A faster exact method for solving the robust multi-mode resource-constrained project scheduling problem

Matthew Bold*[1] and Marc Goerigk[2]

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

Abstract

This paper presents a mixed-integer linear programming formulation for the multi-mode resource-constrained project scheduling problem with uncertain activity durations. We consider a two-stage robust optimisation approach, where activity modes are selected and resource conflicts are resolved under the uncertain activity durations in a first-stage, which is followed by a second-stage where the schedule is evaluated once the activity durations become known. We aim to find solutions to minimise the worst-case project makespan, whilst assuming that activity durations lie in a budgeted uncertainty set.

Computational experiments show that this easy-to-implement formulation is many times faster to solve than the current Benders' decomposition-based state-of-the-art solution approach for this problem, whilst also being able to solve over 40% more instances to optimality over the same set of benchmarking instances.

Keywords: project scheduling; optimisation under uncertainty; robust optimisation; budgeted uncertainty

1 Introduction

The multi-mode resource-constrained project scheduling problem (MRCPSP) is a generalisation of the widely-studied resource-constrained project scheduling problem (RCPSP) to include multiple processing modes for each activity. The inclusion of these modes allows the modelling of situations in which there is more than one way of executing project activities, with each option having its own duration and resource requirements. The MRCPSP consists of selecting the processing modes and start times for a given set

*Corresponding author, email: m.bold1@lancaster.ac.uk

of activities, subject to a set of precedence constraints and limited resource availability, with the objective of minimising the overall project duration, known as the makespan.

In this paper we consider the MRCPSP under uncertain activity durations and model it using a two-stage adjustable robust optimisation framework. In this setting, a first-stage problem is solved to determine activity mode selections and make activity sequencing decisions to resolve resource conflicts. Following this, the actual activity durations are realised and a complete schedule is computed. The aim of the two-stage adjustable robust MRCPSP is to find a feasible first-stage solution (i.e. mode selection and sequencing decisions) in order to minimise the realised worse-case makespan, as computed in the second-stage. We refer to this problem as the robust MRCPSP.

A number of papers in recent years have applied this two-stage robust optimisation approach to the RCPSP. First to use this approach were Artigues et al. (2013), who presented an iterative scenario-relaxation algorithm for this problem with the objective of minimising the worst-case absolute regret. More recently, Bruni et al. (2017) introduced a Benders'-style decomposition approach for solving the robust RCPSP with the objective of minimising the worst-case project makespan. This work was extended in Bruni et al. (2018), which presented a computational study comparing an additional Benders' decomposition approach against a primal decomposition algorithm. Most recently, Bold and Goerigk (2021) introduced a compact reformulation of the robust RCPSP and presented results which showed the superior computational performance of that formulation over the iterative decomposition-based methods developed in the two preceding papers.

The application of this two-stage robust optimisation approach for the MRCPSP is a very recent development. To the best of our knowledge, the only existing paper to consider this problem is Balouka and Cohen (2021), in which the Benders' decomposition approach introduced by Bruni et al. (2017) for the robust RCPSP has been extended for application to the MRCPSP. Mirroring that extension, this paper adapts the compact formulation developed by Bold and Goerigk (2021) to the MRCPSP, with the aim of achieving similarly superior computational performance over the Benders' solution approach.

Following a formal description of the two-stage robust MRCPSP in Section 2, we outline the proposed compact formulation for this problem in Section 3 and present a strengthened version of the Benders' decomposition solution approach from Balouka and Cohen (2021) in Section 4. Results from a computational comparison of these two approaches are detailed in Section 5.

2 Problem description

A project consists of a set of non-preemptive activities $V = \{0, 1, \ldots, n, n+1\}$, where 0 and $n+1$ denote the dummy source and sink activities respectively. Each activity $i \in V$ has a set of available processing modes given by $M_i = \{1, \ldots, |M_i|\}$. The nominal duration of activity i when executed in mode $m \in M_i$ is given by $\bar{d}_{im}$, whilst its worst-case duration is given by $\bar{d}_{im} + \hat{d}_{im}$, where $\hat{d}_{im}$ is its maximum durational deviation. Each mode for activity i, $m \in M_i$, has an associated renewable resource requirement of r_{imk}

for each $k \in K$, where K is the set of renewable resource types involved in the project. Each renewable resource $k \in K$ has an availability of R_k at each time period in the project horizon. As well as renewable resource requirements, mode m of activity i also has a non-renewable resource requirement of r'_{imk} for each non-renewable resource type $k \in K'$, with each non-renewable resource having an overall availability of R'_k for the entire project horizon. Additionally, the project is subject to a set of strict finish-to-start precedence constraints given by E, where $(i, j) \in E$ enforces that activity i must finish before activity j can begin. These form a project network that can be represented using a directed graph $G(V, E)$. Figure 1 shows an example instance involving five non-dummy activities and a single renewable resource.

