An adjustable robust optimization model for the resource-constrained project scheduling problem with uncertain activity durations

M.E. Bruni, L. Di Puglia Pugliese, P. Beraldi, F. Guerriero

www.elsevier.com/locate/omega

PII: S0305-0483(16)30659-4
DOI: http://dx.doi.org/10.1016/j.omega.2016.09.009
Reference: OME1716

To appear in: *Omega*

Received date: 6 March 2016
Revised date: 18 September 2016
Accepted date: 20 September 2016

Cite this article as: M.E. Bruni, L. Di Puglia Pugliese, P. Beraldi and F Guerriero, An adjustable robust optimization model for the resource-constraine project scheduling problem with uncertain activity durations, *Omega* http://dx.doi.org/10.1016/j.omega.2016.09.009

An adjustable robust optimization model for the resource-constrained project scheduling problem with uncertain activity durations

Bruni M.E.[a] Di Puglia Pugliese L.[a, *] Beraldi P.[a]
Guerriero F.[a]

[a]Department of Mechanical, Energy and Management Engineering, University of Calabria, 87036, Rende, Italy

Abstract

This paper addresses the resource-constrained project scheduling problem with uncertain activity durations. An adaptive robust optimization model is proposed to derive the resource allocation decisions that minimize the worst-case makespan, under general polyhedral uncertainty sets. The properties of the model are analyzed, assuming that the activity durations are subject to interval uncertainty where the level of robustness is controlled by a protection factor related to the risk aversion of the decision maker. A general decomposition approach is proposed to solve the robust counterpart of the resource-constrained project scheduling problem, further tailored to address the uncertainty set with the protection factor. An extensive computational study is presented on benchmark instances adapted from the PSPLIB.

Keywords: Project scheduling, Resource constraints, Robust optimization, Benders decomposition.

1 Introduction

The resource-constrained project scheduling problem (RCPSP) consists in sequencing and scheduling project activities usually related by precedence and resource constraints involving renewable scarce resources. As comprehensively investigated in the literature, the RCPSP is an outstanding and challenging problem both in practice, since it arises in many important application fields (construction industry [20, 40], rolling ingots production [55, 57], to mention a few), and in theory.

*Corresponding author: luigi.dipugliapugliese@unical.it

The majority of the scientific contributions considers the model parameters deterministic ([41]). However, in the last years a frank acknowledgment of the uncertainties characterizing the project environment and a growing attention on the project execution, have highlighted the need of incorporating the uncertainty in problem parameters as an inevitable feature of the decision-making process [2, 52]. In fact, due to employees' absenteeism, delays in materials supply, bad weather conditions and many other uncontrollable factors, some project activities may last longer than expected, threatening the operational viability of the planned schedule.

To address these challenges, a flourishing stream of literature has focused on the RCPSP under uncertainty, where the activity durations are assumed uncertain [21, 22]. Two different approaches can be used depending on the genuine interpretation of the RCPSP under uncertainty and the way this uncertainty is tackled. The first approach assumes that uncertainty is represented by random variables with known distribution functions and interprets the RCPSP as a stochastic dynamic optimization problem, where decisions are made each time new information becomes available.

The second approach has mainly dealt with the development of effective and efficient proactive and reactive scheduling procedures. Proactive scheduling aims at generating baseline schedules that incorporate some protection against possible disruptions, whereas reactive scheduling procedures can be invoked during the execution of the project, to repair the baseline schedule by deviating as little as possible from the original one.

Scant attention has been devoted to robust optimization approaches for the RCPSP. The robust optimization methodology ([15, 16, 31]) was first introduced for linear programming problems ([13]) and then extended for mixed-integer linear programming problems. The approach produces solutions that are feasible for all the realizations of the parameters lying within the uncertainty set. The success of this paradigm, in a broad variety of application areas, is mainly due to the fact that this approach is the only reasonable alternative when the distributional information is not readily available. Moreover, it is easy to understand intuitively and it yields computational tractable mathematical programming problems. Originally designed to handle static problems with uncertain parameters, robust optimization was recently extended into a dynamic setting ([14, 23]). In particular, part of the variables must be determined before the realization of the uncertainty, whilst other variables can be adjusted to the realization of the uncertain parameters, hence offering increased flexibility. This framework overcomes the conservativeness of early static robust approaches producing significantly less conservative solutions than the static case and yielding better objective values.

Following this stream, this paper presents an adjustable robust formulation for the RCPSP where in the first stage, sequencing decisions are taken, concerning the order of the project activities. In the second stage, schedul-

ing decisions are made, allowing the activity starting times to depend on the activity delays realized.

The contributions of this work are manifold. A novel robust optimization model for the RCPSP under polyhedral duration uncertainty is presented. A tailored solution approach equipped with problem-specific cuts and lower bound inequalities is provided. Further, a polynomially solvable case is identified with a specific uncertainty structure, which provides an intuitive interpretation for decision makers who might flexibly adjust the level of risk aversion. The resulting problem is then solved with an enhanced approach that exploits the specific structure of the uncertainty set. The results, collected on instances adapted from the PSPLIB, show that the behavior of the proposed solution approach is strongly related to the characteristics of the instances and that not all the cases can be solved within the time limit of 20 minutes. In particular, 15 out of the 48 problem classes of the PSPLIB are computationally demanding, for any risk aversion level. The average computational effort, for the solved instances is, on average, around 173.35 seconds whereas the gap for the unsolved instances is around 38%. The results confirm that robust variants of the RCPSP are computationally demanding, in line with the results presented in [4].

The remainder of the paper is organized as follows. First, we survey the literature in Section 2. A formal definition of the robust RCPSP is given in Section 3. A tailored decomposition approach is presented in Section 4, whereas Section 5 focuses on a specific uncertain set, where the activity durations are subject to interval uncertainty and the level of robustness is controlled by a protection factor. Section 6 discusses the computational results obtained on a set of benchmark problems. Finally, some conclusions are drawn in Section 7. A detailed accounting of the numerical results is given in Appendix A. Formal proofs of theorems are reported in Appendix B.

2 Related literature

The RCPSP has been widely investigated in the academic literature, but the issue of the incorporation of uncertainty has received a growing research attention only in the last fifteen years. Two alternative approaches have been proposed to handle the problem.

In the first one, the duration of each activity is assumed to be a random variable which follows known probability distribution functions and the scheduling problem is viewed as a multistage decision process, where decisions are made each time new information becomes available.

Scheduling is done by policies that define, at decision point, appropriate actions concerning the choice of activities that should be executed next and the objective is typically the minimization of the expected makespan.

First introduced in [56], priority policies schedule activities according to a given priority order. While easy to define and implement, they have been abandoned since the so-called Graham anomalies may occur ([32]).

Preselective policies were then introduced by Igelmund and Radermacher [38]. Germane to these policies is the concept of minimal forbidden set defined as the minimum cardinality set of activities, without precedence constraints, whose total resource consumption exceeds the resource availability. A preselective policy defines for each minimal forbidden set a (preselected) activity to be postponed in order to solve potential resource conflicts. The partial order of precedence constraints induces a digraph which has a node for each activity and for each waiting condition related to preselected activities.

A combination of priority and preselective policies has been proposed by Möhring and Stork [51], who studied the so-called linear preselective policies where an activity is selected to be delayed and the choice respects the order imposed by a priority list. Since each linear preselective policy is also a preselective policy, linear preselective policies inherit the analytic properties of being monotone and continuous. These properties were further exploited in [59] to develop a branch-and-bound procedure equipped with dominance rules and different branching schemes to efficiently compute an optimal preselective policy.

More recently, a new class of policies, called preprocessor policies, have been proposed in [5]. A-priori sequencing decisions resolve some, but not necessarily all, resource conflicts in a preprocessing phase, while the remaining conflicts are dynamically resolved during project execution. In particular, a preprocessor policy is defined by a set of activity pairs (which adds extra precedence relations between activities) and an ordered list used by a priority based policy to solve conflicts during project execution.

Policies have also been used for determining predictive activity starting times, with the objective of minimizing costs related to positive and negative deviations of actual starting times, from the predicted ones, and to penalties/bonuses associated with late/early project completion. Deblaere et al. [24] proposed a methodology for the determination of a project execution policy and a vector of predictive activity starting times. Both expected activity starting time deviations and penalties or bonuses associated with late or early project completion are minimized in objective function. For solving the problem, the authors proposed a combination of four descent procedures that heavily rely on simulation for the evaluation of the objective function.

Since the stochastic RCPSP is challenging from both a theoretical and computational point of view, in the literature it has often been solved by means of tailored heuristic approaches. Golenko et al. [30] presented a heuristic algorithm where resource conflicts are resolved by a zero-one integer programming problem. The problem aims at maximizing the total contribution of the accepted activities to the expected project duration,

where such contribution is defined as the product of the average duration of the activities and their probability of being on the critical path, calculated via simulation.

Starting from the concept of critical chain introduced by Goldratt [29], Rabbani et al. [54] presented a new heuristic implementing backward pass scheduling for feeding-in resources, with the objective of minimizing the expected project duration and its variance. Similarly to the work of [30], the solution of a zero-one integer programming approach is suggested to allocate the resources, considering that the activities with the greatest probability to be on the critical chain and the greatest correlation with the project variance are fed-in first. Tsai and Gemmil [61] proposed a tabu search based heuristic, which uses multiple tabu lists, randomized short-term memory, and multi start diversification mechanism. Later on, Ballestín [7] developed regret-based biased random sampling procedures and embedded them into a genetic algorithm. A GRASP-heuristic able to produce high-quality solutions for multiple possible objective functions is proposed [8]. Bruni et al. [19] presented a chance-constrained based heuristic to build baseline schedules with minimum makespan able to absorb dynamic variations of activity durations.

Within the stochastic programming context, a two-stage integer linear stochastic model has been proposed in [66]. Target times are determined in the first stage followed by the development of a detailed project schedule in the second stage. The two-stage stochastic model aims at minimizing the cost of project completion and expected penalty associated with starting time deviations from the targets. Only one non-renewable resource (the budget) is constrained in the model.

In the second approach, some knowledge of the uncertainty (not necessarily in the form of probability distribution functions) is incorporated in the decision-making process. Proactive scheduling procedures are then used to generate schedules that are in some sense robust (i.e. insensitive) to future adverse events, and reactive scheduling procedures are designed to be invoked during the execution of the project, when disruptions occur. The objective is to build schedules that, under uncertainty, deviate as little as possible from the baseline schedule. Within this research stream, and when abstraction of resource usage is made, Herroelen and Leus [33] developed mathematical programming models for the generation of stable baseline schedules in a project environment without resource consumption. They minimized the expected weighted sum of the absolute deviations between the planned and the actually realized activity starting times when exactly one activity is expected to be delayed during the project execution. Tavares et al. [60] studied the risk of a project as a function of the uncertainty of the duration and the cost of each activity.

When resources come into play, Leus and Herroelen [49], assuming the availability of a feasible baseline schedule, proposed exact and approximate

formulations of the robust resource allocation problem, and a branch-and-bound algorithm for their solution. In the same stream, the problem of minimizing schedule instability, defined as the expected weighted deviation between the computed and the realized schedules, is considered in Lambrechts et al. [45, 46, 47]. Instead of generating one baseline schedule, Fu et al. [26] built a partial order schedule [53], that represents a set of feasible schedules and designed chaining heuristics for improving the robustness. In order to obtain a precedence and resource-feasible schedule, in [33] the resource flow-dependent float factor heuristic which uses information coming from the resource flow network in the calculation of the so-called activity dependent float factor. In [62, 63] the above mentioned heuristic is modified with the aim of preventing resource conflicts. In [64] multiple algorithms are introduced to include time buffers in a given schedule, while a predefined project due date remains respected. The virtual activity duration extension heuristic, presented in [64], relies on the standard deviation of the activity duration to compute a modified duration, whereas the starting time criticality heuristic tries to combine information on activity weights and on the probability that an activity cannot be started at its scheduled starting time. A new procedure for generating a proactive baseline schedule, based on the use of chance constraints, is presented in [44]. A branch-and-cut method is also provided for solving a sample average approximation of the resulting problem. For an extensive review of research in this field, the reader is referred to [25, 34, 35].

Even if related to some extent to our paper, the above mentioned literature is not in the realm of robust optimization, which is the aim of the paper. To our knowledge, the only paper presenting a robust optimization model for the RCPSP is [4]. Assuming that scenarios represent different realizations of the activity durations, a minimax absolute-regret problem is proposed to find a schedule that minimizes the maximum absolute difference between the makespan obtained by the robust solution and the scenario dependent optimal solutions. To solve the problem, the authors proposed both exact and heuristic methods and showed that, within a time limit of 30 minutes, 52 out of 120 randomly generated tests with 20 activities and 9 out of 120 with 30 activities could be exactly solved, whereas the heuristic is able to produce good quality solutions in less time.

3 Problem Definition

The deterministic RCPSP can be described as follows. Let $G = (V, E)$ be a directed precedence graph defined over the node set $V = \{0, \ldots, n+1\}$. The dummy nodes 0 and $n + 1 \in V$ represent the start and end of the project, respectively. Each other node $i \in V$ corresponds to an activity of the project to be performed in a deterministic time duration $d_i \in \mathbb{Z}_+$.

The set E collects the arcs representing the precedence relationships among the activities. While it is processed, each non-dummy activity $i \in V$ requires an amount $r_{ik} \in \mathbb{Z}_{\geq 0}$ of resource k from a given set of resource types K.

A solution to the RCPSP is a vector of non-negative starting times $S_0, \ldots, S_{n+1}$ (referred to as schedule) satisfying the precedence constraints induced by G and the resource constraints related to the presence of a finite capacity R_k for each resource $k \in K$. In particular, a schedule is resource feasible if, at any time and for each resource type, the amount of consumed resource is less than its availability.

Let $F \subseteq V$ be any subset of activities without precedence relations between them such that $\sum_{i \in F} r_{ik} > R_k$, for at least one $k \in K$. The set F is called forbidden since its activities cannot be executed in parallel, because they generate a resource conflict.

