École Doctorale Informatique, Télécommunications et Électronique, Paris
Sorbonne Université, Laboratoire LIP6, équipe Recherche Opérationnelle
EDF R&D, département OSIRIS

Anchored solutions in robust combinatorial optimization

Adèle Pass-Lanneau

Thèse de doctorat en informatique
présentée et soutenue publiquement le 16 mars 2021

Jury

Rapporteurs

Dominique De Werra	Professeur honoraire, EPFL, Suisse
Gerhard J. Woeginger	Professeur, RWTH Aachen, Allemagne

Examinateurs

Bruno Escoffier	Professeur, Sorbonne Université, LIP6, France
Frédéric Meunier	Professeur, École des Ponts, CERMICS, France
Michael Poss	Directeur de recherche, CNRS, LIRMM, France

Encadrants

Pascale Bendotti	Ingénieur-Chercheur HDR, EDF R&D, France
Philippe Chrétienne	Professeur émérite, Sorbonne Université, LIP6, France
Pierre Fouilhoux	Professeur, Université Sorbonne Paris Nord, LIPN, France

Extended abstract

An instance of an optimization problem is often subject to uncertainty and likely to change over time. If the instance changes, an optimal solution to the instance also changes. However, changing a solution could be difficult in practice. The decision maker may be change-averse and prefer solutions that are, in some sense, stable. In this thesis we investigate a stability criterion based on anchored decisions, that are unchanged decisions with respect to a baseline solution. In reoptimization, the goal is to find a new solution with a maximum number of decisions maintained from the previously computed baseline solution. In robust optimization, it is to find in advance a baseline solution along with a subset of anchored decisions. When the solution must be modified to adapt to instance disruptions within a given uncertainty set, anchored decisions are guaranteed not to change.

In Part I, the generic concepts of the thesis are proposed and compared with the literature. In Chapter 1 we first review the state of the art in robust optimization and solution stability. In Chapter 2, the concepts are presented. The anchoring level is defined as the number of identical decisions between two solutions of a discrete optimization problem. Anchor-reoptimization problems are defined, where the anchoring level is maximized. Anchor-robust problems are then defined as 2-stage robust problems where the size of the subset of anchored decisions is maximized. Anchor-robustness fills the gap between static-robustness and adjustable-robustness, as it allows to find a trade-off between the cost of a baseline solution and the number of anchored decisions.

Anchor-reoptimization and anchor-robust problems are studied on two classes of problems: combinatorial problems with only binary variables in Part II, and scheduling problems with continuous variables in Part III and Part IV.

In Part II, we consider combinatorial problems written as integer programs in binary variables, such as matroid bases or matchings. Chapter 3 is devoted to anchor-reoptimization, under the form of k-red problems from the literature. We focus on anchor-reoptimization for matroid bases, which can be solved in polynomial time, either by a lagrangian-based algorithm or using a polyhedral characterization. In Chapter 4, anchor-robustness for combinatorial problems is investigated.

The complexity of the anchor-robust problem is analyzed for various uncertainty sets, and an MIP reformulation is obtained. The anchor-robust problem is shown to be computationally less demanding than the recoverable robust problem from the literature. Finally the so-called price of anchor-robustness is studied, which is the overhead cost of an anchor-robust solution with respect to a recoverable robust solution.

In Part III, we consider project scheduling problems, that involve continuous decision variables. In Chapter 5, reoptimization is first considered, by the means of anchored rescheduling problems. The complexity of rescheduling for various project scheduling problems is analyzed, showing the boundary between polynomial and hard cases of anchored rescheduling problems. In Chapter 6, the anchor-robust approach is developed for project scheduling under precedence constraints, leading to the AnchRobPSP problem. Combinatorial properties of the problem are studied, and dedicated graph models are proposed. We then study the complexity of AnchRobPSP. Several cases of AnchRobPSP are shown NP-hard, including budgeted uncertainty. Algorithms are designed for polynomial cases. For budgeted uncertainty an MIP reformulation is obtained, and its numerical performance is assessed. In Chapter 7, the anchor-robust approach is redesigned to address the Resource-Constrained Project Scheduling Problem (RCPSP), which is already NP-hard in a deterministic setting. We show the connection with the Adjustable-Robust RCPSP from the literature. Algorithmic tools are proposed, building upon the contributions of Chapter 6. Exact MIP-based approaches and dedicated heuristics are devised, for both the Adjustable-Robust RCPSP and the Anchor-Robust RCPSP in a unified way. The proposed exact and heuristic approaches are thoroughly evaluated on benchmark instances. Finally, a use case of anchor-robustness in project scheduling is presented, on a maintenance planning problem arising at EDF.

In Part IV, we further investigate mixed-integer programming techniques for the AnchRobPSP problem introduced in Part III. The formulation from Chapter 6 is investigated, together with new linear formulations valid for a larger variety of uncertainty sets. In Chapter 8, a dominance property is exhibited, and a new compact formulation is proposed. We show that it captures the combinatorial structure of the problem. Indeed it yields a polyhedral characterization of integer solutions of AnchRobPSP in non-trivial cases. Numerical experiments are carried out to show the efficiency of the approach on instances close to characterization cases, and on new uncertainty sets. In Chapter 9, we investigate formulations in the space of anchoring variables only. A complete picture of formulations for AnchRobPSP is given, both in anchoring variables only, and in extended form with additional schedule variables. A facial study of the anchored set polytope is then proposed. In cases where a polyhedral characterization was previously

obtained, we give a minimal description of the anchored set polytope. In general case, new inequalities are exhibited to strengthen inequalities from the proposed linear formulations.

Finally conclusions and research perspectives are presented. We point out some open complexity questions on the anchor-reoptimization and anchor-robust problems studied in the thesis, and give perspectives on the design of efficient algorithms for NP-hard cases. Finally other problems are identified where the concept of anchored solutions is relevant, and would lead to new anchor-reoptimization and anchor-robust problems.

Contents

Part I

Anchored solutions in robust optimization

Preliminaries on optimization problems

We consider discrete optimization problems under the following form:

$$\begin{array}{lll} \text{(P)} & \min & c^T x \\ & \text{s.t.} & x \in \mathcal{X} \end{array}$$

The *decision variables* are coordinates of *decision vector* $x \in \mathbb{R}^I$ with I a set of $|I| = n$ decisions. The objective is to minimize a linear cost function $c^T x$, with cost vector $c \in \mathbb{R}^I$. The feasible set is $\mathcal{X} \subseteq \mathbb{R}^I_+$. An instance of problem (P) is thus $(c, \mathcal{X})$. We assume that problem (P) can be written as a mixed-integer program (MIP) under the following form

$$\begin{array}{lll} \text{(MIP)} & \min & c^T x \\ & \text{s.t.} & Ax \leq b \\ & & x \geq 0 \\ & & x \in \{0,1\}^q \times \mathbb{R}^{n-q} \end{array}$$

where matrix A is the constraint matrix, and vector b the right-hand side. Decision variables x_i, $i \in \{1, \ldots, q\}$ are binary variables, while decision variables x_i, $i \in \{q+1, \ldots, n\}$ are continuous variables. Let $\mathcal{S}$ denote the space where the decision vector lies, that is, $\mathcal{S} = \{0,1\}^q \times \mathbb{R}^{n-q}$. If all decision variables are continuous, i.e., $\mathcal{S} = \mathbb{R}^I$, (MIP) is a linear program. If all decision variables are binary, i.e., $\mathcal{S} = \{0,1\}^I$, (MIP) is an integer program in binary variables. In particular, we say that (P) is a combinatorial problem if it can be written as (MIP) with binary variables only.

Let us introduce three problems: Selection, Min Cost Spanning Tree and PERT Scheduling. These problems will be used in Chapter 1 and Chapter 2 to illustrate the proposed concepts. Selection, Min Cost Spanning Tree are combinatorial problems, while PERT Scheduling has continuous decision variables. All three are polynomially solvable.

SELECTION. Consider a set of n items. Each item $i \in \{1, \ldots, n\}$ is associated with cost $c_i \geq 0$. Let p be an integer, $p \leq n$. The SELECTION problem is to select p items, so as to minimize the total cost of selected items. With binary decision variable $x_i = 1$ if item i is selected, $x_i = 0$ otherwise, the problem writes as

$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & \sum_{i=1}^{n} x_i = p \\ & x \in \{0,1\}^n \end{array}$$

Note that an optimal solution to SELECTION is simply obtained by sorting the items in increasing order of costs c_i, and selecting the first p items.

MIN COST SPANNING TREE. Consider a graph $G = (V, E)$. A *spanning tree* is a subset $T \subseteq E$ of edges such that T is cycle-free and $|T| = |V| - 1$. Consider a vector of costs associated to edges $c \in \mathbb{R}_+^E$. The MIN SPANNING TREE problem is to find a spanning tree $T \subseteq E$ with minimum total cost. Consider binary decision variable $x_e = 1$ if edge e is in the tree, $x_e = 0$ otherwise. The problem writes as $\min\ c^T x$ for $x \in \mathcal{X}$, where $\mathcal{X} \subseteq \{0,1\}^E$ is the set of incidence vectors of spanning trees. Classically the set $\mathcal{X}$ can be described with a family of linear inequalities (see, e.g., (Schrijver, 2003)), and this family has exponential size. A min-cost spanning tree can be found algorithmically in polynomial time with Kruskal's algorithm (see, e.g., (Korte and Vygen, 2012)).

PERT SCHEDULING. Consider a project with a set J of n jobs. Let s and t two dummy jobs representing the beginning and the end of the project, and $\overline{J} = J \cup \{s, t\}$. Consider a set of precedence relations, represented by a *precedence graph* $(\overline{J}, \mathcal{A})$. It is assumed that the precedence graph is acyclic, and s (resp. t) is a predecessor (resp. a successor) of all jobs. Each job has a processing time $p_i \geq 0$. A schedule is a vector of starting times $x_i \geq 0, i \in \overline{J}$ of the jobs. The makespan of schedule is x_t. The PROJECT SCHEDULING PROBLEM, or PERT SCHEDULING PROBLEM, is to find a schedule $x \in \mathbb{R}_+^{\overline{J}}$ such that for every precedence relation $(i, j) \in \mathcal{A}$, job j starts after job i completes, and so as to minimize the makespan x_t. The problem writes as

$$\begin{array}{ll} \min & x_t \\ \text{s.t.} & x_j - x_i \geq p_i \ \forall (i,j) \in \mathcal{A} \\ & x \geq 0 \\ & x \in \mathbb{R}^{\overline{J}} \end{array}$$

It is a linear program, and as such, it is solvable in polynomial time. The optimal value of the problem is equal to the longest s–t path in the precedence graph (Pinedo, 2002). This longest path value can be computed in polynomial time by dynamic programming.

Chapter 1

Robust optimization: state of the art

In this chapter, we present state-of-the-art concepts in robust optimization and solution stability in which the contributions of the thesis fit. In Section 1.1 the general principles of robustness are exposed. In Section 1.2 and Section 1.3, static-robust optimization and two-stage robust optimization are presented. In Section 1.4 related work on solution stability is reviewed.

1.1 Robustness in discrete optimization

Let us first present the core principles of robust optimization, and present the idea of uncertainty sets.

1.1.1 Robust optimization basics

Discrete optimization is commonly used in operations research to represent a real-life problem. It may come from various applicative fields, such as scheduling and planning, workforce assignment, or network design. This real-life problem is represented by an instance of a discrete optimization problem.

However in practice data is often not known exactly. Data can be difficult to measure or estimate: hence it is not known with perfect accuracy. Data can also be dynamic and change over time. Then at some point in time, some data is available which will not be accurate anymore in the future. This occurs when data is acquired in advance, before the solution of the optimization problem is used for practical implementation. Data inaccuracy raises the question of handling uncertainty in optimization problems. It is all the more important because the considered discrete optimization problems are very sensitive to uncertainty. A

disruption in the constraints or objective function, be it very small, may impair the optimality or feasibility of a solution. Therefore one cannot solve the original optimization problem being oblivious to uncertainty, then hope that the solution will behave well if data changes, even "not too much". Uncertainty must be integrated from the start into the optimization process, with full awareness.

The two main trends for optimization under uncertainty are stochastic optimization and robust optimization. The present work lies in the scope of the latter. In stochastic optimization, distributional information of uncertain data is available, and a stochastic criterion (e.g., the expected value) is optimized. The paradigm of robust optimization is to consider that unknown data may take any value in a so-called *uncertainty set*. By contrast with stochastic optimization, no further information is available on distribution of data in the uncertainty set. Robust optimization is then to find solutions that are immune to any realization in the uncertainty set, and to optimize a *worst-case criterion*. Those two ingredients of robust optimization, i.e., an uncertainty set, and a worst-case criterion, were already mentioned in early work, such as the book from Kouvelis and Yu (1996), *Robust Discrete Optimization and Its Applications*:

> *"We suggest the use of the robustness approach to decision making, which assumes inadequate knowledge [...] about the random state of nature and develops a decision that hedges against the worst contingency that may arise."*

1.1.2 Uncertainty sets

Let us formalize the idea of *uncertainty set*. We want to account for uncertainty on numerical data of the mixed-integer program (MIP), that is, coefficients in the constraints or objective function.

The constraint matrix, right-hand side, and cost vector are considered to have *nominal values* A, b, and c. The instance of (MIP) with nominal values as data corresponds to the *deterministic problem*. We then consider that the real data may deviate from the nominal values, to be equal to $A + \delta_A$, $b + \delta_b$, $c + \delta_c$. The *deviations* $\delta_A, \delta_b, \delta_c$ are a matrix and vectors with the same sizes as A, b, c. An *uncertainty realization* is $\delta = (\delta_A, \delta_b, \delta_c)$. The *uncertainty set* Δ is a set containing all uncertainty realizations $\delta = (\delta_A, \delta_b, \delta_c)$ that the decision maker wants to hedge against.

There are a number of options for the geometry of the uncertainty set, as illustrated in Figure 1.1. It can be a discrete set, then every uncertainty realization is said to be a *scenario*. The uncertainty set can also be a convex set, usually containing the element where all deviations are zero. For example, it can be chosen to be a polyhedron. A special case of polyhedron is a *box*, i.e., a cartesian

product of intervals, to represent that every coefficient varies independently within a range. The uncertainty set can also be chosen to be an ellipsoid.

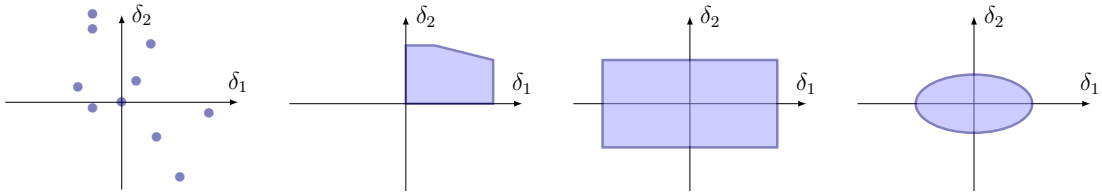


Figure 1.1: Examples of uncertainty sets: a discrete set, a polytope, a box, an ellipsoid.

For a given uncertainty set Δ, each realization $\delta = (\delta_A, \delta_b, \delta_c) \in \Delta$ is associated to an instance $(c^\delta, \mathcal{X}^\delta)$ of original problem (P), where $c^\delta = c + \delta_c$ and $\mathcal{X}^\delta = \{x \in \mathcal{S}\colon\ x \geq 0,\ (A + \delta_A)x \leq b + \delta_b\}$. The uncertainty set Δ corresponds to *cost uncertainty* if it impairs the cost vector only, *right-hand side uncertainty* if it impairs the right-hand side vector only.

Example 1.1 (Uncertainty for Min Cost Spanning Tree). Consider the min cost spanning tree problem. An instance is formed with the graph $G = (V, E)$ and the edge costs $c \in \mathbb{R}^E_+$. A first uncertainty set Δ may correspond to disruptions of cost c^δ, with $c^\delta = c^0 + \delta$. Then the collection of instances is $(c^\delta, \mathcal{X}^0)_{\delta\in\Delta}$. Another uncertainty set Δ may correspond to disruptions on the graph, for example, edge deletions. Then each uncertainty realization $\delta \in \Delta$ corresponds to an edge-set E^δ, and set $\mathcal{X}^\delta$ is the set of spanning trees of $G = (V, E^\delta)$. ◇

Example 1.2 (Uncertainty for PERT Scheduling). Consider project scheduling under processing times uncertainty. Processing times have nominal value $p \in \mathbb{R}^J_+$, but may be disturbed into $p + \delta$ for some $\delta \in \Delta$. Then $\mathcal{X}^\delta$ is the set of schedules of $(J, \mathcal{A}, p + \delta)$ for every δ. The cost of a schedule is its makespan, i.e., the cost vector is $c^\delta = \chi^{\{t\}}$ for every $\delta \in \Delta$. In this case there is no uncertainty on the cost function. ◇

The *robust counterpart* is an optimization problem formulated using nominal values A, b, c and the uncertainty set Δ. In general the robust counterpart extends the deterministic problem, because it coincides with the deterministic problem if the uncertainty set is reduced to singleton $\{0\}$.

A variety of robust approaches were developed in the last 20 years, leading to distinct robust counterparts. In this chapter, we will present the main trends of robust optimization literature, and hopefully, give an insight on the diversity of the proposed approaches.

1.2 Static robustness

Let us first present the first robust approach considered in the literature, which is static robustness.

1.2.1 The static-robust counterpart

Consider an uncertainty set Δ, and the corresponding collection of instances $(c^\delta, \mathcal{X}^\delta)_{\delta\in\Delta}$ of problem (P). Static-robustness is to find a solution that is feasible for every uncertainty realization, i.e., $x \in \mathcal{X}^\delta$ for every $\delta \in \Delta$. Such a solution is said to be a *static-robust solution*. The criterion to minimize is the *worst-case cost* $\max_{\delta\in\Delta} {c^\delta}^T x$.

The static-robust counterpart (SR-P) to problem (P) is the following:

$$\begin{array}{lll} \text{(SR-P)} & \min & \max\limits_{\delta\in\Delta} {c^\delta}^T x \\ & \text{s.t.} & x \in \mathcal{X}^\delta \qquad \forall \delta \in \Delta \end{array}$$

This problem was introduced in a short note from Soyster in 1973 (Soyster, 1973), and later developed by Ben-Tal and Nemirovski (1998). The word "static" is in opposition with 2-stage approaches, detailed in Section 1.3. Static-robustness is also called *strict robustness*.

Note that if the nominal instance is in the collection $(c^\delta, X^\delta)_{\delta\in\Delta}$, which is a standard assumption, the optimal value of (SR-P) is not less than the optimal value of (P). The increase of optimal value from (P) to its robust counterpart is called the *price of robustness* (Bertsimas and Sim, 2004). It is a concern that the price of robustness is not too high, otherwise robust solutions would have a cost that is too high to implement them. In that case, the robust approach is said to be too conservative.

A main concern about the static-robust counterpart is tractability. We use here the word "tractability" for both the computational complexity of the problem, and the fact that it can be efficiently handled with specific algorithms or solvers (for example, an MIP solver). The problem (SR-P) is written with a constraint for each realization $\delta \in \Delta$. Hence it potentially has an infinite number of constraints if Δ has a non-empty interior, e.g., if Δ is a polyhedron. A question is whether this problem can be reformulated to be solved with usual algorithmic tools.

1.2.2 Tractability results

The tractability of the static-robust counterpart depends on both the original problem (P), and the chosen uncertainty set Δ. It was investigated since the 2000's by several works that we now review.

Column-wise convex uncertainty. In (Soyster, 1973) the case of a linear program (LP) was studied. The feasible set of the original problem is $\mathcal{X} = \{x \in \mathbb{R}^n : \ Ax \leq b, x \geq 0\}$. The constraint matrix is uncertain, and may be $A + \delta_A$. The uncertainty set is defined so that the jth column of δ_A lies in a convex set Δ_j. In this case, it can be shown that the constraint $(A + \delta_A)x \leq b$ is satisfied for every $\delta_A \in \Delta$ if and only if $(A + \delta_A^+)x \leq b$, where δ_A^+ is the matrix with jth column $[\delta_A^+]_j = \max_{\Delta_j}[\delta_A]_j$. Assume the matrix δ_A^+ is precomputed. Then the static-robust counterpart reduces to the LP with constraint matrix $A+\delta_A^+$. This is a simple case of uncertainty representation, where uncertainty realizations of two columns on the constraint matrix are uncorrelated. It leads to a robust problem that is just another instance of the original problem. However, it was noted that the obtained robust solutions are very costly: the approach is said to be overly conservative.

Ellipsoidal uncertainty. Other continuous problems were studied in (Ben-Tal and Nemirovski, 1998) for ellipsoidal uncertainty sets. The authors obtained a range of positive results on the nature and the tractability of static-robust problems. They showed that the static-robust counterpart of a convex program under ellipsoidal uncertainty is still a convex program. The static-robust counterpart of an LP under ellipsoidal uncertainty is a conic quadratic program. For a range of quadratic problems, the static-robust counterpart is an SDP program. The static-robust counterpart may have a different nature than that of the original problem. However it is still identified in a class of problems for which generic solvers can be used: convex, quadratic, semi-definite programs.

Discrete uncertainty. The case of combinatorial problems and discrete uncertainty, was investigated in (Kouvelis and Yu, 1996). The authors considered uncertainty sets given as a list of *scenarios*. They studied classical polynomial problems such as minimum assignment, shortest path, or minimum cost spanning tree. Even for an uncertainty set formed with two scenarios, the robust counterparts of such problems are (weakly) NP-hard, while the deterministic problem is polynomial. Pseudo-polynomial algorithms are proposed. When the number of scenarios is unbounded, strong NP-hardness results are obtained for the robust counterparts. The results from (Kouvelis and Yu, 1996) show that robust problems can be harder than the original problems. This calls for the study of uncertainty sets that are more structured than a list of scenarios. Robust problems under discrete uncertainty were further studied, see (Aissi et al., 2009) for a survey on complexity results and approximation algorithms.

Polyhedral and budgeted uncertainty. Another approach is to choose polyhedral uncertainty sets. Using LP duality the static-robust counterpart can be reformulated. Although, there can be an increase in complexity: in (Buchheim and Kurtz, 2018) an example was built of a polynomial combinatorial problem with an NP-hard static-robust counterpart under polyhedral uncertainty. Note that for computational complexity analysis it should be specified whether the uncertainty polytope is given by an outer description (i.e., a list of inequalities) or an inner description (i.e., a list of its extreme points). The definition of the static-robust counterpart implies that it is possible to convexify the uncertainty set without changing the static-robust solutions. Hence it is equivalent to consider a polytope given by an inner description, or the discrete set of its extreme points. For linear programming or (mixed-)integer programming, a major contribution is the one of Bertsimas and Sim (2004) who introduced budgeted uncertainty. Budgeted uncertainty sets are a special case of polytopes. This reference is now presented in a dedicated section.

1.2.3 Budgeted uncertainty and the price of robustness

In (Bertsimas and Sim, 2004, 2003) Bertsimas and Sim introduced budgeted uncertainty, defined as follows. Both the cost vector and the constraint matrix of (MIP) are subject to uncertainty. Each row $[\delta_A]_i$ of the constraint matrix deviation δ_A is supposed to be lying in the discrete set

$$\{u_{ij}\widehat{a}_{ij}:\ \ u_{ij} \in \{-1, 0, 1\}, \textstyle\sum_{j=1}^{n} |u_{ij}| \leq \Gamma_i\}.$$

It means that each coefficient i, j may deviate from its nominal value a_{ij}, in the set $\{a_{ij} - \widehat{a}_{ij}, a_{ij}, a_{ij} + \widehat{a}_{ij}\}$. Integer parameter Γ_i is an *uncertainty budget*: at most Γ_i values may deviate from their nominal value in an uncertainty realization. Note that in (Bertsimas and Sim, 2004) Γ_i could also take non-integer values. Uncertainty on the cost vector is structured similarly.

If the budget is $\Gamma_i = n$, the uncertainty set contains the realization where all coefficients deviate from their nominal values. If the budget is $\Gamma_i = 0$, the uncertainty set is reduced to $\{0\}$. An illustration is provided in Figure 1.2.

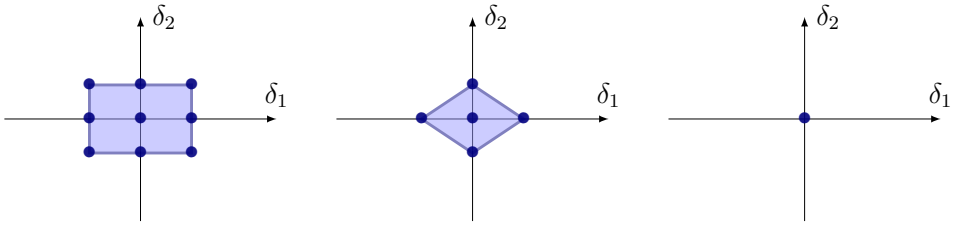


Figure 1.2: Budgeted uncertainty set (dark blue dots) and its convex hull (light blue polytope) for budget $\Gamma = 2, 1, 0$.

Regarding tractability, the authors studied the robust counterpart of an MIP under budgeted uncertainty. They obtained positive results: the robust counterpart of an MIP under budgeted uncertainty can be reformulated into an MIP, with size reasonably increased. They also focused on the case of cost uncertainty for a combinatorial problem. If the original problem is polynomial, the robust counterpart can be solved through a polynomial number of calls to an algorithm for the original problem. An approximation algorithm for the original problem also yields an approximation algorithm for the robust counterpart. Those results advocate for the practical implementability of the robust approach, since the robust counterpart can be solved with the same tools as those for the deterministic problem (either through an MIP solver, or with a dedicated algorithm).

Bertsimas and Sim defined the price of robustness as the increase of the cost of a robust solution, compared to the cost of a solution when there is no uncertainty. The uncertainty budget Γ can be used to tune the price of robustness, by changing the number of deviations in the uncertainty set. Budgeted uncertainty is thus appealing because it yields to tractable robust problems, understandable to practitioners, and with a parameter Γ to control the price of robustness. Note that these results have had a very important influence on the robust optimization literature, and budgeted uncertainty has become a standard uncertainty representation in robust approaches.

1.2.4 Uncertainty modelling in static robustness

Let us finish with a few remarks on the choice of the uncertainty set in static robustness. These remarks carry over to other robust approaches.

When designing uncertainty sets for a robust problem, multiple aspects are to be taken into account, among which the following.

- First, *the relevance of the uncertainty set*, in the sense that the uncertainty set must contain realizations corresponding to what the decision maker wants to hedge against. This is a qualitative appreciation of whether the uncertainty set is convincing. It is also necessary that the parameters defining the uncertainty set (e.g., the range where coefficients lie, the uncertainty budget) can actually be given values by the practitioners.
- *The type of the associated robust problem* obtained for this uncertainty set: is it a convex program? a linear or integer program? can it be solved with some generic solver? It is often wanted that the robust counterpart does not differ too much from the original problem, i.e., that belongs to the same class of problems (e.g., LPs) and that the overhead computational effort to solve it is not too high.
- *The computational complexity of the associated robust problem*: is it com-

putationally tractable? Concerning computational complexity, as noted in (Buchheim and Kurtz, 2018), uncertainty sets with very correlated parameters (e.g., scenarios) lead to difficult problems; uncertainty sets with no correlation (e.g., box uncertainty) lead to easy cases.

- Finally, *the price of robustness*. An important issue of a high price of robustness, in an OR perspective, is that robust solutions that are very costly are likely to be discarded by decision makers. Thus a control on the price of robustness is necessary, to make robust optimization applicable.

1.3 Two-stage robust optimization

Let us now review two-stage and adjustable-robust optimization approaches.

1.3.1 Definition

In the static-robust approach, decisions are made before the uncertainty realization is observed, and there is no possibility of revising them afterwards. The authors of (Ben-Tal et al., 2004) introduced a new class of robust problems, where decisions are made along a 2-stage process:

- First-stage decisions, called *"here and now"* decisions, are made before uncertainty realization is known.
- Second-stage decisions, called *"wait and see"* decisions, are made after uncertainty realization is known. Second-stage decisions are also called *recourse* decisions, or *adjustable* since they adjust to uncertainty realization.

Such a 2-stage decision process is classical in stochastic optimization, see, e.g., (Birge and Louveaux, 2011).

A robust 2-stage problem usually takes the following form:

$$\begin{array}{lllll} \min & \max & \min & & c^\delta(x, y^\delta) \\ x \in \mathcal{X} & \delta \in \Delta & y^\delta \in \mathcal{Y}^\delta(x) & & \end{array}$$

The inner minimization corresponds to the second-stage or recourse problem: given first-stage variables x, and uncertainty realization $\delta \in \Delta$, find the second-stage solution y^δ in feasible set $\mathcal{Y}^\delta(x)$ that minimizes cost $c^\delta(x, \cdot)$. The maximization over $\delta \in \Delta$ is the worst-case criterion. The outer minimization is to optimize first-stage decisions.

In some applications, there can be a natural partition of decision variables into first stage and second stage decisions. For example, some decisions may represent investment decisions to be taken in advance, while other variables are short-term decisions.

Importantly, two-stage robust problems are less conservative that their static-robust counterpart. To see this, the two-stage robust problem can be rewritten with one variable y^δ per realization $\delta \in \Delta$. Then a solution to the two-stage robust problem corresponds to $x \in \mathcal{X}$ and a collection $(y^\delta)_{\delta\in\Delta}$ of second-stage solutions such that $y^\delta \in \mathcal{Y}^\delta(x)$ for every $\delta \in \Delta$. By contrast, in the static-robust counterpart the vector y does not depend on δ, and thus $y \in \mathcal{Y}^\delta(x)$ for every $\delta \in \Delta$. Hence the static-robust optimal value is not less than the two-stage robust optimal value. The price of robustness is thus decreased, if some variables are allowed to be adjustable.

