.73	8
J3031	1.80	1.00	0.70	866.16	7.94	9	245.13	0.96	24	236.46	0.63	25
J3032	1.80	1.00	1.00	199.45	1.47	26	1.00	0.00	30	1.16	0.00	30
J3033	2.10	0.25	0.20	28.35	0.00	30	1.58	0.00	30	2.01	0.00	30
J3034	2.10	0.25	0.50	50.02	0.08	29	0.66	0.00	30	0.79	0.00	30
J3035	2.10	0.25	0.70	144.88	1.10	27	0.54	0.00	30	0.65	0.00	30
J3036	2.10	0.25	1.00	20.52	0.00	30	0.29	0.00	30	0.44	0.00	30
J3037	2.10	0.50	0.20	1131.60	5.59	7	634.14	6.69	18	523.56	2.31	23
J3038	2.10	0.50	0.50	463.92	1.59	23	11.63	0.00	30	12.53	0.00	30
J3039	2.10	0.50	0.70	268.81	0.78	27	3.55	0.00	30	2.56	0.00	30
J3040	2.10	0.50	1.00	257.42	1.66	24	1.29	0.00	30	1.14	0.00	30
J3041	2.10	0.75	0.20	1200.0	7.12	1	1200.00	26.73	0	1193.00	20.67	1
J3042	2.10	0.75	0.50	886.16	7.42	10	282.54	1.43	26	169.89	0.38	27
J3043	2.10	0.75	0.70	823.56	4.69	12	171.58	0.13	27	30.39	0.00	30
J3044	2.10	0.75	1.00	481.39	3.38	19	1.56	0.00	30	1.44	0.00	30
J3045	2.10	1.00	0.20	1164.00	8.10	2	1200.00	34.92	0	1200.00	31.91	0
J3046	2.10	1.00	0.50	1200.00	16.45	0	971.15	4.47	7	836.19	2.79	16
J3047	2.10	1.00	0.70	866.38	7.79	9	259.45	0.35	26	80.91	0.00	30
J3048	2.10	1.00	1.00	181.33	0.83	26	1.33	0.00	30	1.59	0.00	30
				621.09	6.30	767	326.76	5.24	1082	295.32	4.66	1142

Table 2: Comparison of primal method (Bruni et al., 2018), basic model (34)-(44), and extended model including warm-start and transitivity constraints.

Acknowledgements

The authors are grateful for the support of the EPSRC-funded (EP/L015692/1) STOR-i Centre for Doctoral Training.

References

Artigues, C., Demassey, S., and Neron, E., editors (2008). *Resource-Constrained Project Scheduling: Models, Algorithms, Extensions and Applications*. ISTE/Wiley.

Artigues, C., Leus, R., and Nobibon, F. T. (2013). Robust optimization for resource-constrained project scheduling with uncertain activity durations. *Flexible Services and Manufacturing Journal*, 25(1-2):175–205.

Artigues, C., Michelon, P., and Reusser, S. (2003). Insertion techniques for static and dynamic resource-constrained project scheduling. *European Journal of Operational Research*, 149(2):249–267.

Balouka, N. and Cohen, I. (2019). A robust optimization approach for the multi-mode resource-constrained project scheduling problem. *European Journal of Operational Research*.

Bartusch, M., Möhring, R. H., and Radermacher, F. J. (1988). Scheduling project networks with resource constraints and time windows. *Annals of Operations Research*, 16(1):199–240.

Ben-Tal, A., El Ghaoui, L., and Nemirovski, A. (2009). *Robust Optimization*, volume 28. Princeton University Press.

Ben-Tal, A., Goryashko, A., Guslitzer, E., and Nemirovski, A. (2004). Adjustable robust solutions of uncertain linear programs. *Mathematical Programming*, 99(2):351–376.

Ben-Tal, A. and Nemirovski, A. (1998). Robust convex optimization. *Mathematics of Operations Research*, 23(4):769–805.

Ben-Tal, A. and Nemirovski, A. (1999). Robust solutions of uncertain linear programs. *Operations Research Letters*, 25(1):1–13.