A minimal forbidden set is a forbidden set such that each of its subsets is not a forbidden set. Let $\mathcal{F}$ be the set of minimal forbidden sets. Any resource conflict can be removed by adding a set of extra precedence constraints to the original precedence graph in such a way that the makespan can be determined by applying an early start policy on an extended graph.

Formally, following the Main Representation Theorem of Bartusch et al. [9], the solution of the RCPSP can be reduced to the optimal selection of the set $X \subseteq (V \times V) \setminus E$ of extra precedences such that the extended graph $G'(V, (E \cup X))$ is acyclic and $\mathcal{F}(T(E \cup X)) = \emptyset$, where $T(E \cup X)$ is the transitive closure of an extended set of precedence constraints. In line with Balas [6], we can call the set X a sufficient selection. Once a selection has been defined, and the extended set of precedence relations induced by X is added, we can ignore the resource constraints and the makespan can be simply computed by solving a critical-path problem on a graph parameterized by X.

It should be noted that it is often possible to make different selections, for the same schedule. Whilst this is irrelevant in the context of deterministic scheduling, it becomes a compelling issue when uncertainty is explicitly considered since this choice impacts on the performance of the schedule.

Let us consider the network of Figure 1 representing a project with 5 activities; the parameters d_i and $\hat{d}_i$ are the deterministic duration and the possible delay of activity i, respectively.

Now, let us consider two different sufficient selections, i.e., $X_1 = \{(3, 5)\}$ (represented with a dashed edge) and $X_2 = \{(3, 4)\}$ (represented with a dotted edge). For both selections, the deterministic makespan is equal to 18 and corresponds to the critical path composed by the activities 0, 1, 4, 6 (in bold). When we consider the delayed network, the makespan becomes 22 for the selection X_1, and 30 for X_2, respectively. This example highlights the importance of making the right resource allocation decisions under uncertainty. Since resources cannot be easily transferred between activities

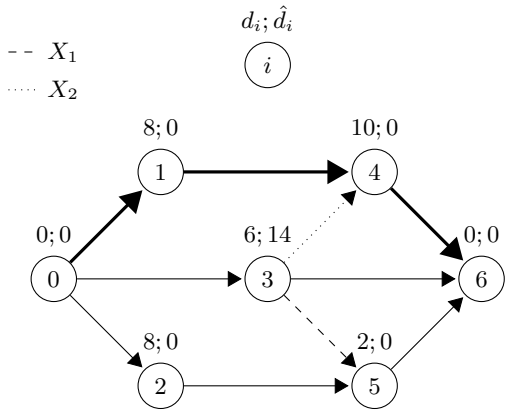

Figure 1: Network project.

at short notice, especially in multi-project environments, it is evident that these decisions should be taken in advance.

On the other hand, whereas the selection is a static here-and-now decision, consistently with the real-world planning and operations of projects, the starting times should adjust to the uncertain durations when they become known.

With the aim of addressing this problem, we define the robust resource constrained project scheduling problem as the following two-stage adjustable robust optimisation problem (TSRCPSP). The objective is to find a sufficient selection that minimizes the worst case makespan under uncertainty:

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

$$S_0(\theta) = 0 \ \forall \theta \in \Theta, \tag{2}$$

$$S_j(\theta) - S_i(\theta) \geq \theta_i \ \forall (i,j) \in \mathcal{E}, \forall \theta \in \Theta. \tag{3}$$

Here, we denote with $\mathcal{X}$ the set of sufficient selections and we assume that the set of arcs E is amended with the extra precedences $X \in \mathcal{X}$, leading to the extended set of precedence relations $\mathcal{E} := E \cup X$. The uncertain duration $\theta_i, \ \forall i \in V$, is defined over the support Θ. Throughout the paper, we shall assume that the support Θ is a nonempty and closed subset of $\mathcal{R}^n$ and is polyhedral. Let $W \in R^{q \times n}$ and $h \in R^q$ be an uncertain matrix and an uncertain vector, respectively, the uncertain polyhedron may be represented as follows $\Theta = \{\theta \in \mathcal{R}^n_+ : W\theta \leq h\}$.

The selection decisions $X \in \mathcal{X}$ represent first-stage decisions to be taken before the activity durations are known, whereas $S_j(\theta)$ denotes the second-stage variables representing the starting time of activity j, under the duration realization $\theta \in \Theta$. It is worthwhile noticing that the starting times also depend on the selection X. The explicit dependency on X is omitted for

the sake of brevity, since it is implicitly included in the set $\mathcal{E}$. To highlight this dependence we define $S : (\mathcal{X} \times \Theta) \rightarrow \mathcal{R}_{\geq 0}^{n+2}$ as the vector of second stage decisions that can be taken after the duration of all activities become known.

It is worthwhile noticing that the robust precedence constraints maintain the same structure of their deterministic counterpart, but they are parameterized in $\theta \in \Theta$.

Remark 1. *The TSRCPSP is not equivalent to the static robust counterpart of the RCPSP, which reads as*

$$\min_{X \in \mathcal{X}, S} S_{n+1} \tag{4}$$

$$S_0 = 0, \tag{5}$$

$$S_j - S_i \geq \theta_i \ \forall (i,j) \in \mathcal{E}, \forall \theta \in \Theta. \tag{6}$$

Intuitively, this problem determines a minimum makespan schedule that is feasible for all anticipated realizations of the activity durations $\theta \in \Theta$ simultaneously since both first-stage and second-stage decisions are selected before the activity durations are known. In other words, the TSRCPSP allows the starting times to be chosen after the activity duration realization is observed, whereas the robust static counterpart requires setting the starting times before any uncertain realization is known.

Whilst the TSRCPSP is an optimistic model, the static counterpart is a worst-case, over-conservative approach. Therefore, the optimal makespan obtained by the solution of the adjustable TSRCPSP (z_{adj}) is less conservative than the one obtained from the solution of the static robust counterpart (z_{static}) hence, in general we have $z_{adj} \leq z_{static}$.

However, in some cases, the optimal solution of the TSRCPSP can be obtained by solving an instance of the static robust RCPSP. Ben-Tal and Nemirovski [12] showed that $z_{adj} = z_{static}$ if the uncertainty set Θ is a Cartesian product of independent compact convex sets. As a result, for instance, solving the TSRCPSP over a hypercube is equivalent to solve the deterministic RCPSP for the worst-case activity duration vector (See Appendix B for a formal proof). Even though computationally attractive, this condition prevents any form of dependency among the activities. Bertsimas and Goyal [17] showed that a static robust solution is a 2-approximation to TSRCPSP if the uncertainty set is symmetric or positive, but the bound can be arbitrarily large for general uncertainty sets.

In order to solve the two-stage problem with general asymmetric uncertainty sets, allowing also correlation among different activities, we present in the next Section an exact solution approach.

4 Exact solution approach

The TSRCPSP presents a specific structure where the set of decision variables can be grouped into two sets, each with a distinctive impact on the global problem. The first one encompasses variables related to the sufficient selection decisions whereas the latter includes variables related to the makespan evaluation.

This structure suggests to adopt a decomposition framework, in a spirit similar to the Benders' approach ([11]). The main idea is to decompose the original problem into a master problem and a subproblem whose solution provides information for valid cuts to append to the Benders' master problem. After introducing the new cut, the master problem is resolved and the algorithm is iterated continuously until the difference between the lower bound and the best upper bound is small enough. Similar approaches have been recently proposed for the solution of two-stage robust problems ([27, 65]). Before presenting the detailed scheme of the algorithm, we introduce in the following subsections, the master problem and the subproblem formulations along with the cut generation scheme adopted.

4.1 The master problem

The master problem should be able to determine sufficient selections. Following the resource-flow formulation (first presented in [3]), the master problem is formulated as a mixed integer program where resource flow variables f_{ijk} are used to model the number of units of resource k directly transferred from activity i to activity j, whereas sequential binary variables y_{ij} are defined to represent all the precedence relations between all the activities (including also the transitive closure $T(E \cup X)$). Moreover, an auxiliary variable η represents a lower bound on the optimal solution of the TSRCPSP.

At a generic iteration t the formulation of the master problem is as

follows:

$$\min \eta \tag{7}$$

$$y_{ij} = 1 \qquad \forall (i,j) \in E, \tag{8}$$

$$y_{ij} + y_{ji} \leq 1 \qquad \forall i,j \in V,\ i < j, \tag{9}$$

$$y_{ip} \geq y_{ij} + y_{jp} - 1 \qquad \forall i,j,p \in V, i \neq j \neq p, \tag{10}$$

$$f_{ijk} - min(\tilde{r}_{ik}, \tilde{r}_{jk}) y_{ij} \leq 0 \qquad \forall i,j \in V, i \neq j, i \neq n+1, j \neq 0, \forall k \in K, \tag{11}$$

$$\sum_{j \in V \setminus \{i,0\}} f_{ijk} = \tilde{r}_{ik} \qquad i \in V \setminus \{n+1\},\ \forall k \in K, \tag{12}$$

$$\sum_{i \in V \setminus \{j,n+1\}} f_{ijk} = \tilde{r}_{jk} \qquad j \in V \setminus \{0\},\ \forall k \in K, \tag{13}$$

$$\eta \geq \Psi(\mathbf{y}, M^{*l}) \qquad 0 \leq l < t, \tag{14}$$

$$f_{ijk} \geq 0 \qquad \forall i,j \in V, i \neq j, i \neq n+1, j \neq 0, \forall k \in K, \tag{15}$$

$$y_{ij} \in \{0,1\} \qquad \forall i,j \in V, i \neq j. \tag{16}$$

Here, $\tilde{r}_{ik}$ is defined as $\tilde{r}_{ik} = \begin{cases} R_k & \text{if } i = 0, n+1 \\ r_{ik} & \forall i \in V \setminus \{0, n+1\} \end{cases}$, $\forall k \in K$.

Preprocessing constraints (8) set the preexisting precedence constraints. Constraints (9) and (10) are valid inequalities preventing cycles. Constraints (11) are big-M constraints linking flow variables with precedence related variables. Constraints (12) and (13) are resource flow-conservation constraints. Inequalities (14) are the optimality cuts that are a function Ψ of the variables y ($\mathbf{y}$ denotes a vector of variables) and of the optimal solution of the subproblem at previous iterations l, $0 \leq l < t$ denoted with M^{*l}. This value, on the other hand, depends on the optimal solution of the master problem at iteration l, $\mathbf{y}^{*\mathbf{l}}$. For the sake of brevity, this dependence is not explicitly indicated. A detailed description of the cuts is presented in subsection 4.3.

Koné et al. [43], by comparing computationally different mathematical programming formulations for the RCPSP, showed that, although the resource flow formulation produces bad relaxations due to the presence of big-M constraints, it may be competitive with time-indexed formulations for problems with a long time horizon (in this respect, it is worth remarking that our master has less big-M constraints than the original flow based formulation where also the precedence relations are formulated as big-M constraints).

Beside this, another advantage of this formulation is the clear separation between the sequencing and the scheduling subproblems, that enables a time-independent formulation of the resource constraints. The incorporation of uncertainty is then confined to the subproblem, where starting times can

be easily determined since resource conflicts have been already resolved in the master.

4.2 The subproblem

The optimal solution of the master problem at a generic iteration t determines an acyclic subgraph $G'(V, \mathcal{E}^t)$ where $\mathcal{E}^t := \{(i,j) \in V \times V : y_{ij}^{*t} > 0\}$; X^t is accordingly set.

Then, a feasible solution for the problem (1) – (3) can be determined by solving the following subproblem

$$\min_{S(\cdot)} \max_{\theta \in \Theta} S_{n+1}(\theta) \tag{17}$$

$$S_0(\theta) = 0, \tag{18}$$

$$S_j(\theta) - S_i(\theta) \geq \theta_i \ \forall (i,j) \in \mathcal{E}^t, \forall \theta \in \Theta. \tag{19}$$

The subproblem at iteration t can be recast in the following equivalent form:

$$\max_{\theta \in \Theta} \min_{S \in \Omega(X^t, \theta)} S_{n+1}, \tag{20}$$

where

$$\Omega(X^t, \theta) = \{S \in R_+^{n+2} : S_0 = 0, \ S_j - S_i \geq \theta_i \ \forall (i,j) \in \mathcal{E}^t\}.$$

Therefore, (20) is a worst case makespan minimization problem with a max-min form, where $\min_{S \in \Omega(X^t, \theta)} S_{n+1}$ determines the makespan over a subgraph defined by the fixed selection X^t and a given $\theta \in \Theta$, which is then maximized over the uncertain set Θ.

Invoking a strong duality result, we can obtain the optimal solution of the robust makespan problem by solving the optimistic counterpart of its dual problem [10, 58]. Loosely speaking, this result states that optimizing under the worst case in the primal is the same as optimizing under the best case in the dual. This strong duality always holds for uncertain polyhedral convex programming problems, where the uncertainty set is a polytope [39].

The dual of the worst-case makespan problem, at a generic iteration t, is the following longest path problem, where the best case criterion is applied.

$$M^{*t} = \max_{\theta \in \Theta, \alpha} \sum_{(i,j) \in \mathcal{E}^t} \theta_i \alpha_{ij} \tag{21}$$

$$\sum_{(i,n+1) \in \mathcal{E}^t} \alpha_{in+1} = 1, \tag{22}$$

$$\sum_{(0,i) \in \mathcal{E}^t} \alpha_{0i} = 1, \tag{23}$$

$$\sum_{(i,j) \in \mathcal{E}^t} \alpha_{ji} - \sum_{(j,i) \in \mathcal{E}^t} \alpha_{ij} = 0 \ \forall i \in V \setminus \{0, n+1\}, \tag{24}$$

$$\alpha_{ij} \in \{0,1\} \ \forall (i,j) \in \mathcal{E}^t. \tag{25}$$

The optimal α values define the longest path π^{*t} of length M^{*t}.

The exact solution of (21) – (25) could be determined by solving with off-the-shelf softwares a mixed-integer linear program obtained by linearizing every bilinear term $\theta_i \alpha_{ij}$ with standard techniques. Whilst optimally solving the subproblem is NP-complete in general, in Section 5, we investigate a polynomially solvable case, for which a tailored solution approach is presented.