An important special case of two-stage problems is the following, where there are no first-stage variables:

$$\text{(Adj-P)} \quad \max_{\delta\in\Delta} \quad \min_{x^\delta \in \mathcal{X}^\delta} \quad c^\delta(x^\delta)$$

In the sequel, we refer to this problem as the adjustable-robust counterpart of problem (P). The value of the adjustable-robust problem is then the worst-case value of an optimal solution of the second-stage instance, with feasible set $\mathcal{X}^\delta$ and objective function c^δ. We mention that in the literature, the terms "two-stage" and "adjustable" are often used as synonyms. In this work adjustable-robustness will more specifically denote the (Adj-P) problem.

1.3.2 Examples

Let us now illustrate the adjustable-robustness concept on examples for the Selection and PERT Scheduling problems.

Example 1.3 (Adjustable robustness for PERT Scheduling). Consider an instance of the PERT Scheduling problem with a set $J = \{1, \dots, 4\}$ of four jobs. The precedence graph is a chain, i.e., it has arc-set $\mathcal{A} = \{(s, 1), (1, 2), (2, 3), (3, 4), (4, t)\}$. Nominal processing times are $p = (1, 1, 1, 1)$. An earliest schedule is thus $x = (0, 0, 1, 2, 3, 4)$, with makespan $x_t = 4$.

Assume now that processing times are uncertain. The real processing times may be $p + \delta$. The uncertainty set is budgeted uncertainty set with budget $\Gamma = 1$ and $\widehat{\delta} = (1, 1, 1, 1)$: that is, we consider that one job may have duration $p_i + \widehat{\delta}_i = 2$. Note that schedule x is infeasible if a job is longer than expected.

A static-robust solution is a schedule feasible for every instance $G(p+\delta)$, $\delta \in \Delta$. The static-robust counterpart is to find a static-robust solution with minimum makespan. It amounts to finding a schedule of $G(p + \widehat{\delta})$ with minimum makespan. The static-robust optimum is 8.

The adjustable-robust counterpart is to find the minimum M such that for every $\delta \in \Delta$ there exists a schedule of $G(p+\delta)$. On the considered instance, it can

be shown that the optimal value of the adjustable-robust counterpart is 5. Indeed, for any $\delta \in \Delta$, the earliest schedule of $G(p+\widehat{\delta})$ has makespan at most 5. It is less that the static-robust optimal value of 8.

For general precedence graph the adjustable-robust problem was studied in (Minoux, 2007a,b). It was shown to be solvable in polynomial time for budgeted uncertainty. The algorithm is to compute by dynamic programming, the length of all longest paths in the precedence graph where $\gamma \leq \Gamma$ deviations of processing times occur. ◇

Example 1.4 (Adjustable robustness for Selection). Consider a toy instance of the Selection problem, with $n = 3$ items and $p = 2$:

$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & \sum_{i=1}^3 x_i = 2 \\ & x_i \in \{0,1\} \qquad i = 1,2,3 \end{array}$$

Consider now that item costs are uncertain. Namely, for each item the nominal cost is 1, but it may go up to 3; however at most one item may have a cost greater than the nominal one. It corresponds to cost uncertainty $c^\delta = c + \delta$ with $c = (1,1,1)$, where deviation δ lies in the budgeted uncertainty set $\Delta = \{\delta_i = 2u_i, u_i \in \{0,1\}^3, \sum_{i=1}^3 u_i \leq 1\}$, with uncertainty budget $\Gamma = 1$.

The static-robust problem is

$$\begin{array}{ll} \min & \max\limits_{\delta\in\Delta} (c+\delta)^T x \\ \text{s.t.} & \sum_{i=1}^3 x_i = 2 \\ & x_i \in \{0,1\} \qquad i = 1,2,3 \end{array}$$

it can be reformulated as one minimization problem with the results of (Bertsimas and Sim, 2004); in the present case since all items are the same, it is clear that an optimal solution is to use any pair of items. Then in the worst case, one of them will have cost 3, and the static-robust value is 4.

The adjustable-robust problem is

$$\begin{array}{lll} \max\limits_{\delta\in\Delta} & \min & (c+\delta)^T x^\delta \\ & \text{s.t.} & \sum_{i=1}^3 x_i^\delta = 2 \\ & & x_i^\delta \in \{0,1\} \qquad i = 1,2,3 \end{array}$$

In this case, the solution x^δ depends on the realization of cost c. Hence an optimal solution is to observe the costs, and select the two items with cost 1, leading to an adjustable-robust value of 2. ◇

1.3.3 Computational complexity of robust two-stage problems

The main challenge associated to 2-stage robust problems is tractability. In (Ben-Tal et al., 2004), it was shown that an adjustable robust problem can be NP-hard, even for a linear program and polyhedral uncertainty set – a case where the static robust problem is easy to solve. The authors note that "unfortunately, there is a price to pay for the flexibility of the adjustable-robust counterpart". Indeed, the decrease of the price of robustness goes hand in hand with an increase of the computational complexity.

1.3.3.1 Two-stage approximation approaches

A trend in the literature is thus the design of approximations for two-stage robust problems that are computationally affordable.

Decision rules. A first idea investigated in the original paper of (Ben-Tal et al., 2004), is to assume that second-stage variables are not free but obey a predetermined dependency in the uncertain data. This is called *decision rules*. For example, second-stage variables may be affine functions of uncertainty realization δ. Then the decision variables are the parameters of the dependency. The idea of affine dependency is carried from the field of optimal control, where a controller applies an action over the system, and affine control laws are commonly used. The authors of (Ben-Tal et al., 2004) showed that the affine adjustable-robust counterpart has good tractability properties. This trend was widely studied in the subsequent years. Empirically it was observed that affine decision rules often give near-optimal solutions. Cases where the optimality of affine decision rules can be proven have been investigated, although not many of them are known (Bertsimas and Goyal, 2012; Housni and Goyal, 2019). More sophisticated decision rules (e.g., piecewise affine) have been developed, but they often are computationally more expensive (Ben-Tal et al., 2020).

K-adaptability. Another idea introduced in (Bertsimas and Caramanis, 2010) is to consider k possible second-stage solutions. These k solutions are precomputed in first stage, and in second stage the best one is picked. This approach is called *k-adaptability*. This solution concept amounts to finding a k-partition of the uncertainty set: each class of the partition is associated with a second-stage solution. Even though a bilinear formulation and tractability result can be obtained under some hypotheses, k-adaptable problems are in general hard to solve. The main interest is that k-adaptable problems form a discrete hierarchy of problems, from robust static ($k = 1$) to "completely adaptable" 2-stage problems ($k \to +\infty$) that is the original 2-stage, where the second-stage solution is any function of the

uncertainty. Subsequent work on k-adaptability include, e.g., (Hanasusanto et al., 2015; Subramanyam et al., 2020).

Note that decision rules and k-adaptability are two approximations that are not comparable (one can be better than another on an instance, and vice versa).

1.3.3.2 Two-stage exact approaches

Another line of research is to design exact approaches to solve 2-stage problems, usually based on decomposition approaches.

Note first that a reformulation can be considered, introducing one decision variable y^δ per uncertainty realization $\delta \in \Delta$. It is often possible to restrict to the extreme points of the uncertainty sets. This leads to a formulation with a finite but exponential number of variables or constraints. It can be solved through row generation, as done for different applications in (Bienstock and Özbay, 2008; Gabrel et al., 2014). The idea was developed to tackle more generic MIPs. In (Billionnet et al., 2014), the authors proposed an approach for solving 2-stage robust MILPs with continuous recourse. The recourse problem can then be dualized, and the worst-case problem turned into a single maximization problem, although bilinear. Linearization techniques can be applied, to get an exponential MILP formulation with a constraint per extreme point of the uncertainty set. The authors of (Billionnet et al., 2014) then propose a constraint-generation scheme. In (Zeng and Zhao, 2013) a column-and-constraint scheme was proposed. In particular, it accommodates with integer variables in the recourse problem. An alternative decomposition approach, where the separation problem is based on Farkas lemma, was studied in (Ayoub and Poss, 2016). For robust project scheduling, dedicated decomposition approaches were proposed in (Bruni et al., 2017, 2018) based on Benders decomposition.

Note that the proposed decomposition approaches (namely, (Billionnet et al., 2014; Zeng and Zhao, 2013)) often require the so-called complete recourse hypothesis: the recourse problem is always feasible, i.e., $\mathcal{Y}^\delta(x) \neq \varnothing$ for every first-stage decision x.

Research on this topic is currently quite active, and methods are being proposed with improved computation times. However, as noted in the review from Rahmaniani et al. (2017), a straightforward implementation of a decomposition approach such as Benders is often time- and memory-consuming. Enhancements strategies are needed to obtain an efficient algorithm, such as improved cut generation, solution generation, or decomposition strategies (Rahmaniani et al., 2017). The implementation of decomposition approaches is more involved than the implementation of a compact MIP formulation. There is thus, at present time, an entry cost for practical implementation of exact two-stage robust optimization.

1.4 Solution stability in reoptimization and robust optimization

In this section, we first present a 2-stage decision process, and point out the question of solution stability arising in this framework. We then review related work on how solution stability can be addressed in both stages, in reoptimization and robust optimization.

1.4.1 A two-stage decision process

Consider that decisions for problem (P) are made along a two-stage process as follows.

- In a first stage, the decision maker has a forecast $(c^0, \mathcal{X}^0)$ of the instance to solve, and computes a baseline solution x^0.
- In a second stage, the decision maker knows the instance $(c, \mathcal{X})$ to solve, and computes a new solution x.

The instance $(c, \mathcal{X})$ can be different from the forecast, either because of a prediction error, or because the instance $(c^0, \mathcal{X}^0)$ has been disrupted in the meantime. Then it is necessary to propose a solution to the new instance, but also if possible to maintain decisions taken in the solution x^0. Finding a solution x that is not too different from the baseline solution x^0 corresponds to *solution stability* between x^0 and x.

In practice, such a decision process is often encountered, because a baseline solution that is precomputed in advance is needed. This is the case, e.g., when processes for implementing a solution are long, technical, or require coordination between multiple entities. The baseline solution then serves as a basis for implementation. If needed, the baseline solution may be adjusted afterwards, but it will be useful to stick to decisions from the baseline solution if possible. This calls for a measure of stability between the baseline solution and an updated solution computed later.

This 2-stage decision process is a special case of the 2-stage framework of Section 1.3, where problem (P) appears twice, in both the first stage and the second stage. Hence one could imagine using the adjustable-robust counterpart of problem (P) to benefit from its low price of robustness. However, this could be contradictory with controlling the stability of second-stage solutions w.r.t. the baseline solution. Consider again Example 1.4 for the Selection problem. The adjustable-robust approach proposes a guarantee on the worst-case value of a solution. However, there is no decision taken in first stage. Consider changing the problem so that a solution is actually decided in first stage. Then it would be necessary to change potentially all decisions in second stage to adjust to uncertainty, and obtain the

low price of adjustable-robustness. Precisely, in Example 1.4, if an item is selected in first-stage with no possibility of changing this decision in second stage, then the price of robustness equals the static-robust one. That is, there is a trade-off between the guarantee of decisions and the price of robustness.

It is thus important in the design of a robust 2-stage approach, to know whether a guarantee on the solution, or on the value of the solution, is needed. Note that in static robustness, we implicitly obtained a guarantee not only on the value of a solution, but also on the solution itself. This does not hold for 2-stage problems. In the project scheduling literature such a choice was emphasized: shall we guarantee the makespan? or the starting times of the jobs?

1.4.2 Solution stability in reoptimization

Reoptimization is the situation that the decision maker faces in the second stage of the decision process. A baseline solution was previously decided, but the instance of the problem has changed, and a new solution is to be found.

The term "reoptimization" was first used in the literature in the following sense: if the baseline solution was an optimal solution to the baseline instance, and the new instance is not far from the baseline instance, can we use the knowledge of the baseline solution to compute more efficiently an optimal solution to the new instance? (see, e.g., (Böckenhauer et al., 2008; Ausiello et al., 2008)). This does not a priori incorporate any stability measure between the new solution and the baseline solution.

Different stability measures have then been considered, to evaluate the distance between the baseline solution and the new solution, and plugged into a reoptimization problem.

In (Şeref et al., 2009), the authors considered so-called *incremental problems*, where the new solution must be at bounded distance from the baseline solution. This distance corresponds to the stability measure, and it can have several definitions depending on the problems. For combinatorial problems a distance between two solutions is the Hamming distance, i.e., the number of elements selected in one solution but not in the other. The authors of (Şeref et al., 2009) studied incremental variants of various network problems (shortest path, min cost flow) and analyzed their complexity. In particular, for a given original problem that is polynomial, different choices of distance lead to incremental problems that can be polynomial or NP-hard.

A similar idea is to consider *transition costs*, so that changing a decision from the solution incurs a given transition cost. In (Schieber et al., 2018), a reoptimization framework was proposed with such transition costs. The authors devised approximate reoptimization algorithms to deal with the trade-off between the cost

of the new solution and the total transition cost. Transition costs appear in other works such as (Gupta et al., 2014; Bampis et al., 2018) where they are aggregated with the cost of the solution into a single objective function.

Reoptimization with a stability measure is also related to bi-objective optimization, because of the trade-off between the quality of the solution and the stability measure. Using transition costs as stability measure often leads to optimization problems with two linear objectives, the cost of the new solution and the total transition cost. Setting one of the criteria as a constraint leads to *budgeted optimization* problems, that are in general NP-hard (see, e.g., (Aggarwal et al., 1982) for the budgeted spanning tree problem, (Berger et al., 2011) for budgeted matching).

For optimization problems with continuous variables, a larger variety of stability measures can be considered. The distance between solutions can be expressed as a L^1-norm, as done in (Nasrabadi and Orlin, 2013) for linear programs.

As for project scheduling problems, a variety of dedicated reoptimization problems were studied. Reoptimization, in that case, is to repair a schedule against disruptions, such as disruptions in jobs durations, or the arrival of new jobs to be inserted. This is called *reactive scheduling*, or *rescheduling*. An approach to repair schedules is to apply heuristics such as decision rules (Smith, 1995; de Vonder et al., 2007). Such heuristics do not a priori guarantee solution stability, but they may be applied only on a subset of the schedule, leaving the rest of the schedule unchanged.

Rescheduling approaches with a stability feature have been proposed, with a variety of criteria. It can be to minimize a continuous deviation measure w.r.t. the baseline schedule (Sakkout and Wallace, 2000; de Vonder et al., 2007), the cost incurred when modifying the solution (Deblaere et al., 2011), or a deviation measure based on the makespan (Artigues and Roubellat, 2000). Another criterion could be the number of modifications in the schedule (Calhoun et al., 2002; Bendotti et al., 2017). Namely in (Bendotti et al., 2017), the anchoring level was introduced as the number of identical starting times between two project schedules. Note that the literature on exact approaches to reactive scheduling is rather scarce, most references investigate heuristics. For more references on reactive scheduling, we refer to the surveys (Herroelen and Leus, 2002; Hazır and Ulusoy, 2020).

1.4.3 Solution stability in robust optimization

After defining a stability measure and the associated reoptimization problem, a question is to compute a baseline solution. This is the problem faced by the decision maker at first stage.

The first-stage problem can be cast in the framework of robust optimization.

Let us now review related work in the field of robust optimization, where in second stage the recourse problem incorporates a stability measure. Note that alternatively, stochastic or multistage models were proposed in the literature, based namely on transition costs (Gupta et al., 2014; Bampis et al., 2018).

In (Liebchen et al., 2009) the authors introduced the concept of *recoverable robustness* as a very general framework for robust optimization with limited recourse. The reasons for limited recourse are multiple.

- *Practical rules.* A first reason is that the recourse should resemble (simple) algorithms that are used in practice for repairing solutions. The authors provide an illustration on timetabling for railway applications, where recourse decisions are, e.g., train delays, or breaking transfers between two trains.
- *Tractability.* Another reason is tractability: the recourse problem is to be solved quickly, and thus it should be reasonably tractable, or solvable through some solver or algorithm.
- *Solution stability.* The last aspect is solution stability, in the sense that it is good practice not to change too much the solution in the recourse problem. In particular, a recoverable robust approach is to bound the number of changes in the solution.

Recoverable robustness is thus a very broad concept, that need to be instantiated on specific problems to be further studied. Recoverable robust railway applications were studied in (Liebchen et al., 2009; D'Angelo et al., 2011).

A research line emerged on combinatorial problems where recoverable robustness is understood in a more restrictive sense: it corresponds to changing at most k decisions in the recourse problem. Then recoverable robustness coincides with *robust optimization with incremental recourse* proposed in (Nasrabadi and Orlin, 2013). The focus was on the computational complexity of recoverable robust variants of famous problems, under standard uncertainty representation such as budgeted uncertainty. This includes the knapsack problem (Büsing et al., 2011), shortest path problems (Büsing, 2012), the min cost spanning tree problem (Hradovich et al., 2017), the selection problem (Kasperski and Zieliński, 2017), among others. As noted in the survey (Kasperski and Zieliński, 2016), most of the results on recoverable robust problems are negative ones. They happen to be very hard, since they retain the computational complexity of both the static-robust problem, and the reoptimization problem. The authors note that the literature is still scarce on approximation algorithms or exact approaches to handle NP-hard recoverable robust problems.

In the project scheduling literature, a related approach is *proactive/reactive scheduling* (Herroelen and Leus, 2002). While reactive scheduling corresponds to the recourse problem, proactive scheduling corresponds to the first-stage problem. The goal is to optimize in a combined way the baseline solution, and the reactive

actions to be taken depending on uncertainty realization. Proactive scheduling problems including a stochastic continuous stability measure were investigated in (Herroelen and Leus, 2004). However, these problems are relative to stochastic optimization, and not robust optimization as defined in the present chapter. The literature on robustness for project scheduling problems, even without any stability measure, is scarce, see (Artigues et al., 2013) for a min regret problem, (Minoux, 2007b) and (Bruni et al., 2017) for adjustable-robust problems. Even though robust project scheduling problems with a stability feature appear to be of clear interest for project managers (Hazır and Ulusoy, 2020), they have been barely addressed in the literature so far. In (Bendotti et al., 2017), a so-called anchor-proactive problem was proposed. The criterion is a guarantee over the number of jobs with stable starting times. The problem was proven polynomial under box uncertainty. The proposed stability criterion is combinatorial, in contrast with most stability measures studied in project scheduling literature.

In this thesis, we build upon the work of (Bendotti et al., 2017). We propose the anchoring level as a combinatorial stability measure for discrete optimization problems, and anchored solutions as a concept for stability in robust optimization. The generic concepts of the thesis are exposed in Chapter 2.

Chapter 2

Anchored solutions: concepts

In this chapter we present the general concepts of anchored solutions, in the framework of the 2-stage decision process described in Section 1.4 of Chapter 1.

In Section 2.1 we first introduce the anchoring level as a criterion to count the number of identical decisions between a baseline solution and a new solution of optimization problem (P). In Section 2.2 we introduce anchor-reoptimization problems, that are arising in second stage to compute the new solution. In Section 2.3 we define anchor-robust problems. These are the robust problems faced in first stage to compute a baseline solution along with a guarantee on a subset of baseline decisions.

Problems defined in this chapter are posed in generic form. They are instantiated and studied in more details in Part II for combinatorial problems, and Part III for project scheduling problems.

2.1 Anchoring level

Let us first introduce the anchoring level, to serve as a stability measure in reoptimization and robust optimization. Recall that $\mathcal{S}$ denotes the space where decision variables lie, e.g., $\mathcal{S} = \{0,1\}^I$ for binary variables, $\mathcal{S} = \mathbb{R}^I$ for continuous variables. Given two vectors $x, y \in \mathcal{S}$, let the *anchoring level* be defined as

$$\sigma(x,y) = |\{i \in I : \ x_i = y_i\}|.$$

Since coordinates of vectors x and y correspond to decisions, the anchoring level is the number of decisions that are identical in x and y. Note also that $\sigma(x,y) = |I| - d(x,y)$ where $d(x,y)$ is the Hamming distance between vectors x an y. The anchoring level was coined as such in (Bendotti et al., 2017) where it was introduced as a stability measure for reoptimization in project scheduling.

Let us propose a slight generalization. The idea is still to count the number of identical decisions in x and y. Coordinates of x and y are either continuous variables ($x_i, y_i \in \mathbb{R}$) or binary variables ($x_i, y_i \in \{0,1\}$). Let $\equiv$ be a binary relation on $\mathbb{R}$ or $\{0,1\}$. For $i \in I$, $x_i \equiv y_i$ indicates that x_i and y_i are identical decisions. Examples are:

- the equality relation $=$
- the relation $=_1$ defined by $a =_1 b$ if $a = b = 1$. It corresponds to logical "and" for binary a and b.
- the relation $=_\varepsilon$ defined by $a =_\varepsilon b$ if $|a - b| \leq \varepsilon$ for $\varepsilon \geq 0$.

Given binary relation $\equiv$ and $x, y \in \mathcal{S}$, let the *anchoring level* associated with relation $\equiv$ be

$$\sigma^{\equiv}(x, y) = |\{i \in I\colon\ x_i \equiv y_i\}|.$$

Then $\sigma^{\equiv}(x, y)$ is the number of decisions that are identical for $\equiv$ in x and y.

Let us now illustrate the definition on two examples: spanning trees and project schedules.

Example 2.1 (Anchoring level for spanning trees). Let $G = (V, E)$ be an undirected graph. Consider two spanning trees X and Y of graph G with incidence vectors x and y. For equivalence relation $=$, the anchoring level $\sigma^{=}(x, y)$ is $|E \setminus (X \cup Y)| + |X \cap Y|$. For equivalence relation $=_1$, the anchoring level is the number of edges belonging to both X and Y, i.e., $\sigma^{=_1}(x, y) = |X \cap Y|$. ◇

Example 2.2 (Anchoring level for project schedules). Consider a set of jobs J, a precedence graph $G = (J, \mathcal{A}, p)$ with processing times $p \in \mathbb{R}^J_+$. Let x and y be two schedules of $G = (J, \mathcal{A}, p)$. The decision x_i (resp. y_i) is the starting time of job i in schedule x (resp. y). For equivalence relation $=$, the anchoring level $\sigma^{=}(x, y)$ is the number of jobs that have the same starting time in x and y. For equivalence relation $=_\varepsilon$, the anchoring level $\sigma^{=_\varepsilon}(x, y)$ is the number of jobs whose starting times in x and y differ from less than tolerance ε. ◇

Note that for the proposed equivalence relations, for a given $a \in \mathbb{R}$, either $a \equiv a$ or a is not equivalent to any other value. This latter case occurs for equivalence relation $=_1$: for binary variable x_i, if $x_i = 0$ it is not equivalent to any other element of $\{0,1\}$. This represents that a zero coordinate in a binary vector is not an actual decision that we want to count in the anchoring level. For all considered equivalence relations, remark that a vector that has the most common decisions with a given vector x is vector x itself, that is, $\sigma^{\equiv}(x, x) = \max_{y \in \mathcal{S}} \sigma^{\equiv}(x, y)$.

A weighted version of the anchoring level can also be defined. Let $w \in \mathbb{R}^I_+$ be *anchoring weights*. Then the *weighted anchoring criterion* is

$$\sigma^{\equiv}_w(x, y) = \textstyle\sum_{i \in I\colon\ x_i \equiv y_i} w_i$$

and clearly $\sigma^{\equiv}_{\mathbf{1}}(x,y) = \sigma^{\equiv}(x,y)$ with $\mathbf{1}$ the all-ones vector of $\mathbb{R}^I$. The anchoring weight w_i is a value representing the benefit of having an identical decision i in two solutions.

In this chapter, unless specified, we will consider the anchoring level:
- for combinatorial problems ($\mathcal{S} = \{0,1\}^I$), $\sigma(x,y) = \{i \in I\colon\ x_i =_1 y_i\}$
- for linear programs ($\mathcal{S} = \mathbb{R}^I$), $\sigma(x,y) = \{i \in I\colon\ x_i = y_i\}$.

2.2 Anchor-Reoptimization

A first situation where the anchoring level comes into play is reoptimization. Given an instance $(c, \mathcal{X})$ of problem (P) and a baseline solution x^0, the goal of reoptimization is to find a new solution $x \in \mathcal{X}$ while taking into account both its cost $c^T x$, and stability w.r.t. the baseline solution, measured by the anchoring level $\sigma(x^0, x)$. In the sequel we formalize anchor-reoptimization problems.

2.2.1 Reoptimization with anchoring level only

A first problem can be defined where only the anchoring level w.r.t. x^0 is to be optimized:

AnchReopt	
Input:	$\mathcal{X}$, $x^0 \in \mathcal{S}$
Problem:	$\max \sigma(x^0, x)$ s.t. $x \in \mathcal{X}$

If $x^0 \in \mathcal{X}$, an optimal solution is $x = x^0$, i.e., to keep the baseline solution, since as noted previously, $\sigma(x^0, x^0) = \max_{x \in \mathcal{S}} \sigma(x^0, x)$. But in general $x^0 \notin \mathcal{X}$, and the AnchReopt problem is to repair solution x^0 into a feasible solution $x \in \mathcal{X}$ by changing a minimum number of decisions. Let us now give more detailed examples.

Combinatorial problems. Consider the case where $\mathcal{S} = \{0,1\}^I$. Then x^0 can be considered as the incidence vector of a subset of elements, say *red* elements. The AnchReopt problem is to find a solution $x \in \mathcal{X}$ with a maximum number of red elements. Regarding the complexity of the problem, note that $\sigma(x^0, x) = {x^0}^T x$ since x^0 and x are binary vectors. that is, the anchoring level writes as a linear function of the new solution x. Hence the problem AnchReopt is an instance of the original problem (P) with cost vector x^0. Therefore for combinatorial problems, the AnchReopt problem is not any harder than the original problem (P).

Linear programming. Consider the case where $\mathcal{S} = \mathbb{R}^I$, and $\mathcal{X}$ is the polyhedron $\{x \in \mathbb{R}^I\colon\ x \geq 0, Ax \leq b\}$. Let $x^0 \in \mathbb{R}^I$. Note that by contrast with combinatorial problems, the anchoring criterion $\sigma(x^0, x) = |\{i \in I\colon\ x^0_i = x_i\}|$ is not a linear function of the new solution x. We now prove that anchor-reoptimization is NP-hard in that case.

Proposition 2.1. AnchReopt *is strongly NP-hard for linear programs.*

Proof. The proof is a reduction from the Stable Set problem: given a graph $G = (V, E)$, integer k, is there a stable set $U \subseteq V$ of size $|U| \geq k$? Let $G = (V, E)$ be a graph, $k \in \mathbb{N}$. Consider the polyhedron $\mathcal{X} = \{x \in \mathbb{R}^V_+\colon\ x \in [0,1]^V,\ x_i + x_j \leq 1\ \forall (i,j) \in E\}$ and the solution $x^0 \in \mathbb{R}^V$ defined by $x^0_v = 1$ for every $v \in V$. Then $\sigma(x^0, x) = |\{v \in V\colon\ x_v = 1\}|$. If there exists a stable set $U \subseteq V$ of size $\geq k$, then its incidence vector χ^U is in $\mathcal{X}$ and $\sigma(x^0, \chi^U) \geq k$. Conversely, assume there exists $x^* \in \mathcal{X}$ such that $\sigma(x^0, x^*) \geq k$. Vector x^* is not an integer vector, but it has more than k coordinates equal to one. Let $U = \{v \in V\colon\ x^*_v = 1\}$. Then $\chi^U \leq x^*$ and $\chi^U \in \mathcal{X} \cap \{0,1\}^V$. Hence U is a stable set of size $\geq k$. This proves that the decision version of AnchReopt is strongly NP-complete. □

The underlying idea is that the anchoring level is a combinatorial criterion: are decisions equal or not? When the variables of the original problem are continuous, there can be an increase in complexity induced by the anchoring level, when solving the AnchReopt problem.

Project scheduling. In project scheduling, the variables are the starting times of jobs, which are continuous. In particular PERT Scheduling writes as a linear program. In Chapter 5 we will study cases where AnchReopt can be solved in polynomial time, in contrast with the general NP-hardness result of Proposition 2.1.

Remark that in the definition of AnchReopt, the baseline solution x^0 is given in the input, but the baseline instance $\mathcal{X}^0$ is not. Hence x^0 can be any element of the space $\mathcal{S}$. For the problems where we study anchor-reoptimization into detail, we were able to obtain polynomiality results for any $x^0 \in \mathcal{S}$: in particular for matroid bases (Chapter 3) and project scheduling problems (Chapter 5). We did not require any further properties on x^0.

2.2.2 Extendable sets in reoptimization

Let us have a closer look to a decision problem related to AnchReopt, arising when $x^0 \notin \mathcal{X}$.

Extendability	
Input:	$\mathcal{X}$, $x^0 \in \mathcal{S}$, $H \subseteq I$
Question:	Is there $x \in \mathcal{X}$ such that $x_i^0 \equiv x_i \ \forall i \in H$?

The problem is to decide if a subset of decisions $(x_i^0)_{i \in H}$ from x^0 can be extended into a new feasible solution $x \in \mathcal{X}$. Let us say that set H is *extendable* if the instance of Extendability problem has answer 'yes'. The problem AnchReopt is to find an extendable set H of maximum size.

Combinatorial problems. Note that for combinatorial problem, Extendability is not any harder than the original problem. Setting anchoring weights $w_i = 1$ if $i \in H$, and $w_i = 0$ otherwise, it comes that H is extendable if and only if AnchReopt for anchoring weights w has optimal value at least $|H|$.