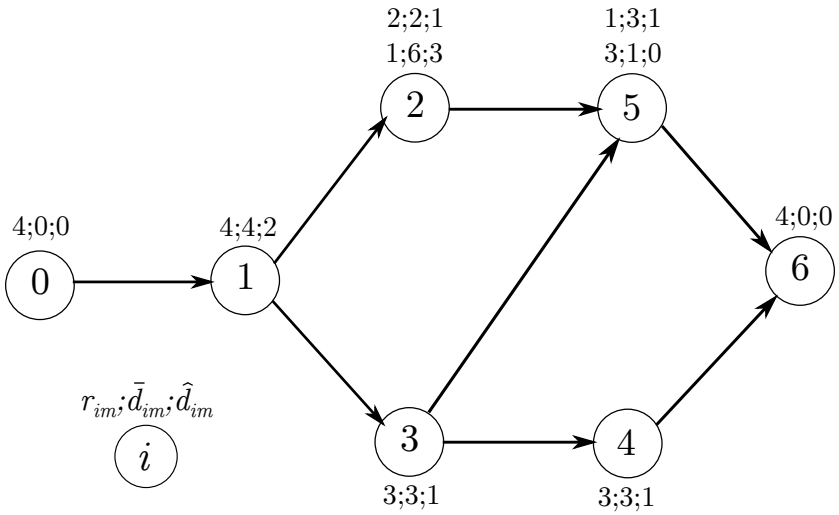


Figure 1: An example instance involving five non-dummy activities and a single renewable resource. Activities 2 and 5 each have two available processing modes, whilst the other activities have only a single mode.

For a given choice of processing modes for each activity $\boldsymbol{m} = (m_1, \ldots, m_n)$, we assume that the activity durations lie somewhere in a budgeted uncertainty set of the form

$$\mathcal{U}_{\boldsymbol{m}}(\Gamma) = \left\{ \boldsymbol{d_m} \in \mathbb{R}_+^{|V|} : d_{im_i} = \bar{d}_{im_i} + \xi_i \hat{d}_{im_i},\ 0 \leq \xi_i \leq 1,\ \forall i \in V,\ \sum_{i \in V} \xi_i \leq \Gamma \right\}.$$

Introduced by Bertsimas and Sim (2004), the motivation of the budgeted uncertainty set is to control the pessimism of the solution by introducing a robustness parameter Γ to limit the number of jobs that can simultaneously achieve their worst-case durations. Observe that when $\Gamma = 0$, each activity takes its nominal duration and the resulting problem is the deterministic MRCPSP. At the other extreme, when $\Gamma = n$, every activity takes its worst-case duration, in which case the uncertainty set becomes equivalent to an interval uncertainty set. In this case the problem can again be solved as a deterministic MRCPSP instance considering only worst-case durations.

For a given set of activity modes $\boldsymbol{m}$, a forbidden set is defined to be any subset of activities $F_{\boldsymbol{m}} \subseteq V$ that are not precedence-related, such that $\sum_{i \in F_{\boldsymbol{m}}} r_{im_i k} > R_k$ for at least one resource $k \in K$. That is, a forbidden set is a collection of activities that cannot be executed in parallel only because of resource limitations. Applying the main representation theorem of Bartusch et al. (1988), for a particular choice of activity modes $\boldsymbol{m}$, a solution to the MRCPSP can be defined by a set of additional precedences $X_{\boldsymbol{m}} \subseteq V^2 \setminus E$ such that the extended precedence network $G(V, E \cup X_{\boldsymbol{m}})$ is acyclic and contains no forbidden sets. Such an extension to the project network is referred to as a *sufficient selection*.

The aim of the two-stage robust MRCPSP is therefore to select set of activity modes and a corresponding sufficient selection in order to minimise the worst-case project makespan. For the case where activity durations lie in a budgeted uncertainty set that we consider in this paper, this problem can be written as

$$\min_{\boldsymbol{m} \in \mathcal{M},\, X_{\boldsymbol{m}} \in \mathcal{X}_{\boldsymbol{m}}} \max_{\boldsymbol{d} \in \mathcal{U}_{\boldsymbol{m}}(\Gamma)} \min S_{n+1} \tag{1}$$

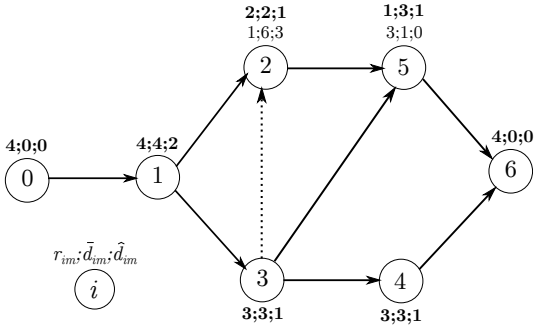
$$S_0 = 0 \tag{2}$$

$$S_j - S_i \geq d_{im_i} \qquad \forall (i,j) \in E \cup X_{\boldsymbol{m}} \tag{3}$$