4.3 Optimality Cuts

Once the subproblem (21) – (25) is solved, a valid cut can be added to the master problem formulation.

Proposition 1. *Given a finite global lower bound L of the problem (1) – (3), and the optimal subproblem solution $\mathbf{y}^{*\mathbf{t}}$, M^{*t} the constraint*

$$\eta \geq (M^{*t} - L) \sum_{(i,j) \in X^t} y_{ij} - [(M^{*t} - L)(|X^t| - 1) - L] \tag{26}$$

is a valid optimality cut. Since there is a finite number of such cuts the algorithm converges to an optimal solution to the TSRCPSP.

Proof. The quantity $\sum_{(i,j) \in X^t} y_{ij}$ is always less than or equal to $|X^t|$, taking exactly the value $|X^t|$ when $\mathbf{y} = \mathbf{y}^{*\mathbf{t}}$ component-wise.
In this case $\sum_{(i,j) \in X^t} y_{ij} - |X^t| = 0$ and the right-hand side takes the value M^{*t}, otherwise the right-hand side takes a value less than or equal to L. Since the first stage decision variables $\mathbf{y}$ are binary, there is only a finite number of feasible first stage solutions and therefore, the number of cuts that can be added to the master is finite. □

This cut can be strengthened to

$$\eta \geq (M^{*t} - L) \sum_{(i,j)\in\pi^{*t}} y_{ij} - [(M^{*t} - L)(|\pi^{*t}| - 1) - L]. \tag{27}$$

In fact, the set of arcs belonging to π^{*t} defines the smallest set of arcs that might influence the makespan.

The general form of (26) is similar to the cut used in the Integer L-shaped algorithm [48] for solving two-stage stochastic programming problems with first stage binary variables. We further notice that, for $L = 0$, the cut resembles the logic cuts used in Logic-based Benders decomposition for scheduling problems [36]. In particular, the optimality cut reads as follows:

$$\eta \geq M^{*t}[1 - (|X^t| - \sum_{(i,j)\in X^t} y_{ij})].$$

This relation asserts that the makespan of the subproblem in the successive iterations cannot be less than M^{*t}, unless at least one extra precedence is removed from X^t.

4.4 Algorithm enhancements

In this Section, we study how to improve the computational performance of the basic method by including a subproblem relaxation in the master problem.

We employed two relaxations within the master problem, since they seem to have somewhat complementary strength. A straightforward relaxation of the subproblem can be obtained by considering the following valid inequalities involving not only the master variables, but also a new set of variables s

$$s_j - s_i \geq d_i^{LB} - B_{ij}(1 - y_{ij}) \ \forall i, j \in V, i \neq j,$$
$$\eta \geq s_{n+1},$$

where d_i^{LB} is a suitable lower bound on the random activity duration, B_{ij} is a sufficiently large number.

We would like to remark that these constraints are written for a given duration vector and, as such, do not depend on the uncertainty set.

These inequalities state that, whenever a precedence is introduced through the variables y_{ij} between activities i and j, the starting time of j should be at least equal to the starting time of the predecessor i plus the activity duration.

Since lower bounds on the activity durations are used, these constraints represent a relaxation of the precedence constraints in the TSRCPSP. Instead of considering the inequalities for all the activity pairs in the network,

it is computationally advantageous to use only a subset of the constraints. In this respect, we have considered a clusterization of the activities in disjoints subsets

$$C_1, \ldots, C_{NC} : \bigcup_{c=1,\ldots,NC} C_c = V, \;\; C_c \cap C_{c'} = \emptyset \;\; c' > c \quad c, c' = 1, \ldots, NC.$$

These clusters follow the topological ordering of the activities in the network in such a way that to a cluster C_c belong smaller numbered activities than activities in C_{c+1}. In each cluster we have considered only one representative activity $n_{C_c} = argmax_{i \in C_c} d_i^{LB}, \; c = 1, \ldots, NC$.

Thus, we have amended the master problem with this set of constraints:

$$s_{n_{C_{c'}}} - s_{n_{C_c}} \geq \frac{(d^{LB}_{n_{C_c}} + d^{LB}_{n_{C_{c'}}})}{2} - B_{n_{C_c} n_{C_{c'}}}(1 - y_{n_{C_c} n_{C_{c'}}}),$$
$$c, c' = 1, \ldots NC : c' > c, \quad (28)$$

$$\eta \geq s_{n_{C_{NC}}}. \quad (29)$$

The second relaxation is based on the following idea. The makespan is obviously greater than or equal to the length of the longest path obtained considering the activity durations equal to their lower bound. Since enumerating all the paths is not viable, we consider an equivalent polynomial size formulation by introducing a flow variable ϕ_{ij} for each pair $(i, j) \in \{V \times V, i \neq j\}$, and by amending the master with the following constraints set:

$$\eta \geq \sum_{i \in V} \sum_{j \in V} d_i^{LB} \phi_{ij} \quad (30)$$

$$\sum_{i \in V} \phi_{ij} - \sum_{j \in V} \phi_{ij} = 0 \;\; \forall i \in V \setminus \{0, n+1\}, \quad (31)$$

$$\sum_{j \in V} \phi_{0j} = 1, \quad (32)$$

$$\sum_{j \in V} \phi_{jn+1} = 1, \quad (33)$$

$$\phi_{ij} \leq y_{ij} \;\; \forall i, j \in V, \;\; i \neq j, \quad (34)$$

$$\sum_{(i,j) \notin E} \phi_{ij} \geq 1, \quad (35)$$

$$\phi_{ij} \geq 0 \;\; \forall i, j \in V, \;\; i \neq j. \quad (36)$$

While equations (31) – (33) are flow conservation constraints, constraints (34) impose that the path should involve only links corresponding to actual or established extra precedence relations in the network. Constraint (35)

enforces the condition that the flow should traverse at least one arc corresponding to an extra precedence. We notice that, beside preventing the lower bound to be zero, this is not restrictive because if any extra precedence is involved the problem turns out to be a critical path problem without resources providing a very weak bound.

4.5 The Benders' algorithm

The Benders' Decomposition algorithm is reported below.

- Set $LB := -\infty$; $UB := +\infty$ $t = 0$, $\pi^{*0} = \{\emptyset\}$, $M^{*0} = null$.
- while $UB > LB$ do
  - Solve the master problem (7) – (13), (15) – (16), (27) – (36).
  - Assign to LB the optimal objective function value of the master problem.
  - Solve the subproblem (21) – (25) and let M^{*t} be the optimal makespan and π^{*t} the corresponding longest path.
  - Update $UB = min(UB, M^{*t})$.
  - Add the cut (27) with $L = LB$ and update $t := t + 1$.
  - end while
- Return LB

We notice that the value of the lower bound L in the cut (27) can assume in each iteration the value LB. In fact, given the peculiar structure of the problem in which the objective function does not include any information related to first stage variables, the optimal solution of the master problem constitutes a global lower bound for the problem (1) – (3).

5 The cardinality constrained uncertainty set

In this section, we focus on a specific dependency structure, inspired by the budget of uncertainty proposed in [18] and used to model uncertain parameters in several combinatorial optimization problems (see, e.g., [1]). This dependency approach provides an intuitive interpretation of uncertainty for a risk averse decision maker who might flexibly adjust the level of risk aversion by tuning the so called budget of uncertainty, representing the cardinality of the activities simultaneously experiencing a delay. It is important to note that this representation of the uncertainty is general enough to account for interactions among activities and can be easily interpreted by the project managers.

Formally, we assume that each activity duration lies between its mean value $\bar{\theta}_i$ and its peak value $\bar{\theta}_i + \hat{\theta}_i$. In practical settings, it is unlikely that all the activities exhibit longer durations than expected. To control the degree of robustness and conservatism we allow only some of the uncertain parameters to simultaneously deviate from their nominal values.

As a consequence, the uncertainty polytope Θ contains all the durations where at most Γ activities can reach their peak values simultaneously. Hence,

$$\Theta = \{\theta_i | i \in V, \theta_i = \bar{\theta}_i + \delta_i \hat{\theta}_i, 0 \leq \delta_i \leq 1, \sum_{i \in V} \delta_i \leq \Gamma\}.$$

Note that Γ lies within the range $[0, |V|]$. In particular, when $\Gamma = 0$ the uncertainty is disregarded, whereas different values can be used to analyze the impact of different levels of uncertainty in the system.

Under this assumption, the subproblem (20) can be written as:

$$\max_{\substack{\sum_{i \in V} \delta_i \leq \Gamma \\ 0 \leq \delta_i \leq 1}} \min_{S \in \Omega_{BS}(X^t, \theta)} S_{n+1} \tag{37}$$

where θ represents the vector of the activity durations θ_i and

$$\Omega_{BS}(X^t, \theta) = \{S \in R^{n+2}_{\geq 0} : S_0 = 0,\ S_j - S_i \geq \bar{\theta}_i + \delta_i \hat{\theta}_i \ \ \forall (i,j) \in \mathcal{E}^t\}.$$

The dual version of problem (37) can be rewritten as:

$$\max_{\alpha, \delta} \sum_{(i,j) \in \mathcal{E}^t} (\bar{\theta}_i + \delta_i \hat{\theta}_i) \alpha_{ij} \tag{38}$$

$$(22) - (24), \tag{39}$$

$$\sum_{i \in V} \delta_i \leq \Gamma, \tag{40}$$

$$\alpha_{ij} \in \{0, 1\}\ \forall (i,j) \in \mathcal{E}^t, \tag{41}$$

$$0 \leq \delta_i \leq 1\ \forall i \in V. \tag{42}$$

Proposition 2. *Problem (38) – (42) is a disjoint program over a polyhedron where the linear constraints concerning the variables α and those representing Θ are disjoint. Hence, there exists an optimal solution that is at the extreme points of the disjoint polyhedra ([37]).*

The model can be linearized as follows.

$$\max_{\alpha,\delta,w} \sum_{(i,j)\in\mathcal{E}^t} (\bar{\theta}_i \alpha_{ij} + \hat{\theta}_i w_{ij}) \tag{43}$$

$$(22) - (24), \tag{44}$$

$$w_{ij} \leq \delta_i \ \forall (i,j) \in \mathcal{E}^t, \tag{45}$$

$$w_{ij} \leq \alpha_{ij} \ \forall (i,j) \in \mathcal{E}^t, \tag{46}$$

$$0 \leq \delta_i \leq 1 \ \forall i \in V, \tag{47}$$

$$\sum_{i\in V} \delta_i \leq \Gamma, \tag{48}$$

$$\alpha_{ij} \in \{0,1\} \ \forall (i,j) \in \mathcal{E}^t, \tag{49}$$

$$w_{ij} \geq 0 \ \forall (i,j) \in \mathcal{E}^t. \tag{50}$$

Proposition 3. *Problem (43) – (50) is equivalent to problem (38)–(42).*

Proof. We need first to prove that $w_{ij} = \delta_i \alpha_{ij}$ at any optimum.
If $\delta_i = 0$ then, in any feasible solution for the constraints (45) $w_{ij} = 0$. Hence the equivalence holds.
If $\delta_i = 1$ we have two possible cases.
If $\alpha_{ij} = 0$ then, in any feasible solution for the constraints (46) $w_{ij} = 0$. Hence the equivalence holds.
If $\alpha_{ij} = 1$ then from the constraints (46) $w_{ij} \leq 1$. Since we are maximizing w in any optimal solution w_{ij} will be exactly 1. Hence the equivalence holds. It is worth observing the classic linearization constraints $w_{ij} \geq \delta_i + \alpha_{ij} - 1$ have been omitted.

Nonetheless, since problem (43) – (50) maximizes $\sum_{(i,j)\in\mathcal{E}^t} \hat{\theta}_i w_{ij}$ and $\hat{\theta}_i \geq 0$, $w_{ij} = 1$ at any optimum. Moreover, $\sum_{(i,j):\alpha_{ij}=1} w_{ij} \leq \Gamma$. □

Proposition 4. *Problem (43) – (50) is polynomially solvable.*

Even though this result is intuitive (see Proposition 2), a formal proof is provided in the Appendix B.

Although (43) – (50) could be solved as a linear problem, when Γ is integer, we have designed a tailored solution approach that relies on the dynamic programming paradigm. The proposed algorithm is based on the following consideration. For each node i of the graph, the algorithm keeps at most Γ paths, starting from the source node 0 and ending at i. Since we are looking for a longest path in the optimistic case, the path with the highest makespan is retained for each node. Each path π_i^γ, $\gamma = 0, \ldots, \Gamma$ from 0 to i is characterized by the presence of exactly γ delayed activities. From a given path π_i^γ, all the possible extensions to any adjacent node j are evaluated. In particular, if the successor activity shows a delay, the path $\pi_j^{\gamma+1}$ is generated to be compared with the existing one, otherwise the path

π_j^γ is considered for comparison. The previous considerations are at the basis of the developed recursive algorithm, formally stated below.

Let $ST(j,\gamma)$ be a state associated with node j at level γ, where exactly γ activities exhibit delay. We denote as $C(ST(j,\gamma))$ the cost associated with state $ST(j,\gamma)$. The dynamic programming recursion follows.

$$\begin{cases} C(ST(0,0)) &= 0, \\ C(ST(j,\gamma)) &= \max_{i:(i,j)\in\mathcal{E}^t} \\ & \left\{\max\left(C(ST(i,\gamma)) + \bar{\theta}_i, C(ST(i,\gamma-1)) + \bar{\theta}_i + \hat{\theta}_i\right)\right\}, \\ & \forall j \in V \setminus \{0\}, \gamma = 1,\ldots,\Gamma, \\ C(ST(j,0)) &= \max_{i:(i,j)\in\mathcal{E}^t} \left\{C(ST(i,0)) + \bar{\theta}_i\right\}, \\ & \forall j \in V \setminus \{0\}. \end{cases} \tag{51}$$

The number of states in equation (51) is equal to Γn and the computation of each state involves, in the worst case, $m = |\mathcal{E}|$ operations. Therefore, the complexity of the algorithm is $O(\Gamma nm)$. For the sake of comprehension, the pseudo-code of the dynamic programming recursion (51) is reported in Appendix C.