Linear programming. Note that for LPs, Extendability is solvable in polynomial time because it writes as an LP with additional equality constraints $x_i^0 = x_i$ for every $i \in H$. Hence deciding if a set is extendable is polynomial, but finding one of maximum size is hard (Proposition 2.1).

Project scheduling. The Extendability problem is studied in Chapter 5 and a combinatorial characterization of extendable sets is provided. This characterization will be an important building block of the robust approaches proposed next, e.g., in Chapter 6.

2.2.3 Reoptimization with anchoring level and cost

If the feasible set does not change, i.e., $\mathcal{X}^0 = \mathcal{X}$, it turns out that the baseline solution x^0 is always feasible and the AnchReopt problem is trivial. However if the cost function changes, the baseline solution x^0 may be sub-optimal, making reoptimization still necessary. A question is thus to find a trade-off between keeping the decisions from the baseline solution; and finding a new solution with reasonable cost. The new solution x is thus to be evaluated under the two criteria $c^T x$ and $\sigma(x^0, x)$. With one as the objective function and the other as a constraint, we obtain two anchor-reoptimization problems:

AnchReopt–Cost	
Input:	$\mathcal{X}$, $x^0 \in \mathcal{S}$, $C \geq 0$
Problem:	$\max \sigma(x^0, x)$ s.t. $x \in \mathcal{X}$, $c^T x \leq C$

AnchReopt–Anch	
Input:	$\mathcal{X}$, $x^0 \in \mathcal{S}$, $A \in \mathbb{N}$
Problem:	$\min c^T x$ s.t. $x \in \mathcal{X}$, $\sigma(x^0, x) \geq A$

Note that in decision version, both lead to the same decision problem:

Dec-AnchReopt-Anch&Cost	
Input:	$\mathcal{X}$, $x^0 \in \mathcal{S}$, $C \geq 0$, $A \in \mathbb{N}$
Question:	Is there $x \in \mathcal{X}$ such that $c^T x \leq C$ and $\sigma(x^0, x) \geq A$?

Combinatorial problems. For combinatorial problems, these problems will be discussed in Chapter 3, with a specific focus on the problem of finding a max-weight basis in a matroid.

Linear programming. Note that for linear programs, AnchReopt–Cost is harder than AnchReopt which was shown to be NP-hard in Section 2.2.1.

Project scheduling. For project scheduling, we obtained that the AnchReopt–Cost is polynomial, as for the AnchReopt problem: see Chapter 5.

2.2.4 Restricted reoptimization for NP-hard problems

Let us now present an additional concept to handle an NP-hard original problem (P). Similarly to what was noted in (Liebchen et al., 2009), it is useful to design reoptimization problems that are computationally easy, so that they can be integrated as the recourse problem of robust 2-stage approaches. In the anchor-reoptimization problems proposed in Section 2.2.1 and Section 2.2.3, the new solution x is sought in the feasible set $\mathcal{X}$. In general these problems retain the whole complexity of the original problem (P), and thus are NP-hard whenever (P) is NP-hard.

An approach is to consider that only a subset of variables describing the solution can be re-optimized. Consider that feasible solutions of problem (P) are represented not only with the decision vector x, but also with some additional variables $\phi \in \mathbb{R}^J$. Then a feasible solution is $(x, \phi) \in \mathcal{X}$. Given a baseline solution (x^0, ϕ), *restricted reoptimization* is to search for a second-stage solution $(x, \phi) \in \mathcal{X}$. The ϕ variables are thus forced to be the same in the baseline solution and the second-stage solution. The anchoring level is still evaluated between vectors x^0 and x.

Variables ϕ may capture the structure of the solution: then the second-stage solution (x, ϕ) keeps the same structure as the baseline solution (x^0, ϕ). Restricted reoptimization may be computationally attractive, since it is not necessary to re-optimize the ϕ variables. Note that it corresponds, in a robust 2-stage perspective, to decide that variables ϕ are first-stage variables. The recourse problem is then to re-optimize only x variables.

Combinatorial problems. In the present work, we consider only original combinatorial problems that are solvable in polynomial time. The investigation of restricted anchor-reoptimization for combinatorial problems would be an interesting research perspective, especially to deal with NP-hard graph problems.

Project scheduling. An example of restricted reoptimization is considered in Chapter 5 for the Resource-Constrained Project Scheduling Problem (RCPSP), which is NP-hard and computationally demanding. Variables ϕ then correspond to sequencing decisions related to resource allocations, while x variables are starting times. It is used in Chapter 7 to design efficient robust approaches for the RCPSP.

2.3 Anchor-Robust optimization

Let us now present a robust optimization approach including an anchoring criterion, to compute a baseline solution.

2.3.1 Framework

Let us first present the general framework with the chosen uncertainty representation, then recall robust approaches from the literature.

In first stage, the baseline instance is $(c^0, \mathcal{X}^0)$. In second stage, the new instance is (c^δ, X^δ). We consider that the baseline instance corresponds to the nominal instance of problem (P). We assume, as in the robust optimization paradigm presented in Chapter 1, that we want to hedge against second-stage instances $(c^\delta, \mathcal{X}^\delta)_{\delta \in \Delta}$, with Δ the uncertainty set. It is assumed that $\mathcal{X}^\delta \neq \varnothing$ for every $\delta \in \Delta$, i.e., all second-stage instances admit a feasible solution. The baseline solution x^0 is a first-stage decision, while the new solution x^δ is a second-stage decision, adjustable to uncertainty realization δ.

Let us now formally recall static-robust and adjustable-robust approaches from the literature, within this framework. These two approaches correspond to two extreme situations where all variables are in first stage (for static-robustness) or all variables are in second stage (for adjustable-robustness).

The static-robust problem associated with (P) , as stated in Chapter 1, is

$$\begin{array}{lllll} \text{(SR-P)} & \min & \max\limits_{\delta\in\Delta} {c^\delta}^T x^0 & & \\ & \text{s.t.} & x^0 \in \mathcal{X}^\delta & \forall \delta \in \Delta \end{array}$$

The solution x^0 is the baseline solution, chosen before uncertainty realization. There is no explicit second-stage solution; implicitly, $x^\delta = x^0$ for every $\delta \in \Delta$. (SR-P) is overly conservative because of the worst-case cost $\max\limits_{\delta\in\Delta} {c^\delta}^T x^0$, and the constraint that x^0 is feasible for every second-stage instance. It might even be that no such static-robust solution exists if $\cap_{\delta\in\Delta}\mathcal{X}^\delta = \varnothing$.

The adjustable-robust counterpart of (P) involves one second-stage solution x^δ for every $\delta \in \Delta$ and it writes

$$\begin{array}{llll} \text{(Adj-P)} & \max & \min & {c^\delta}^T x^\delta \\ & \delta \in \Delta & \text{s.t.} & x^\delta \in \mathcal{X}^\delta \end{array}$$

In this case, there is no explicit baseline solution x^0 in the optimization problem.

2.3.2 Anchored solutions

Let us now introduce an important concept of the thesis, which is the definition of anchored sets and anchored solutions.

Definition 2.1 (Anchored set)**.** *For given Δ, a subset $H \subseteq I$ of decisions is anchored w.r.t. a solution $x^0 \in \mathcal{X}^0$ if for every $\delta \in \Delta$, there exists $x^\delta \in \mathcal{X}^\delta$ such that $x^0_i \equiv x^\delta_i$ for every $i \in H$.*

An anchored set is thus a subset of decisions from solution x^0 that can be, in some sense, guaranteed, since it is always possible to repair x^0 into a feasible second-stage solution x^δ without changing the decisions x^0_i, $i \in H$ from the anchored set. The fact that a decision is unchanged is represented by equivalence relation $\equiv$.

Let us now define an anchored solution.

Definition 2.2 (Anchored solution)**.** *For given Δ, an anchored solution is a pair (x^0, H) where $x^0 \in \mathcal{X}^0$ is a baseline solution and $H \subseteq I$ is anchored w.r.t. x^0.*

Hence an anchored solution is a baseline solution x^0 for problem (P), with a guarantee on decisions of the anchored set H against uncertainty Δ.

Alternatively, we may give an anchored solution as a triplet $(x^0, H, (x^\delta)_{\delta\in\Delta})$, where $x^0 \in \mathcal{X}^0$, $x^\delta \in \mathcal{X}^\delta$ for every $\delta \in \Delta$, and $x^0_i \equiv x^\delta_i$ for every $i \in H$, $\delta \in \Delta$. For an anchored solution given as a pair (x^0, H) the existence of second-stage solutions such that $x^0_i = x^\delta_i$ for every $i \in H$, is granted. It may be useful to give explicitly the second-stage solutions in the anchored solutions, e.g., to evaluate the worst-case cost of second-stage solutions.

Let us now point out the connection with solutions of the static-robust and adjustable-robust approaches.

Observation 2.1 (Static means all anchored)**.** *A solution x^0 is static-robust if and only if all decisions are anchored w.r.t. x^0, that is, $H = \{i \in I\colon\ x^0_i \equiv x^0_i\}$.*

Note that the set $\{i \in I\colon\ x^0_i \equiv x^0_i\}$ is simply I for equivalence relations $=$ and $=_\varepsilon$; it is $\{i \in I\colon\ x^0_i = 1\}$ for binary x and equivalence relation $=_1$.

Observation 2.2 (Adjustable means none anchored)**.** *A solution $(x^\delta)_{\delta\in\Delta}$ of the adjustable-robust problem corresponds to an anchored solution with $H = \varnothing$.*

In a solution $(x^\delta)_{\delta\in\Delta}$ of the adjustable-robust problem, it may be that some decisions coincide for every $\delta \in \Delta$. But since there is no a priori guarantee of that, these are not anchored decisions.

For anchored solutions, criteria of interest are:
- cost of the baseline solution ${c^0}^T x^0$
- worst-case cost of second-stage solutions $\max_{\delta\in\Delta} {c^\delta}^T x^\delta$
- size (or weight) of the anchored set

These criteria are conflicting. Namely, optimizing the size of the anchored set leads to a static-robust solution by Observation 2.1; optimizing the worst-case cost leads to an adjustable-robust solution by Observation 2.2.

We propose anchored solutions as a middle ground between static-robust and adjustable-robust solutions. Let us illustrate the difference between static-robust, adjustable-robust and anchored solutions on an example.

Example 2.3 (Anchored solutions). Consider the Selection problem with a set I of n items available. Assume that all items are the same, with nominal cost $c_i = c$ for every $i \in I$. The problem is to select p items among n with minimum total cost. Consider budgeted uncertainty with budget Γ and deviation $\widehat{\delta}_i = 1$ for every $i \in I$. Note that all items play the same role.

A static-robust solution is to select any p items. Then the worst-case cost of this solution is attained when deviations occur on the selected items, leading to a cost of $Cost_{Stat}(n, p, \Gamma) = pc + \min\{\Gamma, p\}$.

In an adjustable-robust solution, the selected items depend on the uncertainty realization. An optimal solution is to select items among the ones with no cost deviations ($\min\{p, n-\Gamma\}$ such items can be selected) then to complete the solution if necessary with items with cost deviation. This leads to a worst-case cost of $Cost_{Adj}(n, p, \Gamma) = pc + p - \min\{p, n - \Gamma\} = pc + \max\{0, p - n + \Gamma\}$.

An anchored solution is formed by a first-stage selection x^0 and a set of k items which is anchored (with $k \leq p$). These anchored items remain in the second-stage solution. What is the worst-case cost of such a solution? The worst-case

uncertainty realization is when the k anchored items have a cost deviation, leading to a cost of $kc + \min\{\Gamma, k\}$. Then the $p - k$ other items may be changed: in that sense, the non-anchored items can be selected as in an adjustable-robust solution. It leads to a cost $Cost_{Adj}(n-k, p-k, \Gamma - \min\{\Gamma, k\})$ for the non-anchored items. The total cost is $Cost^k_{Anch}(n, p, \Gamma) = pc + \max\{\min\{\Gamma, k\}, p - n + \Gamma\}$.

It is then clear that $Cost_{Adj}(n, p, \Gamma) = Cost^0_{Anch}(n, p, \Gamma) \leq Cost^k_{Anch}(n, p, \Gamma) \leq Cost^p_{Anch}(n, p, \Gamma) = Cost_{Stat}(n, p, \Gamma)$ for any number k of anchored items. $\diamond$

Throughout the thesis, we will illustrate that anchored solutions bridge the gap between static-robustness and adjustable-robustness, see, e.g., Chapter 4 or Chapter 6.

2.3.3 Anchor-Robust optimization problems

Let us now formulate robust problems involving anchored solutions. The above mentioned criteria are conflicting: namely, the costs versus the size of the anchored set. Similarly to anchor-reoptimization, we now define variants, depending on whether the objective function is to optimize the cost of solutions or the size of the anchored set.

Anchored set maximization variant. A first problem is to optimize the size of the anchored set, while looking for a baseline solution with bounded cost. Let $C^0 \geq 0$. The Anchor-Robust (AnchRob) problem is

$$\begin{array}{lll} \text{(AnchRob)} & \max & |H| \\ & x^0 \in \mathcal{X}^0 & \\ & {c^0}^T x^0 \leq C^0 & \\ & H \subseteq I \text{ anchored w.r.t. } x^0 & \end{array}$$

(AnchRob) can be seen as a robust 2-stage optimization problem. Let us write the problem under the following form:

$$\begin{array}{llll} \max & \min & \max & |H| \\ x^0 \in \mathcal{X}^0 & \delta \in \Delta & x^\delta \in \mathcal{X}^\delta & \\ {c^0}^T x^0 \leq C^0 & & x^0_i = x^\delta_i \ \forall i \in H & \\ H \subseteq I & & & \end{array}$$

The recourse problem is then an instance of the Extendability problem: given x^0, $H \subseteq I$ and $\mathcal{X}^\delta$, decide the existence of $x^\delta \in \mathcal{X}^\delta$ such that $x^0_i = x^\delta_i$ $\forall i \in H$. It is only a feasibility problem since the cost $|H|$ does not depend on second-stage variables. Hence the inner min/max problem has finite value $|H|$ if set H is anchored, and infinite value otherwise.

Importantly, (AnchRob) outputs a static-robust solution x^0 with $H = I$ if there exists one respecting the baseline cost condition ${c^0}^T x^0 \leq C^0$. If no static-robust solution with such a cost exists, an optimal solution of (AnchRob) has a max-size subset of anchored jobs $H \subsetneq I$. Note that the condition on the baseline cost could be replaced by a condition on the worst-case cost of second-stage solutions.

The Anchor-Robust problem under this Anchored set maximization variant is well-suited for problems with uncertainty on the feasible set. It is studied for project scheduling under precedence constraints in Chapter 6, Chapter 8, and for the Resource-Constrained Project Scheduling Problem in Chapter 7.

Worst-case cost minimization. Let us now propose a cost minimization variant, which is particularly adapted to the case with cost uncertainty only. Let $k \in \{0, \dots, |I|\}$ be an integer. The problem is to minimize the worst-case cost of second-stage solutions, while ensuring that the anchored set has size at least k.

$$\begin{array}{lllll} \text{(AnchRob)} & \min & \max & \min\ {c^\delta}^T x^\delta \\ & x^0 \in \mathcal{X}^0 & \delta \in \Delta & x^\delta \in \mathcal{X}^\delta \\ & H \subseteq I & & x^0_i = x^\delta_i\ \forall i \in H \\ & |H| \geq k \end{array}$$

In this case the problem clearly writes as a robust 2-stage problem. The set H is anchored w.r.t. x^0 as soon as the inner max/min has a finite value.

The Anchor-Robust under this worst-case cost minimization variant is studied for combinatorial problems with cost uncertainty in Chapter 4.

2.3.4 Connection with recoverable robustness

The concepts of anchor-robustness and recoverable robustness (Liebchen et al., 2009) are closely related, as both are 2-stage concepts with a feature to ensure that first-stage and second-stage solutions are similar.

Solutions of the recoverable robust approach are formed with a baseline solution x^0 and a collection $(x^\delta)_{\delta \in \Delta}$ of second-stage solutions. For a given integer $k \in \{0, \dots, |I|\}$ a recoverable robust approach is to impose a lower bound on the number of common decisions in x^0 and x^δ. Formally, for every $\delta \in \Delta$, it is imposed that $|\{i \in I\colon\ x^0_i \equiv x^\delta_i\}| \geq k$, or equivalently, $\sigma(x^0, x^\delta) \geq k$.

Let us now highlight the difference between anchor-robustness and recoverable robustness. Consider a baseline solution x^0 and a collection of second-stage solutions $(x^\delta)_{\delta \in \Delta}$.

- x^0 and $(x^\delta)_{\delta \in \Delta}$ form a recoverable robust solution if for every $\delta \in \Delta$, there exists a H^δ such that $|H^\delta| \geq k$ and $x^0_i \equiv x^\delta_i$ for every $i \in H^\delta$.

– x^0 and $(x^\delta)_{\delta\in\Delta}$ form an anchored solution with set H of size $|H| \geq k$ if for every $\delta \in \Delta$, $x^0_i \equiv x^\delta_i$ for every $i \in H$.

Hence any anchored solution is a recoverable robust solution. The difference is that in an anchored solution the set H of common decisions between x^0 and x^δ is the same for every $\delta \in \Delta$, while for recoverable robust solutions it depends on δ.

Knowing set H in first stage strengthen the guarantee of decisions: indeed the decision maker knows not only that a small number of changes may occur in second stage, but also that these changes will occur only on non-anchored decisions.

Example 2.4 (Anchored v.s. recoverable robust solutions). Consider solutions to the Selection problem, where 5 items must be selected out of a total of 8. Consider that the baseline solution x^0, and second-stage solutions $x^{\delta_1}, x^{\delta_2}, x^{\delta_3}$ are the following vectors. Let $\Delta = \{\delta_1, \delta_2, \delta_3\}$. Since solutions are binary vectors, recall that the anchoring level is defined as the number of common 1's coordinates.

The solutions x^0, $(x^\delta)_{\delta\in\Delta}$ form a recoverable solution for $k = 4$, since $\sigma(x^0, x^\delta) \geq 4$ for every $\delta \in \Delta$. The common decisions are indicated by the **1**'s in bold.

With set $H = \{e_5, e_6\}$, $(x^0, H, (x^\delta)_{\delta\in\Delta})$ is an anchored solution. The anchored decisions are indicated by the *1*'s in red. There is no anchored set of size larger than 2 for these baseline and second-stage solutions. E.g., $\{e_3, e_5, e_6\}$ is not anchored since $x^{\delta_3}_3 = 0$.

$$
\begin{array}{cccc}
x^0 & x^{\delta_1} & x^{\delta_2} & x^{\delta_3} \\
\begin{bmatrix} 0 \\ \mathbf{1} \\ \mathbf{1} \\ 0 \\ \textit{1} \\ \textit{1} \\ 0 \\ \mathbf{1} \end{bmatrix} &
\begin{bmatrix} 1 \\ \mathbf{1} \\ \mathbf{1} \\ 0 \\ \textit{1} \\ \textit{1} \\ 0 \\ 0 \end{bmatrix} &
\begin{bmatrix} 0 \\ 0 \\ \mathbf{1} \\ 1 \\ \textit{1} \\ \textit{1} \\ 0 \\ \mathbf{1} \end{bmatrix} &
\begin{bmatrix} 0 \\ \mathbf{1} \\ 0 \\ 0 \\ \textit{1} \\ \textit{1} \\ 1 \\ \mathbf{1} \end{bmatrix}
\end{array}
$$

◇

We mention that research works on recoverable robustness usually employ a different definition, where a lower bound is set on the number of different decision in x^0 and x^δ (see, e.g., the survey (Kasperski and Zieliński, 2016)). However we point out that with equality as equivalence relation, such a lower bound is equivalent to $\sigma(x^0, x^\delta) \geq k$. Indeed $|\{i \in I\colon\ x^0_i \equiv x^\delta_i\}| + |\{i \in I\colon\ x^0_i \not\equiv x^\delta_i\}| = |I|$, where $|I|$ is constant.

Finally, we compare the approach to the Anchor-Proactive problem introduced in (Bendotti et al., 2017). The Anchor-Proactive problem is to find a baseline

solution $x^0 \in \mathcal{X}^0$ such that ${c^0}^T x^0 \leq C^0$, maximizing $\min_{\delta\in\Delta} \max_{x^\delta\in\mathcal{X}^\delta} \sigma(x^0, x^\delta)$. An anchor-proactive solution with objective value k is thus $(x^0, (x^\delta)_{\delta\in\Delta})$ such that for every $\delta \in \Delta$, $\sigma(x^0, x^\delta) \geq k$. Introducing set H^δ of common decisions, this writes: for every $\delta \in \Delta$, there exists H^δ such that $|H^\delta| \geq k$ and $x^0_i \equiv x^\delta_i$ for every $i \in H^\delta$. Thus anchor-proactive solutions corresponds exactly to recoverable solutions, as previously defined. The difference between the Anchor-Proactive problem and recoverable robust problems is mainly the choice of the criterion: in the former the parameter k is maximized while the baseline cost is bounded, in the latter the solution cost is minimized for fixed parameter k. These problems are thus variants, similarly to the two variants of the Anchor-Robust problem defined in Section 2.3.

2.3.5 Uncertainty with a worst-case element

All problems were introduced without making any assumption on the structure and properties of the uncertainty. Let us now have a look at the case where the uncertainty set has a so-called worst-case element. This situation is commonly encountered for simple uncertainty sets, and we will show that anchored solutions have a specific form in that case. The worst-case element is formally defined as follows.

Definition 2.3 (Worst-case element of uncertainty set)**.** *Let $\delta^+ \in \Delta$. Then δ^+ is a* worst-case element *of the uncertainty set Δ if $\mathcal{X}^{\delta^+} \subseteq \mathcal{X}^\delta$ for every $\delta \in \Delta$ and $c^{\delta^+} \geq c^\delta$ for every $\delta \in \Delta$.*

In the sequel, we use the shorthand notation $c^+ = c^{\delta^+}$, $x^+ = x^{\delta^+}$, $\mathcal{X}^+ = \mathcal{X}^{\delta^+}$. Note that the definition of the worst-case element implies $\mathcal{X}^+ \subseteq \bigcap_{\delta\in\Delta} \mathcal{X}^\delta$ and $\mathcal{X}^+$ is unique. A simple uncertainty set with a worst-case element is a box. For example with $c^\delta = c + \delta$ and $\delta \in \Delta = \Pi_{i\in I}[\delta^-_i, \delta^+_i]$, it comes that δ^+ is a worst-case element of Δ.

Proposition 2.2. *Assume the uncertainty set Δ has a worst-case δ^+. Then a solution (x^0, H) is anchored for set Δ if and only if it is anchored for set $\{\delta^+\}$.*

Proof. Since $\{\delta^+\} \subseteq \Delta$, it is clear that if (x^0, H) is anchored for Δ, it is anchored for the subset $\{\delta^+\}$. Conversely, assume there exists $x^+ \in \mathcal{X}^+$ such that $x^0_i = x^+_i$ for every $i \in H$. This solution x^+ is thus in $\mathcal{X}^\delta$ for every $\delta \in \Delta$ by assumption on the worst-case element. Hence (x^0, H) is anchored for Δ. □

This observation can be interpreted as a dominance of the worst-case element in the uncertainty set. Then the Anchor-Robust optimization problems can be simplified into one-level optimization problems. For example for the Anchored set maximization variant, with Proposition 2.2 the condition H anchored w.r.t. x^0 for

set Δ reduces to the existence of $x^+ \in \mathcal{X}^+$ such that $x^0_i = x^+_i$ for every $i \in H^+$. Hence the (AnchRob) problem is

$$\begin{array}{ll} \max & |H^+| \\ \text{s.t.} & x^0 \in \mathcal{X}^0 \\ & {c^0}^T x^0 \leq C^0 \\ & x^+ \in \mathcal{X}^+ \\ & x^0_i = x^+_i \ \forall i \in H^+ \end{array}$$

This problem is studied, e.g., in Section 6.4.1 for anchor-robust project scheduling.

Consider now the worst-case cost minimization variant. Consider an anchored solution $(x^0, H, (x^\delta)_{\delta\in\Delta})$ where every x^δ is an optimal solution to the recourse problem of (AnchRob). Then it holds that the worst-case cost of $(x^0, H, (x^\delta)_{\delta\in\Delta})$ is ${c^+}^T x^+$: indeed for every $\delta \in \Delta$ the optimal value of the recourse problem is at most ${c^\delta}^T x^+$ since $x^+ \in \mathcal{X}^\delta$, which is upper-bounded by ${c^+}^T x^+$ by assumption on δ^+. The (AnchRob) problem is thus

$$\begin{array}{ll} \min & {c^+}^T x^+ \\ \text{s.t.} & x^0 \in \mathcal{X}^0 \\ & x^+ \in \mathcal{X}^+ \\ & x^0_i = x^+_i \text{ for every } i \in H^+ \\ & |H^+| \geq k \end{array}$$

This problem is considered in Section 4.2.3 for combinatorial problems.

2.4 Main research directions on Anchor-Reoptimization and Anchor-Robust problems

In this chapter we presented anchored solutions, and the associated anchor-reoptimization and anchor-robust problems. The main issues relative to these problems that are investigated throughout the manuscript are the following.

▶ **Computational complexity** of anchor-reoptimization and anchor-robust problems. A focus is to analyze when there is an increase in complexity w.r.t. the original problem, and when there is none. We also investigate algorithmic approaches for polynomial cases.

For anchor-reoptimization, the complexity depends on the original problem, but also on the chosen anchoring criterion. We will study several variants (with different anchoring weights, equivalence relations, or restricted reoptimization). See Chap. 3, Chap. 5.

For anchor-robust optimization, the complexity depends on the original problem, but also heavily on the uncertainty representation: what is the uncertain data? what is the uncertainty set? Importantly an anchor-robust problem also inherits from the hardness of the inner anchor-reoptimization problem. See Chap. 4, Chap. 6.

▶ **Combinatorial properties**. We analyze the combinatorial properties of extendable or anchored sets. Such a combinatorial study is useful even when the variables of the original problem are continuous, because the anchoring criterion itself is combinatorial. See Chap. 5, Chap. 6, Chap. 7. The combinatorial nature of anchored solutions also motivates polyhedral approaches, investigated in Chap. 3, Chap. 8, Chap. 9.

▶ **Existence of reformulations for Anchor-Robust problems**. In the spirit of the literature on robust optimization, we investigate reformulations for anchor-robust problems, that are naturally written as imbricated max/min/max problems. Such mathematical programs are hard to manage. By a dedicated approach, we establish MIP reformulations in several cases, see Chap. 4, Chap. 6, Chap. 7.

▶ **Design of efficient algorithms**. Finally for hard cases, a goal is to design algorithmic tools for solving the problem. These tools are exact approaches, either based on MIP reformulations and polyhedral study (Chap. 7, Chap. 8, Chap. 9); or on heuristic algorithms for the most challenging problems (Chap. 7).

Part II

Anchored solutions to combinatorial problems

Preliminaries on combinatorial problems

In this part, we consider combinatorial problems, defined as follows. Let E be a set of elements. Let $\boldsymbol{X} \subseteq 2^E$ be a collection of *feasible* subsets of E. Let $w \in \mathbb{R}^E$ be a vector of weights associated to elements. The considered combinatorial problem (P) is to find a feasible subset $X \in \boldsymbol{X}$ of elements that maximizes the total weight $w(X) = \sum_{i \in X} w_i$.

$$\begin{array}{lll} (\mathrm{P}) & \max & w(X) \\ & \text{s.t.} & X \in \boldsymbol{X} \end{array}$$

Given a subset $U \subseteq E$, let χ^U denote its incidence vector. Let us denote by $\mathcal{X}$ the set $\{\chi^U : \ U \in \boldsymbol{X}\}$ containing all incidence vectors of feasible sets. The combinatorial problem (P) is then represented by the program with binary decision vector $x \in \{0,1\}^E$:

$$\begin{array}{lll} (\mathrm{P}) & \max & w^T x \\ & \text{s.t.} & x \in \mathcal{X} \end{array}$$

We assume that the feasible set $\mathcal{X}$ can be described by integrality constraints and linear constraints, so that (P) is represented by the integer program

$$\begin{array}{lll} (\mathrm{P}) & \max & w^T x \\ & \text{s.t.} & Ax \leq b \\ & & x \geq 0 \\ & & x \in \{0,1\}^E \end{array}$$

The family of constraints $Ax \leq b$ may have exponential size.

Additionally, we may consider combinatorial problems for which a polyhedral characterization is known. In that case, a collection of linear inequalities $A'x \leq b'$ is known so that

$$\mathrm{conv}(\mathcal{X}) = \{x \in \mathbb{R}^E : \ A'x \leq b', \ x \geq 0\}$$

Note that it is a strong assumption on problem (P).

Let us now define classical combinatorial problems that will be used in the sequel.

MAX-WEIGHT MATROID BASIS PROBLEM. A *matroid* is a pair $M = (E, \mathcal{I})$ composed with a ground-set E and a non-empty collection $\mathcal{I}$ of subsets of E called *independent*, such that:
(i) if $X \in \mathcal{I}$ and $X' \subseteq X$, then $X' \in \mathcal{I}$
(ii) if X and X' are independent and $|X'| < |X|$, then there exists $e \in X \setminus X'$ such that $X' \cup \{e\} \in \mathcal{I}$.

An inclusion-wise maximal independent set is a *basis*. Given a matroid $M = (E, \mathcal{I})$ and weights $w \in \mathbb{R}^E$, the MAX-WEIGHT MATROID BASIS PROBLEM is to find a basis X with maximum total weight $w(X)$.