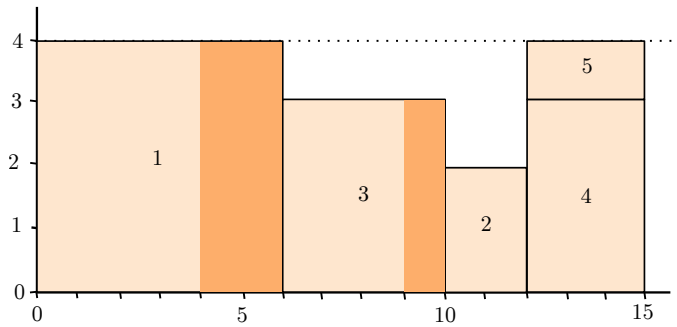
$$S_i \geq 0 \qquad \forall i \in V, \tag{4}$$

where $\mathcal{M} \subseteq \mathbb{N}^n$ represents the set of all possible combinations of activity processing mode selections, and $\mathcal{X}_{\boldsymbol{m}}$ is the set of all possible sufficient selections for the choice of processing modes given by $\boldsymbol{m}$.

An optimal solution to the instance shown in Figure 1 is given by Figure 2a, where the mode choices are shown in bold, and the sufficient selection is given by $\{(3,2)\}$. The worst-case schedule for this optimal solution is shown in Figure 2b, where activities 1 and 3 have been delayed to achieve a worst-case makespan of 15 (note that delaying a combination of activity 1 and any other activity results in the same worst-case makespan).



(a) Activity mode choices are highlighted in bold and sufficient selection is given by dashed arc $\{(3,2)\}$.



(b) Worst-case schedule corresponding to optimal robust solution, where activities 1 and 3 have been delayed.

Figure 2: Optimal solution to the example instance from Figure 1.

3 A compact formulation

We propose solving the robust MRCPSP using an extended version of the mixed-integer programming formulation developed by Bold and Goerigk (2021) for solving the two-stage adjustable robust RCPSP. This formulation combines the first and second-stage problems into a single compact formulation and was shown to significantly outperform the strongest iterative decomposition-based methods for that problem.

This formulation is constructed by recasting the adversarial subproblem of determining the worst-case activity durations for a given set of mode choices and sufficient selection as a longest path problem on an augmented project network. This augmented project network is formed of $\Gamma + 1$ connected copies of the original project network, with each level being used to account for a delay to a single activity in the project. Having reformulated the adversarial subproblem as a longest-path problem, strong duality can be employed to enable its insertion it into a formulation for the first-stage problem, and ultimately arrive at a complete compact formulation for the full problem. Given that the derivation of this formulation is more or less identical to the derivation of the corresponding formulation for the robust RCPSP, we omit it here and instead refer the reader to Bold and Goerigk (2021).

In the resulting compact formulation, y_{ij} variables are used to define a complete set of transitive precedences between the project activities that are formed by the original project precedence constraints and the additional precedences introduced to resolve resource conflicts, i.e. $\{(i,j),\, i,j \in V : y_{ij} = 1\} = T(E \cup X_{\boldsymbol{m}})$, where $T(\cdot)$ is used to denote the transitive closure of a set. Variables x_{im} determine the processing mode for each activity, whilst resource flow variables f_{ijk} track the amount of renewable resource k that is transferred from activity i to activity j upon its completion. The $S_{i\gamma}$ variables are dual variables associated with the longest-path subproblem, which in this formulation are used to track the start time of activities under the worst-case activity durations. Using these variables, the compact formulation for the robust MRCPSP can be written as

$$\min\ S_{n+1,\Gamma} \tag{5}$$

$$\text{s.t. } S_{00} = 0 \tag{6}$$

$$S_{j\gamma} - S_{i\gamma} \geq \bar{d}_{im} x_{im} - N(1 - y_{ij}) \qquad \forall (i,j) \in V^2,\, \forall m \in M_i,\, \gamma = 0, \ldots, \Gamma \tag{7}$$

$$S_{j,\gamma+1} - S_{i\gamma} \geq (\bar{d}_{im} + \hat{d}_{im}) x_{im} - N(1 - y_{ij}) \qquad \forall (i,j) \in V^2,\, m \in M_i,\, \gamma = 0, \ldots, \Gamma - 1 \tag{8}$$

$$y_{ij} = 1 \qquad \forall (i,j) \in E \cup \{(n+1, n+1)\} \tag{9}$$

$$y_{ij} + y_{ji} \leq 1 \qquad \forall i,j \in V,\, i < j \tag{10}$$

$$y_{ij} = y_{ip} + y_{pj} - 1 \qquad \forall i,j,p \in V,\, i \neq j \neq p \tag{11}$$

$$f_{ijk} \leq P_{ijk} y_{ij} \qquad \forall (i,j) \in V^2,\, i \neq n+1,\, j \neq 0,\, \forall k \in K \tag{12}$$