Example 1. *The purpose of this example is to illustrate the application of the decomposition approach and the generation of the cuts. Let us consider the 9 nodes network represented in Figure 2, where the resource consumption r_i, the nominal duration $\bar{\theta}_i$ and the deviation $\hat{\theta}_i$ are associated with each node. We consider one resource with $R_1 = 4$ and $\Gamma = 2$.*

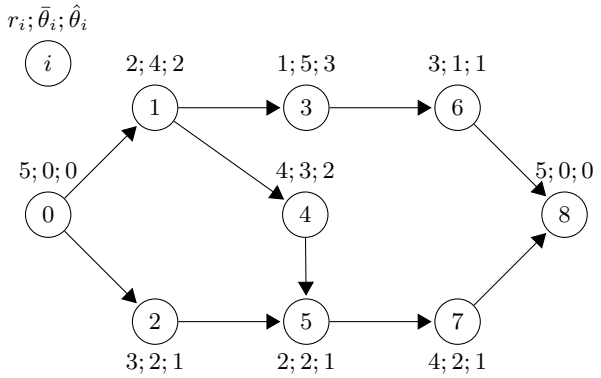

Figure 2: Network example

We show the solution of the subproblem in term of path π^{*t} and makespan and the cut added for each iteration t.

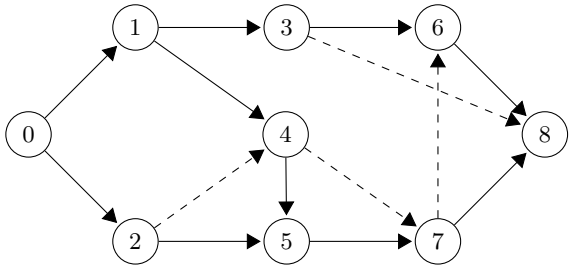

Figure 3: Network represented the selection X^{*0}

- *First iteration* $t = 0$. *The Master is solved. The subgraph* $G'(V, \mathcal{E}^0)$ *is represented in Figure 3, where the extra precedence relationships are represented in dashed lines. The optimal objective function is* $\eta^{*0} = 9$. *Hence,* $LB = 9$.

  - *The solution of the sub-problem on the network defined by* X^{*0} *is* $\pi^{*0} =< 0, 1, 4, 5, 7, 6, 8 >$ *with* $M^{*0} = 16$. *Then,* $UB = 16$
  - *Add the cut* $\eta \geq (16-9)(y_{01} + y_{14} + y_{45} + y_{57} + y_{76} + y_{68}) - [(16-9)(6-1) - 9]$

- *Second iteration* $t = 1$. *The subgraph* $G'(V, \mathcal{E}^1)$ *is represented in Figure 4. The optimal objective function is* $\eta^{*1} = 11$, *hence* $LB = 11$.

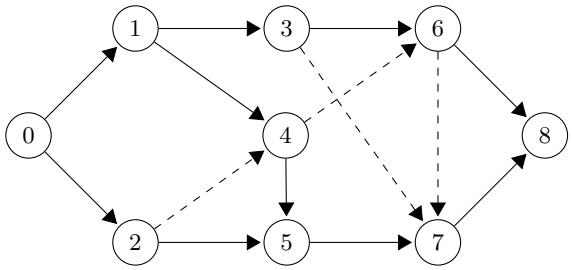

Figure 4: Network represented the selection X^{*1}

  - *The solution of the sub-problem on the network defined by* X^{*1} *is* $\pi^{*1} =< 0, 1, 3, 6, 7, 8 >$ *with* $M^{*1} = 17$, $UB = 16$
  - *Add cut* $\eta \geq (17-11)(y_{01} + y_{13} + y_{36} + y_{67} + y_{78}) - [(17-11)(5-1) - 11]$

- *Third iteration* $t = 2$. *The solution of the master problem is* $\eta^{*2} = 16$, *then* $LB = 16$. *Stop* $UB = LB = 16$

6 Computational experiments

The aim of this section is to assess the computational behavior of the exact solution approach in terms of both efficiency and effectiveness. The proposed algorithm has been coded in Java and the master problem has been solved by CPLEX 12.5.1. The experiments have been carried out on an Intel Core i7-470HQ CPU 2.50 GHz RAM 16 GB, under the Windows 8 operation system.

6.1 Instances

The computational results have been collected on problems generated from the 30 activities benchmark instances of the PSPLIB (http://www.om-db.wi.tum.de/psplib). We refer the interested readers to [42] for a detailed description of the test problems. The instances differ essentially for three main parameters, namely

> $NC = \{1.5, 1.8, 2.1\}$ (Network complexity); representing the network complexity, i.e., the average number of non-redundant arcs per nodes including the dummy activities;
>
> $RF = \{0.25, 0.5, 0.75, 1\}$ (Resource factor); measuring how many different resources are used on average by the activities; an instance with a RF equal to 0 is trivial, since no resources are requested. On the other hand, if the RF is 1 all activities request all resource types.
>
> $RS = \{0.2, 0.5, 0.7, 1\}$ (Resource strength); measuring the size of resource conflicts. It computes the ratio of the resource capacity to the highest peak resource usage on a single resource. If the RS is 0 at least one of the activities uses all the amount available of a resource. On the other hand, a RS equal to 1 means that the instance is trivial.

Starting from the 480 deterministic instances we have considered three different values for Γ in the set $\{3, 5, 7\}$, thus generating 1440 robust counterparts. The nominal value $\bar{\theta}$ is equal to the deterministic duration, whereas, $\hat{\theta} := 0.5\ \bar{\theta}$. A time limit of 20 minutes has been imposed to the Benders' algorithm.

6.2 Computational results

The numerical results are collected in Tables A.1–A.8 of Appendix A.

The behavior of the proposed solution approach is strongly related to the characteristics of the instances. In particular, the parameters RF and RS have a strong impact on the performance of the proposed strategy.

The higher the value of the RF, the harder are the instances. This confirms the behavior observed in the deterministic case, since when very

few resource types are required by the activities (low values for RF) it is easy to take good resource allocation decisions. An opposite trend can be observed for the parameter RS. In fact, high RS values result in less sharp resource constraints whereas low values correspond to problems with very scarce resources. We notice that this has a direct impact on the master problem, where a high number of extra precedence relations need to be added in order to solve the frequent resource conflicts.

Figure 5 shows the percentage of solved instances to optimality by varying the parameters RF and RS.

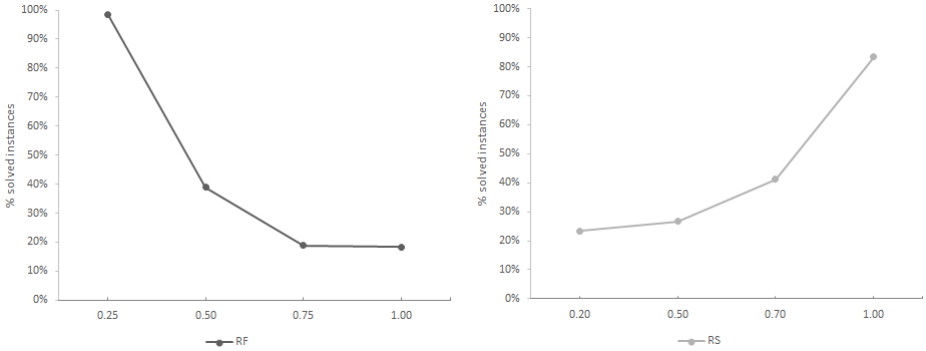

Figure 5: Average percentage of solved instances for different values of RF and RS.

The NC value does not exhibit a strong influence on the number of instances solved to optimality. Indeed, the proposed solution approach is able to solve 191, 211, and 226 instances for value of NC equal to 1.50, 1.80, and 2.10, respectively. The average computational effort is, on average, 173.35 seconds and the proposed solution approach performs 127.43 iterations (see column $\#iter$ of Table A.1), considering the instances solved within the imposed time limit.

The dynamic programming scheme is very efficient and solves the subproblem in 0.07 seconds on average, finding the optimal solution in the first 79% of the iterations. In addition, on average, the upper bound is updated 5.89 times.

For the instances that are not solved to optimality within the fixed time limit, the average optimality gap is 38% (see Table A.5). These are difficult instances, since the proposed approach performs half of the total number of iterations (49.79%) without improving the upper bound values. Nonetheless, we should notice that the initial upper bound is finally improved (20.8%).

In the foregoing, we present a detailed analysis of the computational performance of the proposed approach as a function of the network characteristics and of the degree of uncertainty.

Analysis based on the parameter NC The first part is devoted to the analysis of the impact of the network complexity on the performance of the solution method. Table 1 shows the average results over all solved instances as a function of NC. The total computational time, the average time per iteration and the average time spent for solving the subproblems are indicated in the Table with headers $time$, $\frac{time}{iter}$ and $timeSP$, respectively.

NC	$time$	$\frac{time}{iter}$	$timeSP$
1.50	134.81	2.00	0.03
1.80	198.93	1.60	0.07
2.10	179.91	0.97	0.10

Table 1: Average computational results over all the instances solved within the time limit as a function of NC.

The analysis of the results shows that, on average, the higher the value of NC, the higher the number of instances solved within the time limit (see Table A.1 of the Appendix A). This behavior can be explained by observing that the inclusion of precedence relations between activities reduces the number of feasible schedules and, thus, the size of the solution space. Indeed, for $NC = 2.10$, the proposed solution approach requires on average 0.97 seconds per iteration (see Table 1). This value is 1.64 and 2.05 times lower than the average time per iteration for problems with NC equal to 1.80 and 1.50, respectively. The same behavior has been observed for each value of Γ (see Figure 6).

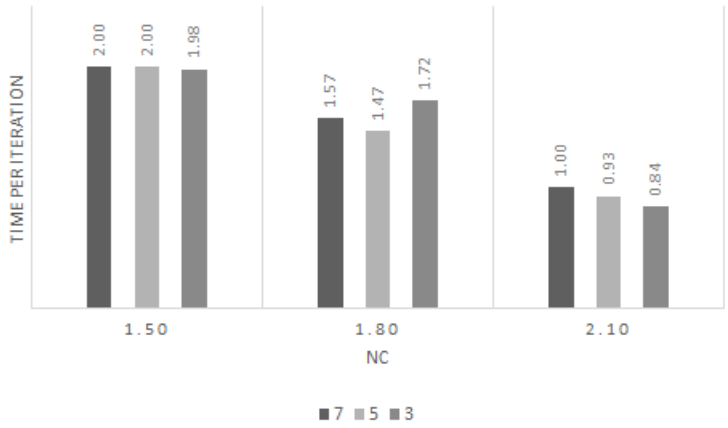

Figure 6: Average execution time per iteration over all the instances solved within the time limit for different values of NC and Γ.

The higher the value of NC, the higher the computational burden for solving the recourse problem. This is an expected trend. Indeed, the behav-

ior of the dynamic programming approach is strongly related to the density of the network, since the higher the number of arcs incident to the nodes, the higher the number of possible paths for each node. The average execution time is 0.03, 0.07 and 0.10 for the instances with NC equal to 1.50, 1.80, and 2.10, respectively.

Analysis based on the parameter RF Parameter RF strongly influences the behavior of the proposed solution approach. Indeed, the number of instances with $RF = 0.25$ solved to optimality is 5.36 times higher than the number of instances with $RF = 1.00$. This performance is justified by the execution time per iteration (see Table 2).

RF	$time$	$\frac{time}{iter}$	$timeSP$
0.25	51.74	0.85	0.03
0.50	246.77	1.13	0.13
0.75	297.84	1.55	0.09
1.00	223.28	2.94	0.04

Table 2: Average computational results over the instances solved within the time limit as a function of RF.

Figure 7 shows the execution time as a function of the parameter RF. The performance of the algorithm is similar for each value of Γ considering RF equal to 0.25 and 0.5. It is worth noting that the worst performance is shown for $\Gamma = 7$.

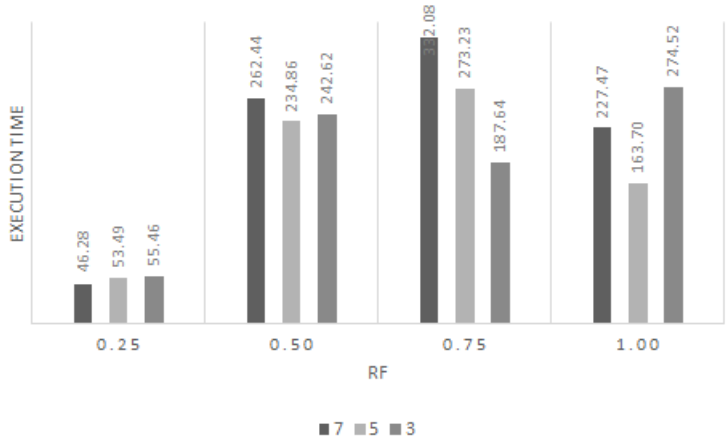

Figure 7: Average total execution time over all the instances solved within the time limit for different values of RF and Γ.

As already observed, increasing the RF value produces a severe deterioration of the algorithmic performance in terms of execution time per

iteration (see Figure 8).

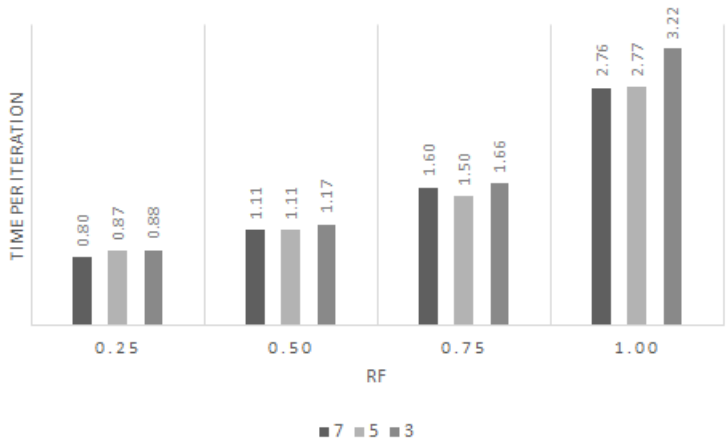

Figure 8: Average execution time per iteration over all the instances solved within the time limit for different values of RF and Γ.