Matroids are classical structures in combinatorial optimization, see, e.g., Schrijver's book (Schrijver, 2003) for an extensive background. The problem MAX-WEIGHT MATROID BASIS PROBLEM generalizes two problems defined in Preliminaries of Part I: SELECTION and MIN COST SPANNING TREE. The problem MAX-WEIGHT MATROID BASIS PROBLEM is solvable in polynomial time by a greedy algorithm, similar to Kruskal's algorithm for spanning trees. Matroids also enjoy remarkable polyhedral properties, namely: a polyhedral characterization for the base polytope $\text{conv}\{\chi^B : B \text{ base}\}$ was obtained by Edmonds (1971).

MAX-WEIGHT BIPARTITE MATCHING. Let $G = (V_1, V_2, E)$ be a bipartite graph. A matching is a subset of edges $X \subseteq E$ containing no adjacent edges. The MAX-WEIGHT BIPARTITE MATCHING PROBLEM is to find a matching X with maximum weight $w(X)$.

It can be solved in polynomial time, e.g., by the Hungarian algorithm. A characterization of $\text{conv}(\mathcal{X})$, where $\mathcal{X}$ is the set of incidence vectors of matchings, is known (see (Schrijver, 2003)).

Chapter 3

Anchor-Reoptimization for combinatorial problems: a case study on matroid bases

In this chapter, we consider the combinatorial problem over the set of elements E, with feasible solutions $\boldsymbol{X} \subseteq 2^E$ and weights $w \in \mathbb{R}^E$, that is

$$\begin{array}{lll} \text{(P)} & \max & w(X) \\ & \text{s.t.} & X \in \boldsymbol{X} \end{array}$$

Let us recall general definitions, as given in Chapter 2, specified for the combinatorial problem (P). Given two subsets of elements X and X', an element in $X \cap X'$ is said to be *anchored*. The *anchoring level* is $|X \cap X'|$, i.e., the number of elements that are selected in both X and X'. If anchoring weights $a \in \mathbb{R}^E_+$ are considered, the *weighted anchoring level* is $a(X \cap X')$. The main motivation for anchoring problems is that in practice, the decision maker does not solve the optimization problem (P) from scratch. The baseline solution is a subset X_0 of elements initially selected. Hence the solution X of (P) can be evaluated on two criteria: its total weight $w(X)$ and its (weighted) anchoring level with the baseline $|X_0 \cap X|$ (or $a(X_0 \cap X)$).

Let us consider that the elements of the baseline X_0 are colored *red*. Let $\rho \in \{0,1\}^E$ denote the incidence vector of X_0, i.e., of red elements. The anchoring level $|X_0 \cap X|$ (or $a(X_0 \cap X)$) is then the total number (or total anchoring weight) of red elements selected in solution X.

Let us recall the considered anchor-reoptimization problems. The first problem AnchReopt is to find a feasible set of elements maximizing the number of elements from the baseline.

AnchReopt			
Input:	$\rho \in \{0,1\}^E$, $\boldsymbol{X} \subseteq 2^E$		
Problem:	max	$\rho(X)$	
	s.t.	$X \in \boldsymbol{X}$	

Problem AnchReopt is simply the original problem (P) with weights $\rho \in \{0,1\}^E$. Hence it is polynomially solvable if (P) is polynomially solvable for unit weights. For a problem (P) that is NP-hard even for unit weights (e.g., stable set problem) then AnchReopt is also NP-hard, since vector ρ is allowed to be any vector in $\{0,1\}^E$. We will focus on the second class of problems, involving both the anchoring level $\rho(X)$ and the weight $w(X)$ of the solution.

AnchReopt–Cost			
Input:	$\rho \in \{0,1\}^E$, $\boldsymbol{X} \subseteq 2^E$, $w \in \mathbb{R}^E$, $W \geq 0$		
Problem:	max	$\rho(X)$	
	s.t.	$X \in \boldsymbol{X}$	
		$w(X) \geq W$	

AnchReopt–Anch, a.k.a. k-red variant			
Input:	$\rho \in \{0,1\}^E$, $\boldsymbol{X} \subseteq 2^E$, $w \in \mathbb{R}^E$, $k \in \mathbb{N}$		
Problem:	max	$w(X)$	(P-kRed)
	s.t.	$X \in \boldsymbol{X}$	
		$\rho(X) \geq k$	

The problem AnchReopt–Anch is to find a max-weight solution with at least k red elements: it is the *k-red variant* of original problem (P), denoted by (P-kRed).

Note that a polynomial algorithm for AnchReopt–Cost gives a polynomial algorithm for AnchReopt–Anch, and vice versa. Namely, the optimal value W^* of an instance $(\rho, \boldsymbol{X}, w, k)$ of AnchReopt–Anch is the largest W such that the optimal value of AnchReopt–Cost on instance $(\rho, \boldsymbol{X}, w, W)$ is at least k. The optimum W^* can be found by dichotomy on the value of W, which requires at most $\log(W_{\max})$ calls to the algorithm for AnchReopt–Cost where $W_{\max}$ is a bound on W^* (e.g., $W_{\max} = n \max_{e \in E} w_e$). This is polynomial in the size of the instance of AnchReopt–Anch.

In the sequel we will focus on the k-red variant (P-kRed). Let us illustrate the k-red variant of the Min Cost Spanning Tree problem in Example 3.1.

Example 3.1 (k-red spanning trees). Consider a baseline instance of the Min Cost Spanning Tree problem. The graph $G = (V, E)$ has 6 nodes. Baseline edge costs are represented in Figure 3.1, along with the baseline spanning tree in red. It has minimum cost.

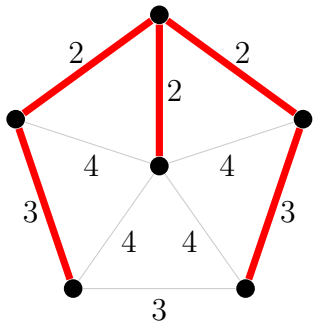
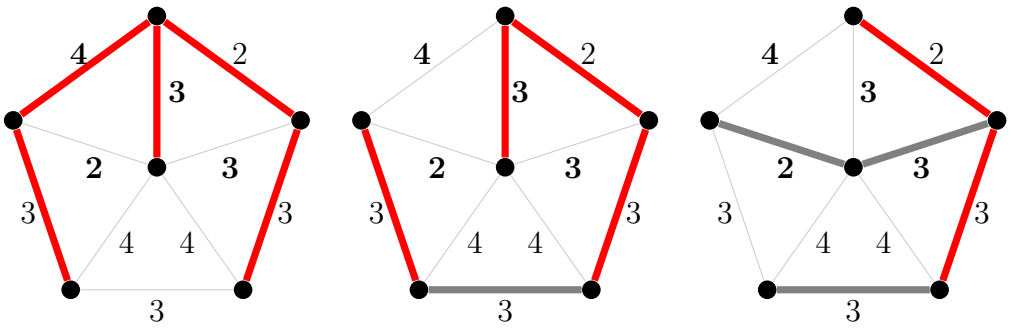


Figure 3.1: Baseline instance of the Min Cost Spanning Tree Problem and an optimal solution in red.

Consider now that some edge costs change, as indicated in bold in Figure 3.2. Three solutions are represented. Solution (a) is to keep the baseline: it has new cost 15. Solution (c) is an optimal solution to the new instance: it has a cost of 13, but only 2 edges in common with the baseline red solution. Solution (b) has cost 14, and 4 red edges.



(a)
cost 15
5 red edges (baseline)

(b)
cost 14
4 red edges

(c)
cost 13 (opt)
2 red edges

Figure 3.2: New instance of the Min Cost Spanning Tree Problem, and three distinct solutions.

This shows that solution (b) with intermediate cost, and 4 red edges out of 5, can be found by solving the k-red Spanning Tree problem with $k = 4$. ◇

In this chapter, we investigate the complexity of the k-red variant (P-kRed) of polynomial combinatorial problems, and especially matroid bases. Let us review some related work. For combinatorial problems, the anchoring level coincides with transition costs: if element e from the baseline is included in the solution, it yields a profit a_e. This model has been studied, e.g., in (Schieber et al., 2018), where the authors considered a special case of the AnchReopt–Cost problem. Spanning trees with a side linear constraint have been widely studied (Aggarwal et al., 1982; Ravi and Goemans, 1996; Ahuja et al., 1993). If the side constraint is any linear constraint, the problem is weakly NP-hard (Aggarwal et al., 1982). In particular, this implies that the Anchor-Reoptimization problem is NP-hard for non-unit anchoring weights. The biobjective spanning tree has also received attention (Hamacher and Ruhe, 1994; Sourd and Spanjaard, 2008).

The chapter is structured as follows. Section 3.1 is devoted to a case study on the problem of finding a max-weight k-red basis in a matroid. Two approaches are proposed. The first one is an algorithm based on Lagrangian relaxation of the k-red constraint, presented in Section 3.1.2. The second one is a polyhedral characterization, presented in Section 3.1.3. Both approaches yield a proof that the k-red matroid basis problem is polynomial. In Section 3.2 the result is illustrated for spanning trees, along with a consequence on the Pareto front of the k-red biobjective spanning tree. In Section 3.3 the max-weight k-red matching problem in a bipartite graph is considered. We show that a longstanding open problem, Exact Perfect Matching, reduces to this k-red variant.

3.1 Max-weight k-red bases in matroids

Let us study the complexity of finding a max-weight k-red matroid basis. Let $M = (E, \mathcal{I})$ be the matroid and $\mathcal{B}$ its set of bases. Let $\rho \in \{0,1\}^E$ be the incidence vector of red elements, and $k \in \mathbb{N}$. Let $w \in \mathbb{R}^E$. Let $\mathcal{B}^{k\text{-red}}$ denote the set of bases with at least k-red elements, i.e., $\mathcal{B}^{k\text{-red}} = \{X \in \mathcal{B} \colon \rho(X) \geq k\}$. We study the problem (Bas-kRed) of finding a max-weight solution in $\mathcal{B}^{k\text{-red}}$. It can be written as the integer program

$$\begin{array}{rl} \max & w^T x \\ \text{s.t.} & x \in \mathcal{X} \\ & \rho^T x \geq k \end{array} \qquad \text{(Bas-}k\text{Red)}$$

Recall that $\mathcal{X}$ denote the set of incidence vectors of $\mathcal{B}$, i.e., the feasible set of original problem (P).

In this section, we propose two approaches to solve the (Bas-kRed)problem. Both give a proof of the following result:

Theorem 3.1. *The problem (Bas-kRed) of finding a max-weight basis with at least k red elements is solvable in polynomial time.*

The result can be found for spanning trees in (Ahuja et al., 1993; Şeref et al., 2009). We here propose a natural extension to matroid bases, and the double algorithmic/polyhedral perspective to the problem.

3.1.1 Preliminaries on matroids

Let us give additional definitions and properties on matroids. We refer to Schrijver's book (Schrijver, 2003) for details and proofs of the main theorems.

A basis of matroid M is defined as an inclusion-wise maximal independent set. By (ii) in matroid definition (see Page 52), equivalently a basis is an independent set of maximum cardinality. The *rank* is a function $\mathrm{r}_M \colon 2^E \longrightarrow \mathbb{N}$ defined by $\mathrm{r}_M(F) = \max\{|I| \colon I \subseteq F, I \in \mathcal{I}\}$ for every subset $F \subseteq E$ of elements. Then a basis X has cardinality $|X| = \mathrm{r}_M(E)$. For $X \subseteq E$ and $e \in E$, let $X + e$ (resp. $X - e$) denote $X \cup \{e\}$ (resp. $X \setminus \{e\}$). Matroids satisfy the following exchange lemma:

Lemma 3.1 (Exchange lemma)**.** *Given two bases X and X', then for all $e \in X \setminus X'$, there exists $f \in X' \setminus X$ such that $X - e + f$ and $X' - f + e$ are bases.*

Matroids also enjoy nice polyhedral properties. The *independent set polytope* of matroid M is $\mathrm{conv}\{\chi^I \colon I \in \mathcal{I}\}$ and the *base polytope* of matroid M is $\mathrm{conv}\{\chi^B \colon B \in \mathcal{B}\}$. The incidence vector x of an independent set I, and thus any element x of the independent set polytope, satisfies inequalities

$$x_e \geq 0 \qquad \forall e \in E \tag{3.1}$$

$$x(F) \leq \mathrm{r}_M(F) \qquad \forall F \subseteq E \tag{3.2}$$

Inequality (3.2) is valid since for every $I \in \mathcal{I}$ and $F \subseteq E$, it holds that $I \cap F \in \mathcal{I}$ hence $|I \cap F| \leq \mathrm{r}_M(F)$, by definition of the rank. The incidence vector x of a basis B also satisfies

$$x(E) = \mathrm{r}_M(E) \tag{3.3}$$

An important result due to Edmonds is that these inequalities characterize the base polytope and the independent set polytope.

Theorem 3.2 (Edmonds, see Corollary 40.2b in (Schrijver, 2003))**.** *The polytopes $\{x \in \mathbb{R}^E \colon (3.1)\text{–}(3.2)\}$ and $\{x \in \mathbb{R}^E \colon (3.1)\text{–}(3.2)\text{–}(3.3)\}$ are integer. They characterize the independent set polytope and the base polytope of matroid M, respectively.*

Importantly, Theorem 3.2 implies that the Max-Weight Matroid Basis problem is polynomial, by linear programming. The family of inequalities (2) is exponential, but a polynomial separation algorithm exists (Theorem 40.4 in (Schrijver, 2003)). Hence the maximum of $w^T x$ over $\{x \in \mathbb{R}^E \colon (3.1)\text{–}(3.2)\text{–}(3.3)\}$ can be found in polynomial time.

Another result is for the intersection of two matroids.

Theorem 3.3 (Edmonds, see Corollary 41.12a in (Schrijver, 2003))**.** *Let $M_1 = (E, \mathcal{I}_1)$ and $M_2 = (E, \mathcal{I}_2)$ be two matroids on the same ground-set. The polytope of common independent sets* $\operatorname{conv}\{\chi^I \colon I \in \mathcal{I}_1 \cap \mathcal{I}_2\}$ *is described by inequalities* (3.1) *and* (3.2) *associated with both matroids.*

Let us finally note that the Min Cost Spanning Tree problem is a special case of Max-Weight Basis. With a graph $G = (V, E)$, a matroid $M^G = (E, \mathcal{I}^G)$ is associated where a subset $X \subseteq E$ is an independent set if the set X is cycle-free. Spanning trees are bases of matroid M^G. Spanning trees have cardinality $\mathrm{r}_{M^G}(E) = |V| - 1$.

3.1.2 Lagrangian-based algorithm

Let us present the Lagrangian-based approach to (Bas-kRed). In Section 3.1.2.1 we present the general framework applicable to any k-red problem (P-kRed). In Section 3.1.2.2 we give further properties specific to k-red matroid bases.

Lagrangian relaxation was used in (Ravi and Goemans, 1996) for the NP-hard constrained spanning tree and in Example 16.6 in (Ahuja et al., 1993) for a polynomial degree-constrained spanning tree problem. The algorithm from Section 3.1.2.2 is adapted from this technique, and extended to matroid bases. While preparing the present work, we found out that in (Şeref et al., 2009) the same proof technique was used for k-red spanning trees and when red edges form a spanning tree.

3.1.2.1 General approach

Consider Lagrangian relaxation of the k-red constraint $\rho(X) \geq k$. Let $\lambda \geq 0$ be the dual Lagrange multiplier associated with the constraint $\rho(X) \geq k$, and let $\mathscr{G}(\lambda)$ be the dual problem

$$\mathscr{G}(\lambda) : \max_{X \in \boldsymbol{X}} w(X) + \lambda(\rho(X) - k)$$

Let $\mathcal{G}(\lambda)$ be the dual function, i.e., the optimal value of problem $\mathscr{G}(\lambda)$. For a given $\lambda \geq 0$, the value $\mathcal{G}(\lambda)$ can be computed by solving the problem (P) for modified weights $\widetilde{w}_e = w_e + \lambda\rho_e$. The k-red problem is $\max_{X \in \boldsymbol{X}} \min_{\lambda \geq 0} w(X) + \lambda(\rho(X) - k)$.

Hence it holds that $\mathcal{G}(\lambda)$ is an upper bound to the problem for any value of multiplier $\lambda \geq 0$. The best of such upper bounds is the *dual bound*

$$\min_{\lambda \geq 0} \mathcal{G}(\lambda).$$

In general, the dual bound is not equal to the optimal value of the problem: there is a Lagrangian duality gap.

A classical optimality condition is as follows. Given $\lambda \geq 0$ and X an optimal solution of the dual problem $\mathscr{G}(\lambda)$, if λ and X satisfy the complementarity conditions $\rho(X) \geq k$, ($\rho(X) = k$ or $\lambda = 0$), then X is an optimal solution of (P-kRed). Indeed this condition implies $w(X) + \lambda(\rho(X) - k) = w(X)$, hence the weight of X is equal to the upper bound $\mathcal{G}(\lambda)$. The solution X being feasible for (P-kRed), it is an optimal solution to (P-kRed).

3.1.2.2 Polynomial algorithm for matroid bases

Let us now give results specific to (Bas-kRed).

Proposition 3.1. *For matroid bases, the dual bound* $\min_{\lambda \geq 0} \mathcal{G}(\lambda)$ *is equal to*

$$\min_{\lambda \in \Lambda} \mathcal{G}(\lambda) \text{ where } \Lambda = \{w_f - w_e \colon e, f \in E,\ \rho_e = 1, \rho_f = 0\}$$

and thus it can be computed in polynomial time.

Proof. A solution to the dual problem for multiplier λ is a solution to (P) for modified weights $\widetilde{w} = w + \lambda\rho$. Recall that an instance of (P) with weights $\widetilde{w}$ can be solved using the greedy algorithm, which compare weights of elements. We assume that if elements have the same weight, the one with smallest index is picked first. Consider for any instance of (P), the list of elements of E sorted lexicographically by increasing $\widetilde{w}_e$, then by increasing index. Hence the greedy algorithm always returns the same solution for all instances whose such list is the same. Let λ^* be a value of the multiplier such that the list associated to modified weights $\widetilde{w}$ changes at λ^*. Then there exists $e, f \in E$ such that $\widetilde{w}_e \leq \widetilde{w}_f$ if and only if $\lambda \leq \lambda^*$. Hence e is red, f is not red, and $\lambda^* = w_f - w_e$.

The dual function is piecewise affine, and every breakpoint of $\mathcal{G}(\lambda)$ corresponds to a value of the multiplier where the output of the greedy algorithm changes. By the previous remark, every breakpoint is in the set $\Lambda = \{w_f - w_e \colon e, f \in E,\ \rho_e = 1, \rho_f = 0\}$. This set having quadratic size, the minimum of $\mathcal{G}(\lambda)$ can be found by evaluating the function at each point of Λ. Note that in case (Bas-kRed) is unfeasible, the minimum of $\mathcal{G}$ is unbounded, which can be detected by computing the value of $\mathcal{G}$ in one additional point greater than all points in Λ. Hence the dual bound $\min_{\lambda \geq 0} \mathcal{G}(\lambda)$ can be found in polynomial time as claimed. □

By Proposition 3.1, the value λ^* attaining the dual bound $\mathcal{G}(\lambda^*)$ and an optimal solution X^* to the dual problem can be found in polynomial time. In general solution X^* does not contain k red edges. In order to show that we can output a k-red solution satisfying the optimality condition, we need the following lemma.

Lemma 3.2. *Let $w \in \mathbb{R}^E$, and W^* be the maximum weight of a basis of M.*

Let k be an integer. Given two bases of weight W^* with respectively less and more than k red elements, there always exists a basis of weight W^* with exactly k red elements, and it is computable in polynomial time by Algorithm 1.

Algorithm 1:

Input: $X_1, X_2 \in \mathcal{B}$, $k \in \mathbb{N}$ such that $w(X_1) = w(X_2) = W^*$ and $\rho(X_1) < k < \rho(X_2)$.
Output: $X^= \in \mathcal{B}$ such that $w(X^=) = W^*$ and $\rho(X^=) = k$.

Let $X := X_2$;
while $\rho(X) > k$ **do**
  Let $e \in X \setminus X_1$ be a red element;
  Let $f \in X_1 \setminus X$ such that $X - e + f \in \mathcal{B}$ and $X_1 - f + e \in \mathcal{B}$;
  Let $X := X - e + f$.
end
Return X;

Proof. Let us prove validity of Algorithm 1. The element e exists because at each iteration, X has more red elements that X_1. The element f exists by the exchange lemma. At each iteration, the set X is a basis. Also, the number of red edges in X is decremented of at most one at each iteration. Hence when the algorithm terminates, it returns a basis X such that $\rho(X) = k$.

Let us show that the equality $w(X) = W^*$ remains valid at each iteration, i.e., that $w(X - e + f) = w(X)$ before the update of X. Since $X - e + f$ is a basis it comes $w(X - e + f) \leq W^* = w(X)$ hence $w_f \leq w_e$. Similarly since $X_1 - f + e$ is a basis it comes $w(X_1 - f + e) \leq W^* = w(X_1)$ and $w_e \leq w_f$. Thus $w_e = w_f$ and the weight of X remains equal to W^*.

Finally it remains to show that this algorithm terminates after a polynomial number of iterations. At each update of X, $|X_1 \cap X|$ is incremented. Assume the algorithm has not terminated after $|X \setminus X_1|$ iterations: then $X = X_1$ hence $\rho(X) < k$, a contradiction. Thus there are at most $|X \setminus X_1|$ iterations and the algorithm is polynomial. □

Let us now present a proof of Theorem 3.1 stated Page 57.

Proof. Let $\lambda^0 = 0 < \lambda^1 < \ldots < \lambda^m$ be the points of Λ. Let $\lambda^{t^*} \in \Lambda$ be a point where $\mathcal{G}$ attains its minimum, computed in polynomial time thanks to Proposition 3.1 by exhaustive search on Λ. For every $\lambda^t \in \Lambda$, let X^t denote the optimal

solution of (P) computed by the greedy algorithm in interval $[\lambda^t, \lambda^{t+1}]$. In this interval the dual function coincides with the linear function $\lambda \mapsto w(X^t) + \lambda(\rho(X^t) - k)$.

Let us now exhibit a pair λ, X satisfying the complementarity conditions recalled in Section 3.1.2.1, that is: $\lambda \geq 0$, X is an optimal solution to the dual problem in λ, and $\rho(X) \geq k$, ($\rho(X) = k$ or $\lambda = 0$).

(i) Assume first $\lambda^{t^*} = 0$. Then $\mathcal{G}$ is non-decreasing, and the slope of $\mathcal{G}$ on $[0, \lambda^1]$ is non-negative. This slope equals $\rho(X^0) - k$. Hence the pair $\lambda = 0$, X^0 satisfy the condition and X^0 is an optimal solution of (Bas-kRed).

(ii) Assume $\lambda^{t^*} > 0$. Consider the two solutions X^{t^*-1} and X^{t^*}: both are optimal solutions of the dual problem $\mathscr{G}(\lambda^{t^*})$. Since the dual function attains its minimum in λ^{t^*} it comes $\rho(X^{t^*-1}) - k \leq 0$ and $\rho(X^{t^*}) - k \geq 0$. If one of them has k red edges, we are done. Otherwise, X^{t^*-1} and X^{t^*} satisfy the conditions of Lemma 3.2 for the modified weights associated with λ^{t^*}. Using Algorithm 1, a solution $X^=$ can be computed in polynomial time, such that $X^=$ is also an optimal solution of $\mathscr{G}(\lambda^{t^*})$, and $\rho(X^=) = k$. Then the condition is satisfied by λ^{t^*} and $X^=$, hence $X^=$ is an optimal solution of (Bas-kRed).

□

In particular, for (Bas-kRed) problem this implies that there is no Lagrangian duality gap.

3.1.3 Polyhedral characterization

Let us now propose a different approach to solve the (Bas-kRed) problem. Let $\mathcal{Q} = \text{conv}\{\chi^B\colon\ B \in \mathcal{B}\}$ denote the base polytope of M. By Edmonds's result stated in Theorem 3.2, $\mathcal{Q}$ is characterized by inequalities (3.1)–(3.2)–(3.3). Let $\mathcal{Q}^{k\text{-red}} = \text{conv}\{\chi^B\colon\ B \in \mathcal{B}^{k\text{-red}}\}$ be the polytope of problem (Bas-kRed). We are interested in the characterization of the polytope $\mathcal{Q}^{k\text{-red}}$. It holds that $\mathcal{Q}^{k\text{-red}}$ is included in the intersection $\mathcal{Q} \cap \{x \in \mathbb{R}^E\colon\ \rho^T x \geq k\}$ of the base polytope with the k-red half-space. In general, intersecting integer polytope $\mathcal{Q}$ with a half-space $\{x \in \mathbb{R}^E\colon\ a^T x \geq k\}$, $a \in \mathbb{R}^E$ would lead to a set with non-integer extreme points. In general for $\mathcal{X} \subseteq \{0,1\}^E$ and $a \in \mathbb{R}^E$, the set $\mathcal{X} \cap \{x \in \mathbb{R}^E\colon\ a^T x \geq k\}$ does not admit an efficient polyhedral characterization. Indeed the problem of maximizing $w^T x$ over this set corresponds to AnchReopt–Anch with non-unit anchoring weights, which is NP-hard.

In the case of the k-red constraint, the following polyhedral result holds.

Theorem 3.4. *Polytope $\mathcal{Q}^{k\text{-}red}$ is defined by*

$$x_e \geq 0 \qquad \forall e \in E \tag{3.1}$$
$$x(F) \leq \mathrm{r}_M(F) \qquad \forall F \subseteq E \tag{3.2}$$
$$x(E) = \mathrm{r}_M(E) \tag{3.3}$$
$$\rho^T x \geq k \tag{3.4}$$

That is, $\mathcal{Q}^{k\text{-}red} = \mathcal{Q} \cap \{x \in \mathbb{R}^E \colon \ \rho^T x \geq k\}$.

Adding the k-red inequality (3.4) here preserves the integrality of the polytope.

Proof. The proof relies on the identification of the problem (Bas-kRed) as an intersection of matroids. Let $m = \mathrm{r}_M(E)$. Let $\bar{\rho}$ denote the incidence vector of non-red elements, i.e., $\bar{\rho} = 1 - \rho$. Let $M^{k\text{-red}} = (E, \mathcal{I}^{k\text{-red}})$ be a second matroid defined by: $X \in \mathcal{I}^{k\text{-red}}$ if $\bar{\rho}(X) \leq m-k$. That is, a subset of elements is independent if it has at most $m - k$ non-red elements. It is a partition matroid.

Then $\mathcal{B}^{k\text{-red}} = \mathcal{B} \cap \mathcal{I}^{k\text{-red}}$: indeed a basis $X \in \mathcal{B}$ has cardinality $|X| = m$, hence $\bar{\rho}(X) \leq m - k$ if and only if $\rho(X) \geq k$. Therefore $\mathcal{Q}^{k\text{-red}} = \mathrm{conv}\{\chi^B \colon \ B \in \mathcal{B} \cap \mathcal{I}^{k\text{-red}}\}$. Using Theorem 3.3, it comes that $\mathcal{Q}^{k\text{-red}} = \mathcal{Q} \cap \mathcal{P}$, where $\mathcal{P}$ is the independent set polytope of matroid $M^{k\text{-red}}$.

Let us show that $\mathcal{Q} \cap \{x \in \mathbb{R}^E \colon \ \rho^T x \geq k\} \subseteq \mathcal{Q} \cap \mathcal{P}$. Let us compute the rank $\mathrm{r}_{M^{k\text{-red}}}(F)$ for $F \subseteq E$. In set F an independent set of max size is formed with: all red elements from F, and a number $\min(\bar{\rho}(F), m-k)$ of non-red elements. Hence $\mathrm{r}_{M^{k\text{-red}}}(F) = \rho(F) + \min(\bar{\rho}(F), m-k)$.

Let $x \in \mathcal{Q} \cap \{x \in \mathbb{R}^E \colon \ \rho^T x \geq k\}$. Let $F \subseteq E$. Let F_r and F_n denote the subset of red and non-red elements of F respectively. Then $x(F) = x(F_r) + x(F_n)$. It holds that $x(F_r) \leq |F_r| = \rho(F)$, and similarly $x(F_n) \leq \bar{\rho}(F)$. Moreover $x(F_n) \leq \bar{\rho}^T x = (1-\rho)^T x \leq m - k$, since $x(E) = m$ and $\rho^T x \geq k$. Thus $x(F) \leq \rho(F) + \min(\bar{\rho}(F), m-k)$. By Theorem 3.2, $\mathcal{P}$ is described by inequalities (1)–(2), here satisfied by x, hence $x \in \mathcal{P}$.

Finally this proves $\mathcal{Q} \cap \{x \in \mathbb{R}^E \colon \ \rho^T x \geq k\} \subseteq \mathcal{Q} \cap \mathcal{P} = \mathcal{Q}^{k\text{-red}}$, and $\mathcal{Q}^{k\text{-red}} = \mathcal{Q} \cap \{x \in \mathbb{R}^E \colon \ \rho^T x \geq k\}$ as claimed. □

The polyhedral result from Theorem 3.4 yields another proof of the polynomiality of (Bas-kRed) by linear programming. Indeed solving (Bas-kRed) is equivalent to maximizing $w^T x$ over the polytope $\mathcal{Q}^{k\text{-red}}$. The inequalities defining $\mathcal{Q}^{k\text{-red}}$ are in an exponential number, but again the existence of a polynomial separation algorithm for inequalities (3.2) ensures that the LP can be solved in polynomial time.