Ben-Tal, A. and Nemirovski, A. (2000). Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical Programming*, 88(3):411–424.

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252.

Bertsimas, D., Brown, D. B., and Caramanis, C. (2011). Theory and applications of robust optimization. *SIAM Review*, 53(3):464–501.

Bertsimas, D. and Sim, M. (2004). The price of robustness. *Operations Research*, 52(1):35–53.

Bruni, M. E., Beraldi, P., and Guerriero, F. (2015). The stochastic resource-constrained project scheduling problem. In *Handbook on Project Management and Scheduling*, volume 2, pages 811–835. Springer.

Bruni, M. E., Pugliese, L. D. P., Beraldi, P., and Guerriero, F. (2017). An adjustable robust optimization model for the resource-constrained project scheduling problem with uncertain activity durations. *Omega*, 71:66–84.

Bruni, M. E., Pugliese, L. D. P., Beraldi, P., and Guerriero, F. (2018). A computational study of exact approaches for the adjustable robust resource-constrained project scheduling problem. *Computers & Operations Research*, 99:178–190.

Dolan, E. D. and Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213.

Drezet, L.-E. and Billaut, J.-C. (2008). Employee scheduling in an IT company. In Artigues, C., Demassey, S., and Neron, E., editors, *Resource-Constrained Project Scheduling: Models, Algorithms, Extensions and Applications*, pages 243–255. ISTE/Wiley.

Ghouila-Houri, A. (1962). Caractérisation des matrices totalement unimodulaires. *Comptes Redus Hebdomadaires des Séances de l'Académie des Sciences (Paris)*, 254:1192–1194.

Goerigk, M. and Schöbel, A. (2016). Algorithm engineering in robust optimization. In *Algorithm Engineering*, pages 245–279. Springer.

Gorissen, B. L., Yanıkoğlu, İ., and den Hertog, D. (2015). A practical guide to robust optimization. *Omega*, 53:124–137.

Gourgand, M., Grangeon, N., and Norre, S. (2008). Assembly shop scheduling. In Artigues, C., Demassey, S., and Neron, E., editors, *Resource-Constrained Project Scheduling: Models, Algorithms, Extensions and Applications*, pages 229–242. ISTE/Wiley.

Herroelen, W. and Leus, R. (2005). Project scheduling under uncertainty: Survey and research potentials. *European Journal of Operational Research*, 165(2):289–306.

Igelmund, G. and Radermacher, F. J. (1983a). Algorithmic approaches to preselective strategies for stochastic scheduling problems. *Networks*, 13(1):29–48.

Igelmund, G. and Radermacher, F. J. (1983b). Preselective strategies for the optimization of stochastic project networks under resource constraints. *Networks*, 13(1):1–28.

Kim, J.-L. (2013). Genetic algorithm stopping criteria for optimization of construction resource scheduling problems. *Construction Management and Economics*, 31(1):3–19.

Kolisch, R. (1996). Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. *European Journal of Operational Research*, 90(2):320–333.

Kolisch, R. and Sprecher, A. (1997). PSPLIB - A project scheduling problem library: OR software - ORSEP operations research software exchange program. *European Journal of Operational Research*, 96(1):205–216.

Kouvelis, P. and Yu, G. (1997). Robust discrete optimization and its applications (nonconvex optimization and its applications). In *Nonconvex Optimization and Its Applications*, volume 14. Kluwer Academic Publishers.

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

Li, H. and Womer, N. K. (2015). Solving stochastic resource-constrained project scheduling problems by closed-loop approximate dynamic programming. *European Journal of Operational Research*, 246(1):20–33.

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

Pass-Lanneau, A., Bendotti, P., and Brunod-Indrigo, L. (2020). Exact and heuristic methods for Anchor-Robust and Adjustable-Robust RCPSP. *arXiv preprint arXiv:2011.02020*. https://arxiv.org/abs/2011.02020.

Pritsker, A. A. B., Waiters, L. J., and Wolfe, P. M. (1969). Multiproject scheduling with limited resources: A zero-one programming approach. *Management Science*, 16(1):93–108.