$$\sum_{i \in V \setminus \{n+1\}} f_{ijk} = \sum_{m \in M_j} r_{jmk} x_{jm} \qquad \forall j \in V \setminus \{0\},\, \forall k \in K \tag{13}$$

$$\sum_{j \in V \setminus \{0\}} f_{ijk} = \sum_{m \in M_i} r_{imk} x_{im} \qquad \forall i \in V \setminus \{n+1\}, \forall k \in K \tag{14}$$

$$\sum_{m \in M_i} x_{im} = 1 \qquad \forall i \in V \tag{15}$$

$$\sum_{m \in M_i} r'_{imk} x_{im} \leq R'_k \qquad \forall i \in V, k \in K' \tag{16}$$

$$S_{i\gamma} \geq 0 \qquad \forall i \in V, \gamma \in 0, \ldots, \Gamma \tag{17}$$

$$y_{ij} \in \{0,1\} \qquad \forall (i,j) \in V^2 \tag{18}$$

$$f_{ijk} \geq 0 \qquad \forall (i,j) \in V^2, \forall k \in K \tag{19}$$

$$x_{im} \in \{0,1\} \qquad \forall i \in V, m \in M_i, \tag{20}$$

where $N = \sum_{i \in V} \max_{m \in M_i} (\bar{d}_{im} + \hat{d}_{im})$ is an upper bound on the minimum makespan, and $P_{ijk} = \min\{\max_{m \in M_i} r_{imk}, \max_{m \in M_j} r_{jmk}\}$ is the maximum possible flow of resource k from i into j.

Constraints (6)-(8) are the dual constraints corresponding to the longest-path variables in the adversarial subproblem, and serve to ensure that the activity start times respect the project precedences as well as any delays to activity durations. Constraints (9) capture the original project precedence relationships. Although not required for the correctness of the model, constraints (10) and (11) are additional transitivity constraints that have been included because they were shown in Bold and Goerigk (2021) to provide significant improvements to the computational performance of the model. Constraints (12)-(14) are resource flow constraints which ensure that the transfer of resources between activities follows precedence constraints and that resources are conserved as they flow into and out of each activity in the network. Constraints (15) allows only one processing mode to be selected for each activity. Finally, non-renewable resource constraints are enforced by (16).

With polynomially many variables and constraints this formulation can be implemented straightforwardly using standard mathematical optimisation software such as Gurobi or CPLEX.

4 A Benders' decomposition approach

In this section we outline an alternative approach to solving the robust MRCPSP based on Benders' decomposition. A Benders'-type approach for solving the robust MRCPSP was first presented in Balouka and Cohen (2021), and as mentioned previously, this itself was based on the approach for solving the robust RCPSP used in Bruni et al. (2017).

The main idea behind this approach is to decompose the full problem into its two stages: 1. a master problem that determines activity processing modes and resolves resource conflicts, and 2. a subproblem that takes the solution from the master problem and evaluates it by finding the worst-case makespan for the resulting network by solving a longest path problem. Since the master problem does not account for all the problem uncertainty at once, its solution forms a lower bound to the optimal objective value

of the original problem. Meanwhile, the solution to the subproblem is feasible to the original problem, and therefore provides an upper bound. These two problems are solved iteratively, with optimality cuts being added to the master problem each iteration based on the solution to the subproblem until the lower and upper bounds meet.

After initially implementing the Benders' algorithm as it is presented in Balouka and Cohen (2021), it was discovered that its performance could be strengthened by replacing the master problem formulation used in that implementation with a stripped-down version of the compact formulation (5)-(20) in which the uncertainty is removed. In this section we present our strengthened implementation of the Benders' approach for solving the robust MRCPSP. A comparison of these two Benders' implementations is included in the results in Section 5. For the details of the original implementation of the Benders' decomposition algorithm for the robust MRCPSP, see Balouka and Cohen (2021).

4.1 The master problem

The role of the master problem is to determine a choice of activity processing modes $\boldsymbol{m}$, as well as a sufficient selection $X_{\boldsymbol{m}}$ to resolve any resulting resource conflicts. This problem is solved without the direct consideration of the uncertain activity durations, and instead, uncertainty is accounted for in the subproblem and communicated back to the master problem with the use of optimality cuts. Hence, to remove the uncertainty from the model, (5)-(20) is modified by considering only a single level of the augmented project network. Using this formulation, the master problem can be written as

$$\min \ \eta \tag{21}$$

$$\text{s.t. } \eta \geq S_{n+1} \tag{22}$$

$$S_0 = 0 \tag{23}$$

$$S_j - S_i \geq \bar{d}_{im} x_{im} - N(1 - y_{ij}) \qquad \forall (i,j) \in V^2, \forall m \in M_i \tag{24}$$

$$\eta \geq \lambda(x, y, M^{*\ell}) \qquad \forall \ell = 1, \ldots, t-1 \tag{25}$$

$$(9)\text{-}(16), (18)\text{-}(20) \tag{26}$$

$$S_i \geq 0 \qquad \forall i \in V, \tag{27}$$

where (25) are the optimality cuts generated from the solutions of the resulting subproblems at each of the previous iterations.