3.2 Illustration for the k-red spanning tree

Let us now illustrate the results from Section 3.1 for the Min Cost Spanning Tree problem. Note first that it is equivalent to look for a min cost or a max weight spanning tree, since all spanning trees have the same cardinality. For the spanning tree problem, the polyhedral result from Theorem 3.4 can be declined as follows. Given $W \subseteq V$, let $E[W]$ denote the set of edges with both extremities in W. It is known (Edmonds, see Theorem 6.13 in (Korte and Vygen, 2012)) that the spanning tree polytope is defined by

$$x_e \geq 0 \qquad \forall e \in E \tag{3.1}$$
$$x(E[W]) \leq |W| - 1 \qquad \forall W \subseteq V \tag{3.5}$$
$$x(E) = |V| - 1 \tag{3.6}$$

Corollary 3.1. *The polytope of k-red spanning trees is defined by inequalities* (3.1)–(3.5)–(3.6) *and k-red inequality* (3.4)*.*

Let us now consider the k-red spanning tree problem under a bi-objective perspective. Consider the bi-objective problem of finding $x \in \mathcal{X}$ with the two objectives to maximize: total weight $w^T x$ and number of red elements $\rho^T x$. A solution $x \in \mathcal{X}$ is *dominated* by solution $x' \in \mathcal{X}$ if $w^T x \leq w^T x'$ and $\rho^T x \leq \rho^T x'$, and one of the inequalities is strict. The Pareto front is formed by all non-dominated solutions of $\mathcal{X}$, usually represented in the objective space: that is, a solution x is represented by point $(\rho^T x, w^T x)$. For k-red problems, since $\rho^T x$ takes only integer values, the Pareto front is formed with a polynomial number of points $(k, w^T x^k)$, $k \in \{0, \ldots, |E|\}$. Then x^k is an optimal solution of the max-weight k-red problem. A solution x^* is said to be *supported* if it maximizes a convex combination of the objectives $w^T x + \lambda \rho^T x$, for some $\lambda \geq 0$. In general for the bi-objective spanning tree, the Pareto front in the objective space is a set with exponential size, and not all non-dominated solutions are supported (Sourd and Spanjaard, 2008). We show now:

Proposition 3.2. *For k-red matroid bases, every point of the Pareto front is supported.*

Proof. Consider a point (k, W) of the Pareto front. The value W is the optimum of the k-red problem. By the previous result from Section 3.1.2, there is no Lagrangian duality gap: W equals the dual bound $\mathcal{G}(\lambda^*)$ for some $\lambda^* \in \Lambda$. Following the proof of Theorem 3.1, we consider the following cases:

(i) $\lambda^* = 0$. Then the dual problem is simply the original problem with weights w. An optimal solution to the dual problem is a maximizer of w over $\boldsymbol{X}$. Thus (k, W) is supported.

(ii) $\lambda^* > 0$. Among solutions of the dual problem at λ^*, there exists a solution with less than k red edges and a solution with more than k edges. Then with Lemma 3.2, there exists a solution to the dual problem $X^=$ with exactly k red edges. Hence $\rho(X^=) = k$, and $w(X^=) = \mathcal{G}(\lambda^*)$. That is, solution $X^=$ corresponds to point (k, W) of the Pareto front. Moreover, $X^=$ is a maximizer of the convex combination of objectives $w + \lambda^*\rho$ over $\boldsymbol{X}$. Thus (k, W) is supported.

□

Let us illustrate the result on an instance of the Min Cost Spanning Tree problem. Consider a grid graph with $6 \times 6 = 36$ nodes. A vector of edge-costs c^0 is randomly generated in range $\{1, \ldots, 5\}$. The baseline solution is a tree x^0 with minimum cost ${c^0}^T x^0$, represented in red in left-top picture of Figure 3.4. A new vector of edge-cost c is defined by $c_e = 6 - c^0_e$ for every $e \in E$. Then the values of c are also in $\{1, \ldots, 5\}$, but the cheapest edges in c^0 are now the most expensive for costs c. For edge cost c, the baseline tree has cost $c^T x^0 = 268$ while the optimal cost is 120. In Figure 3.4 are represented 6 different solutions of the new Min Cost Spanning Tree instance. The corresponding Pareto front is represented in Figure 3.3.

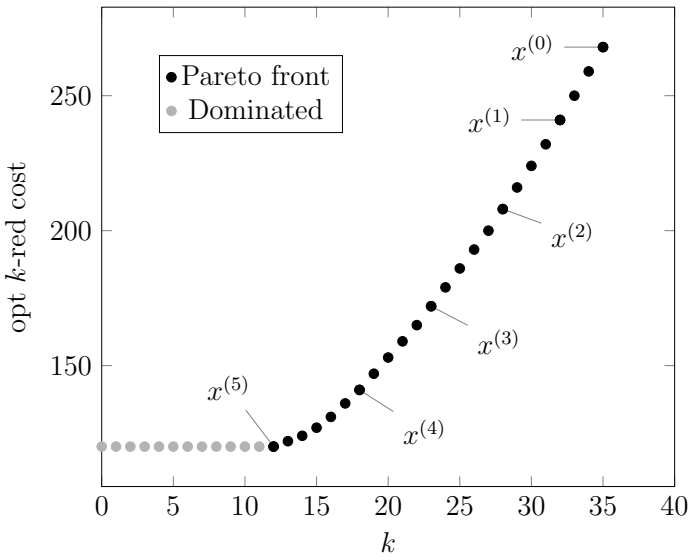


Figure 3.3: Pareto front for the considered grid graph instance, and points associated to solutions of Figure 3.4.

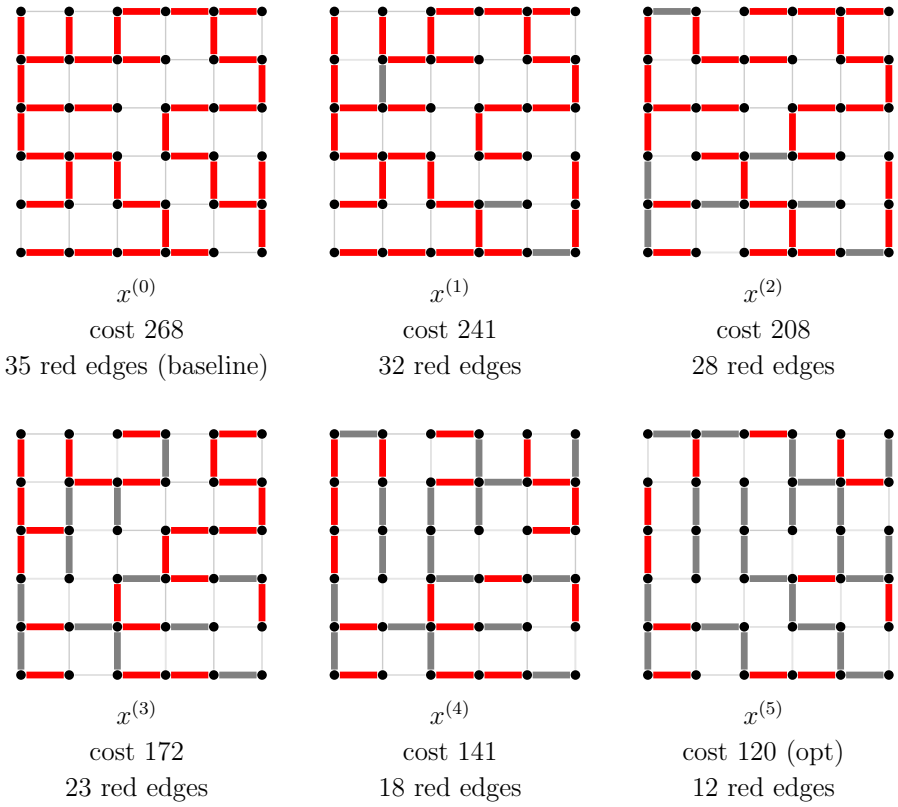


Figure 3.4: Six different solutions from the Pareto front.

3.3 k-red bipartite matching

Let us now consider k-red bipartite matchings. Given a bipartite graph $G = (V_1 \cup V_2, E)$, let $\boldsymbol{X}$ denote the set of matchings of G. Consider edge weights $w \in \mathbb{R}^E$. The Max-Weight Bipartite Matching Problem is solvable in polynomial time, by a flow algorithm or the Hungarian algorithm. Consider now the k-red variant. Given $\rho \in \{0,1\}^E$ and $k \in \mathbb{N}$, the k-red variant (Match-kRed) is to find a max-weight matching $X \in \boldsymbol{X}$ such that $\rho(X) \geq k$.

If all weights are unit or all edges are red, the problem is solvable in polynomial time using a min cost flow algorithm. In general case, the complexity of this problem is here left as open question. Indeed we prove a connection with the Exact Perfect Matching problem, whose complexity is an open problem. A perfect matching is a matching that matches all vertices of the graph. Let $\nu(G)$ denote the size of a maximum matching.

Exact Perfect Matching	
Input:	A bipartite graph $G = (V_1 \cup V_2, E)$, a subset $R \subseteq E$ of red edges, an integer k
Question:	Is there a perfect matching of the graph with exactly k red edges?

Finding a polynomial algorithm to solve the Exact Perfect Matching is a long-standing open problem since it was introduced in (Papadimitriou and Yannakakis, 1982). Non deterministic algorithms are known (Mulmuley et al., 1987), along with algorithms for complete graphs (Yi et al., 2002). Yuster (2012) provides an algorithm that either answers that an exact perfect matching does not exists, or returns a matching with k red edges of size at least $\nu(G) - 1$.

Theorem 3.5. *There is a polynomial reduction from* Exact Perfect Matching *to the decision problem associated with (Match-kRed).*

Proof. Let G be a bipartite graph with a subset of edges red, k an integer: they form an instance of the Exact Perfect Matching problem. An instance (G', ρ, w, W) of the decision version of (Match-kRed) is built as follows: let $G' = G$, let ρ be the indicator of red edges of G, let the weight w be the indicator vector of non-red edges of G, and $W = \nu(G) - k$, where $\nu(G)$ is the size of a perfect matching in G.

Assume there exists a perfect matching X of G with exactly k red edges. Then $w(X) = \nu(G) - k \geq W$ and $\rho(X) = k$, hence the instance (G', ρ, w, W) has a 'yes' answer. Conversely, let X' be a matching of G' such that $\rho(X') \geq k$ and $w(X') \geq \nu(G) - k$. Hence $\rho(X') + w(X') \geq \nu(G)$. By definition of ρ and w it holds that $\rho(X') + w(X') = |X'| \leq \nu(G)$. Hence X' is a perfect matching and equalities $\rho(X') = k$ and $w(X') = \nu(G) - k$ are both satisfied. Therefore there exists a perfect matching with exactly k red edges in G. □

The Lagrangian approach used in the matroid case can also be used for matchings. It was done, e.g., for the Exact Perfect Matching problem in Yuster (2012), and in Berger et al. (2011) for a budgeted matching problem. However a result similar to Lemma 3.2, providing an optimal solution from two Lagrangian solutions, is not to be expected: indeed there may exist a Lagrangian duality gap.

Let us recall a classical result in Lagrangian relaxation, that shows that the two approaches for matroid bases are closely related. Consider any k-red problem $\max w^T x$ for $x \in \mathcal{X}$, $\rho^T x \geq k$. Let $\mathcal{Q} = \operatorname{conv} \mathcal{X}$. For $\lambda \geq 0$, let $\mathcal{G}(\lambda) = \max_{x \in \mathcal{X}} w^T x + \lambda(\rho^T x - k)$.

Proposition 3.3 (Geoffrion, see Theorem 5.36 in (Korte and Vygen, 2012))**.** *The polytope $\mathcal{Q} \cap \{x \in \mathbb{R}^E : \ \rho^T x \geq k\}$ is integer if and only if for any value of the weights $w \in \mathbb{R}^E$, the dual bound $\min_{\lambda \geq 0} \mathcal{G}(\lambda)$ is equal to the optimal value of the problem.*

Proof. Geoffrion's theorem states – roughly speaking – that the dual bound corresponds to the value of the problem, where undualized constraints are convexified. Let us recall the proof for completeness. Consider the value $V = \max\{w^T x\colon\ x \in \mathcal{Q},\ \rho^T x \geq k\}$. Dualizing the k-red constraints, it comes that $V = \max_{x\in\mathcal{Q}} \min_{\lambda\geq 0} w^T x + \lambda(\rho^T x - k)$. The max/min can be inverted since both feasible sets are polytopes. Hence $V = \min_{\lambda\geq 0} \max_{x\in\mathcal{Q}} w^T x + \lambda(\rho^T x - k) = \min_{\lambda\geq 0} \mathscr{G}(\lambda)$. Hence the absence of Lagrangian duality gap is equivalent to $V = \max\{w^T x\colon\ x \in \mathcal{X}, \rho^T x \geq k\}$. This holds for every weight vector $w \in \mathbb{R}^E$ if and only if $\mathcal{Q} \cap \{x \in \mathbb{R}^E\colon\ \rho^T x \geq k\} = \operatorname{conv}\{x \in \mathcal{X}, \rho^T x \geq k\}$, i.e., $\mathcal{Q} \cap \{x \in \mathbb{R}^E\colon\ \rho^T x \geq k\}$ is integer. □

In Example 3.2, a small instance of k-red matching is given where there is a Lagrangian duality gap, and equivalently, the matching polytope intersected with the k-red constraint is not integer anymore.

Example 3.2 (Lagrangian duality gap for k-red matchings). Consider a bipartite graph $G = (U, V, E)$ with $U = \{u, u'\}$, $V = \{v, v', v''\}$ and edge-set $\{e_1, e_2, e_3, e_4\}$ as represented in Figure 3.5. Weights are $w = (1, 1, 2, 1)$. Edges e_2 and e_4 are red. Note here that there are 3 inclusion-wise maximal matchings: $M_{13} = \{e_1, e_3\}$ has weight 3 and 0 red edges; $M_{14} = \{e_1, e_4\}$ has weight 2 and 1 red edge; $M_{24} = \{e_2, e_4\}$ has weight 2 and 2 red edges. Let $k = 1$. The optimum of the k-red problem is thus 2 (attained by M_{14} and M_{24}). Consider now the vector $x^* = \frac{1}{2}(\chi^{M_{13}} + \chi^{M_{24}})$. It is in the matching polytope $\mathcal{Q}$. Also it satisfies $\rho^T x^* = 1 \geq k$. It has weight $\frac{1}{2}(1 + 1 + 2 + 1) = 2.5$. This shows that the intersection $\mathcal{Q} \cap \{x \in \mathbb{R}^E\colon\ \rho^T x \geq k\}$ contains x^*, which is not an element of $\mathcal{Q}^{k\text{-red}}$. The dual function is $\mathscr{G}(\lambda) = \max\{3 - \lambda, 2 + \lambda\}$. Its minimum is 2.5, while the optimum of the k-red problem is 2. ◇

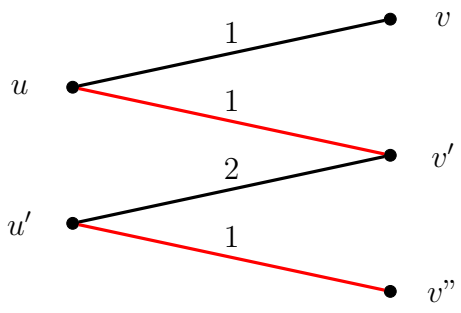


Figure 3.5: k-red matching instance from Example 3.2.

Conclusion

We studied k-red variants of polynomial combinatorial problems and especially matroid bases. This latter problem is polynomial. While the complexity result cannot be regarded as a new result, we proposed a complete picture of the problem with an algorithmic and a polyhedral perspective. In particular, the polyhedral characterization will serve on its own for robust approaches.

The approaches are both built upon the properties of matroid bases. We pointed out that the result cannot be extended in a straightforward manner to other problems, even for bipartite matchings that have a close structure. A perspective would be to investigate polyhedral characterizations of other k-red problems, first of all, k-red matchings.

Chapter 4

Anchor-Robustness for combinatorial problems

In this chapter we consider a combinatorial problem (P) with ground-set E, defined by $\min c(X)$ subject to $X \in \boldsymbol{X}$, where $\boldsymbol{X} \subseteq 2^E$ and $c(X) = \sum_{e \in X} c_e$ with cost vector $c \in \mathbb{R}^E$. Problem (P) is represented by integer program $\min c^T x$ subject to $x \in \mathcal{X}$, with $\mathcal{X} \subseteq \{0,1\}^E$.

We consider that uncertainty may impair the cost function. Nominal cost vector is $c \in \mathbb{R}^E$. Real cost vector may be $c^\delta = c + \delta$ for $\delta \in \Delta$, with Δ the uncertainty set. We assume that costs are non-negative for every uncertainty realization, that is, $c + \delta \geq 0$ for every $\delta \in \Delta$. By contrast the feasible set $\mathcal{X}$ is not subject to uncertainty.

The concept of anchor-robustness introduced in Chapter 2 is investigated for combinatorial problems under cost uncertainty. In Section 4.1, the Anchor-Robust problem (AnchRob) is formally defined, and compared to the recoverable robust problem (RecovRob) studied in the literature for combinatorial problems. In Section 4.2 we analyze the complexity of (AnchRob) for discrete and polyhedral uncertainty sets. We point out the connection with other robust approaches. In Section 4.3 we investigate the existence of MIP reformulations for (AnchRob) and (RecovRob). Finally in Section 4.4 we assess the price of anchor-robustness, that is, the decrease of the optimal value between (AnchRob) and (RecovRob).

4.1 Definitions

Let us first give definitions of anchored solutions and the Anchor-Robust problem. These correspond to the general definitions given in Chapter 2, now specialized to combinatorial problems.

4.1.1 Anchored sets

As in Chapter 2, we consider a baseline solution $X^0 \in \boldsymbol{X}$, and a collection of second-stage solutions $X^\delta \in \boldsymbol{X}$ $\forall \delta \in \Delta$ so that the second-stage solution is chosen after uncertainty realization $\delta \in \Delta$ is known. A collection of second-stage collections $(X^\delta)_{\delta\in\Delta}$ can be interpreted as a strategy of a player, who chooses solution X^δ depending on the adversarial realization of $\delta \in \Delta$. Given a collection of solutions $(X^\delta)_{\delta\in\Delta}$, the *worst-case cost* is $\max_{\delta\in\Delta} c^\delta(X^\delta)$. An anchored set is defined as follows.

Definition 4.1 (Anchored set)**.** *Let X^0 be a baseline solution and $(X^\delta)_{\delta\in\Delta}$ second-stage solutions. A subset $H \subseteq E$ is* anchored *w.r.t. X^0, $(X^\delta)_{\delta\in\Delta}$ if $H \subseteq X^0$ and $H \subseteq X^\delta$ for every $\delta \in \Delta$.*

Remark that it is Definition 2.1 from Chapter 2, rewritten in the special case of binary vectors and equivalence relation $=_1$. An interpretation of Definition 4.1 is that elements of H are guaranteed against uncertainty set Δ, since they are selected in the baseline X^0 and in every second-stage instance X^δ.

An anchored solution is a triplet $(X^0, H, (X^\delta)_{\delta\in\Delta})$ with $H \subseteq E$ a set that is anchored w.r.t. X^0, $(X^\delta)_{\delta\in\Delta}$.

Anchored solutions coincide with static-robust solutions when the anchored set is $H = X^0$, as noted in Observation 2.1. Note that since we are considering cost uncertainty only, any solution $X^0 \in \boldsymbol{X}$ is trivially a static-robust solution, i.e., feasible for any uncertainty realization. The coincidence with static-robust solutions is useful for the evaluation of worst-case costs of anchored solutions. With $H = X^0$ is anchored, the worst-case cost of the second-stage instances $\max_{\delta\in\Delta} c^\delta(X^\delta)$ is equal to the worst-case cost of the baseline $\max_{\delta\in\Delta} c^\delta(X^0)$. Indeed, recall that second-stage costs are non-negative, thus an optimal second-stage solution containing the anchored set $H = X^0$ is just $X^\delta = X^0$ for every $\delta \in \Delta$.

4.1.2 The Anchor-Robust problem

Let us now formulate the Anchor-Robust problem, derived from the general setting presented in Chapter 2. Consider a 2-stage robust optimization problem where:

- in first stage is decided a baseline solution X^0 and an anchored set $H \subseteq X^0$
- in second stage is decided a new solution X^δ such that $H \subseteq X^\delta$

Consider anchoring weights $a \in \mathbb{R}^E_+$. Given $k \geq 0$, it will be required that the anchored set has total anchoring weight at least k. Solutions of the Anchor-Robust problem will be anchored solutions $(X^0, H, (X^\delta)_{\delta\in\Delta})$ with anchored set H satisfying $a(H) \geq k$.

Consider first-stage cost vector $C \in \mathbb{R}^E$, second-stage cost vector $c \in \mathbb{R}^E$ and uncertainty set Δ. The objective function is the sum of the first-stage cost of the

baseline and the worst-case cost of second-stage solutions: $C(X^0)+\max_{\delta\in\Delta} c^\delta(X^\delta)$. This criterion was used in the literature for other robust approaches (see, e.g., (Hradovich et al., 2017)).

Given feasible set $\boldsymbol{X}$, costs $C, c \in \mathbb{R}^E$, uncertainty set Δ, anchoring weights $a \in \mathbb{R}^E_+$ and $k \geq 0$, the Anchor-Robust (AnchRob) problem is to find an anchored solution $(X^0, (X^\delta)_{\delta\in\Delta}, H)$ such that $a(H) \geq k$, and $C(X^0) + \max_{\delta\in\Delta} c^\delta(X^\delta)$ is minimized. This writes as the following program:

$$\begin{array}{llll} \text{(AnchRob)} & \min & C(X^0) + \max & \min\ c^\delta(X^\delta) \\ & X^0 \in \boldsymbol{X} & \delta \in \Delta & X^\delta \in \boldsymbol{X} \\ & H \subseteq X^0 & & H \subseteq X^\delta \\ & a(H) \geq k & & \end{array}$$

Note that the inner minimization problem is always feasible, since X^0 is a feasible solution that can be used in second stage. But for a given realization $\delta \in \Delta$ the cost $c^\delta(X^0)$ can be suboptimal, hence a solution X^δ with better cost $c^\delta(X^\delta) < c^\delta(X^0)$ is chosen.

A special case of interest is when anchoring weights are unitary. Then the anchored set must satisfy $|H| \geq k$ where k is an integer in $\{0, \ldots, |E|\}$. Another special case is when $C = 0$. Then the objective function of (AnchRob) coincides exactly with the static-robust objective function, that is, the worst-case cost. Note that for unit anchoring weights and $C = 0$, the problem is exactly the Worst-case minimization variant of the anchor-robust problem defined in Chapter 2.

Let us now propose a mathematical program for (AnchRob). Recall that $\mathcal{X}$ denotes the set of incidence vectors of elements of $\boldsymbol{X}$. The decision variables are as follows:

- $x \in \mathcal{X}$: incidence vector of baseline solution X^0;
- $h \in \{0,1\}^E$: incidence vector of the anchored set H;
- $y \in \mathcal{X}$: incidence vector of the second-stage solution X^δ. Note that y that is not implicitly indexed with δ, but does depend on the uncertainty realization.

The fact that h represents an anchored set can be enforced by the linear constraints: $h_e \leq x_e$ for every $e \in E$, and $h_e \leq y_e$ for every $e \in E$. The (AnchRob) problem is thus:

$$\begin{array}{llll} \text{(AnchRob)} & \min\ C^T x + & \max & \min\ (c+\delta)^T y \\ & x \in \mathcal{X} & \delta \in \Delta & y \in \mathcal{X} \\ & h \in \{0,1\}^E & & h \leq y \\ & h \leq x & & \\ & a^T h \geq k & & \end{array}$$

4.1.3 Connection with the Recoverable-Robust Problem

Let us now present a recoverable robust approach inspired from the literature for the considered combinatorial problem. Similarly to the (AnchRob) problem, a baseline solution X^0 is decided, then a second-stage solution X^δ is decided after the uncertainty realization is known. In the recoverable robust approach, the second-stage solution must share at least k elements with the first-stage solutions.

We consider the following recoverable-robust problem (RecovRob). The inputs are exactly the same as for (AnchRob), namely, feasible set $\boldsymbol{X}$, costs $C, c \in \mathbb{R}^E$, uncertainty set Δ, anchoring weights $a \in \mathbb{R}^E_+$ and $k \geq 0$. The (RecovRob) problem is

$$\begin{array}{llll} \text{(RecovRob)} & \min C(X^0) + & \max & \min c^\delta(X^\delta) \\ & X^0 \in \boldsymbol{X} & \delta \in \Delta & X^\delta \in \boldsymbol{X} \\ & & & a(X^0 \cap X^\delta) \geq k \end{array}$$

The constraint $a(X^0 \cap X^\delta) \geq k$ can be interpreted as a constraint on the distance between the baseline solution and the second-stage solution. It can be written as the existence of a set H^δ, $H^\delta \subseteq X^0$, $H^\delta \subseteq X^\delta$ such that $a(H^\delta) \geq k$. The difference with the Anchor-Robust problem is thus the dependency of the set of common decisions to uncertainty realization δ. In the case of unit anchoring weights, (RecovRob) is the problem studied in the literature (Kasperski and Zieliński, 2016; Hradovich et al., 2017; Kasperski and Zieliński, 2017).

Using binary vectors x and y to represent sets X^0 and X^δ, the constraint $|X^0 \cap X^\delta| \geq k$ (resp. $a(X^0 \cap X^\delta) \geq k$) can be cast as the quadratic constraint $x^T y \geq k$ (resp. $\sum_{e \in E} a_e x_e y_e \geq k$). The (RecovRob) problem can thus be written as the following mathematical program:

$$\begin{array}{llll} \text{(RecovRob)} & \min C^T x + & \max & \min (c+\delta)^T y \\ & x \in \mathcal{X} & \delta \in \Delta & y \in \mathcal{X} \\ & & & \sum_{e \in E} a_e x_e y_e \geq k \end{array}$$

As a consequence from the connection between (AnchRob) and (RecovRob), the optimal value of (RecovRob) is always better (lower) than the optimal value of (AnchRob). A case where the solutions are not the same, and thus the difference between optimal values is strict, was given in Chapter 2.

4.2 Complexity for discrete and polyhedral uncertainty sets

In this section, the complexity of (AnchRob) is investigated for a variety of uncertainty sets. Note first that (AnchRob) is always harder than the original problem (P). Thus we mostly focus on cases where (P) is solvable in polynomial time, to see whether there is an increase in complexity in the anchor-robust problem.

We make the even stronger assumption that an efficient polyhedral characterization is known for the original problem. It means that a description of $\text{conv}(\mathcal{X})$ is known, and either $\text{conv}(\mathcal{X})$ is described by a polynomial number of inequalities or these inequalities can be separated in polynomial time. Hence the original problem (P) can be solved as the linear program $\min\ c^T x$, $x \in \text{conv}(\mathcal{X})$ in polynomial time. This is the case, e.g., for spanning trees, shortest paths, or bipartite matchings.

4.2.1 Uncertainty sets

Let us present the considered uncertainty sets. Let $\widehat{\delta} \in \mathbb{R}^E_+$ be a vector, where $\widehat{\delta}_e$ is the worst-case deviation of the cost of element e. Set Δ can be a polytope, given by an outer description

$$\Delta = \{(\widehat{\delta}_e u_e) \colon\ u \geq 0, Mu \leq \beta\}$$

where M is a matrix and β a vector with κ lines. In particular we assume (w.l.o.g. up to a change of $\widehat{\delta}$) that $\max_{u \geq 0, Mu \leq \beta} u_e = 1$ for every $e \in E$. We assume κ polynomial in $|E|$, so that optimizing a linear function over Δ can be done in polynomial time. Two particular cases are as follows. Given $\Gamma \in \{0, \dots, |E|\}$, set Δ is a *polyhedral budgeted uncertainty set* if

$$\Delta = \{(\widehat{\delta}_e u_e) \colon\ u_e \in [0,1]^E, \sum_{e \in E} u_e \leq \Gamma\}$$

Set Δ is a *box uncertainty set* if

$$\Delta = \{(\widehat{\delta}_e u_e) \colon\ u_e \in [0,1]^E\}$$

which corresponds to the special case where $\Gamma = |E|$. Set Δ can also be defined as a discrete set. In particular, set Δ is a *discrete budgeted set* if

$$\Delta = \{(\widehat{\delta}_e u_e) \colon\ u_e \in \{0,1\}^E, \sum_{e \in E} u_e \leq \Gamma\}$$

This is budgeted uncertainty as defined in (Bertsimas and Sim, 2004).

Importantly, note that the uncertainty set cannot be convexified without changing (AnchRob) optimal value. This is to be contrasted with other robust approaches, e.g., static-robustness. For that reason, we consider both polyhedral budgeted and discrete budgeted sets, though the former is the convex hull of the latter. Let us illustrate this point on an example.

Example 4.1 (Do not convexify Δ). Consider the Selection problem with n items and $p = 1$. Consider nominal costs $c = 0$. Let $\widehat{\delta}_e = 1$ for every $e \in E$ and $\Gamma = 1$. Consider Δ (resp. $\text{conv}(\Delta)$) the discrete (resp. polyhedral) budgeted uncertainty set associated with $\widehat{\delta}$ and Γ. Consider the case where there is no first stage, and the (AnchRob) problem is $\max_{\delta\in\Delta} \min_{x\in\mathcal{X}}(c+\delta)^T x$. It can be seen as a game where uncertainty player plays first, then the optimizer player plays second.

- If the uncertainty set is discrete Δ, the uncertainty player adds cost $\widehat{\delta}_e = 1$ to one edge e, then the second player finds a solution with cost 0 by selecting any other element $f \neq e$.
- If the uncertainty set is polyhedral $\text{conv}(\Delta)$, the uncertainty player can now "split" uncertainty, as $\delta_e = \frac{1}{n}$ for every $e \in E$. The second player is then forced to select a solution with cost $\frac{1}{n}$.