In addition, Figure 8 shows that the execution time per iteration is not influenced by the values of Γ except for a peak with Γ equal to 3 for the harder instances.

Analysis based on the parameter RS The last part of this Section is devoted to the analysis of the impact of the resource strength on the performance of the proposed approach. As shown in Figure 5, the higher the value of RS, the higher the number of instances solved to optimality, revealing the difficulty of the instances with lower values of RS. The average results are reported in Table 3.

RS	$time$	$\frac{time}{iter}$	$timeSP$
0.20	156.45	0.95	0.07
0.50	105.46	0.94	0.06
0.70	246.51	1.49	0.08
1.00	144.91	1.57	0.05

Table 3: Average computational results over the instances solved within the time limit as a function of RS.

The analysis of the results shows that the most time consuming instances are those with $RS = 0.70$, whereas the computational cost is almost the same for $RS = 0.20$ and $RS = 1.00$. The CPU time required by the solution of the instances with $RS = 0.50$ is 2.34 times less than that needed for solving the instances with $RS = 0.70$.

Table 3 shows a relation among the parameter RS and the computational effort per iteration. Similar values are observed for $RS = 0.20$ and $RS = 0.50$ and for $RS = 0.70$ and $RS = 1.00$ (see Figure 9). The average execution time of the latter is 1.67 times higher than the former. In general, the higher RS, the higher the time per iteration.

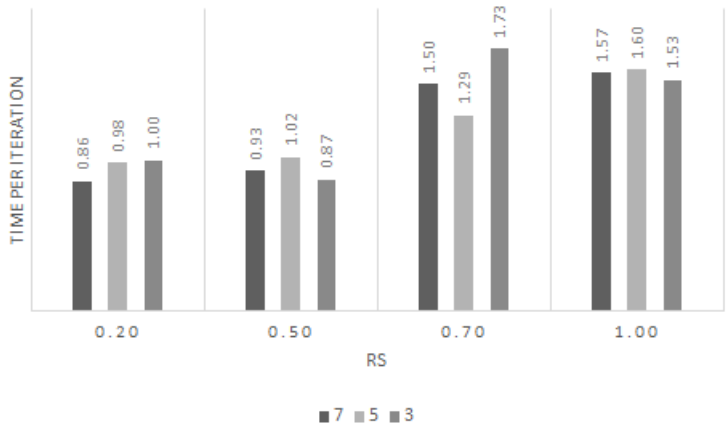

Figure 9: Average execution time per iteration over all the instances solved within the time limit for different values of RS and Γ.

The efficiency of the dynamic programming approach is not influenced by RS (see column $timeSP$ of Table 3).

Analysis based on the parameter Γ The value of Γ does not influence the performance of the proposed solution approach. Indeed, the number of iterations performed is almost equal for each value of Γ. This is not a surprising result. The parameter Γ is an input of the dynamic programming approach defined to solve the robust makespan problem. Indeed, its execution time increases when Γ increases (see column $timeSP$ of Table 4). Since the time spent for the subproblem is very limited, slight variations in the execution time for higher values of Γ do not influence the performance of the proposed solution strategy.

In order to investigate the sensitivity of the model to different values of Γ, Table 4 reports the average makespan computed over all the instances of the corresponding Γ parameter setting.

Different Γ values correspond to different levels of robustness. As expected, there is a monotone increase of the objective function value when the value of Γ is incremented as well. The relative extended makespan can be interpreted as the level of protection provided by the robust model.

In particular, when Γ is increased from 3 to 5 the makespan increases by around 10.76%, whereas a further increase of Γ to 7 produces a makespan

Γ	$Makespan$	$time$	$\#iter$	$timeSP$
7	81.73	118.78	105.58	0.063
5	76.43	117.19	105.13	0.056
3	69.03	124.31	105.59	0.046

Table 4: Average computational results over the instances solved to optimality for each value of Γ.

growth of 6.87%. These two figures highlight the capability of the model to produce different robust solutions for different levels of conservatism, according to the risk perception of the decision maker and providing, at the same time, a guarantee of protection of the schedule.

7 Conclusions

In this paper we have studied the robust RCPSP under polyhedral uncertainty. To address this case, we have proposed an adjustable robust optimization approach, where resource allocation decisions are taken in advance, whereas the starting times can be adjusted to face uncertainty.

A specific decomposition algorithm has been designed. It isolates the resource allocation decisions from the scheduling decisions. The algorithm has been tested on a set of instances generated from the 30 activities deterministic benchmark counterparts. The analysis of the collected results suggests that the algorithmic performance strongly depends on the model parameters. In particular, for RF ranging from 0.5 to 1 and $RS = 0.2, 0.5$ most of the instances cannot be solved within the time limit of 20 minutes. On the other hand, the average computational effort for the solved instances is quite low.

As possible directions for future work, we plan to improve the running times of the master problem solution, using tailored solution approaches, such has branch-and-cut algorithm, and to develop tailored heuristic approaches for solving the hardest instances.

Our investigation led to identify a polynomially solvable case, while strong NP-hardness characterizes the general case of polyhedral uncertainty sets ([50]). The identification of other polynomially solvable cases might represent an interesting direction for future research. Finally, future research efforts should be focused on extensions that consider ellipsoidal uncertainty sets.

References

[1] Alves Pessoa A, Di Puglia Pugliese L, Guerriero F, Poss M (2015) Robust Constrained Shortest Path Problems Under Budgeted Uncer-

tainty. Networks, 66(2): 99–111

[2] Arsham H (1993) Managing project activity-duration uncertainties. Omega, 21(1): 111–122

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

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

[5] Ashtiani B, Leus R, Aryanezhad, MB (2011) New competitive results for the stochastic resource constrained project scheduling problem: Exploring the benefits of preprocessing. Journal of Scheduling 14: 157–171

[6] Balas E. Project scheduling with resource constraints. In E.M.L. Beale, editor, Applications of Mathematical Programming Techniques, pages 187-200. The English Universities Press Ltd, 1971.

[7] Ballestín F (2007) When it is worthwhile to work with the stochastic RCPSP? Journal of Scheduling 10: 153–166

[8] Ballestín, F and Leus, R (2009) Resource-constrained project scheduling for timely project completion with stochastic activity durations decisions in resource-constrained projects. Decision Sciences 38: 1–37

[9] Bartusch M, Möhring RM, Radermacher FJ (1988) Scheduling project networks with resource constraints and time windows. Annals of Operations Research 16: 201-240

[10] Beck A, Ben-Tal A (2009) Duality in robust optimization: Primal worst equals dual best. Operations Research Letters 37(1): 1-6

[11] Benders JF (1962) Partitioning procedures for solving mixed-variables programming problems. Numerische Mathematik 4: 238-252

[12] Ben-Tal A, Nemirovski A (1999) Robust solutions of uncertain linear programs. Operations Research Letters 25(1): 1-14

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

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

[15] Ben-Tal A, El Ghaoui L, Nemirovski A (2009) Robust optimization. (Princeton University Press)

[16] Bertsimas D, Brown DB, Caramanis C (2011) Theory and applications of robust optimization. SIAM Review 53: 464-501

[17] Bertsimas D, Goyal V (2010) On the power of robust solutions in two-stage stochastic and adaptive optimization problems. Mathematics of Operations Research 35: 284-305

[18] Bertsimas D, Sim M (2004) The price of robustness. Operations Research 52 (1): 35-53

[19] Bruni ME, Beraldi P, Guerriero F, Pinto E (2011) A heuristic approach for resource constrained project scheduling with uncertain activity durations. Computers and Operations Research 38: 1305–1318

[20] Bruni ME, Beraldi P, Guerriero F, Pinto E (2011) A scheduling methodology for dealing with uncertainty in construction projects. Engineering Computation 28(8): 1064–1078

[21] Bruni ME, Beraldi P, Guerriero F (2015) The stochastic resource-constrained project scheduling problem, in Handbook on Project Management and Scheduling 2, 811-835

[22] Creemers F (2015) Minimizing the expected makespan of a project with stochastic activity durations under resource constraints. Journal of Scheduling 18(3): 263–273

[23] Chen X, Zhang Y (2009) Uncertain linear programs: Extended affinely adjustable robust counterparts. Operations Research 57(6): 1469–1482

[24] Deblaere F, Demeulemeester E, Herroelen W (2010) Generating Proactive Execution Policies for Resource-Constrained Projects with Uncertain Activity Durations. Research Report KBI 1006, K.U.Leuven, 29 pp.

[25] Demeulemeester E, Herroelen W (2011) Robust Project Scheduling, Foundations and Trends in Technology, Information and Operations Management: Vol. 3: No. 34, pp 201-376. *http* : *//dx.doi.org*/10.1561/0200000021

[26] Fu N, Lau H.C, Varakantham P (2015) Robust execution strategies for project scheduling with unreliable resources and stochastic durations. Journal of Scheduling 18: 607–622

[27] Gabrel V, Lacroix M, Murat C, Remli N (2014) Robust location transportation problems under uncertain demands. Discrete Applied Mathematics 164(1): 100-111

[28] Ghouila-Houri A (1962) Caractérisation des matrices totalement unimodulaires. Comptes Rendus de l'Académie des Sciences 254: 1192

[29] Goldratt EM (1997) Critical Chain. The North River Press, Great Barrington

[30] Golenko-Ginzburg D, Gonik A (1997) Stochastic network project scheduling with non-consumable limited resources. International Journal of Production Economics 48: 29–37

[31] Gorissen BL, Yanikoğlu $\dot{I}$, den Hertog D (2015) A practical guide to robust optimization. Omega, 53: 124–137

[32] Graham RL (1966) Bounds on multiprocessing timing anomalies. Bell System Technical Journal 45, 1563–1581

[33] Herroelen W, Leus R (2004a) The construction of stable project baseline schedules. European Journal of Operational Research 156: 550-565

[34] Herroelen W, Leus R (2004b) Robust and reactive project scheduling: a review and classification of procedures. International Journal of Production Research 42(8): 1599-620

[35] Herroelen W, Leus R (2005) Project scheduling under uncertainty-survey and research potentials. European Journal of Operational Research 165(2): 289-306

[36] Hooker JN (2004) A Hybrid Method for Planning and Scheduling. Principles and Practice of Constraint Programming-Lecture Notes in Computer Science, 3258: 305–316

[37] Horst R, Tuy H (1996) Special Problems of Concave Minimization. Global Optimization: Deterministic Approaches, Springer-Verlag, Berlin, Heidelberg, New York.

[38] Igelmund G, Radermacher FJ (1983a) Algorithmic approaches to preselective strategies for stochastic scheduling problems. Networks 13: 29–48

[39] Jeyakumar V, Lisiam G (2010) Strong Duality in Robust Convex Programming: Complete Characterizations. SIAM Journal on Optimization 20(6): 3384-3407

[40] Kerkhove L-P, Vanhoucke M (2016) Optimised scheduling for weather sensitive offshore construction projects. Omega, http://dx.doi.org/10.1016/j.omega.2016.01.011

[41] Kolisch R, Padman R (2001) An integrated survey of deterministic project scheduling. Omega, 29(3): 249–272

[42] Kolisch R, Sprecher A (1996) Psplib - a project scheduling problem library. European Journal of Operational Research 96(1): 205–216

[43] Koné O, Artigues C, Lopez P, Mongeau M (2013) Comparison of mixed integer linear programming models for the resource-constrained project scheduling problem with consumption and production of resources. Flexible Services and Manufacturing Journal 25(1): 25–47

[44] Lamas P, Demeulemeester E (2015) A purely proactive scheduling procedure for the resource-constrained project scheduling problem with stochastic activity durations. Journal of Scheduling 19(4): 409–428

[45] Lambrechts O, Demeulemeester E, Herroelen W (2008) Proactive and reactive strategies for resource-constrained project scheduling with uncertain resource availabilities. Journal of Scheduling 11: 121–136

[46] Lambrechts O, Demeulemeester E, Herroelen W (2008) A tabu search procedure for developing robust predictive project schedules. International Journal of Production Economics 111(2): 493–508

[47] Lambrechts O, Demeulemeester E, Herroelen W (2011) Time slack-based techniques for robust project scheduling subject to resource uncertainty. Annals of Operations Research 186(1): 443–464

[48] Laporte G, Louveaux FV (1993) The integer L-shaped method for stochastic integer programs with complete recourse. Operations Research Letters 13(3): 133-142

[49] Leus R, Herroelen W (2004) Stability and resource allocation in project planning. IIE Transactions 36: 667-82

[50] Minoux M (2011) On 2-stage robust LP with RHS uncertainty: complexity results and applications. Journal of Global Optimization 49: 521-537

[51] Möhring RH and Stork F (2000) Linear preselective strategies for stochastic project scheduling. Mathematical Methods of Operations Research 52(3): 501–515

[52] Pich MT, Loch CH and De Meyer A (2002) On Uncertainty, Ambiguity, and Complexity in Project Management. Management Science 48(8): 1008–1023

[53] Policella N, Cesta A, Oddi A, Smith S (2009) Solve-and-robustify. Journal of Scheduling 12: 299–314

[54] Rabbani M, Fatemi Ghomi SMT, Jolai F, Lahiji NS (2007) A new heuristic for resource-constrained project scheduling in stochastic networks using critical chain concept. European Journal of Operational Research 176: 794–808

[55] Schwindt C, Trautmann N (2003) Scheduling the production of rolling ingots: Industrial context, model, and solution method. International Transactions in Operational Research 10: 547-563

[56] Radermacher FJ (1981) Optimale Strategien für stochastische Scheduling–Probleme. Habilitationsschrift, RWTH Aachen. In: Schriften zur Informatik und angewandten Mathematik 98, RWTH Aachen