The optimal objective value of the master problem (21)-(27), denoted by η^*, forms a lower bound to the original robust MRCPSP problem. Note that for the first iteration, the solution to the master problem corresponds to an optimal solution to the MRCPSP with nominal activity durations.

4.2 The subproblem

The subproblem at iteration t computes the worst-case makespan for the mode choices and sufficient selection from the master problem at iteration t, denoted by $\boldsymbol{m}^{*t}$ and $X_{\boldsymbol{m}^{*t}}$ respectively. This is done by solving a longest-path problem through the extended

project network $T(E \cup X_{\boldsymbol{m}^{*t}})$, in which the activity durations (i.e. arc lengths) can be chosen from the budgeted uncertainty set $\mathcal{U}_{\boldsymbol{m}^{*t}}(\Gamma)$. This problem can be formulated as

$$V^{*t} = \max \sum_{(i,j)\in T(E\cup X_{\boldsymbol{m}^{*t}})} \bar{d}_{im_i^{*t}}\alpha_{ij} + \hat{d}_{im_i^{*t}}w_{ij} \tag{28}$$

$$\sum_{(i,n+1)\in T(E\cup X_{\boldsymbol{m}^{*t}})} \alpha_{i,n+1} = 1 \tag{29}$$

$$\sum_{(0,i)\in T(E\cup X_{\boldsymbol{m}^{*t}})} \alpha_{0,i} = 1 \tag{30}$$

$$\sum_{(i,j)\in T(E\cup X_{\boldsymbol{m}^{*t}})} \alpha_{ij} - \sum_{(j,i)\in T(E\cup X_{\boldsymbol{m}^{*t}})} \alpha_{ji} = 0 \qquad \forall i \in V \setminus \{0, n+1\} \tag{31}$$

$$w_{ij} \leq \xi_i \qquad \forall (i,j) \in T(E \cup X_{\boldsymbol{m}^{*t}}) \tag{32}$$

$$w_{ij} \leq \alpha_{ij} \qquad \forall (i,j) \in T(E \cup X_{\boldsymbol{m}^{*t}}) \tag{33}$$

$$\sum_{i\in V} \xi_i \leq \Gamma \tag{34}$$

$$\alpha_{ij} \in \{0,1\} \qquad \forall (i,j) \in T(E \cup X_{\boldsymbol{m}^{*t}}) \tag{35}$$

$$w_{ij} \geq 0 \qquad \forall (i,j) \in T(E \cup X_{\boldsymbol{m}^{*t}}) \tag{36}$$

$$0 \leq \xi_i \leq 1 \qquad \forall i \in V. \tag{37}$$

Variables α_{ij} define the longest path through the network, which we denote as $\pi^{*t} = \{(i,j), i,j \in V : \alpha_{ij} = 1\}$, and has length V^{*t}. Variables ξ_i are the activity delay variables, whilst w_{ij} are used to linearise the formulation. Note that this formulation of the subproblem is identical to the one implemented by Balouka and Cohen (2021).

4.3 Optimality cuts

Following the solution to the subproblem, a valid cut is generated and added to the master problem for the next iteration. This cut simply forces an alternative solution in the master problem if it is to achieve a better objective value in the next iteration.

Given the current best lower bound for the original problem, LB, the mode selection from the master problem, $\boldsymbol{m}^{*t}$, and the optimal objective value of the subproblem, V^{*t}, the cut at iteration t can be written as

$$\eta \geq (V^{*t} - LB) \cdot \sum_{(i,j)\in\pi^{*t}} \Big(1/3(y_{ij} + x_{i,m_i^{*t}} + x_{j,m_j^{*t}}) - (3 - y_{ij} - x_{i,m_i^{*t}} - x_{j,m_j^{*t}})\Big) - (V^{*t} - LB) \cdot (|\pi^{*t}| - 1) + LB. \tag{38}$$

Balouka and Cohen (2021) show that this constraint is a valid optimality cut for the problem, and that the number of these cuts that need to be added to the master problem before finding an optimal solution is finite.

An overview of the implementation of the Benders' solution approach outlined in this section is presented in Algorithm 1.

Algorithm 1 Benders' decomposition algorithm.