Optimal values of (AnchRob) for Δ and $\text{conv}(\Delta)$ are thus different. ◇

4.2.2 Complexity for extreme values of k

Let us start by analyzing the (AnchRob) problem depending on the value of k, to see when it coincides with other robust problems, and obtain first complexity results.

4.2.2.1 Case $k = 0$

Consider first the case where $k = 0$. No anchored element is required, hence there is no linking constraint between the baseline solution and the second-stage solution. The (AnchRob) problem thus boils down to solving independently two problems:

- Finding a baseline solution x that minimizes $C^T x$ (if there are first-stage costs). This is simply an instance of the original problem.
- Solving $\max_{\delta\in\Delta} \min_{y\in\mathcal{X}}(c+\delta)^T y$.

This latter problem is the adjustable-robust problem (Adj-P), with no first-stage variable. It is also referred to in the literature as the adversarial problem (see (Nasrabadi and Orlin, 2013; Kasperski and Zieliński, 2016)). It can also be seen as an interdiction problem, where a first player (the uncertainty player) tries to increase the optimal value of the original problem, by disrupting the elements costs (see (Frederickson and Solis-Oba, 1999)).

Note that in that case $k = 0$, (AnchRob) and (RecovRob) coincide.

Proposition 4.1. *For $k = 0$, the (AnchRob) problem is:*
- *polynomial for polyhedral uncertainty;*
- *NP-hard for discrete budgeted uncertainty.*

Proof. Consider polyhedral uncertainty. For $k = 0$, the (AnchRob) problem reduces to $\max_{\delta\in\Delta} \min_{y\in\text{conv}(\mathcal{X})}(c+\delta)^T y$. Since Δ and $\text{conv}(\mathcal{X})$ are polytopes, max and min can be inverted. Replacing $\max_{\delta\in\Delta}(c+\delta)^T y$ by its dual, we obtain a minimization problem $\min c^T y + \eta^T \beta$ for $y \in \text{conv}(\mathcal{X})$, $\widehat{\delta} y \leq M^T \eta$, $\eta \geq 0$, which is an LP. It is solvable in polynomial time by assumption on $\text{conv}(\mathcal{X})$.

Consider discrete budgeted uncertainty. The NP-hardness result is shown in the special case of spanning trees. In (Frederickson and Solis-Oba, 1999) it was shown that the following problem Spanning Tree Interdiction is NP-complete: given graph $G(V, E)$, integer $\Gamma \in \mathbb{N}$, edge weight $w \in \{0, 1\}^E$, integer W, decide if there exists a subset $S \subseteq E$, $|S| \leq \Gamma$ such that the min weight spanning tree in the graph $(V, E \setminus S)$ has weight $\geq W$. Let $(G = (V, E), \Gamma, w, W)$ be an instance of Spanning Tree Interdiction. Let $\mathcal{X}$ denote the set of spanning trees of graph G. Let cost vector $c = w$, $\widehat{\delta}_e = (|V| - 1) \max_{e\in E} w_e$, budget Γ. Remark that in the graph G, for an uncertainty realization δ, edges have either cost w_e or cost $w_e + \widehat{\delta}_e$. Due to the value of $\widehat{\delta}_e$, a min spanning tree for costs $c+\delta$ uses only edges with cost w_e if possible. The (AnchRob) problem is $\max_{\delta\in\Delta} \min_{x\in\mathcal{X}}(c+\delta)^T x$. Let $W \geq 0$. Then it holds that the optimal value of (AnchRob) is at least W if and only if the answer to the Spanning Tree Interdiction problem on instance $(G = (V, E), \Gamma, w, W)$ is 'yes'. Indeed, an uncertainty realization $\delta \in \Delta$ corresponds to a set S of Γ edges that are disrupted to cost $\widehat{\delta}_e$. There exists a spanning tree in the graph (V, E) with cost W if and only if there exists a spanning tree in the graph $(V, E \setminus S)$ with cost W. The claimed NP-hardness result follows. □

4.2.2.2 Case $k = k^{\max}$

Let us now consider that the original problem satisfies the assumption

All solutions of (P) have the same cardinality $k^{\max}$. (FixedCard)

This assumption holds for matroid bases.

Consider unit anchoring weights and $k = k^{\max}$. Then the inequality $\sum_{e\in E} h_e \geq k$ implies $h = x$. Given a first-stage solution x, since costs are non-negative a second-stage optimal solution is $y = x$. (AnchRob) problem is thus to find a static-robust solution x minimizing the worst-case cost $\max_{\delta\in\Delta}(c+\delta)^T x$.

Note that (AnchRob) and (RecovRob) coincide in this case.

Proposition 4.2. *Under assumption* (FixedCard)*, (AnchRob) is polynomial for $k = k^{\max}$ for polyhedral budgeted uncertainty and discrete budgeted uncertainty.*

Proof. The complexity of (AnchRob) is the complexity of the static-robust problem for the considered uncertainty sets. Consider discrete budgeted Δ. Then the problem is $\min_{x\in\mathcal{X}} \max_{\delta\in\Delta}(c+\delta)^T x$. Using the technique from Bertsimas and Sim (Bertsimas and Sim, 2004), it holds that this problem can be solved through $|E|+1$ instances of the original problem (P). Consider now a polyhedral uncertainty set Δ. In $\min_{x\in\mathcal{X}} \max_{\delta\in\Delta}(c+\delta)^T x$ the inner max is on a linear function of δ. Hence set Δ can be convexified without changing the optimal value. The polynomiality result follows. □

4.2.3 Complexity depending on the uncertainty set

Let us now examine how the complexity of (AnchRob) depends on the uncertainty set: box uncertainty, polyhedral uncertainty, and discrete budgeted uncertainty.

4.2.3.1 Box uncertainty

For box uncertainty, the worst case over Δ in (AnchRob) is attained for the realization $\delta = \widehat{\delta}$: for every $y \in \mathcal{X}$, $\max_{\delta\in\Delta}(c+\delta)^T y = (c+\widehat{\delta})^T y$. Problem (AnchRob) is then

$$\begin{aligned} \min \quad & C^T x + (c+\widehat{\delta})^T y \\ & x \in \mathcal{X} \\ & y \in \mathcal{X} \\ & h \in \{0,1\}^E \\ & h \le x,\ h \le y \\ & a^T h \ge k \end{aligned}$$

It coincides with (RecovRob).

Consider first the case where there are no first-stage costs ($C = 0$). Then w.l.o.g. the baseline and the second-stage solution can be set equal. With $x = y$ the problem corresponds to the AnchReopt–Anch problem with cost function $c + \widehat{\delta}$. For general C and unit anchoring weights, the problem was specifically studied for spanning trees (Hradovich et al., 2017) and matroid bases (Lendl et al., 2019) and shown to be solvable in polynomial time. These results are wrapped up in the following proposition:

Proposition 4.3. *Under box uncertainty, (AnchRob) for matroid bases is:*
- *NP-hard for non-unit anchoring weights, even for $C = 0$;*
- *polynomial for unit anchoring weights.*

4.2.3.2 Polyhedral uncertainty

Let us now consider polyhedral uncertainty, which subsumes box uncertainty. Proposition 4.3 implies that (AnchRob) is NP-hard even for $C = 0$. However this relies on non-unit anchoring weights. An open question is the following

Question 4.1. *Under polyhedral budgeted uncertainty, what is the complexity of (AnchRob) for matroid bases with unit anchoring weights?*

Note that the corresponding question for the (RecovRob) problem, i.e., the complexity under polyhedral uncertainty and unit weights, is open even for the spanning tree problem (Kasperski and Zieliński, 2016). It is not clear whether the complexity status of (AnchRob) would directly imply the answer to this open question. However the same proof technique may be used for both problems.

4.2.3.3 Discrete budgeted uncertainty

In the case of discrete budgeted uncertainty, the result for $k = 0$ from Proposition 4.1 implies

Proposition 4.4. *Under discrete budgeted uncertainty, (AnchRob) is NP-hard.*

Note that the result does not rely on anchoring weights.

4.3 MIP reformulations

In this section, we consider polyhedral uncertainty. As shown in Section 4.2, it is a case where the (AnchRob) problem can be NP-hard. To address the problem we investigate exact approaches based on mixed-integer programming.

MIP reformulations of static-robust problems (Soyster, 1973; Bertsimas and Sim, 2004) were important results advocating for the practical implementability of robust optimization. The class of robust 2-stage problems is more challenging. MIP formulations for robust 2-stage problems are often based on an enumeration of extreme points of the uncertainty set, as noted in the literature review of Chapter 1. By contrast, we investigate MIP formulations where there is no enumeration over the uncertainty set.

In some special cases, a compact MIP reformulation of (AnchRob) was obtained, namely box uncertainty (see Section 4.2.3). The question is to see whether a similar reformulation can be found for more general polyhedral uncertainty sets. We first give a positive result by providing an MIP reformulation for (AnchRob) in Section 4.3.1. Then we discuss the existence of a similar MIP reformulation for (RecovRob) in Section 4.3.2.

4.3.1 Efficient MIP reformulation for (AnchRob)

Consider polyhedral uncertainty $\Delta = \{(\widehat{\delta}_e u_e) \colon \ u \geq 0, Mu \leq \beta\}$. Let $\eta \in \mathbb{R}^{\kappa}_{+}$ denote a vector of dual variables associated to the inequalities $Mu \leq \beta$ in the definition of Δ. Let $\mathrm{diag}(\widehat{\delta})$ denote the diagonal matrix of $\mathbb{R}^{E \times E}$ with (e,e) coefficient equal to $\widehat{\delta}_e$. Then for any $y \in [0,1]^E$, the dual of the maximization problem $\max_{\delta \in \Delta} \widehat{\delta}^T y$ is to minimize $\eta^T \beta$ for $\mathrm{diag}(\widehat{\delta}) y \leq M^T \eta$, $\eta \in \mathbb{R}^{\kappa}_{+}$.

For the (AnchRob) problem under polyhedral uncertainty, we obtain the following result

Theorem 4.1. *The (AnchRob) problem under polyhedral uncertainty $\Delta = \{(\widehat{\delta}_e u_e) \colon u \geq 0, Mu \leq \beta\}$ and anchoring weights $a \in \mathbb{R}^E$ admits the following MIP reformulation (F_{Anch}).*

$$
\begin{aligned}
(F_{\mathrm{Anch}}) \quad & \min \ \ C^T x + c^T y + \eta^T \beta \\
& x \in \mathcal{X} \\
& y \in \mathrm{conv}(\mathcal{X}) \\
& h \in \{0,1\}^E \\
& h \leq x \\
& h \leq y \\
& a^T h \geq k \\
& \mathrm{diag}(\widehat{\delta}) \ y \leq M^T \eta \\
& \eta \in \mathbb{R}^{\kappa}_{+}
\end{aligned}
$$

Proof. Consider the second-stage minimization problem in (AnchRob): $\min(c+\delta)^T y$ for $y \in \mathcal{X}, h \leq y$. Note that $h \in \{0,1\}^E$ is fixed. The set $F_h = \{y \in [0,1]^E \colon \ y \geq h\}$ is a face of the hypercube $[0,1]^E$. The second-stage problem is to minimize the linear function $(c+\delta)^T y$ for $y \in \mathcal{X} \cap F_h$, or equivalently, for $y \in \mathrm{conv}(\mathcal{X} \cap F_h)$.

Let us show that $\mathrm{conv}(\mathcal{X} \cap F_h) = \mathrm{conv}(\mathcal{X}) \cap F_h$. Since $\mathrm{conv}(\mathcal{X}) \subseteq [0,1]^E$, the set $\mathrm{conv}(\mathcal{X}) \cap F_h$ is a face of $\mathrm{conv}(\mathcal{X})$. The convex hull $\mathrm{conv}(\mathcal{X})$ is an integer polytope, hence its restriction to face $\mathrm{conv}(\mathcal{X}) \cap F_h$ is also an integer polytope.

Thus the inner minimization is the linear program $\min \ (c+\delta)^T y$ for $y \in \mathrm{conv}(\mathcal{X}) \cap F_h$. Because Δ is a polytope, strong duality gives $\max_{\delta \in \Delta} \min_{y \in \mathrm{conv}(\mathcal{X}) \cap F_h} (c+\delta)^T y = \min_{y \in \mathrm{conv}(\mathcal{X}) \cap F_h} \max_{\delta \in \Delta} (c+\delta)^T y$. The maximum $\max_{\delta \in \Delta} \delta^T y$ can be dualized into $\eta^T \beta$ for $\mathrm{diag}(\widehat{\delta}) \ y \leq M^T \eta$, $\eta \in \mathbb{R}^{\kappa}_{+}$. Plugging into the first-stage minimization problem of (AnchRob), we obtain the claimed MIP reformulation (F_{Anch}). □

Importantly the proof of Theorem 4.1 does not require any further property on the set $\mathcal{X}$, except that it is contained in the hypercube.

The reformulation (F_{Anch}) is an MIP formulation whenever linear inequalities defining $\mathcal{X}$ and $\text{conv}(\mathcal{X})$ are known. The size of (F_{Anch}) is polynomial in the size of the original MIP and the number of constraints κ defining set Δ. Indeed if q denotes the number of constraints defining $\text{conv}(\mathcal{X})$, the MIP reformulation (F_{Anch}) has $3|E|$ binary variables, $|E| + \kappa$ continuous variables, and $2q + 3|E| + 1$ linear constraints.

Note that we cannot in general impose $y \in \mathcal{X}$. The optimal solution of (F_{Anch}) may have a y-vector that is a fractional point in $\text{conv}(\mathcal{X})$, thus not an extreme point of $\text{conv}(\mathcal{X})$.

To see this, consider again the instance of Example 4.1. Consider the MIP reformulation (F_{Anch}) in that case. Since $k = 0$, first-stage solution x can be dropped (set $x = y$ w.l.o.g.). The obtained MIP is to minimize $\eta_0 + \sum_{e\in E} \eta_e$ for $y \in \text{conv}(\mathcal{X}) = \{y \in [0,1]^E \colon \ \sum_{e\in E} y_e = 1\}$, $y_e \leq \eta_0 + \eta_e$ for every $e \in E$, $\eta_0 \geq 0$, $\eta \in \mathbb{R}^E_+$. An optimal solution of cost $\frac{1}{n}$ is attained for the fractional point $y_e = \frac{1}{n}$ for every $e \in E$ (setting $\eta_0 = \frac{1}{n}$, $\eta_e = 0$ for every $e \in E$). By contrast an integer y would lead to a solution of cost 1.

In particular, this highlights that a description of $\text{conv}(\mathcal{X})$ is necessary to use (F_{Anch}). For the considered combinatorial problems, a valid formulation is known, i.e., a polytope $\mathcal{P}$ such that $\mathcal{X} = \mathcal{P} \cap \{0,1\}^E$. However replacing $\text{conv}(\mathcal{X})$ by $\mathcal{P}$ in (F_{Anch}) would not lead to a valid reformulation of (AnchRob).

4.3.2 Existence of MIP reformulation for (RecovRob)

Let us now make a connection with (RecovRob), for which an MIP reformulation would also be of interest. Further assumption on the problem is needed. In Chapter 3, we showed that when the original problem (P) is to find a max-weight basis in a matroid, its feasible set $\mathcal{X}$ satisfies the following property: $\forall x \in \mathcal{X}$, $\text{conv}(\mathcal{X} \cap \{y \colon \ \sum_{e\in E} x_e y_e \geq k\}) = \text{conv}(\mathcal{X}) \cap \{y \colon \ \sum_{e\in E} x_e y_e \geq k\}$. Here x plays the role of the red edges incidence vector. The equality corresponds to the integrality of the k-red polytope from Theorem 3.4.

Let us consider problems (P) and anchoring weights $a \in \mathbb{R}^E$ that satisfy a similar property, denoted by (Integ):

$$\forall x \in \mathcal{X}, \quad \text{conv}\left(\mathcal{X} \cap \{y \colon \ \sum_{e\in E} a_e x_e y_e \geq k\}\right) = \text{conv}(\mathcal{X}) \cap \{y \colon \ \sum_{e\in E} a_e x_e y_e \geq k\}. \tag{Integ}$$

Note that the left-hand side set is always included in the right-hand side set. The assumption (Integ) holds when the polytope $\text{conv}(\mathcal{X}) \cap \{y \colon \ \sum_{e\in E} a_e x_e y_e \geq k\}$ is integer.

Let (F_{Rel}) denote the relaxation of (F_{Anch}) where $h \in \{0,1\}^E$ is relaxed into $h \in [0,1]^E$.

Proposition 4.5. *If the problem satisfies assumption* (Integ)*, formulation* (F_{Rel}) *is an MIP reformulation for (RecovRob) under polyhedral uncertainty.*

Proof. For fixed $x \in \{0,1\}^E$, let $F_x = \{y \in \mathbb{R}^E : \sum_{e\in E} a_e x_e y_e \geq k\}$. Consider the second-stage minimization problem in (RecovRob): for fixed $x \in \mathcal{X}$, it is to minimize $(c+\delta)^T y$ for $y \in \mathcal{X} \cap F_x$. By the assumption (Integ), it is min $(c+\delta)^T y$ for $y \in \text{conv}(\mathcal{X}) \cap F_x$. Because Δ is a polytope, strong duality gives $\max_{\delta\in\Delta} \min_{y\in\text{conv}(\mathcal{X})\cap F_x} (c+\delta)^T y = \min_{y\in\text{conv}(\mathcal{X})\cap F_x} \max_{\delta\in\Delta} (c+\delta)^T y$. The inner maximum can again be dualized. We obtain the (quadratic) reformulation

$$\begin{aligned}
&\min \;\; C^T x + c^T y + \eta^T \beta \\
&x \in \mathcal{X} \\
&y \in \text{conv}(\mathcal{X}) \\
&\textstyle\sum_{e\in E} a_e x_e y_e \geq k \\
&\text{diag}(\widehat{\delta})\; y \leq M^T \eta \\
&\eta \in \mathbb{R}^{\kappa}_+
\end{aligned}$$

Now consider linearizing the quadratic constraints. Let $h \in [0,1]^E$. The constraint $\sum_{e\in E} a_e x_e y_e \geq k$ can be replaced by: $\sum_{e\in E} a_e h_e \geq k$, $h_e \leq x_e$, $h_e \leq y_e$. Then $h_e \leq \min(x_e, y_e)$ which is equal to $x_e y_e$ since $x_e \in \{0,1\}$. Plugging the linearization constraints into the reformulation, we obtain (F_{Rel}) as claimed. □

Hence (AnchRob) and (RecovRob) have reformulations (F_{Anch}) and (F_{Rel}) that vary only by whether h variables are binary or continuous. However for (RecovRob) assumption (Integ) was needed to dualize the recourse problem.

We now show that obtaining a valid reformulation for (RecovRob) as (F_{Rel}) is more an exception, rather than a general rule. Assumption (Integ) was necessary in the proof of Proposition 4.5; if this assumption is not satisfied, the validity of the reformulation may fall down. Let us now formalize this idea. Note first that (F_{Rel}) can be written as $\min_{x\in\mathcal{X}} \phi(x)$, where $\phi(x)$ is the optimal value of a linear program that can be solved in polynomial time. This holds since $x \in \mathcal{X}$ are the only binary variables in (F_{Rel}).

Proposition 4.6. *If $P \neq NP$, the problem (RecovRob) cannot be reformulated as $\min_{x\in\mathcal{X}} \phi(x)$, where $\phi(x)$ is the optimal value of a minimization linear program that is solvable in polynomial time.*

In particular, formulation (F_{Rel}) *is not always a valid reformulation for (RecovRob).*

Proof. Let us show that for a given first-stage solution $x \in \mathcal{X}$, computing its optimal value in a solution of the (RecovRob) can be NP-hard. Consequently if $P \neq NP$, (RecovRob) cannot be written as $\min_{c\in\mathcal{X}} \phi(x)$ Consider the following example. The underlying problem is matroid basis. The uncertainty set is box, and anchoring weights are $a \in \mathbb{R}^E$, not unitary. For a given $x \in \mathcal{X}$ the polytope $\text{conv}(\mathcal{X}) \cap F_x$ is not integer, as shown in Chapter 3; thus assumption (Integ) does not hold. The (RecovRob) is to minimize $C^T x + (c + \widehat{\delta})^T y$ for $x, y \in \mathcal{X}$, $\sum_{e\in E} a_e x_e y_e \geq k$. This problem is NP-hard for fixed $x \in \mathcal{X}$, as noted in Proposition 4.3. By contrast, for fixed x formulation (F_{Rel}) is a linear program, thus solvable in polynomial time. □

The results from Theorem 4.1 and Proposition 4.6 show that (AnchRob) and (RecovRob) do not have the same properties w.r.t. MIP reformulations. Problem (AnchRob) always has a MIP reformulation, regardless of the problem and anchoring weights. For (RecovRob) the existence of such a reformulation is not always satisfied. This reveals that (RecovRob) is computationally harder than (AnchRob). It advocates for the use of anchor-robustness, at the expense of solutions that have a higher cost w.r.t. recoverable robust solutions. Let us now assess this increase in cost, referred to as the price of anchor-robustness.

4.4 The price of anchor-robustness

Consider an instance of (AnchRob) (or (RecovRob)) problem, formed with $\mathcal{X}$, costs c, C, set Δ, unit anchoring weights and integer k. We assume that the problem (P) satisfies the assumption that all solutions have the cardinality $k^{\max}$. Let $Cost_{Anch}(k)$ and $Cost_{Recov}(k)$ denote the optimal values of the (AnchRob) and (RecovRob) respectively. Anchor-robustness and recoverable robustness are both a middle ground between adjustable robustness (case $k = 0$) and static robustness (case $k = k^{\max}$). Let $Cost_{Adj} = Cost_{Anch}(0)$ and $Cost_{Stat} = Cost_{Anch}(k^{\max})$. It holds that for every $k \in \{0, \ldots, k^{\max}\}$

$$Cost_{Adj} \leq Cost_{Recov}(k) \leq Cost_{Anch}(k) \leq Cost_{Stat}.$$

We now investigate the gap

$$PoAR(k) = \frac{Cost_{Anch}(k) - Cost_{Recov}(k)}{Cost_{Recov}(k)}$$

that we call the *price of anchor-robustness*. In the sequel, the price of anchor-robustness is examined both theoretically and numerically.

4.4.1 Unboundedness

A negative result is that

Proposition 4.7. *The price of anchor-robustness is unbounded, even for unit anchoring weights and polyhedral budgeted uncertainty.*

Proof. Consider the Selection problem, with feasible set $\mathcal{X} = \{x \in \{0,1\}^E : \sum_{e\in E} x_e = p\}$ for some $p \leq |E|$. Consider unit anchoring weights and $k = p$. Selection is a special case of matroid bases, hence assumption (Integ) is satisfied. Let $C = 0$.

Consider problem (AnchRob). Note that the existence of $h \in \{0,1\}^E$, $\sum_{e\in E} h_e \geq k$, $y \geq h$ implies that $h = y$, since $k = p$. Hence y must be an integer solution. Problem (AnchRob) thus writes: $\min_{y\in\mathcal{X}} \max_{\delta\in\Delta}(c+\delta)^T y$. Hence problem (AnchRob) coincides with the static-robust problem.

Consider now problem (RecovRob). Note that for any $y \in \text{conv}(\mathcal{X})$, the equality $\sum_{e\in E} y_e = p$ is satisfied. Setting $h = y$, we get $h \in [0,1]^E$ such that $\sum_{e\in E} h_e \geq k$. Problem (RecovRob) thus writes: $\min_{y\in\text{conv}(\mathcal{X})} \max_{\delta\in\Delta}(c+\delta)^T y$. Inverting min and max, it is equal to $\max_{\delta\in\Delta} \min_{y\in\text{conv}(\mathcal{X})}(c+\delta)^T y = \max_{\delta\in\Delta} \min_{y\in\mathcal{X}}(c+\delta)^T y$. Hence problem (RecovRob) coincides with the adjustable-robust problem.

Let us now give an example where

$$Cost_{Adj} \leq Cost_{Recov}(k) = \tfrac{1}{n} < 1 = Cost_{Anch}(k) \leq Cost_{Stat}.$$

Let $c = 0$. Consider polyhedral budgeted uncertainty with $\Gamma = 1$ and $\widehat{\delta}_e = 1$ for every $e \in E$. Let $p = k = 1$. The problem is thus to pick one item. Note that all items are the same. In problem (AnchRob), one item should be chosen in first stage. An optimal solution has cost $\max_{\delta\in\Delta} \widehat{\delta}_e = 1$ hence $Cost_{Anch}(k) = 1$. In problem (RecovRob), an optimal second-stage solution is to pick the item $e \in E$ with minimum deviation δ_e. Such an item has cost at most $\frac{1}{n}$, attained for uncertainty realization $\delta_e = \frac{1}{n}$ for every $e \in E$. Hence $Cost_{Recov}(k) = \frac{1}{n}$. Hence $PoAR(k) = n - 1$. □

4.4.2 Numerical experiments

Let us now evaluate the price of anchor-robustness in numerical experiments, on problem Selection.

Given integer p, the problem has feasible set $\mathcal{X} = \{x \in \{0,1\}^E : \sum_{e\in E} x_e = p\}$. Its convex hull is $\text{conv}(\mathcal{X}) = \{x \in [0,1]^E : \sum_{e\in E} x_e = p\}$. Consider first-stage

costs $C \in \mathbb{R}^E$, second-stage costs $c \in \mathbb{R}^E$, integer k, and polyhedral budgeted uncertainty with budget Γ and deviations $\widehat{\delta} \in \mathbb{R}^E$.

Using Theorem 4.1, a reformulation for the (AnchRob) problem is

$$\begin{array}{l}
\min \;\; C^T x + c^T y + \eta\Gamma + \sum_{e\in E} \rho_e \\
x \in \{0,1\}^E \\
\sum_{e\in E} x_e = p \\
y \in [0,1]^E \\
\sum_{e\in E} y_e = p \\
h \in \{0,1\}^E \\
h \leq x \\
h \leq y \\
\sum_{e\in E} h_e \geq k \\
\widehat{\delta}_e y_e \leq \rho_e + \eta \;\; \forall e \in E \\
\eta \geq 0 \\
\rho \in \mathbb{R}^E_+
\end{array}$$

By Proposition 4.5, a reformulation for (RecovRob) is the MIP where integrality of h is relaxed into $h \in [0,1]^E$.

We consider instances of Selection with $E = \{1, \ldots, n\}$ a set of $n = 50$ items and $p = 10$. Costs C_e and c_e are randomly generated in range $\{1, \ldots, 5\}$ for every $e \in E$. The uncertainty set is polyhedral budgeted uncertainty set, where deviation $\widehat{\delta}_e$ is also randomly generated in range $\{1, \ldots, 5\}$ for every $e \in E$. A total number of 100 random triplets $(C, c, \widehat{\delta})$ is generated.

The (RecovRob) and (AnchRob) problems can be solved by the MIP reformulation derived from Theorem 4.1 which is compact. The corresponding MIP formulations are solved using Julia, JuMP v0.18 and Cplex 12.8. We first mention that all instances are solved within a few seconds.

Let us illustrate the optimal values of (RecovRob) and (AnchRob) depending on k and Γ. In Figure 4.1, we represent, for each budget $\Gamma = 1, 3, 5, 10, 15, 20, 50$:
– in blue, the optimal value of (AnchRob) depending on the value of $k \in \{0, \dots, p\}$;
– in green, the optimal value of (RecovRob) depending on the value of $k \in \{0, \dots, p\}$;
– in gray the area between the two curves, corresponding to the price of anchor-robustness.

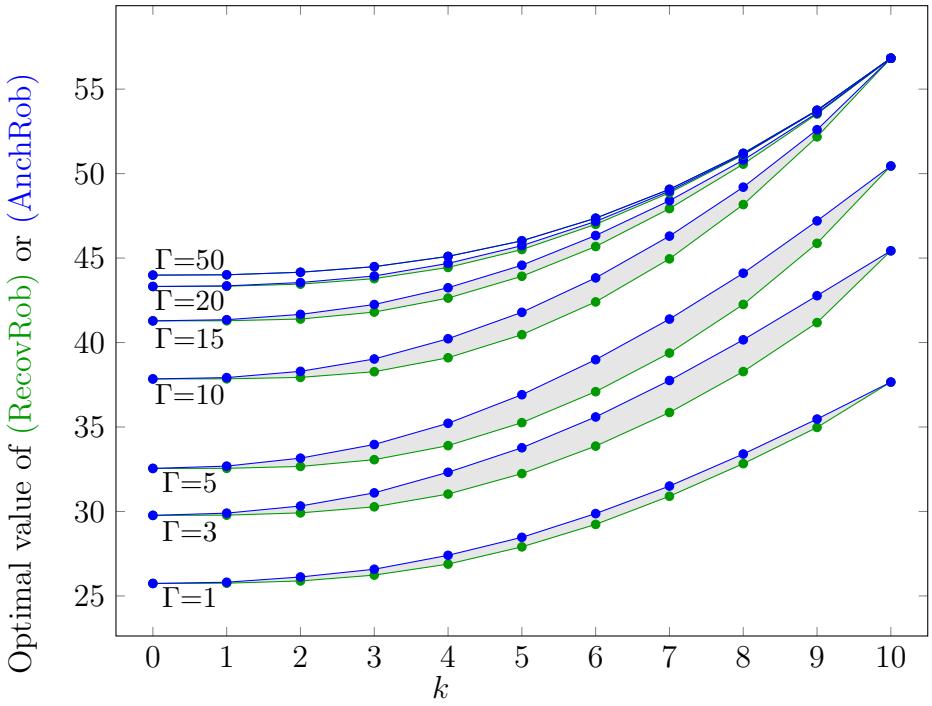


Figure 4.1: Optimal values of (RecovRob) and (AnchRob) depending on k, for budget $\Gamma = 1, 3, 5, 10, 15, 20, 50$.