[57] Rom WO, Tukel OI, Muscatello JR (2002) MRP in a job shop environment using a resource constrained project scheduling model. Omega, 30: 275–286

[58] Soyster AL, Murphy FH (2013) A unifying framework for duality and modeling in robust linear programs. Omega, 41: 984–997

[59] Stork F (2001) Stochastic resource-constrained project scheduling. PhD Dissertation, Technische Universität Berlin, Berlin, Germany

[60] Tavares LV, Ferreira JAA, Coelho JS (1998) On the optimal management of project risk. European Journal of Operational Research 107: 451-69

[61] Tsai YW, Gemmil DD (1998) Using tabu search to schedule activities of stochastic resource-constrained projects. European Journal of Operational Research 111: 129–141

[62] Van de Vonder S, Demeulemeester E, Herroelen W, Leus R (2005) The use of buffers in project management: the trade-off between stability and makespan. International Journal of Production Economics 97: 227-40

[63] Van De Vonder S, Demeulemeester E, Herroelen W, Leus R (2006) The trade-off between stability and makespan in resource-constrained project scheduling. International Journal of Production Research 44(2): 215-36

[64] Van de Vonder S, Demeulemeester E, Herroelen W (2008) Proactive heuristic procedures for robust project scheduling: an experimental analysis. European Journal of Operational Research 189: 723-33

[65] Zeng B, Zhao L (2013) Solving two-stage robust optimization problems using a column-and-constraint generation method. Operations Research Letters 41(5): 457-461

[66] Zhu G, Bard JF, Yu G (2007) A two-stage stochastic programming approach for project planning with uncertain activity durations. Journal of Scheduling 10: 167–180

Appendix A - Numerical results

Tables A.1 – A.4 report the average computational results for the instances solved to optimality within the time limit. Each row of Table A.1 reports the average results over 30 instances with the same value of NC, RF, and RS. Tables A.2, A.3, and A.4 show average results over ten instances with Γ equal to 7, 5, and 3, respectively.
Column *time* gives the CPU time in seconds, column $\#iter$ reports the number of cuts added during the execution of the algorithm, column $1^{st}UB$ indicates the optimal solution of the subproblem at the first iteration, $\#Impr$ reports how many times the values of the UB has been improved, column *itUbest* shows the iteration in which the best value of UB is obtained. $timeSP$ reports the time in seconds required by the dynamic programming algorithm for solving the subproblem, column $\#solv$ shows the number of instances solved within the time limit.

NC	*RF*	*RS*	*time*	*#iter*	$1^{st}UB$	*#impr*	*itUbest*	*timeSP*	*#solv*
1.50	0.25	0.20	275.78	171.79	91.00	6.63	89.29	0.08	24
1.50	0.25	0.50	68.53	61.17	85.00	6.03	44.27	0.02	30
1.50	0.25	0.70	12.68	17.23	98.90	4.10	10.67	0.01	30
1.50	0.25	1.00	10.42	16.60	85.80	4.23	12.33	0.01	30
1.50	0.50	0.20							0
1.50	0.50	0.50							0
1.50	0.50	0.70	384.94	183.17	76.17	5.83	78.50	0.09	6
1.50	0.50	1.00	109.87	69.43	94.97	6.90	49.80	0.03	30
1.50	0.75	0.20							0
1.50	0.75	0.50							0
1.50	0.75	0.70							0
1.50	0.75	1.00	187.21	49.50	97.36	5.57	41.14	0.02	14
1.50	1.00	0.20							0
1.50	1.00	0.50							0
1.50	1.00	0.70	52.87	10.67	87.00	6.67	8.67	0.00	3
1.50	1.00	1.00	110.96	27.33	73.71	3.79	21.71	0.01	24
1.80	0.25	0.20	129.89	166.30	100.63	6.63	87.70	0.07	30
1.80	0.25	0.50	19.66	38.07	91.03	5.30	22.67	0.01	30
1.80	0.25	0.70	16.41	30.43	89.63	5.10	18.70	0.01	30
1.80	0.25	1.00	4.40	11.13	88.03	3.97	9.03	0.00	30
1.80	0.50	0.20							0
1.80	0.50	0.50	84.30	45.67	138.67	9.00	34.00	0.04	3
1.80	0.50	0.70	498.83	432.93	111.33	9.87	339.53	0.25	15
1.80	0.50	1.00	228.79	208.70	94.70	7.77	166.87	0.11	30
1.80	0.75	0.20							0
1.80	0.75	0.50							0
1.80	0.75	0.70	204.85	93.17	89.17	3.67	66.67	0.05	6
1.80	0.75	1.00	201.25	95.95	94.20	5.20	91.65	0.05	20
1.80	1.00	0.20							0
1.80	1.00	0.50							0
1.80	1.00	0.70	389.44	83.67	89.67	3.00	81.67	0.06	3
1.80	1.00	1.00	410.46	164.07	82.64	5.50	160.00	0.08	14
2.10	0.25	0.20	63.67	158.57	101.83	5.90	105.83	0.07	30
2.10	0.25	0.50	11.57	32.90	92.53	4.57	22.93	0.01	30
2.10	0.25	0.70	4.92	15.53	104.07	5.33	11.57	0.01	30
2.10	0.25	1.00	2.95	11.00	90.40	3.87	8.73	0.01	30
2.10	0.50	0.20							0
2.10	0.50	0.50	343.21	381.00	115.67	12.00	343.33	0.23	3
2.10	0.50	0.70	238.76	319.78	104.39	8.30	270.43	0.19	23
2.10	0.50	1.00	85.43	109.43	103.37	7.13	86.40	0.06	30
2.10	0.75	0.20							0
2.10	0.75	0.50							0
2.10	0.75	0.70	661.37	472.50	68.50	4.00	396.00	0.25	2
2.10	0.75	1.00	234.52	251.46	92.92	7.04	222.19	0.15	26
2.10	1.00	0.20							0
2.10	1.00	0.50							0
2.10	1.00	0.70							0
2.10	1.00	1.00	152.69	93.77	87.09	3.73	82.18	0.06	22
avg			173.35	127.43	94.01	5.89	99.48	0.07	
tot									628

Table A.1: Average computational results for the instances solved to optimality.

NC	RF	RS	$time$	$\#iter$	$1^{st}UB$	$\#impr$	$itUbest$	$timeSP$	$\#solv$
1.50	0.25	0.20	217.19	152.13	98.00	7.13	85.63	0.08	8
1.50	0.25	0.50	64.69	53.80	91.60	5.70	44.90	0.02	10
1.50	0.25	0.70	11.82	14.60	106.50	4.00	8.30	0.01	10
1.50	0.25	1.00	9.41	15.20	92.50	4.50	12.00	0.01	10
1.50	0.50	0.20							0
1.50	0.50	0.50							0
1.50	0.50	0.70	504.24	250.50	82.00	7.50	73.00	0.14	2
1.50	0.50	1.00	91.59	61.00	102.30	7.00	53.30	0.03	10
1.50	0.75	0.20							0
1.50	0.75	0.50							0
1.50	0.75	0.70							0
1.50	0.75	1.00	245.85	61.80	106.40	6.40	50.60	0.03	5
1.50	1.00	0.20							0
1.50	1.00	0.50							0
1.50	1.00	0.70	41.58	9.00	95.00	6.00	7.00	0.00	1
1.50	1.00	1.00	109.85	29.63	79.50	3.75	24.00	0.01	8
1.80	0.25	0.20	135.64	175.60	108.50	7.50	119.60	0.09	10
1.80	0.25	0.50	18.24	35.50	97.70	5.10	22.00	0.02	10
1.80	0.25	0.70	13.62	26.70	96.20	4.70	14.00	0.01	10
1.80	0.25	1.00	4.77	11.60	94.50	3.80	9.60	0.00	10
1.80	0.50	0.20							0
1.80	0.50	0.50	65.68	42.00	148.00	5.00	11.00	0.04	1
1.80	0.50	0.70	436.50	393.00	120.00	9.80	336.60	0.26	5
1.80	0.50	1.00	222.07	204.00	101.90	8.10	153.20	0.12	10
1.80	0.75	0.20							0
1.80	0.75	0.50							0
1.80	0.75	0.70	242.76	108.00	96.00	4.00	46.00	0.07	2
1.80	0.75	1.00	190.37	92.71	102.14	5.00	88.71	0.06	7
1.80	1.00	0.20							0
1.80	1.00	0.50							0
1.80	1.00	0.70	442.35	94.00	97.00	2.00	92.00	0.08	1
1.80	1.00	1.00	341.55	159.25	90.00	5.50	157.25	0.10	4
2.10	0.25	0.20	61.38	155.20	109.70	5.50	102.00	0.08	10
2.10	0.25	0.50	11.56	31.80	99.90	5.20	22.50	0.02	10
2.10	0.25	0.70	4.53	15.50	111.60	5.80	11.20	0.01	10
2.10	0.25	1.00	2.45	9.50	97.50	3.70	7.50	0.00	10
2.10	0.50	0.20							0
2.10	0.50	0.50	448.45	494.00	124.00	11.00	492.00	0.32	1
2.10	0.50	0.70	266.22	365.00	111.75	8.25	305.63	0.25	8
2.10	0.50	1.00	64.82	83.70	111.20	7.10	67.80	0.06	10
2.10	0.75	0.20							0
2.10	0.75	0.50							0
2.10	0.75	0.70	769.43	543.00	70.00	4.00	541.00	0.31	1
2.10	0.75	1.00	212.02	229.88	98.38	6.88	214.38	0.15	8
2.10	1.00	0.20							0
2.10	1.00	0.50							0
2.10	1.00	0.70							0
2.10	1.00	1.00	202.00	120.00	94.25	3.88	110.00	0.08	8

Table A.2: Average computational results for the instances solved to optimality with $\Gamma = 7$.

NC	*RF*	*RS*	*time*	*#iter*	$1^{st}UB$	*#impr*	*itUbest*	*timeSP*	*#solv*
1.50	0.25	0.20	307.57	187.75	91.63	5.88	79.75	0.09	8
1.50	0.25	0.50	59.24	56.80	85.70	6.20	31.50	0.02	10
1.50	0.25	0.70	14.04	20.90	99.40	4.30	13.70	0.01	10
1.50	0.25	1.00	10.35	16.20	86.30	4.30	12.00	0.01	10
1.50	0.50	0.20							0
1.50	0.50	0.50							0
1.50	0.50	0.70	271.93	130.50	76.50	5.00	65.50	0.06	2
1.50	0.50	1.00	131.31	86.30	95.50	6.70	39.10	0.04	10
1.50	0.75	0.20							0
1.50	0.75	0.50							0
1.50	0.75	0.70	0.00	0.00	0.00	0.00	0.00	0.00	0
1.50	0.75	1.00	180.81	45.80	99.60	5.60	39.80	0.02	5
1.50	1.00	0.20							0
1.50	1.00	0.50							0
1.50	1.00	0.70	61.81	12.00	87.00	7.00	10.00	0.01	1
1.50	1.00	1.00	131.36	28.00	74.25	3.63	22.88	0.01	8
1.80	0.25	0.20	124.40	156.70	101.00	5.60	56.60	0.07	10
1.80	0.25	0.50	19.00	36.20	91.60	5.20	15.60	0.01	10
1.80	0.25	0.70	17.89	30.10	90.30	5.40	20.30	0.01	10
1.80	0.25	1.00	4.01	10.50	88.60	4.10	8.50	0.00	10
1.80	0.50	0.20							0
1.80	0.50	0.50	150.50	71.00	139.00	11.00	69.00	0.06	1
1.80	0.50	0.70	459.18	421.20	112.40	10.00	279.00	0.24	5
1.80	0.50	1.00	222.61	204.60	95.20	7.10	164.20	0.11	10
1.80	0.75	0.20							0
1.80	0.75	0.50							0
1.80	0.75	0.70	238.49	108.00	89.50	4.00	106.00	0.06	2
1.80	0.75	1.00	133.63	65.50	93.17	5.00	59.50	0.03	6
1.80	1.00	0.20							0
1.80	1.00	0.50							0
1.80	1.00	0.70	39.52	8.00	90.00	4.00	6.00	0.00	1
1.80	1.00	1.00	491.69	185.00	84.00	5.80	180.00	0.10	5
2.10	0.25	0.20	66.09	164.70	102.20	5.90	106.50	0.08	10
2.10	0.25	0.50	11.56	34.00	92.90	4.70	23.10	0.02	10
2.10	0.25	0.70	5.05	14.80	104.60	5.20	11.00	0.01	10
2.10	0.25	1.00	2.66	10.20	90.90	3.80	7.70	0.01	10
2.10	0.50	0.20							0
2.10	0.50	0.50	368.54	401.00	116.00	13.00	356.00	0.25	1
2.10	0.50	0.70	232.13	317.50	104.50	7.75	288.75	0.19	8
2.10	0.50	1.00	42.70	65.40	103.70	7.70	60.90	0.04	10
2.10	0.75	0.20							0
2.10	0.75	0.50							0
2.10	0.75	0.70	553.31	402.00	67.00	4.00	251.00	0.20	1
2.10	0.75	1.00	259.91	289.20	93.80	7.40	243.80	0.19	10
2.10	1.00	0.20							0
2.10	1.00	0.50							0
2.10	1.00	0.70							0
2.10	1.00	1.00	94.12	62.29	86.86	3.71	57.29	0.04	7

Table A.3: Average computational results for the instances solved to optimality with $\Gamma = 5$.