Initialise: Set $LB = -\infty$, $UB = +\infty$ and $t = 1$.
while $UB > LB$ **do**
    Solve master problem (21)-(27)
        Get objective value η^{*t}, processing modes $\boldsymbol{m^{*t}}$ and precedences $\boldsymbol{y_{m^{*t}}}$
        If $\eta^{*t} > LB$, update $LB \leftarrow \eta^{*t}$
    Solve subproblem (28)-(37)
        Get objective value V^{*t} and longest path π^{*t}
        If $V^{*t} < UB$, update $UB \leftarrow V^{*t}$
    Add cut (38) to master problem
    Update $t \leftarrow t + 1$
end while
return UB

4.4 Example

To demonstrate its implementation, we use the Benders' decomposition approach to solve the example shown in Figure 1 with $R_1 = 4$ and $\Gamma = 2$. The algorithm solves this instance in 6 iterations, and the solution information from the master and subproblem at each of these iterations is shown in Table 1.

The first iteration of the master problem solves the nominal instance, assuming no delays to the activity durations. This solution opts to schedule activities 2 and 3 in parallel, which can be achieved by setting $m_2 = 2$, and the sufficient selection for is given by $\{(4, 5)\}$. This first solution is evaluated in the subproblem to have a worst-case makespan of 16. In the second iteration, having added the first optimality cut, the master problem finds the solution shown in Figure 2a. By calculating the schedule shown in Figure 2b, the subproblem evaluates the objective value of this solution to be 15. Although this solution is optimal, the algorithm requires a further four iterations to prove that this is the case. Note that there is no need to solve the subproblem in the final iteration since after solving the master problem and updating UB, we have that $LB = UB$.

t	LB	UB	η^{*t}	V^{*t}	$\boldsymbol{m}^{*t}$	π^{*t}
1	11	16	11	16	1,2,1,1,2	0,1,2,5,6
2	12	15	12	15	1,1,1,1,1	0,1,3,2,4,6
3	12	15	12	15	1,1,1,1,1	0,1,2,3,4,6
4	13	15	13	18	1,2,1,1,1	0,1,2,5,6
5	13	15	13	16	1,1,1,2,1	0,1,3,4,2,5,6
6	15	15	15	-	1,1,1,1,1	-

Table 1: Solution information at each of the six iterations of the Benders' algorithm required to solve example instance in Figure 1.

5 Computational experiments and results

In this section we compare results from using the compact formulation and the Benders' decomposition approach to solve uncertain MRCPSP instances. A complete set of the raw results used to generate the tables and plots presented here, in addition to the source code used to implement these experiments, can be found at https://github.com/boldm1/robust-mrcpsp.

The instances used in this computational study have been created from the deterministic $j10$, and $j20$ MRCPSP instances from the PSPLIB (Kolisch and Sprecher (1997), https://www.om-db.wi.tum.de/psplib/). The $j10$ set contains a total of 536 instances each involving 10 activities, and the $j20$ set contains a total of 554 instances each involving 20 activities. We introduce uncertainty into these deterministic instances by setting the maximum durational deviation for each activity to be $\hat{d}_{im} = \lfloor 0.7 \times \bar{d}_{im} \rfloor$ for each mode $m \in M_i$. These uncertain instances have then been solved using both methods for a range of robustness levels Γ. In particular we solve the $j10$ instances for $\Gamma \in \{0, 3, 5, 7\}$ and the $j20$ instances for $\Gamma \in \{0, 5, 10, 15\}$.

Both the compact reformulation and Benders' decomposition approach have been implemented in Python 3.9.2 and solved using Gurobi 9.0.1 running on a single core of a 2.30 GHz Intel Xeon CPU. A time limit of 2 hours per instance was imposed on each of the solution methods.

We begin by observing the effect of the robustness parameter Γ on the average optimal objective values shown in Tables 2a and 2b. The values in these tables have been computed only over the instances for which an optimal solution was found for all the values of Γ (i.e. all 536 instances in the $j10$ set, and 477 instances of the $j20$ set). As we would expect, the solution cost increases in a concave manner as Γ increases.

Γ	0	3	5	7
obj.	16.84	25.34	26.35	26.46

(a) $j10$

Γ	0	5	10	15
obj.	24.61	37.89	38.69	38.69

(b) $j20$

Table 2: Average optimal objective values across instances in the $j10$ and $j20$ instance sets for different values of Γ.

Table 3 reports the percentage of instances solved to optimality (*%sol.*), the average percentage optimality gap over instances for which a feasible solution was found (*gap*), and the average solution time in seconds (*sol. time*) for the different choices of Γ across the two instance sets, for both the Benders' decomposition approach and the compact formulation. Note that if no optimal solution was found within the time limit by one of the solution methods, the solution time was recorded as the time limit value of 7200 seconds. Additionally, for the Benders' approach, the average number of (completed) iterations (*it.*) and the average time per (completed) iteration (*it. time*) are also reported. The average of the reported values across the different values for Γ are also given for each instance set.

Firstly, the results in Table 3 show that for both methods, the instances tend to increase in difficulty as Γ increases. We can also see that for the nominal problems, i.e. when $\Gamma = 0$, the Benders' approach has a slight edge on the compact formulation. This is what we would expect given the relationship between the Benders' master problem formulation and the compact formulation. However, for all the non-zero values of Γ across both instance sets, the compact formulation performs significantly better than the Benders' approach, solving a greater proportion of instances with dramatically reduced computation times.