Each pair of curves (blue and green) illustrates how recoverable and anchor-robustness bridge the gap between adjustable robustness ($k = 0$) and static-robustness ($k = p$). Also optimal values of (RecovRob) and (AnchRob) coincide at these extreme values of k. For the static-robust case ($k = p$), note that all curves coincide whenever $\Gamma \geq p$. Indeed, for any budget $\Gamma \geq p$, the worst-case cost of a solution is when all items costs are disrupted. The gray area corresponds to the difference between (RecovRob) and (AnchRob). It is very limited for small budget ($\Gamma = 1$) and high budget ($\Gamma > 10$). The cases where optimal values of (RecovRob) and (AnchRob) differ the most are for intermediate value of k and Γ.

In Figure 4.2 we represent, for each budget $\Gamma = 1, 3, 5, 10, 15, 20, 50$, the price of anchor-robustness $PoAR(k) = \frac{Cost_{Anch}(k) - Cost_{Recov}(k)}{Cost_{Recov}(k)}$.

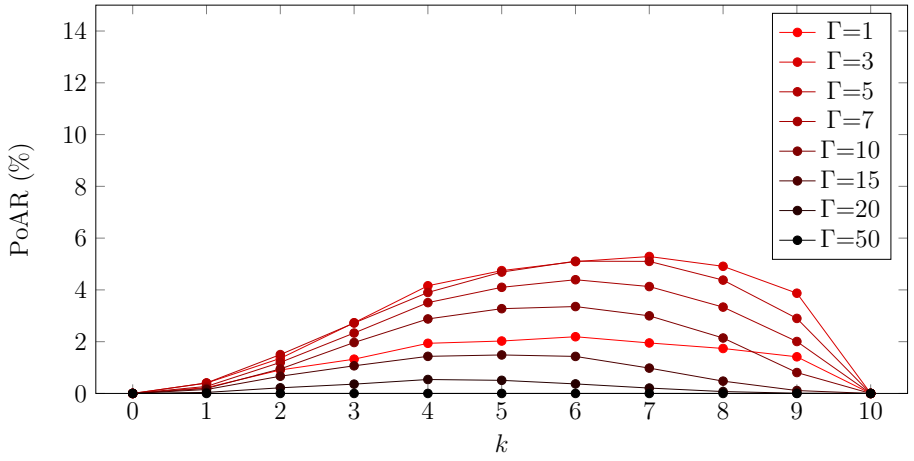


Figure 4.2: Value of $PoAR(k)$ depending on k, for budget $\Gamma = 1, 3, 5, 10, 15, 20, 50$.

On the considered instances, the $PoAR$ value remains small: at most 5%. The largest values of $PoAR$ are obtained for $k \in \{4, \dots, 8\}$ and small budget $\Gamma = 3, 5$. The $PoAR$ for $\Gamma = 1$ is smaller, while for budget $\Gamma \geq 5$, it decreases with Γ. Recall that the theoretical result of Proposition 4.7 is that $PoAR$ is unbounded. By contrast we observe here that in random instances, the price of anchor-robustness remains very small.

Importantly in our random instances, the costs are non-zero. Hence the gap between the costs C_e and c_e and the worst-case deviation $\widehat{\delta}_e$, is bounded. In the instance built in Proposition 4.7, the costs were 0, hence the gap between costs and worst-case deviation was unbounded. This suggests that further results could be obtained to bound the price of anchor-robustness, with more assumptions on $C, c, \widehat{\delta}$.

Another question is whether a low price of anchor-robustness would also be observed empirically on other combinatorial problems, e.g., for the Min-Cost Spanning Tree problem. When investigating this question for spanning trees, it appeared that designing instances (especially, graphs) with a non-zero $PoAR$ was uneasy. Random instances often had no $PoAR$, or even no gap between the optimal costs of static-robust and adjustable-robust problems. These preliminary results are not presented in the manuscript.

Conclusion

In this work we studied the Anchor-Robust approach for combinatorial problems under cost uncertainty. The focus was set on original polynomial problems with a polyhedral characterization. We first studied the complexity of the (AnchRob) problem depending on the uncertainty set and the number of anchored decisions required. The problem was shown NP-hard for discrete budgeted uncertainty, and some cases of polyhedral uncertainty. The hardness results for (AnchRob) are to be contrasted to existing robust approaches that are tractable for these uncertainty sets. MIP techniques for (AnchRob) were then investigated. An MIP reformulation was obtained for (AnchRob), relying on a description of the polytope of the original problem. In parallel, the recoverable robust approach was also examined. It is close to (AnchRob) and similarly provides intermediate solutions between adjustable and static-robustness. Most complexity results are common to both problems. Concerning MIP reformulations, a strong assumption on the problem is needed to obtain a reformulation for (RecovRob). This reveals the different nature of problems (AnchRob) and (RecovRob). We then investigated the price of anchor-robustness. While it is in general unbounded, numerical computations showed that it was small in practice on random instances.

There are multiple research perspectives to the work presented in this chapter. The complexity of (AnchRob) could be studied more specifically for some problems. An open question is the complexity of the (AnchRob) problem for spanning trees, polyhedral budgeted uncertainty and unit anchoring weights. The big picture of the complexity of (AnchRob) on classical polynomial problems is far from being complete. On MIP reformulations, it would then be necessary to assess the numerical performance of the proposed MIP for NP-hard cases of (AnchRob). A research direction is also to extend the MIP tools to new cases, e.g., to design an approach for (AnchRob) when an efficient description of the problem polytope is not known. It can be expected that the proposed reformulation could serve as a basis for the design of approximation or decomposition-based exact approaches. Finally, investigation on the price of anchor-robustness could be further carried out. Better bounds could be searched for, depending on costs and deviations values or properties of the combinatorial problem, and compared to numerical computations.

Part III

Anchored solutions in project scheduling

Preliminaries on project scheduling

In this part, we study project scheduling problems, a family of problems containing PERT Scheduling.

We consider a project with a set J of n jobs. Let s and t two dummy jobs representing the beginning and the end of the project, and $\overline{J} = J \cup \{s, t\}$. A job $j \in J$ has processing time $p_j \geq 0$. A solution of a project scheduling problem is a schedule, i.e., a vector of starting times $x_i \geq 0, i \in \overline{J}$ of the jobs. The starting time of s is usually $x_s = 0$. The makespan of the schedule is x_t.

Various constraints can be imposed on the schedule, leading to distinct project scheduling problems.

Precedence constraints. Let $(\overline{J}, \mathcal{A})$ be a *precedence graph*. It is assumed that the precedence graph is acyclic, and s (resp. t) is a predecessor (resp. a successor) of all jobs. Then precedence constraints are of form

$$x_j - x_i \geq p_i \quad \forall (i,j) \in \mathcal{A}$$

representing that for every precedence arc $(i,j) \in \mathcal{A}$, job j must start after job i is completed. The problem of finding a schedule under precedence constraints, so as to minimize the makespan, is the PERT scheduling problem introduced in Part I.

Generalized precedence constraints. A generalization is to consider constraints of form

$$x_j - x_i \geq a_{ij} \quad \forall (i,j) \in \mathcal{A}$$

where a_{ij} is any real number, possibly negative. Such generalized precedence constraints can be used to cast time-window constraints, or deadlines. For exemple, if job i should start before date d, this can be represented by the generalized precedence constraint $x_s - x_i \geq -d$. Let the precedence graph be $(\overline{J}, \mathcal{A})$ and let $(\overline{J}, \mathcal{A}, a)$ denote its arc-weighted version. The problem of finding a schedule under generalized precedence constraints, so as to minimize the makespan, is the Gen Prec Project Scheduling Problem. A schedule satisfying the constraints of $(\overline{J}, \mathcal{A}, a)$ exists if and only if there exists no positive circuit in $(\overline{J}, \mathcal{A}, a)$. The minimum makespan is the length of the longest $s-t$ path in $(\overline{J}, \mathcal{A}, a)$, which can be found in polynomial time by dynamic programming (Pinedo, 2002).

Resource constraints. Consider a set of renewable resources $\mathcal{R}$. Assume that each job $i \in J$ has a resource requirement r_{ik} of resource $k \in \mathcal{R}$ during its execution. The total amount of resource k available is limited, equal to R_k. Resource constraints associated to (r, R) are that at every date $d \geq 0$, the total quantity of resource required by jobs under execution is less than the resource availability. This writes

$$\sum_{i \in J:\ x_i \leq d < x_i + p_i} r_{ik} \leq R_k \quad \forall d \geq 0$$

Note that schedule x satisfies the resource constraints at all dates if and only if it satisfies them at every date $x_j, j \in J$ where a job starts. The problem of finding a schedule under precedence constraints and resource constraints, so as to minimize the makespan, is the Resource-Constrained Project Scheduling Problem (RCPSP). It has been widely studied in the scheduling literature. It is NP-hard in the strong sense and computationally challenging (Artigues et al., 2008).

A special case is when there is only one resource type ($|\mathcal{R}| = 1$), every job has a resource requirement $r_i = 1$, and the resource availability is an integer m. Then a schedule x satisfies the resource constraints if and only if at every date, there is at most m jobs under execution. This corresponds to executing jobs on m identical machines. In that special case, finding a feasible schedule with minimum makespan is still an NP-hard problem (Garey and Johnson, 1979).

Chapter 5

Anchored Rescheduling problems for project scheduling

In this chapter we study anchor-reoptimization problems for project scheduling, called anchored rescheduling problems, and analyze their computational complexity. The present results have been published in (Bendotti et al., 2020b). Following the general definitions of Chapter 2, we study two anchoring levels. The first one is the number of identical starting times between both schedules. The second one is the number of starting times that differ from less than a tolerance threshold. The anchored rescheduling problem was previously investigated in (Bendotti et al., 2017) for PERT Scheduling with no tolerance, and proved polynomial. We extend the result to GenPrec Project Scheduling, with a tolerance. Generalized precedences allow for a larger variety of constraints such as deadlines or time windows. The introduction of the tolerance feature also leads to a more realistic anchored rescheduling problem. We then consider a machine scheduling case, where the anchored rescheduling problem is shown NP-hard. We define a variant where the sequence of jobs on machines is fixed, resorting to restricted reoptimization as introduced in Section 2.2.4. It is shown that this simpler, yet practically attractive, variant is solvable in polynomial time.

5.1 Anchored rescheduling under generalized precedence

We consider a project scheduling problem Π where a solution is a schedule $x \in \mathbb{R}^{\overline{J}}$. Given a subset $I \subseteq J$, a partial assignment of starting times $(x_i)_{i \in I}$ is called a baseline. Given an instance $\mathcal{I}$ of project scheduling problem Π, a baseline $(x_i)_{i \in I}$ and a solution y of $\mathcal{I}$, the anchoring level is $\sigma(x, y) = |\{i \in I : x_i = y_i\}|$, i.e., the

number of anchored jobs that have the same starting time in the baseline and in the new solution y. This definition was introduced in (Bendotti et al., 2017) and it follows the general definition of Chapter 2 with equivalence relation $=$. In practice the decision maker may consider that a change of the starting time of a job is negligible within some tolerance. Given a tolerance vector $\varepsilon \in \mathbb{R}^J_+$, baseline $(x_i)_{i\in I}$ and solution y, job $i \in I$ is *ε-anchored* if $|x_i - y_i| \leq \varepsilon_i$. The *$\varepsilon$-anchorage level* is $\sigma^\varepsilon(x, y) = |\{i \in I : |x_i - y_i| \leq \varepsilon_i\}|$. It corresponds to the general definition of Chapter 2 with equivalence relation $=_\varepsilon$; note however that we allow the tolerance parameter to depend on the job.

The corresponding Anchored Rescheduling problems are as follows.

AnchRe(Π)	
Input:	instance $\mathcal{I}$ of problem Π, baseline $(x_i)_{i\in I}$
Problem:	find y a schedule of $\mathcal{I}$ such that $\sigma(x, y)$ is maximized.

ε-AnchRe(Π)	
Input:	instance $\mathcal{I}$ of problem Π, baseline $(x_i)_{i\in I}$, tolerance $\varepsilon \in \mathbb{R}^J_+$
Problem:	find y a schedule of $\mathcal{I}$ such that $\sigma^\varepsilon(x, y)$ is maximized.

Note that the baseline $(x_i)_{i\in I}$ may be issued from the solution of a previous instance $\mathcal{I}^0$ of problem Π. However in the sequel no specific assumption is made on the properties of baseline x.

Let us now give additional background on the GenPrec Project Scheduling problem, denoted by (GenPrec). The set of jobs is $J = \{1, \ldots, n\}$, and jobs s and t are indexed 0 and $n + 1$ respectively, so that $\overline{J} = \{0, \ldots, n + 1\}$. Consider a directed graph $G = (\{0, \ldots, n + 1\}, \mathcal{A})$. Let $G(a)$ be the weighted digraph obtained by adding arc weights $a \in \mathbb{R}^{\mathcal{A}}$ to the digraph G. Let us denote by (i, j, a_{ij}) a weighted arc of $G(a)$. The weighted digraph $G(a)$ defines an instance of the (GenPrec) scheduling problem if it satisfies the following assumptions:
(i) there is no circuit of positive length in $G(a)$
(ii) for every job $i \in \{1, \ldots, n\}$ there exists at least a path of non-negative length from 0 to i and from i to $n + 1$ in $G(a)$.

The (GenPrec) problem is to find a schedule x of jobs $\{0, \ldots, n + 1\}$ so that

$$x_j - x_i \geq a_{ij} \text{ for every arc } (i, j, a_{ij}) \text{ of } G(a).$$

Note that w.l.o.g. we set $x_0 = 0$ in every schedule. Assumption (i) ensures the existence of a feasible schedule for the instance $G(a)$ (see e.g. (Pinedo, 2002)).

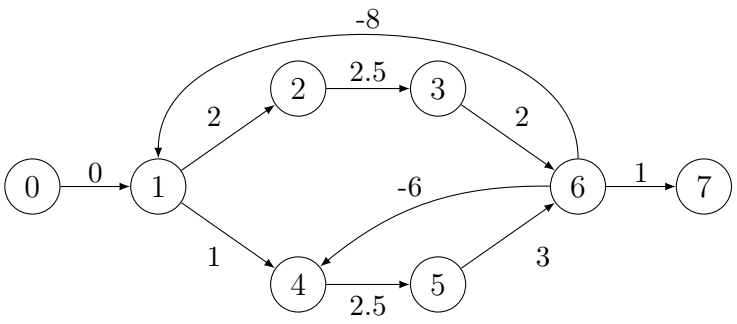


Figure 5.1: Generalized precedence graph $G(a)$ with 6 jobs.

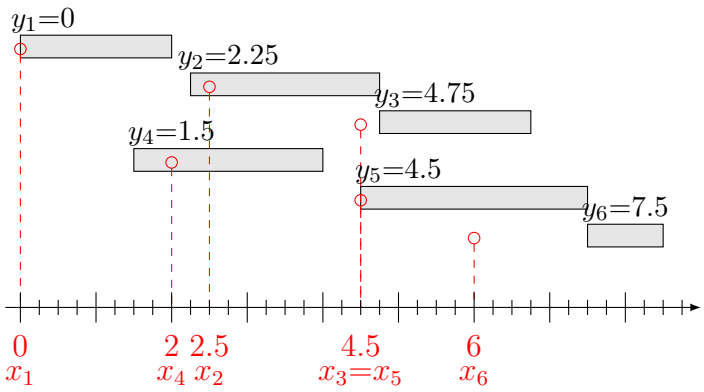


Figure 5.2: Schedule $y = (0, 2.25, 4.75, 1.5, 4.5, 7.5)$ feasible for $G(a)$ with 6 jobs.

From assumption (ii), it comes that in every feasible schedule and for every $i \in \{1, \ldots, n\}$, the inequality $0 \leq x_i \leq x_{n+1}$ holds: job 0 and job $n+1$ then represent the beginning and the end of the schedule respectively.

Various constraints can be modeled within this framework. A classical precedence constraint $x_j - x_i \geq p_i$ can be represented by an arc (i, j, p_i), where p_i is the processing time of job i. The special case of acyclic precedence graph $G(p)$ is denoted by (Prec). For illustrative purpose, an example of a generalized precedence graph $G(a)$ with $n = 6$ jobs is represented in Figure 5.1. It features circuits, and arcs with negative weights, e.g., the arc $(6, 4, -6)$ corresponds to constraint $y_6 - y_4 \leq 6$.

A feasible schedule y is represented in Figure 5.2. Job i is represented as a rectangle of length $\max_{(i,j)\in\mathcal{A}} a_{ij}$, note e.g. that $a_{12} \neq a_{14}$. Baseline $x = (0, 2.5, 4.5, 2, 4.5, 6)$ is also represented. For baseline x and no tolerance, only jobs $\{1, 5\}$ are anchored. For the same baseline and with tolerance $\varepsilon = (0, 0.25, 0.25, 0, 0, 0)$, jobs $\{1, 2, 3, 5\}$ are ε-anchored.

The rest of the chapter is structured as follows. In Section 5.2 the problem ε-

AnchRe(GenPrec) is proven to be solvable in polynomial time. In Section 5.3 we correct a flawed complexity result from (Bendotti et al., 2017) in the case of time-window constraints. In Section 5.5 we use our framework in a machine scheduling variant. In Section 5.4 we analyze the impact of tolerance ε on the optimum of the anchored rescheduling problem.

5.2 Polynomiality of ε-AnchRe(GenPrec)

In this section we prove

Theorem 5.1. *ε-AnchRe(GenPrec) is solvable in polynomial time.*

Let $((x_i)_{i\in I}, G(a), \varepsilon)$ be an instance of ε-AnchRe(GenPrec). By testing assumptions (i) and (ii), it can be checked in polynomial time that the weighted digraph $G(a)$ is a valid instance of the (GenPrec) problem.

As done in Chapter 2, we can now define extendable sets, that are subsets of decisions that can be maintained in the new solution. Let $H \subseteq I$ be a subset of jobs. Formally, the set H is *x-extendable* if there exists a schedule x of $G(a)$ such that all jobs in H are ε-anchored with respect to x. Solving ε-AnchRe(GenPrec) is exactly finding a set H x-extendable of maximum size and an associated solution y.

Consider an *auxiliary graph* G^{aux} defined by copying the graph $G(a)$, then adding for every job $i \in H$ two new arcs $(0, i, x_i - \varepsilon_i)$ and $(i, 0, -(x_i + \varepsilon_i))$. The auxiliary graph defines an instance of the (GenPrec) problem, whose constraints are the constraints from $G(a)$, and the new arcs constraints $y_i \geq x_i - \varepsilon_i$ and $-y_i \geq -(x_i + \varepsilon_i)$, that is exactly $|y_i - x_i| \leq \varepsilon_i$. Hence there is a one-to-one correspondence between the schedules of the auxiliary graph G^{aux}, and the schedules of the instance $G(a)$ in which all jobs in H are ε-anchored with respect to the baseline x. Thus we obtain

Proposition 5.1. *The set H is x-extendable if and only if the auxiliary graph G^{aux} has no positive circuit.*

Let us now show that the absence of positive circuit in G^{aux} is also equivalent to $H \cup \{0\}$ being an antichain in an appropriate poset. Let us denote by $\ell(P)$ the length of a directed path P in $G(a)$. For every pair of distinct jobs $i, j \in \{0, 1, \ldots, n\}$, let $L_{G(a)}(i, j)$ be the maximum length of a directed path from i to j in $G(a)$. By convention it is equal to $-\infty$ if there is no such path. A relation $\mathscr{R}$ on the set of jobs $\{0, 1, \ldots, n\}$ is defined by:

$$i \mathscr{R} j \text{ if and only if } i = j \text{ or } x_i - \varepsilon_i + L_{G(a)}(i, j) > x_j + \varepsilon_j$$

where we define $\varepsilon_0 = 0$ for the simplicity of notation. In particular, if $i \neq j$ and $i\mathscr{R}j$, it implies that value $L_{G(a)}(i,j)$ is finite and there exists a path from i to j in $G(a)$.

Lemma 5.1. *The relation $\mathscr{R}$ is a partial order on the set of jobs $\{0,1,\ldots,n\}$.*

Proof. Relation $\mathscr{R}$ is clearly reflexive. Relation $\mathscr{R}$ is antisymmetric: if $i\mathscr{R}j$ and $j\mathscr{R}i$ with $i \neq j$, then there exists a longest path P_{ij} from i to j and a longest path P_{ji} from j to i in $G(a)$. The circuit obtained by closing P_{ij} with P_{ji} has length $L_{G(a)}(i,j) + L_{G(a)}(j,i)$, which is non-positive by assumption (i) on $G(a)$. Furthermore $L_{G(a)}(i,j) + L_{G(a)}(j,i) > x_j + \varepsilon_j - (x_i - \varepsilon_i) + x_i + \varepsilon_i - (x_j - \varepsilon_j) = 2(\varepsilon_i + \varepsilon_j) \geq 0$, a contradiction. Finally relation $\mathscr{R}$ is transitive: consider three pairwise distinct jobs i,j,k such that $i\mathscr{R}j$ and $j\mathscr{R}k$. There exists two paths P_{ij} and P_{jk} in $G(a)$, hence their concatenation (P_{ij}, P_{jk}) forms a path from i to k. It comes $x_i - \varepsilon_i + L_{G(a)}(i,k) \geq x_i - \varepsilon_i + L_{G(a)}(i,j) + L_{G(a)}(j,k) > x_j + \varepsilon_j + L_{G(a)}(j,k) > x_k + \varepsilon_k + 2\varepsilon_j \geq x_k + \varepsilon_k$. Hence $i\mathscr{R}k$. □

Proposition 5.2. *The auxiliary graph G^{aux} has no positive circuit if and only if $H \cup \{0\}$ is an antichain of the poset $(\{0,1,\ldots,n\}, \mathscr{R})$.*

Proof. Assume that $H \cup \{0\}$ is not an antichain. Then there exists two distinct jobs i and j in $H \cup \{0\}$ such that $i\mathscr{R}j$. Hence there exists a longest path P_{ij} from i to j in $G(a)$. Consider the circuit C obtained by closing the path P_{ij} with the new arc $(0, i, x_i - \varepsilon_i)$ if $i \neq 0$ and with the new arc $(j, 0, -(x_j + \varepsilon_j))$ if $j \neq 0$. Then the length of C is $L_{G(a)}(i,j) + x_i - \varepsilon_i - (x_j + \varepsilon_j)$. Note that it is valid even if $i = 0$ or $j = 0$. This length is positive from the definition of $\mathscr{R}$, hence the auxiliary graph contains a positive circuit.

Conversely, assume that there exists a positive circuit in the auxiliary graph. Then there exists a positive circuit C that contains every vertex at most once. Since $G(a)$ contains no positive circuit, the circuit C contains at least one new arc, and consequently it contains vertex 0 exactly once. It follows that C contains one or two successive new arcs. Let P be the path obtained by removing the new arcs from the circuit C. Let i and j be the first and last vertex of P respectively. The length of C can be written $x_i - \varepsilon_i + \ell(P) - (x_j + \varepsilon_j)$ (again it is valid even if $i = 0$ or $j = 0$). By assumption this length is positive, then with $\ell(P) \leq L_{G(a)}(i,j)$ it comes $x_i - \varepsilon_i + L_{G(a)}(i,j) > x_j + \varepsilon_j$ and $i\mathscr{R}j$. □

Remark that all results presented here can be extended to the case of asymmetric tolerance intervals, that is, the case where the starting time of job i is considered as unchanged if $y_i \in [x_i - \varepsilon_i^-, x_i + \varepsilon_i^+]$ with two distinct parameters $\varepsilon_i^-, \varepsilon_i^+ \geq 0$.

We now deduce the proof of the polynomiality result.

Proof of Theorem 5.1. With Proposition 5.1 and Proposition 5.2, a set H is x-extendable if and only if $H \cup \{0\}$ is an antichain of the poset $(\{0, 1, \ldots, n\}, \mathscr{R})$. Hence solving the reactive problem is tantamount to finding a maximum size antichain $H \cup \{0\}$ of the poset. The latter problem can be solved in polynomial time (Dilworth, 1950). Note also that given a set H^* of maximum size, a corresponding reactive solution y^* can be found in polynomial time by computing any schedule of G_{H^*}. Hence the problem ε-AnchRe(GenPrec) is polynomial-time solvable. □

Algorithmically, a max-size antichain can be found through a combinatorial algorithm such as Dilworth's algorithm (Dilworth, 1950), or through linear programming. A non-compact characterization of the associated polytope is known, together with a polynomial separation algorithm (Schrijver, 2003). Furthermore anchoring weights may be introduced so that the objective is to maximize the total weight of ε-anchored jobs. Then the linear programming approach allows to search for a max-weight antichain for any weight function, while Dilworth's algorithm requires integer weights.

Beyond the complexity result, the combination of Proposition 5.1 and Proposition 5.2 constitutes a combinatorial characterization of x-extendable sets. This characterization will be of prominent importance in the robust approach developped in Chapter 6. Thus we write it explicitly, in the case $\varepsilon = 0$, in the following corollary.

Corollary 5.1 (Characterization of extendable sets)**.** *The set H is x-extendable if and only if*

$$x_j - x_i \geq L_{G(a)}(i, j) \qquad \forall i, j \in H \cup \{0\}$$

5.3 Anchored rescheduling with a deadline constraint

In Chapter 2 we defined two classes of anchor-reoptimization problems, depending on whether the cost of the new solution is taken into account. For scheduling problems, this amounts to setting an upper bound on the makespan of schedule y, i.e., to impose a deadline constraint $y_{n+1} \leq B$. The AnchReopt–Cost problem introduced in Chapter 2 can then be declined as the following anchored rescheduling problem

ε-AnchRe(Π)-Deadline	
Input:	instance $\mathcal{I}$ of problem Π, baseline $(x_i)_{i\in I}$, tolerance $\varepsilon \in \mathbb{R}_+^J$, deadline B
Problem:	find y a schedule of $\mathcal{I}$ such that $y_{n+1} \leq B$ and $\sigma^\varepsilon(x, y)$ is maximized.

Using generalized precedence constraints, time window constraints of form $y_i \in [l_i, u_i]$ can be modelled with two arcs $(0, i, l_i)$ and $(i, 0, -u_i)$ (recall that $y_0 = 0$ w.l.o.g.). In particular, the deadline constraint $y_{n+1} \leq B$ can be represented by the generalized precedence constraint $(n + 1, 0, -B)$. Consequently

Corollary 5.2. *ε-AnchRe(Π)-Deadline is solvable in polynomial time.*

We now correct a flawed complexity result from (Bendotti et al., 2017). In this previous paper, the so-called Anchor-Reactive CPM-Scheduling Problem with Time Windows (ARSPTW) was defined as a variant of AnchRe(Prec) where the baseline is a complete schedule ($I = J$), and a deadline constraint is imposed on the new schedule: $y_{n+1} \leq B$. Deadline B is part of the instance of the ARSPTW. It follows that the ARSPTW is a special case of ε-AnchRe(Π)-Deadline.

Corollary 5.3. *The ARSPTW is polynomial-time solvable.*

The proof of Theorem 4.3 from (Bendotti et al., 2017) incorrectly stated the NP-hardness of the ARSPTW. It relied on a reduction from the so-called Maximum Complete Bipartite Subgraph problem (MCBS): given a bipartite graph $G = (L \cup R, E)$ with n non isolated nodes and an integer k, is there a complete bipartite subgraph of G with at least k nodes? The correct reference to Garey and Johnson (1979) requires that the complete bipartite subgraph is *balanced*, i.e., it has the same number of nodes in L and R. Without this condition, the MCBS problem is polynomial-time solvable.

5.4 Sensitivity analysis of ε-AnchRe(GenPrec) with respect to tolerance

Regarding tolerance ε as a new input of the rescheduling problem, a natural question is the sensitivity of the optimal value of ε-AnchRe(GenPrec) to ε. In this section, we study the behavior of the rescheduling optimal value in the case where tolerance is given by a single parameter $\epsilon \geq 0$, that is, $\varepsilon_i = \epsilon$ for every $i \in \{1, \ldots, n\}$. Similar results hold if every ε_i is any affine function of ϵ. Given a baseline x and an instance $G(a)$, let

$$OPT(\epsilon) = \max_{\substack{y \text{ schedule} \\ \text{of } G(a)}} \sigma_\epsilon(x, y)$$

For every pair of distinct jobs $i, j \in \{0, \ldots, n\}$, define $b_{ij} = \frac{1}{2}(L_{G(a)}(i,j) - (x_j - x_i))$ if $i \neq 0$ and $j \neq 0$, and $b_{ij} = L_{G(a)}(i,j) - (x_j - x_i)$ if $i = 0$ or $j = 0$. Let $B = \{b_{ij}, i, j \in \{0, \ldots, n\}, i \neq j\}$.