Vanhoucke, M. (2006). Scheduling an R&D project with quality-dependent time slots. In Gavrilova, M., Gervasi, O., Tan, C. J. K., Taniar, D., and Laganá, A., editors, *International Conference on Computational Science and Its Applications - ICCSA 2006*, volume 3982 of *Lecture Notes in Computer Science*, pages 621–630. Springer.

Yanıkoğlu, İ., Gorissen, B. L., and den Hertog, D. (2019). A survey of adjustable robust optimization. *European Journal of Operational Research*, 277(3):799–813.

Zhu, G., Bard, J. F., and Yu, G. (2007). A two-stage stochastic programming approach for project planning with uncertain activity durations. *Journal of Scheduling*, 10(3):167–180.

A Non-integrality of the adversarial sub-problem

Here, we show that the constraint matrix of model (13)-(22) is not totally unimodular, contrary to the claim made in Bruni et al. (2017). In the following, we define $\mathcal{E} := E \cup X$. The constraint matrix of (13)-(22) can be written in matrix notation as:

$$C = \begin{array}{c} \begin{matrix} \alpha & w & \delta \end{matrix} \\ \begin{pmatrix} A & 0 & 0 \\ 0 & I_{\mathcal{E}} & -B \\ -I_{\mathcal{E}} & I_{\mathcal{E}} & 0 \\ 0 & 0 & e_V^T \\ 0 & 0 & I_V \end{pmatrix} \end{array} \begin{array}{l} \\ \text{Group 1 (14)-(16)} \\ \text{Group 2 (17)} \\ \text{Group 3 (18)} \\ \text{Group 4 (19)} \\ \text{Group 5 (20)} \end{array}$$

where A is a $|V| \times |\mathcal{E}|$ arc-node incidence matrix, B is a $|\mathcal{E}| \times |V|$ matrix where $B_{(i,j),i} = 1$ for each $(i,j) \in \mathcal{E}$, and 0 otherwise, I_V and $I_{\mathcal{E}}$ are identity matrices of dimension $|V|$ and $|\mathcal{E}|$ respectively, and e_V^T is a $|V| \times 1$ vectors of 1's. The rows of this matrix have been grouped according to the constraints that they represent, and similarly, the columns have been grouped by the variables that they represent.

Ghouila-Houri (1962) showed that a matrix A is totally unimodular if and only if for every subset of rows R, there exists a partition of R into two disjoint subsets R_1 and R_2 such that

$$\sum_{i \in R_1} a_{ij} - \sum_{i \in R_2} a_{ij} \in \{-1, 0, 1\}, \quad \forall j = 1, \ldots, n.$$

Therefore, finding a subset of rows of matrix C for which this condition cannot hold will prove that C is not totally unimodular.

Consider the constraint matrix of the example shown in Figure 3:

$$
C = \begin{array}{c}
\overbrace{}^{\alpha}\ \overbrace{}^{w}\ \overbrace{}^{\delta} \\
\begin{array}{cccc|cccc|cccc}
C1 & C2 & C3 & C4 & C5 & C6 & C7 & C8 & C9 & C10 & C11 & C12
\end{array} \\
\left(\begin{array}{cccc|cccc|cccc}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
-1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots
\end{array}\right)
\end{array}
\begin{array}{l}
R1 \\ \vdots \\ R2 \\ R3 \\ \vdots \\ R4 \\ R5 \\ \vdots \\ \vdots
\end{array}
\begin{array}{l}
\left.\begin{array}{l}\\ \\ \end{array}\right\} \text{Group 1} \\
\left.\begin{array}{l}\\ \\ \\ \end{array}\right\} \text{Group 2} \\
\left.\begin{array}{l}\\ \\ \\ \end{array}\right\} \text{Group 3} \\
\\
\text{Group 4 + Group 5}
\end{array}
$$

Take R to be the subset of rows consisting of the first row of Group 1, and first two rows of Groups 2 and 3. We will refer to these rows as $R1, \ldots, R5$. To ensure that the sum of column $C9$ is in $\{-1, 0, 1\}$, $R2$ and $R3$ must be assigned opposite signs. $R4$ and $R5$ must have opposite signs to $R2$ and $R3$, respectively to ensure that the sum of columns $C5$ and $C6$ are in $\{-1, 0, 1\}$. Then, whatever the choice of sign for $R1$, the sum of column $C1$ and the sum of column $C2$ cannot both be in $\{-1, 0, 1\}$. Hence, there exists a subset of rows for which the Ghouila-Houri characterisation of total unimodularity does not hold, thus proving that matrix C is not totally unimodular, and that model (13)-(22) is not equivalent to its linear relaxation.