		Benders'					Compact formulation		
	Γ	*%sol.*	*gap*	*its.*	*it. time*	*sol. time*	*%sol.*	*gap*	*sol. time*
$j10$	0	100.0	0.00	1	0.96	1.1	100.0	0.00	1.8
	3	81.7	3.13	67	3.76	1551.4	100.0	0.00	5.0
	5	79.7	4.20	76	3.89	1688.4	100.0	0.00	4.6
	7	79.5	4.52	78	3.98	1701.1	100.0	0.00	6.5
		85.2	2.96	56	3.15	1235.5	100.0	0.00	4.47
$j20$	0	89.9	0.00	1	171.08	885.5	89.5	1.89	925.1
	5	50.9	10.21	220	133.32	4169.3	88.6	1.87	1041.5
	10	50.4	10.99	225	138.49	4244.6	87.9	2.58	1118.5
	15	49.5	11.21	216	146.26	4300.3	86.3	2.91	1260.6
		60.2	8.10	165	147.28	3399.9	88.1	2.31	1086.4

Table 3: Comparison of the Benders' decomposition approach and the compact reformulation across the $j10$ and $j20$ instance sets and for different values of Γ.

Figures 3 and 4 show performance profiles and optimality gap plots for the compact formulation and both implementations of the Benders' decomposition solution approach (Algorithm 1 and Balouka and Cohen (2021)), across the $j10$ and $j20$ sets respectively. Note that only the results for the instances with non-zero values for Γ were used to generate these plots, resulting in a total of 1608 $j10$ instances and 1664 $j20$ instances.

Performance profiles (Dolan and Moré (2002)) provide a visual comparison of competing solution approaches using their *performance ratios*. The performance ratio of algorithm $a \in \mathcal{A}$ for instance $i \in \mathcal{I}$ is defined to be

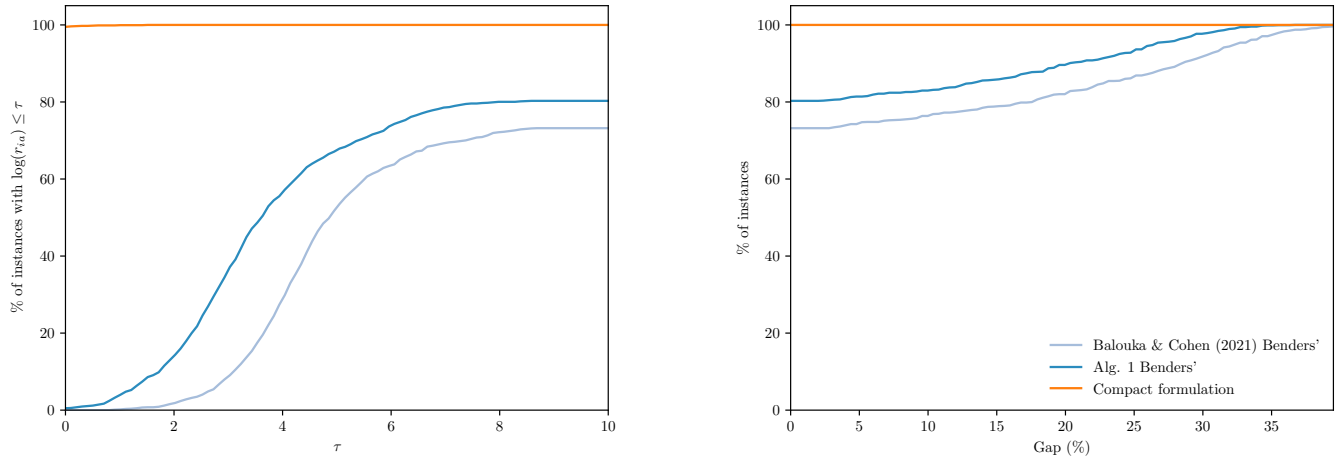
$$r_{ia} = \frac{t_{ia}}{\min_{a \in \mathcal{A}} t_{ia}},$$

where t_{ia} is the time required to solve instance i using algorithm a. If method a is unable to solve an instance i within the 2-hour time limit, then $r_{ia} = R$, where $R \geq \max_{i,m} r_{im}$. The performance profiles in Figures 3a and 4a show the percentage of instances that have a performance ratio within a factor of τ of the best algorithm. These are plotted using a log scale for clarity. The y-intercepts of each method give the percentage of instances for which that method found the optimal solution in the fastest time, whereas the right-most value gives the overall percentage of instances that were solved to optimality within the time limit by that method.

These performance profiles show that whenever an instance was able to be solved to optimality by either solution approach, the compact formulation was always the faster method. If we specifically compare the solution times of the compact formulation with the stronger of the two Benders' implementations, across the instances in $j10$ for which both methods found the optimal solution, the average solution time of the compact formulation (1.4s) was almost 200 times faster than for the Benders' approach (283.5s). And when both methods found the optimal solution for instances in the $j20$ set, the average solution time of the compact formulation (13.7s) was almost 100 times faster than for the Benders' approach (1304.5s).