NC	RF	RS	$time$	$\#iter$	$1^{st}UB$	$\#impr$	$itUbest$	$timeSP$	$\#solv$
1.50	0.25	0.20	302.58	175.50	83.38	6.88	102.50	0.07	8
1.50	0.25	0.50	81.67	72.90	77.70	6.20	56.40	0.02	10
1.50	0.25	0.70	12.17	16.20	90.80	4.00	10.00	0.01	10
1.50	0.25	1.00	11.49	18.40	78.60	3.90	13.00	0.01	10
1.50	0.50	0.20							0
1.50	0.50	0.50							0
1.50	0.50	0.70	378.64	168.50	70.00	5.00	97.00	0.07	2
1.50	0.50	1.00	106.72	61.00	87.10	7.00	57.00	0.02	10
1.50	0.75	0.20							0
1.50	0.75	0.50							0
1.50	0.75	0.70							0
1.50	0.75	1.00	121.91	38.75	83.25	4.50	31.00	0.02	4
1.50	1.00	0.20							0
1.50	1.00	0.50							0
1.50	1.00	0.70	55.22	11.00	79.00	7.00	9.00	0.00	1
1.50	1.00	1.00	91.68	24.38	67.38	4.00	18.25	0.01	8
1.80	0.25	0.20	129.63	166.60	92.40	6.80	86.90	0.05	10
1.80	0.25	0.50	21.74	42.50	83.80	5.60	30.40	0.01	10
1.80	0.25	0.70	17.72	34.50	82.40	5.20	21.80	0.01	10
1.80	0.25	1.00	4.43	11.30	81.00	4.00	9.00	0.00	10
1.80	0.50	0.20							0
1.80	0.50	0.50	36.73	24.00	129.00	11.00	22.00	0.01	1
1.80	0.50	0.70	600.80	484.60	101.60	9.80	403.00	0.25	5
1.80	0.50	1.00	241.70	217.50	87.00	8.10	183.20	0.09	10
1.80	0.75	0.20							0
1.80	0.75	0.50							0
1.80	0.75	0.70	133.30	63.50	82.00	3.00	48.00	0.03	2
1.80	0.75	1.00	270.09	125.29	87.14	5.57	122.14	0.06	7
1.80	1.00	0.20							0
1.80	1.00	0.50							0
1.80	1.00	0.70	686.43	149.00	82.00	3.00	147.00	0.09	1
1.80	1.00	1.00	384.35	147.00	75.40	5.20	142.20	0.06	5
2.10	0.25	0.20	63.55	155.80	93.60	6.30	109.00	0.06	10
2.10	0.25	0.50	11.60	32.90	84.80	3.80	23.20	0.01	10
2.10	0.25	0.70	5.19	16.30	96.00	5.00	12.50	0.00	10
2.10	0.25	1.00	3.73	13.30	82.80	4.10	11.00	0.00	10
2.10	0.50	0.20							0
2.10	0.50	0.50	212.65	248.00	107.00	12.00	182.00	0.13	1
2.10	0.50	0.70	214.97	270.71	95.86	9.00	209.29	0.13	7
2.10	0.50	1.00	148.77	179.20	95.20	6.60	130.50	0.09	10
2.10	0.75	0.20							0
2.10	0.75	0.50							0
2.10	0.75	0.70							0
2.10	0.75	1.00	225.27	225.88	86.38	6.75	203.00	0.11	8
2.10	1.00	0.20							0
2.10	1.00	0.50							0
2.10	1.00	0.70							0
2.10	1.00	1.00	154.89	95.29	79.14	3.57	75.29	0.04	7

Table A.4: Average computational results for the instances solved to optimality with $\Gamma = 3$.

Tables A.5 – A.8 collect the average results related to the instances not solved to optimality. In particular, we show the value of the UB at the end of the computation, the optimality gap given by $(UB - LB)/UB$ under column *gap*, and column $\#!solv$ gives the number of instances not solved within the time limit.

NC	RF	RS	UB	$\#iter$	$1^{st}UB$	$\#impr$	$itUbest$	$timeSP$	gap	$\#!solv$
1.50	0.25	0.20	63.00	964.83	82.67	9.83	459.67	0.39	0.25	6
1.50	0.25	0.50								0
1.50	0.25	0.70								0
1.50	0.25	1.00								0
1.50	0.50	0.20	107.60	354.83	143.83	7.87	126.87	0.29	0.37	30
1.50	0.50	0.50	80.53	528.07	105.57	7.53	231.70	0.33	0.38	30
1.50	0.50	0.70	71.17	574.79	89.63	7.21	366.67	0.32	0.32	24
1.50	0.50	1.00								0
1.50	0.75	0.20	131.93	192.43	162.30	7.10	89.03	0.19	0.43	30
1.50	0.75	0.50	90.93	231.17	111.23	5.07	102.93	0.20	0.45	30
1.50	0.75	0.70	90.00	268.67	123.67	7.40	142.27	0.18	0.39	30
1.50	0.75	1.00	73.19	286.69	89.25	6.63	154.75	0.15	0.35	16
1.50	1.00	0.20	145.13	94.20	173.60	6.63	46.83	0.16	0.51	30
1.50	1.00	0.50	108.60	148.33	131.03	5.40	51.00	0.19	0.53	30
1.50	1.00	0.70	96.67	167.85	120.59	6.26	66.26	0.14	0.44	27
1.50	1.00	1.00	74.00	180.50	86.50	4.17	82.67	0.13	0.33	6
1.80	0.25	0.20								0
1.80	0.25	0.50								0
1.80	0.25	0.70								0
1.80	0.25	1.00								0
1.80	0.50	0.20	106.57	709.77	138.53	8.57	266.83	0.60	0.36	30
1.80	0.50	0.50	80.63	795.70	112.48	8.30	328.74	0.47	0.35	27
1.80	0.50	0.70	78.20	904.40	113.33	10.60	592.93	0.53	0.31	15
1.80	0.50	1.00								0
1.80	0.75	0.20	131.13	286.60	162.17	7.43	152.00	0.30	0.42	30
1.80	0.75	0.50	94.53	373.97	128.33	7.80	151.87	0.31	0.40	30
1.80	0.75	0.70	86.92	468.50	103.75	5.38	199.50	0.34	0.36	24
1.80	0.75	1.00	80.40	553.40	123.10	9.50	324.30	0.36	0.33	10
1.80	1.00	0.20	163.07	154.47	186.03	5.93	71.87	0.18	0.46	30
1.80	1.00	0.50	113.00	226.53	130.17	4.20	80.43	0.22	0.51	30
1.80	1.00	0.70	93.74	257.30	106.26	3.89	99.07	0.23	0.43	27
1.80	1.00	1.00	87.56	310.31	109.31	5.25	138.56	0.23	0.35	16
2.10	0.25	0.20								0
2.10	0.25	0.50								0
2.10	0.25	0.70								0
2.10	0.25	1.00								0
2.10	0.50	0.20	110.60	1107.37	143.23	9.27	425.70	944.70	0.30	30
2.10	0.50	0.50	88.33	1381.22	120.89	9.74	841.22	929.48	0.31	27
2.10	0.50	0.70	83.86	1287.00	132.29	12.43	698.57	725.86	0.28	7
2.10	0.50	1.00								0
2.10	0.75	0.20	141.93	505.57	172.67	9.27	277.70	570.90	0.36	30
2.10	0.75	0.50	103.10	637.63	133.17	7.10	280.90	573.70	0.40	30
2.10	0.75	0.70	93.61	774.32	134.57	8.32	323.21	647.71	0.37	28
2.10	0.75	1.00	74.50	1275.00	95.25	8.75	1050.75	916.25	0.28	4
2.10	1.00	0.20	157.40	329.50	182.30	7.27	151.50	415.33	0.39	30
2.10	1.00	0.50	113.63	398.00	127.53	3.50	91.17	413.87	0.48	30
2.10	1.00	0.70	97.77	524.93	106.97	3.67	180.07	556.60	0.42	30
2.10	1.00	1.00	85.50	609.75	111.63	5.88	229.50	411.38	0.32	8
avg			99.96	525.40	126.29	7.15	261.09	209.18	0.38	
tot										812

Table A.5: Average computational results for the instances not solved within the time limit.

NC	RF	RS	UB	$\#iter$	$1^{st}UB$	$\#impr$	$itUbest$	$timeSP$	gap	$\#!solv$
1.50	0.25	0.20	67.00	979.00	89.00	11.50	812.50	0.44	0.34	2
1.50	0.25	0.50								0
1.50	0.25	0.70								0
1.50	0.25	1.00								0
1.50	0.50	0.20	115.20	355.90	152.60	7.60	110.90	0.35	0.41	10
1.50	0.50	0.50	87.00	538.00	113.30	7.20	319.30	0.39	0.43	10
1.50	0.50	0.70	76.88	558.75	96.38	6.75	365.38	0.37	0.35	8
1.50	0.50	1.00								0
1.50	0.75	0.20	138.70	192.20	169.30	7.20	90.00	0.22	0.46	10
1.50	0.75	0.50	96.70	231.30	117.90	4.80	68.80	0.19	0.49	10
1.50	0.75	0.70	96.70	269.90	130.30	6.60	150.80	0.22	0.44	10
1.50	0.75	1.00	78.80	289.80	95.40	7.20	157.40	0.18	0.40	5
1.50	1.00	0.20	151.60	94.60	182.60	6.90	43.50	0.13	0.53	10
1.50	1.00	0.50	116.80	147.80	139.30	5.10	55.00	0.16	0.56	10
1.50	1.00	0.70	105.22	169.56	128.44	5.67	46.67	0.17	0.48	9
1.50	1.00	1.00	80.00	190.00	93.50	4.50	87.00	0.17	0.38	2
1.80	0.25	0.20								0
1.80	0.25	0.50								0
1.80	0.25	0.70								0
1.80	0.25	1.00								0
1.80	0.50	0.20	113.20	710.30	147.10	8.40	361.40	0.69	0.40	10
1.80	0.50	0.50	87.44	797.33	120.33	8.56	179.56	0.56	0.41	9
1.80	0.50	0.70	84.40	889.00	121.20	10.60	669.40	0.61	0.36	5
1.80	0.50	1.00								0
1.80	0.75	0.20	138.40	286.00	170.00	7.40	150.40	0.34	0.45	10
1.80	0.75	0.50	101.80	370.30	133.40	7.10	127.60	0.36	0.45	10
1.80	0.75	0.70	93.88	470.00	111.63	5.88	169.13	0.39	0.41	8
1.80	0.75	1.00	84.33	552.33	132.67	8.67	328.33	0.42	0.38	3
1.80	1.00	0.20	170.10	155.10	195.60	6.20	90.10	0.21	0.49	10
1.80	1.00	0.50	121.30	226.00	138.80	4.00	76.60	0.25	0.54	10
1.80	1.00	0.70	100.00	256.89	113.89	3.89	91.22	0.26	0.47	9
1.80	1.00	1.00	93.67	322.33	113.83	5.00	114.50	0.28	0.40	6
2.10	0.25	0.20								0
2.10	0.25	0.50								0
2.10	0.25	0.70								0
2.10	0.25	1.00								0
2.10	0.50	0.20	119.30	1116.90	151.70	9.60	424.10	1.11	0.36	10
2.10	0.50	0.50	95.11	1383.11	129.00	9.78	770.89	1.12	0.36	9
2.10	0.50	0.70	92.00	1245.50	148.50	14.50	490.50	0.86	0.34	2
2.10	0.50	1.00								0
2.10	0.75	0.20	150.70	506.60	181.90	9.70	314.30	0.65	0.39	10
2.10	0.75	0.50	112.20	638.80	141.80	6.20	185.10	0.68	0.45	10
2.10	0.75	0.70	102.67	769.67	146.11	8.67	298.78	0.79	0.42	9
2.10	0.75	1.00	81.00	1232.00	108.00	9.50	903.00	1.07	0.34	2
2.10	1.00	0.20	165.60	330.30	191.30	7.10	174.70	0.48	0.42	10
2.10	1.00	0.50	121.30	396.60	136.10	3.50	83.90	0.47	0.51	10
2.10	1.00	0.70	105.20	525.20	114.80	3.70	191.70	0.52	0.47	10
2.10	1.00	1.00	93.00	587.00	127.00	6.50	233.00	0.51	0.38	2

Table A.6: Average computational results for the instances not solved within the time limit with $\Gamma = 7$.

NC	RF	RS	UB	$\#iter$	$1^{st}UB$	$\#impr$	$itUbest$	$timeSP$	gap	$\#!solv$
1.50	0.25	0.20	64.00	1047.00	83.00	9.50	410.00	0.42	0.31	2
1.50	0.25	0.50								0
1.50	0.25	0.70								0
1.50	0.25	1.00								0
1.50	0.50	0.20	109.20	352.40	144.20	7.50	81.00	0.29	0.38	10
1.50	0.50	0.50	80.80	521.30	105.80	7.50	169.30	0.33	0.38	10
1.50	0.50	0.70	71.63	579.63	90.13	7.75	376.50	0.32	0.35	8
1.50	0.50	1.00								0
1.50	0.75	0.20	134.60	194.60	163.10	7.00	85.10	0.19	0.45	10
1.50	0.75	0.50	91.40	231.30	114.40	5.50	102.30	0.28	0.46	10
1.50	0.75	0.70	89.90	265.10	127.30	7.90	145.40	0.17	0.39	10
1.50	0.75	1.00	73.60	289.20	87.40	6.00	215.80	0.15	0.36	5
1.50	1.00	0.20	145.20	93.90	173.90	6.80	45.10	0.24	0.51	10
1.50	1.00	0.50	108.40	149.10	131.30	5.80	52.60	0.14	0.53	10
1.50	1.00	0.70	96.89	168.56	121.00	6.33	59.33	0.15	0.44	9
1.50	1.00	1.00	74.50	180.50	87.00	4.50	81.00	0.14	0.34	2
1.80	0.25	0.20								0
1.80	0.25	0.50								0
1.80	0.25	0.70								0
1.80	0.25	1.00								0
1.80	0.50	0.20	107.20	719.70	138.90	8.20	198.80	0.61	0.36	10
1.80	0.50	0.50	80.89	799.00	113.00	8.22	325.11	0.47	0.36	9
1.80	0.50	0.70	78.20	896.00	113.00	9.40	416.00	0.53	0.31	5
1.80	0.50	1.00								0
1.80	0.75	0.20	132.50	288.40	162.40	7.30	149.80	0.30	0.43	10
1.80	0.75	0.50	94.60	371.20	131.60	8.10	123.30	0.30	0.41	10
1.80	0.75	0.70	87.63	466.88	104.88	5.63	225.63	0.33	0.37	8
1.80	0.75	1.00	84.00	548.75	121.00	9.25	290.50	0.37	0.33	4
1.80	1.00	0.20	164.30	153.50	186.20	5.60	44.50	0.18	0.47	10
1.80	1.00	0.50	112.40	226.70	130.40	4.50	89.90	0.22	0.51	10
1.80	1.00	0.70	92.89	257.11	106.56	4.33	124.44	0.23	0.42	9
1.80	1.00	1.00	86.40	305.60	111.00	5.40	179.20	0.23	0.34	5
2.10	0.25	0.20								0
2.10	0.25	0.50								0
2.10	0.25	0.70								0
2.10	0.25	1.00								0
2.10	0.50	0.20	111.20	1113.50	143.50	9.20	356.50	0.93	0.31	10
2.10	0.50	0.50	89.00	1378.78	121.44	9.89	856.33	0.91	0.32	9
2.10	0.50	0.70	87.50	1237.00	139.00	14.00	828.00	0.74	0.31	2
2.10	0.50	1.00								0
2.10	0.75	0.20	141.50	503.30	172.90	9.20	258.00	0.57	0.35	10
2.10	0.75	0.50	102.80	638.70	133.70	7.10	326.00	0.56	0.40	10
2.10	0.75	0.70	95.22	773.22	137.67	7.78	323.78	0.66	0.38	9
2.10	0.75	1.00								0
2.10	1.00	0.20	157.90	329.70	182.60	7.30	125.70	0.42	0.39	10
2.10	1.00	0.50	114.20	398.40	127.80	3.90	122.10	0.41	0.48	10
2.10	1.00	0.70	98.70	524.10	107.50	3.60	191.60	0.78	0.43	10
2.10	1.00	1.00	89.67	611.67	111.33	5.00	147.00	0.41	0.35	3

Table A.7: Average computational results for the instances not solved within the time limit with $\Gamma = 5$.