B Example of the warm-start procedure

As an example, we apply the warm-start procedure from Section 4.2 to the project given in Figure 1. We set $T^{\max} = 20$ as an arbitrary upper bound on the minimum makespan of the corresponding deterministic project. The earliest-start-times ES_i, $i \in V$ are calculated via a forward-pass of the project precedence network, whilst the latest-finish-times LF_i, $i \in V$ are calculated relative to $T^{\max}$ via a backward-pass of the project precedence network. These values are shown in Figure 7.

The LFT priority-rule heuristic orders the project activities according to increasing latest-finish-times to get $\sigma = (0, 2, 1, 4, 5, 3, 6, 7, 8)$, and then obtains a feasible schedule by scheduling these activities one-at-a-time in the order that they appear in σ using the serial schedule generation scheme (Kolisch, 1996). This results in the following feasible schedule: $S_0 = 0$, $S_1 = 0$, $S_2 = 0$, $S_3 = 6$, $S_4 = 3$, $S_5 = 3$, $S_6 = 11$, $S_7 = 12$, $S_8 = 16$.

From this schedule, the set of feasible y-variables obtained is shown in Table 3. Formulation (34)-(44) is solved with y-variables fixed to these values to obtain feasible values for the S-variables and f-variables. The resulting solution is the feasible warm-start solution we use.

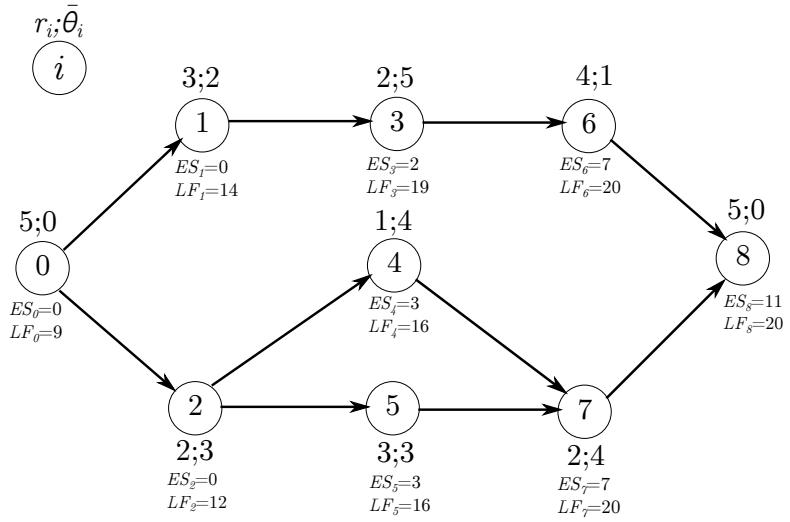


Figure 7: Earliest-start-times and latest-finish-times for the deterministic project corresponding to the example project shown in Figure 1. Latest-start-times have been calculated relative to an arbitrary upper bound on the minimum project makespan given by $T^{\max} = 20$.

		j								
		0	1	2	3	4	5	6	7	8
i	0	1	1	1	1	1	1	1	1	1
	1	0	0	0	1	1	1	1	1	1
	2	0	0	0	1	1	1	1	1	1
	3	0	0	0	0	0	0	1	1	1
	4	0	0	0	0	0	0	1	1	1
	5	0	0	0	1	0	0	1	1	1
	6	0	0	0	0	0	0	0	1	1
	7	0	0	0	0	0	0	0	0	1
	8	0	0	0	0	0	0	0	0	1

Table 3: The set of feasible y-variables corresponding to the solution found by the LFT priority-rule heuristic for the example project in Figure 7. The ij-th element of this table gives the value of variable y_{ij}.