Figures 3b and 4b show the percentage of instances solved by each method to within a given optimality gap within the 2-hour time limit. These plots serve as a continuation of the performance profiles beyond just the instances that were solved to optimality. Summarising the data from these plots across both the $j10$ and $j20$ instance sets, the Balouka and Cohen (2021) Benders' implementation solves 55.1% of instances to optimality and finds feasible solutions for a further 35.6% of instances. Our Benders' implementation solves 65.0% of instances and finds feasible solution solutions for a further 30.0% of instances. The compact formulation on the other hand finds a feasible solution to every instance, solving 93.1% these to optimality.

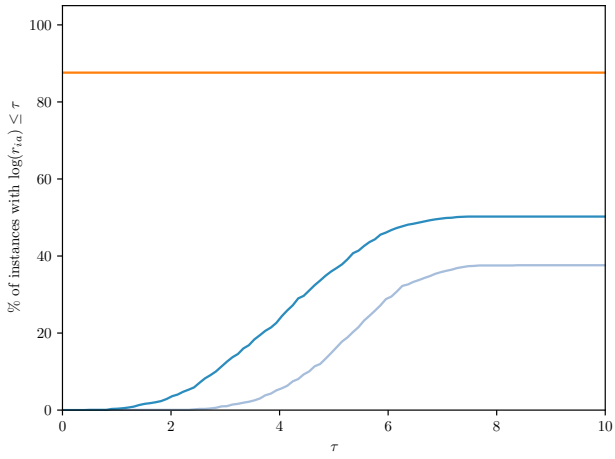
The results presented here show the clear improvements to the Benders' algorithm afforded by amending the master problem to use the formulation (21)-(27). More significantly however, these results demonstrate the complete dominance of the compact formulation over both Benders' methods.



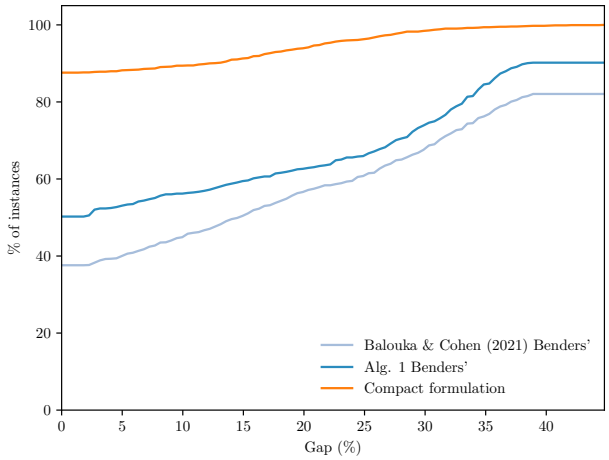
(a) Performance profiles showing percentage of instances with a performance ratio of within τ. Plotted on a log scale.

(b) Percentage of instances solved to within given optimality gap within the two-hour time limit.

Figure 3: Comparison of the Benders' approaches and compact formulation over instances in the $j10$ set.



(a) Performance profile showing percentage of instances with a performance ratio of within τ. Plotted on a log scale.



(b) Percentage of instances solved to within given optimality gap within the two-hour time limit.

Figure 4: Comparison of the Benders' approaches and compact formulation over instances in the $j20$ set.

6 Conclusions

The work presented in this paper extends the compact mixed-integer programming formulation first introduced by Bold and Goerigk (2021), for application to the two-stage adjustable robust MRCPSP with uncertain activity durations. The computational performance of this formulation has been examined over a total of 3270 uncertain MRCPSP instances of varying size and difficulty, and compared against an improved version of the current state-of-the-art for solving this problem, based on a Benders' decomposition approach. The improved Benders' approach is the result of replacing the original master problem with a new formulation based on a simplified version of the compact formulation we present for the full problem.

Results presented in Section 5 show that the compact formulation completely dominates the enhanced Benders' approach, solving over 43% more instances to optimality, and doing so with dramatically reduced computation times. In addition to these strong computational improvements, the proposed compact formulation has the significant added benefit of being simpler to implement than the iterative Benders' approach.

Despite these strong results, the instances used in these experiments contain only up to 20 activities. To enable the solving of larger scale instances, the development of heuristic solution approaches should be a primary focus of future research on this problem.

Acknowledgements

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

References

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

Balouka, N. and Cohen, I. (2021). A robust optimization approach for the multi-mode resource-constrained project scheduling problem. *European Journal of Operational Research*, 291(2):457–470.

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.

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

Bold, M. and Goerigk, M. (2021). A compact reformulation of the two-stage robust resource-constrained project scheduling problem. *Computers & Operations Research*, 130:105232.

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.

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.