NC	RF	RS	UB	$\#iter$	$1^{st}UB$	$\#impr$	$itUbest$	$timeSP$	gap	$\#!solv$
1.50	0.25	0.20	58.00	868.50	76.00	8.50	156.50	0.31	0.11	2
1.50	0.25	0.50								0
1.50	0.25	0.70								0
1.50	0.25	1.00								0
1.50	0.50	0.20	98.40	356.20	134.70	8.50	188.70	0.25	0.31	10
1.50	0.50	0.50	73.80	524.90	97.60	7.90	206.50	0.28	0.32	10
1.50	0.50	0.70	65.00	586.00	82.38	7.13	358.13	0.27	0.28	8
1.50	0.50	1.00								0
1.50	0.75	0.20	122.50	190.50	154.50	7.10	92.00	0.16	0.39	10
1.50	0.75	0.50	84.70	230.90	101.40	4.90	137.70	0.14	0.41	10
1.50	0.75	0.70	83.40	271.00	113.40	7.70	130.60	0.15	0.35	10
1.50	0.75	1.00	68.17	282.00	85.67	6.67	101.67	0.13	0.29	6
1.50	1.00	0.20	138.60	94.10	164.30	6.20	51.90	0.10	0.49	10
1.50	1.00	0.50	100.60	148.10	122.50	5.30	45.40	0.26	0.49	10
1.50	1.00	0.70	87.89	165.44	112.33	6.78	92.78	0.12	0.38	9
1.50	1.00	1.00	67.50	171.00	79.00	3.50	80.00	0.09	0.27	2
1.80	0.25	0.20								0
1.80	0.25	0.50								0
1.80	0.25	0.70								0
1.80	0.25	1.00								0
1.80	0.50	0.20	99.30	699.30	129.60	9.10	240.30	0.50	0.31	10
1.80	0.50	0.50	73.56	790.78	104.11	8.11	481.56	0.39	0.30	9
1.80	0.50	0.70	72.00	928.20	105.80	11.80	693.40	0.43	0.24	5
1.80	0.50	1.00								0
1.80	0.75	0.20	122.50	285.40	154.10	7.60	155.80	0.25	0.38	10
1.80	0.75	0.50	87.20	380.40	120.00	8.20	204.70	0.27	0.36	10
1.80	0.75	0.70	79.25	468.63	94.75	4.63	203.75	0.28	0.30	8
1.80	0.75	1.00	71.67	560.67	116.33	10.67	365.33	0.31	0.27	3
1.80	1.00	0.20	154.80	154.80	176.30	6.00	81.00	0.16	0.44	10
1.80	1.00	0.50	105.30	226.90	121.30	4.10	74.80	0.19	0.47	10
1.80	1.00	0.70	88.33	257.89	98.33	3.44	81.56	0.19	0.39	9
1.80	1.00	1.00	81.40	300.60	102.20	5.40	126.80	0.19	0.30	5
2.10	0.25	0.20								0
2.10	0.25	0.50								0
2.10	0.25	0.70								0
2.10	0.25	1.00								0
2.10	0.50	0.20	101.30	1091.70	134.50	9.00	496.50	0.79	0.24	10
2.10	0.50	0.50	80.89	1381.78	112.22	9.56	896.44	0.76	0.25	9
2.10	0.50	0.70	76.00	1348.00	117.00	10.00	751.00	0.62	0.23	3
2.10	0.50	1.00								0
2.10	0.75	0.20	133.60	506.80	163.20	8.90	260.80	0.49	0.32	10
2.10	0.75	0.50	94.30	635.40	124.00	8.00	331.60	0.48	0.35	10
2.10	0.75	0.70	84.00	779.50	121.40	8.50	344.70	0.51	0.31	10
2.10	0.75	1.00	68.00	1318.00	82.50	8.00	1198.50	0.76	0.22	2
2.10	1.00	0.20	148.70	328.50	173.00	7.40	154.10	0.35	0.35	10
2.10	1.00	0.50	105.40	399.00	118.70	3.10	67.50	0.36	0.44	10
2.10	1.00	0.70	89.40	525.50	98.60	3.70	156.90	0.38	0.37	10
2.10	1.00	1.00	76.33	623.00	101.67	6.33	309.67	0.35	0.24	3

Table A.8: Average computational results for the instances not solved within the time limit with $\Gamma = 3$.

Appendix B - Formal proofs

The case with hypercube uncertainty set

If the uncertainty set is an hypercube, $z_{adj} \geq z_{static}$. As a corollary this implies that $z_{adj} = z_{static}$.

Proof. We have seen that the static solution is an upper bound on the optimal value of the adjustable robust problem. Therefore, $z_{adj} \leq z_{static}$.
We would like to show that, if the uncertainty set is an hypercube, $z_{adj} \geq z_{static}$. As a corollary this implies that $z_{adj} = z_{static}$.

$$z_{adj} = \min_{X \in \mathcal{X}, S(\cdot)} \max_{\theta \in \Theta} S_{n+1}(\theta) \tag{52}$$

$$S_0(\theta) = 0, \tag{53}$$

$$S_j(\theta) - S_i(\theta) \geq \theta_i \ \forall (i,j) \in \mathcal{E}, \forall \theta \in \Theta. \tag{54}$$

Let Θ be a hypercube, i.e.,

$$\Theta = [l_1, u_1] \times [l_2, u_2] \times, \ldots, \times [l_n, u_n],$$

where l_i and u_i, $\forall i \in V \backslash \{0, n+1\}$ represent, respectively, lower bounds and upper bounds for the activity durations, respectively.
Suppose that (X^*, S^*) is an optimal solution for the static robust RCPSP defined as follows:

$$z_{static} = \min_{x \in \mathcal{X}, S} S_{n+1} \tag{55}$$

$$S_0 = 0, \tag{56}$$

$$S_j - S_i \geq \theta_i \ \forall (i,j) \in \mathcal{E}, \forall \theta \in \Theta. \tag{57}$$

Therefore, the solution $X = X^*$, $S(\theta) = S^*$, $\forall \theta \in \Theta$ is a feasible solution for the TSRCPSP, which implies

$$z_{adj} \leq \max_{\theta \in \Theta} S^*_{n+1}(\theta).$$

Now consider $\hat{X}$ and $\hat{S}(\theta), \forall \theta \in \Theta$ an optimal fully adjustable solution for the TSRCPSP and consider the following realization of the uncertain parameters:

$$\theta_{i,max} = \max_{\theta \in \Theta} \theta_i = \max[l_i, u_i] = u_i, \ \forall i \in V \setminus \{0, n+1\}.$$

We denote the corresponding vector with θ_{max}. Clearly,

$$z_{adj} = \max_{\theta \in \Theta} \hat{S}_{n+1}(\theta) \geq \hat{S}_{n+1}(\theta_{max}).$$

The solution $\hat{X}, \hat{S}_{n+1}(\theta_{max})$ is a feasible solution for the static robust problem. In fact,

$$\hat{S}_j(\theta_{max}) - \hat{S}_i(\theta_{max}) \geq \theta_{i,max} \geq \theta_i \ \forall (i,j) \in \mathcal{E}, \forall \theta \in \Theta.$$

This follows from the feasibility of $\hat{X}$ and $\hat{S}(\theta_{max})$ for the scenario θ_{max} and from the fact that $\theta_{max} \geq \theta$, for all $\theta \in \Theta$.
Therefore,

$$z_{static} \leq \hat{S}_{n+1}(\theta_{max}).$$

Since

$$z_{adj} \geq \hat{S}_{n+1}(\theta_{max}),$$

$$z_{static} \leq z_{adj}.$$

Therefore, the fact that

$$z_{static} \geq z_{adj}$$

implies

$$z_{static} = z_{adj}.$$

□

Complexity of problem (43) – (50)

Problem (43) – (50) is polinomially solvable.

Proof. We may express the constraint (44) – (48) in a matrix-vector notation: where A is a $|\mathcal{E}| \times |V|$ arc-node incidence matrix, B is a

$$\begin{array}{c} \\ \text{Group 1} \\ \text{Group 2} \\ \text{Group 3} \\ \text{Group 4} \\ \text{Group 5} \end{array} \begin{array}{c} \begin{array}{ccc} \alpha & w & \delta \end{array} \\ \begin{pmatrix} \text{A} & 0 & 0 \\ 0 & I_{\mathcal{E}} & -B \\ -I_{\mathcal{E}} & I_{\mathcal{E}} & 0 \\ 0 & 0 & I_V \\ 0 & 0 & e_V^T \end{pmatrix} \end{array}$$

$|\mathcal{E}| \times |V|$ matrix where each element corresponding let say to row (i,j) and column i is equal to 1, indicating whether or not the arc (i,j) leaves the node i. I_V and $I_{\mathcal{E}}$ are identity matrix of dimension $|V|$ and $|\mathcal{E}|$, respectively and e_V^T is a $|V| \times 1$ vector with all elements being unity.

The rows in the Group 1 correspond to constraints (44), the rows in the Group 2 and 3 to constraints (45) and (46), respectively. The rows in the last two groups are concerned with constraints (47) and (48), respectively.
Ghouila-Houri [28] showed that a matrix is totally unimodular if and only if for any subset of rows there exists a partition such that for each column the sum of the row elements belong to one partition minus the sum of the row elements belonging to the other partition is in $\{0, 1, -1\}$.
For any collection of rows of the above constraint matrix, we can construct two partitions such that the sum of rows in one partition minus the sum of rows in the other partition has only -1, 0, +1 in each column.
Now, if the rows belonging to the Group 2 are absent, it is sufficient for any combination of rows to multiply the rows of Group 4, if present, by -1.
In order to obtain a similar partition when the Group 2 is present, it is necessary to split them into two sub-classes C_1 and C_2. In particular, for each column i of the matrix B we should consider the rows whose indexes correspond to the outgoing edges of i, so that the difference between the rows in C_1 and C_2 is $\{0, 1, -1\}$. We notice that each column i of the matrix B may have an even or odd number of nonzero entries, depending on the cardinality of the forward star related to the node i. Then, if the number of these rows is even, we can evenly split the rows into C_1 and C_2, otherwise we can put half of the rows plus one in C_1 and the remaining ones in C_2.
If a collection of rows of Group 2 is present and the Group 3 is absent, if we multiply (-1) to the rows belonging to the Group C_2 and 4, the current sum of rows belongs to $\{0, 1, -1\}$.
If a collection of rows from both Groups 2 and 3 are present, we can appropriately multiply the rows of Group 3 and 4, to keep $\{0, 1, -1\}$ in each column of the sum of rows.
Therefore, the constraint matrix is totally unimodular.

□

Appendix C - Pseudo-code of Dynamic Programming (51)

Let $\mathcal{L}$ be the list of nodes to be processed. The steps of the dynamic programming are depicted in Algorithm 1.

Algorithm 1 . $\mathcal{DP}$ scheme

Step 0 *(Initialization Phase)*
Set: $\mathcal{L} = \{0\}$, $mod = false$.

Step 1 *(Node Selection)*
Select and delete from $\mathcal{L}$ a node i.
Set $mod = false$.

Step 2 *(State Generation)*
for all $j : (i, j) \in \mathcal{E}^t$ **do**
  Set $mod = false$.
  Set $\alpha = \max\left\{(C(ST(i,\gamma)) + \bar{\theta}_i, C(ST(i,\gamma-1)) + \bar{\theta}_i + \hat{\theta}_i\right\}, \gamma = 1, \ldots, \Gamma$.
  if $C(ST(j,\gamma)) < \alpha$ **then**
    Set $C(ST(j,\gamma)) = \alpha$.
    Set $mod = true$.
  end if
  Set $\beta = C(ST(i,0)) + \bar{\theta}_i$.
  if $C(ST(j,0)) < \beta$ **then**
    Set $C(ST(j,0)) = \beta$.
    Set $mod = true$.
  end if
  if *mod* **then**
    Add node j to $\mathcal{L}$.
  end if
end for

Step 3 *(Termination check)*
if $\mathcal{L} = \emptyset$ **then**
  STOP
else
  Go to **Step 1**.
end if