Proposition 5.3. *The function $OPT(\cdot)$ is piecewise constant and non-decreasing on $\mathbb{R}_+$. Moreover every breakpoint ϵ^* of function OPT belongs to set B.*

Proof. The function $OPT(\cdot)$ is integer-valued. Moreover the function $\epsilon \mapsto \sigma_\epsilon(x, y)$ is non-decreasing hence $OPT(\cdot)$ is also non-decreasing. Let $\epsilon \geq 0$. Define $b^\epsilon = \max\{b \in B,\ b \leq \epsilon\}$. The claim is that $OPT(\epsilon) = OPT(b^\epsilon)$. Consider the poset $(\{0, \ldots, n\}, \mathscr{R})$ introduced in Section 5.2. The relation $\mathscr{R}$ depends on the tolerance and it can be rewritten as follows: for two distinct jobs $i, j \in \{0, \ldots, n\}$, $i \mathscr{R} j$ if and only if $b_{ij} > \epsilon$. Moreover, from the definition of b^ϵ, for every pair i, j, the inequality $b_{ij} > \epsilon$ is equivalent to $b_{ij} > b^\epsilon$. Hence, the relation $\mathscr{R}$ is the same for tolerance ϵ and for tolerance b^ϵ. From Proposition 5.2, for every ϵ, the value $OPT(\epsilon)$ is the maximum size of an antichain containing job 0 in the poset $(\{0, \ldots, n\}, \mathscr{R})$. Since the poset remains the same for tolerance ϵ and tolerance b^ϵ, it comes $OPT(\epsilon) = OPT(b^\epsilon)$. From the claim, it follows that OPT is constant on every interval of form $[b, b'[$ where b and b' are two successive points of B. Hence any breakpoint of the function must be a point in B. □

Consider now the problem of minimizing the tolerance while ensuring that at least k jobs are ε-anchored:

MinTolerance	
Input:	precedence graph $G(a)$, baseline x, integer k
Problem:	find ϵ such that $OPT(\epsilon) \geq k$ and ϵ is minimized.

From Proposition 5.3 the optimum of MinTolerance can be found in set B. Values in B can be computed in polynomial time, and $|B| \leq n^2 + n$, which implies:

Corollary 5.4. MinTolerance *can be solved in polynomial time.*

Consider the instance with 6 jobs from Figure 5.1. Figure 5.3 shows its associated rescheduling optimal value OPT for the baseline $x = (0, 2.5, 4.5, 2, 4.5, 6)$. The computation of the set B for this instance leads to four possible breakpoint values ($B = \{0, 0.25, 0.5, 0.75\}$), but only three of them are actual breakpoints of OPT. Namely, changing the tolerance within the range $[0.25; 0.75[$ has no impact on the number of ε-anchored jobs.

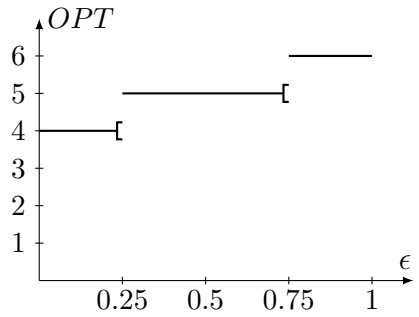


Figure 5.3: Anchored rescheduling optimal value $OPT(\epsilon)$ for instance from Figure 5.1 and baseline $x = (0, 2.5, 4.5, 2, 4.5, 6)$.

5.5 Towards machine rescheduling

A question is to benefit from the polynomiality result of Theorem 5.1 in a machine environment. Consider a set of m machines, and a set of jobs J to be scheduled under precedence constraints represented by precedence graph $G(p)$. The problem is denoted by $(m|\text{Prec})$. A solution of $(m|\text{Prec})$ is then formed with a vector of starting times $x \in \mathbb{R}^J_+$, and an assignment of jobs on machines.

5.5.1 NP-hardness on one machine

The anchored rescheduling problem can be considered as previously, that is,

ε-AnchRe$(m\|\text{Prec})$	
Input:	integer m, precedence graph $G(p)$, baseline $(x_i)_{i\in I}$, tolerance $\varepsilon \in \mathbb{R}^J_+$
Problem:	find y a schedule of $G(p)$ on m machines such that $\sigma^\varepsilon(x, y)$

We first note that

Proposition 5.4 (Chrétienne (2018))**.** AnchRe*$(m|Prec)$ is NP-hard, even on a single machine ($m = 1$).*

Indeed, it was proven in (Chrétienne, 2018) that the AnchRe$(m|\text{Prec})$ problem is NP-complete for a single machine, even when there is no precedence constraints, by a reduction from 3-Partition.

By contrast with the project scheduling problem studied in Section 5.2, for a single machine it is even difficult to decide whether a set H is x-extendable. Note

that Corollary 5.1 gives a polynomial algorithm to decide if a set is extendable in project scheduling, by computing longest paths in the precedence graph. On a single machine, we prove

Proposition 5.5. *On a single machine, deciding if a set H is x-extendable is NP-hard.*

Proof. The proof is a reduction from 2-PARTITION. Consider the following instance of the scheduling problem. There are $n+2$ jobs: n normal jobs with processing time p_j, one middle job with processing time 1, and one final job with processing time 1. The precedence constraints are that all jobs must be executed before the final job. The baseline is such that $x_{mid} = \frac{1}{2}(\sum_{j=1}^{n} p_j)$ and $x_{final} = 1 + \sum_{j=1}^{n} p_j$. The set H is formed with the middle job and the final job. The instance is illustrated in Figure 5.4.

Then the following equivalence holds. There exists a schedule y of jobs on the machine such that the middle and the final jobs are scheduled at their baseline starting times if and only if there exists a 2-partition of the p_j's into two equally valued subsets. The claimed NP-hardness result follows. □

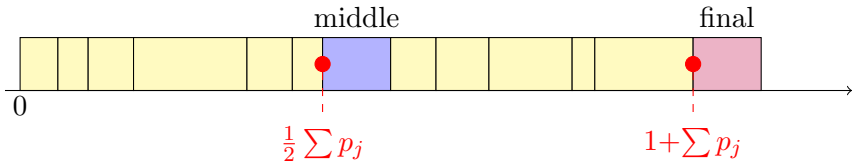


Figure 5.4: Schedule on a single machine from the reduction, with middle job (resp. final job) in blue (resp. red) and normal jobs in yellow.

5.5.2 Rescheduling with fixed sequences

Consider now the following variant. Given a baseline solution, let $\mathscr{S} = (S_1, \ldots, S_m)$ be a collection of sequences, where sequence S_k is the ordered list of jobs processed on machine k in the baseline. We consider the anchored rescheduling problem where it is required that the new solution y is consistent with the sequences from $\mathscr{S}$, i.e.,

ε-AnchRe(m|Prec)-fixedSeq	
Input:	integer m, precedence graph $G(p)$, baseline $(x_i)_{i\in I}$, sequences $\mathscr{S}$, tolerance $\varepsilon \in \mathbb{R}_+^J$
Problem:	find y a schedule of $G(p)$ on m machines with sequences $\mathscr{S}$ such that $\sigma^\varepsilon(x, y)$ is maximized

Theorem 5.2. ε-AnchRe(m|Prec)-fixedSeq *is solvable in polynomial time.*

Proof. Let $G^{\mathscr{S}}(p)$ denote the precedence graph formed with $G(p)$ and additional arcs (i, j) for every pair i, j of successive jobs in a sequence of $\mathscr{S}$. Then y is a schedule of $G(p)$ on m machines with sequence $\mathscr{S}$ if and only if it is a schedule of the instance $G^{\mathscr{S}}(p)$ of (Prec). Hence solving ε-AnchRe(m|Prec)-fixedSeq is equivalent to solving ε-AnchRe(Prec) for the precedence graph $G^{\mathscr{S}}(p)$. From Theorem 5.1, it comes that ε-AnchRe(m|Prec)-fixedSeq can be solved in polynomial time. □

This fits within the framework of restricted reoptimization, presented in Section 2.2.4 of Chapter 2. In rescheduling with fixed sequences, rescheduling does not impair neither the assignment of jobs on machines nor the sequences of jobs on machines. The schedule y being much more constrained than in ε-AnchRe(m|Prec), less jobs can be ε-anchored. However the sequences of jobs on machines can be regarded as decisions that are maintained during rescheduling. Rescheduling with fixed sequences thus serves a similar purpose as anchored jobs, by the stabilization of decisions. A practical justification for rescheduling with fixed sequences is that the order of jobs on machines may be difficult or costly to revise if the instance changes. The sequence of jobs on a machine may require preparation, while it is easier to adjust only starting times of jobs.

Note that in the NP-hardness result of Proposition 5.5, the reduction is based on sequencing decisions: should a job be scheduled before or after the middle job?

Rescheduling while maintaining the structure of the schedule w.r.t. resources (e.g., the sequence of jobs on machines) was considered in the literature in the context of robust approaches for the resource-constrained project scheduling problem (Bruni et al., 2017). However the authors of (Bruni et al., 2017) did not consider any criterion to maintain starting times.

Conclusion

We studied a rescheduling problem with the objective of maximizing the number of jobs whose starting times correspond to the baseline, within tolerance ε. The problem was shown to be polynomial, even under generalized precedence constraints, including deadlines or time windows constraints. It was also shown that this polynomiality result can be used in the case of machine anchored rescheduling with fixed sequences of jobs on machines. Fixing sequence decisions seem to be the keypoint for designing polynomial rescheduling variants. In Chapter 7 the RCPSP problem is dealt with. The idea of restricted reoptimization, where sequencing decisions are fixed, is further developed.

If the anchored rescheduling problem of an NP-hard problem is bound to remain NP-hard, it is interesting to stress that a polynomial variant can be of practical interest. It could be integrated into a robust problem, in order to find a good baseline schedule that could be marginally modified if disruptions occur. A perspective is thus to identify such polynomial variants of rescheduling problems, under other resource constraints.

Chapter 6

The Anchor-Robust Project Scheduling Problem

We consider robust optimization for Project Scheduling under uncertain processing times. In project management, the decision maker often needs to compute a baseline schedule in advance. If processing times are later disrupted, the baseline schedule may become infeasible. Then rescheduling is necessary to recover a feasible solution. The need for stability in rescheduling was pointed out in the literature review of Chapter 1 and anchor-rescheduling problems were investigated in Chapter 5. In this chapter we investigate the anchor-robust approach to limit adverse effects of rescheduling by guaranteeing the starting times of some jobs, against realizations of processing times in a given uncertainty set.

For a given uncertainty set, standard robust approaches either produce a schedule with a very large makespan, or offer no guarantee on starting times of the jobs. The robust-static approach produces a schedule with very large makespan, that would be inapplicable in practice. On the opposite, an adjustable-robust approach is to find the minimum worst-case makespan, at the expense of deciding the schedule after the uncertainty realization. Thus there is no baseline solution, and no guarantee on starting times.

In this work, corresponding to (Bendotti et al., 2019), we propose the anchor-robust approach as a middle ground between guaranteeing starting times and guaranteeing the makespan. The output is twofold: a baseline schedule with bounded makespan, and a subset of *anchored jobs*, that corresponds to guaranteed starting times in the baseline schedule. Starting times of non-anchored jobs may need to be adjusted according to the uncertainty realization to recover a feasible schedule. Every job is associated with an anchoring weight, representing how beneficial it is to have a guarantee on the starting time of the job. The Anchor-Robust Project Scheduling Problem (AnchRobPSP) is to find a baseline schedule

with bounded makespan along with a maximum weight subset of anchored jobs.

The aim of this chapter is to provide an in-depth study of AnchRobPSP, to propose algorithms based on its combinatorial properties. The main contributions are as follows.

AnchRobPSP is first formulated using graph models. Two graph models are provided: the first one is applicable to general uncertainty sets, while the second one is dedicated to budgeted uncertainty sets. These graph models are shown to be convenient tools to both exhibit hard cases and devise algorithms. In particular, the existence of a compact MIP formulation for AnchRobPSP under budgeted uncertainty is obtained.

Complexity results and algorithms are given for AnchRobPSP under the following uncertainty sets. For box uncertainty sets, we show that AnchRobPSP is solvable in polynomial time. For upper-bounded uncertainty sets, we show that AnchRobPSP is co-NP-hard. For budgeted uncertainty sets, we first show that the decision version of AnchRobPSP is NP-complete, even in restricted cases with fixed budget. (Pseudo-)polynomial algorithms are then given for special cases, depending on the uncertainty set and the precedence graph.

A computational evaluation highlights the characteristics of optimal solutions of AnchRobPSP and the impact of parameters. The proposed approaches to solve AnchRobPSP are shown to be competitive with classical affine decision rules from the literature.

6.1 The Anchor-Robust Project Scheduling Problem

In this section we first define formally the AnchRobPSP problem, the considered uncertainty sets, and point out connections with classical robust approaches.

6.1.1 Problem definition

Let us recall some notation for the original Project Scheduling problem. A set of jobs $J = \{1, \dots, n\}$ must be scheduled while respecting precedence constraints, represented by a directed acyclic graph $G = (\overline{J}, \mathcal{A})$. The vertex-set of G is $\overline{J} = J \cup \{s, t\}$ where s (resp. t) is a predecessor (resp. successor) of all jobs, representing the beginning (resp. the end) of the schedule. Each job $i \in J$ has a processing time $p_i \in \mathbb{R}_+$, and $p_s = 0$. Given a vector $p \in \mathbb{R}_+^J$, let $G(p)$ be the weighted digraph obtained from G by weighting every arc (i, j) with p_i. A schedule of $G(p)$

is a vector of starting times $x \in \mathbb{R}_+^{\overline{J}}$ such that $x_j - x_i \geq p_i$ for every arc (i,j) of $G(p)$, $x_s = 0$. Then x_t is the makespan of the schedule.

We consider a 2-stage decision process to deal with uncertain processing times of jobs. At first stage, the decision maker has a baseline instance $G(p)$ to solve, and a deadline $M \geq 0$. He must choose a *baseline schedule*, i.e., a schedule of $G(p)$ with makespan at most M. At second stage, the decision maker will know the real instance $G(p+\delta)$ where $\delta \in \mathbb{R}_+^J$ is the vector of *disruptions* on processing times. Then a second-stage schedule x^δ of $G(p+\delta)$ will be determined. We consider that the decision maker wants to hedge against a family $\{G(p+\delta)\}_{\delta \in \Delta}$ of instances, where the set Δ is the uncertainty set. In order to have guarantees on starting times, we are interested in jobs for which the starting time in the baseline schedule x^0 will be the same as the starting time in any second-stage schedule x^δ. Formally

Definition 6.1. *Given a baseline schedule x^0, a subset H of jobs is* anchored *with respect to x^0 if for every $\delta \in \Delta$, there exists a schedule x^δ of $G(p+\delta)$ such that $x_i^0 = x_i^\delta$ for every $i \in H$.*

Note that it corresponds to Definition 2.1 from Chapter 2 in the case of continuous variables with equivalence relation $=$.

Each job $i \in J$ is associated with an anchoring weight $a_i \in \mathbb{R}_+$. Given a baseline instance $G(p)$, an uncertainty set Δ, anchoring weights $a \in \mathbb{R}_+^J$, and a deadline M, the Anchor-Robust Project Scheduling Problem (AnchRobPSP) is to find a baseline schedule x^0 and a subset $H \subseteq J$ anchored with respect to x^0, so that the total weight $a(H) = \sum_{i \in H} a_i$ is maximized.

(AnchRobPSP) is a robust 2-stage problem that can be written as the following program:

$$
\begin{array}{llll}
\text{(AnchRobPSP):} & \max & \min & \max\ a(H) \\
& x^0 \text{ schedule of } G(p) & \delta \in \Delta & x^\delta \text{ schedule of } G(p+\delta) \\
& x_t^0 \leq M & & x_j^\delta = x_j^0\ \forall j \in H \\
& H \subseteq J & &
\end{array}
$$

Uncertainty appears in the right-hand side of precedence constraints. Also, since H is decided in first stage, the recourse problem is a linear program with constant objective function, i.e., it is a feasibility problem. Set H is an anchored set if and only if the inner min/max has value $a(H)$; otherwise it has value $-\infty$. It corresponds to the Anchored set maximization variant of the Anchor-Robust problem, proposed in generic form in Chapter 2.

A previous reference is the proactive problem denoted by DAPSP in (Bendotti et al., 2017). It is also a robust 2-stage problem that can be written similarly as a max/min/max program, but with the set H being a recourse variable. In other words the set H would appear in the inner maximization problem for DAPSP, rather than in the outer maximization problem for AnchRobPSP. The complexity of DAPSP was studied in (Bendotti et al., 2017) for the special case of box uncertainty. From a practical point of view, AnchRobPSP offers a stronger guarantee on starting times, since the anchored jobs are known in first stage. We refer to Section 2.3.4 for a generic comparison between the two problems.

6.1.2 Uncertainty sets

Let us now give definitions of the considered uncertainty sets. Let $\Delta \subseteq \mathbb{R}^J_+$. The set Δ is referred to as a *general uncertainty set* when no further condition is imposed. The set Δ is an *upper-bounded uncertainty set* if there exists $\delta^+ \in \mathbb{R}^J_+$ such that for every $\delta \in \Delta$, $\delta^+_i \geq \delta_i$ for every $i \in J$, i.e., the duration of job i in a second-stage instance is at most $p_i + \delta^+_i$. Note that δ^+ is not necessarily an element of Δ because it is unlikely that all jobs have maximum duration at the same time. The set Δ is a *box uncertainty set* if $\Delta = \Pi_{i\in J}[\delta^-_i, \delta^+_i]$ for some $\delta^-, \delta^+ \in \mathbb{R}^J_+$. The set Δ is a *budgeted uncertainty set* associated with vector $\widehat{\delta} \in \mathbb{R}^J_+$ and integer $\Gamma \in \{1, \dots, n\}$ if $\Delta = \left\{(\widehat{\delta}_i u_i)_{i\in J} \colon\ u_i \in \{0,1\}^J,\ \sum_{i\in J} u_i \leq \Gamma\right\}$. Budget Γ corresponds to the maximum number of disruptions that can occur at the same time (Bertsimas and Sim, 2004).

Note that the uncertainty set Δ can be convexified without changing feasible solutions. This result is well-known for robust 2-stage linear programs, see e.g., Theorem 2.2 in (Ben-Tal et al., 2004). It can be checked that it applies similarly to AnchRobPSP. It stems from the fact that combinatorial decisions (for deciding set H) are taken in first stage. Consequently, it is equivalent to consider the budgeted uncertainty set defined by Γ and $\widehat{\delta}$, or its convex hull $\left\{(\widehat{\delta}_i u_i)_{i\in J} \colon\ u_i \in [0,1]^J,\ \sum_{i\in J} u_i \leq \Gamma\right\}$. In this latter case, more than Γ jobs may have a non-zero disruption δ_i.

Let us illustrate AnchRobPSP on a simple example for these uncertainty sets. Consider a project scheduling instance with 5 jobs. The precedence graph is represented in Figure 6.1. Nominal processing times are $p = (1, 1, 1, 1, 2)$. Each job is also associated with a worst-case deviation $\widehat{\delta} = (0.5, 1, 0.5, 0.5, 0.5)$. In Figure 6.1, each arc (i, j) is weighted with value $p_i + \widehat{\delta}_i$. Let the deadline be $M = 4.5$.

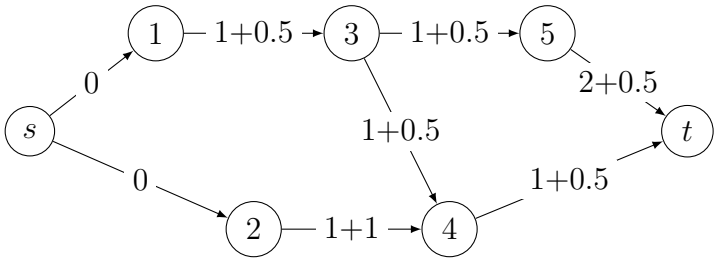


Figure 6.1: Precedence graph for an instance with 5 jobs, with arc-weights $p + \widehat{\delta}$.

Two different uncertainty sets will be considered, to give a flavor of their respective impact on solutions of AnchRobPSP.

A first uncertainty set is the box $\Delta = \Pi_{i \in J}[0, \widehat{\delta}_i]$. Then every job i may have any duration in range $[p_i,\ p_i{+}\widehat{\delta}_i]$. In general a schedule for processing times $p{+}\widehat{\delta}$ has a makespan larger than M, hence such a schedule is infeasible for AnchRobPSP. Consider the schedule $x = (0, 1, 1, 3, 2.5)$ and set $H = \{1, 2, 4\}$. Figure 6.2 shows schedule x. Each job is represented by a rectangle of length p_i, and deviation $\widehat{\delta}_i$ is represented with dotted rectangle. Jobs from H are represented in dark gray. Then (x, H) is a solution of AnchRobPSP for box Δ. First x has makespan $M = 4.5$. Also the set H is anchored w.r.t. x, since for any value of jobs durations, it is possible to repair the schedule by moving jobs 3 and 5 only (in this case, just by right-shifting).

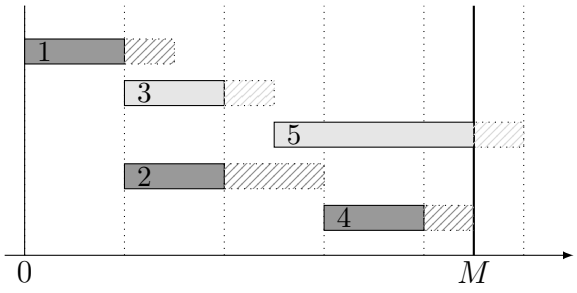


Figure 6.2: Schedule $x = (0, 1, 1, 3, 2.5)$ and set $H = \{1, 2, 4\}$ in dark gray, anchored for box uncertainty set $\Delta = \Pi_{i \in J}[0, \widehat{\delta}_i]$.

A second uncertainty set is $\Delta = \{(\widehat{\delta}_i u_i)_{i \in J} \colon\ u \in \{0,1\}^J,\ \sum_{i \in J} u_i \leq 1\}$. It corresponds to an uncertainty budget $\Gamma = 1$, i.e., at most one processing time deviates. Consider the same schedule x, and the set $H' = \{1, 2, 4, 5\}$, represented in Figure 6.3. Set H' is anchored w.r.t. x for budgeted uncertainty set Δ: indeed for any value of $\delta \in \Delta$, the schedule can be repaired by moving only job 3.

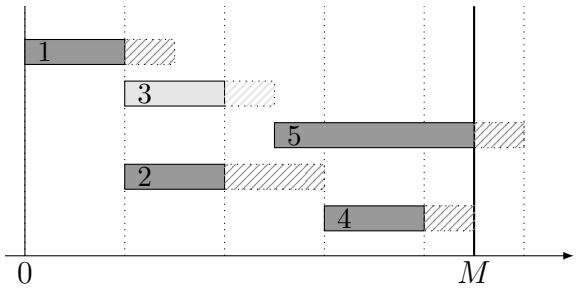


Figure 6.3: Schedule $x = (0, 1, 1, 3, 2.5)$ and set $H' = \{1, 2, 4, 5\}$ in dark gray, anchored for budgeted uncertainty set $\Delta = \{(\widehat{\delta}_i u_i)_{i \in J} : u \in \{0,1\}^J,\ \sum_{i \in J} u_i \leq 1\}$.

6.1.3 Anchoring weights vs makespan

The anchoring weight a_i represents the value for guaranteeing the starting time of job i, or the cost incurred if job i must be rescheduled. Then the weight of H represents a saving obtained from anchored jobs. Anchoring weights can also be used to enforce some jobs into the anchored set H.

Solutions (x^0, H) can be evaluated with respect to two conflicting criteria: makespan of the baseline schedule x^0_t, and weight of the anchored set $a(H)$. In AnchRobPSP, the objective is $a(H)$ and x^0_t is upper bounded in the constraints. Alternatively one may define the following variant A-AnchRobPSP. The input is $(G(p), \Delta, a, A)$ with some *anchoring target* $A \geq 0$, and the problem is to find (x^0, H) with x^0 a baseline schedule, H anchored w.r.t. x^0, $a(H) \geq A$, and the makespan x^0_t is minimized. Regarding algorithmic complexity, studying either AnchRobPSP or A-AnchRobPSP is equivalent, and the algorithmic approaches we propose for AnchRobPSP are applicable to A-AnchRobPSP. This variant corresponds to the Cost minimization variant of the Anchor-Robust problem, presented in Chapter 2.

6.1.4 The price of anchor-robustness

Let us highlight the connection between anchor-robustness and standard robust approaches, as noted in Chapter 2. In the adjustable-robust approach, all starting times are adjustable, i.e., there are no anchored jobs ($H = \varnothing$). In the static-robust approach, all starting times are guaranteed, i.e., all jobs are anchored ($H = J$). Note that AnchRobPSP produces a solution where starting times are guaranteed as much as possible, contrarily to the adjustable-robust problem; AnchRobPSP is able to output a baseline schedule respecting deadline M even when a static-robust schedule could not.

Let us show that the price of robustness introduced in (Bertsimas and Sim, 2004) can be generalized to anchor-robustness.

In AnchRobPSP, second-stage schedules are not explicit, however the definition of anchored set ensures that there for every $\delta \in \Delta$ there exists a second-stage schedule such that $x_i^0 = x_i^\delta$ for every $i \in H$. In particular, the makespan of second-stage schedules is not subject to any constraint. Only the makespan of the baseline schedule is bounded by M. In order to compare anchored solutions with other robust approaches, let us consider anchored solutions given as a triplet $(x^0, H, (x^\delta)_{\delta\in\Delta})$, where H is anchored w.r.t. x^0 and x^δ.

The *worst-case makespan* is $\max_{\delta\in\Delta} x_t^\delta$. It is the objective function of static-robust and adjustable-robust problems. For $\alpha \in [0, 100]$ let us define M_α as the minimum value of the worst-case makespan $\max_{\delta\in\Delta} x_t^\delta$, where $(x^0, H, (x^\delta)_{\delta\in\Delta})$ is an anchored solution and $|H| \geq \frac{\alpha}{100}n$. That is, M_α is the minimum worst-case makespan that can be guaranteed if $\alpha\%$ of the jobs are anchored. Then $M_{\alpha=0}$ is exactly the optimal value M_{AR} of the adjustable-robust problem, and $M_{\alpha=100}$ is exactly the optimal value M_{RS} of the static-robust problem.

Importantly, M_α can easily be computed with algorithmic tools designed for AnchRobPSP. Consider adding to the precedence graph a terminal fictitious job t^*, with $p_{t^*} = 0$. If t^* is anchored, the starting time of job t^* corresponds the worst-case makespan. Define unit anchoring weights. Then M_α is the optimal value of problem A-AnchRobPSP in the new instance, with t^* enforced anchored and with anchoring target $A = 1 + \frac{\alpha}{100}n$.

Let $M_{\min}$ denote the minimum makespan of a schedule of $G(p)$. The price of robustness for the adjustable-robust problem and the static-robust problem are classically defined as $\frac{M_{AR}}{M_{\min}}$ and $\frac{M_{RS}}{M_{\min}}$ respectively, and it holds that $\frac{M_{AR}}{M_{\min}} \leq \frac{M_{RS}}{M_{\min}}$. For $\alpha \in [0,\ 100]$, let the *price of anchor-robustness* $PoAR_{\alpha\%}$ be the ratio $\frac{M_\alpha}{M_{\min}}$. By varying the anchoring target α in $[0, 100]$, it is possible to tune the price of (anchor-)robustness, and get intermediary solutions between adjustable-robust solutions and static-robust solutions. Some computations of the price of anchor-robustness are carried out in Section 6.6.

6.2 Graph Models for the Anchor-Robust Project Scheduling Problem

Let us first restate a technical result that will be used in the sequel. Let $\widetilde{G}$ be a directed acyclic graph with vertex-set $\overline{J} = J \cup \{s, t\}$ and let $\widetilde{G}(\pi)$ be the weighted digraph obtained by adding arc-weight $\pi_{ij} \in \mathbb{R}_+$ to every arc (i, j) of $\widetilde{G}$. Note that $\widetilde{G}(\pi)$ is a precedence graph for project scheduling under generalized precedence. Given a path P in $\widetilde{G}(\pi)$, the length of the path P in $\widetilde{G}(\pi)$ is denoted by $\ell_{\widetilde{G}(\pi)}(P)$. Given two vertices $i, j \in \overline{J}$, let $L_{\widetilde{G}(\pi)}(i, j)$ be the length of the longest $i-j$ path in $\widetilde{G}(\pi)$ (or $-\infty$ if there is no $i-j$ path in $\widetilde{G}(\pi)$). Given $x \in \mathbb{R}_+^{\overline{J}}$ and $H \subseteq J$, let x_H denote the restriction of vector x to H, i.e., $x_H = (x_i)_{i \in H}$.

In Chapter 5 we gave a characterization of extendable sets, i.e., subsets of starting times from a baseline that can be extended into a new solution of a project scheduling instance. Let us restate this result as a lemma.

Lemma 6.1. *Given $\widetilde{G}(\pi)$, a set $H \subseteq J$ and x^0_H a vector of $\mathbb{R}_+^H$, there exists a schedule x of $\widetilde{G}(\pi)$ such that $x_H = x^0_H$ if and only if $x^0_j - x^0_i \geq L_{\widetilde{G}(\pi)}(i, j)$ for every $i, j \in H \cup \{s\}$.*

Let $(G(p), \Delta, M, a)$ be an instance of AnchRobPSP. It is assumed throughout the paper that $L_{G(p)}(s, t) \leq M$, namely, a baseline schedule always exists.

6.2.1 A General-Purpose Graph Model

Let Δ be a general uncertainty set. For every $i, j \in \overline{J}$, let $L^{\Delta}_{ij} = \max_{\delta \in \Delta} L_{G(p+\delta)}(i, j)$. Let G^{Δ} be the weighted digraph obtained by taking the transitive closure of the precedence graph G, and adding the arc-weight L^{Δ}_{ij} on each arc (i, j) with $j \neq t$, and the arc-weight $L_{G(p)}(i, t)$ on each arc (i, t). Given $H \subseteq J$, let $G^{\Delta}[H]$ denote the subgraph of G^{Δ} induced by $H \cup \{s, t\}$. An example of an instance with 5 jobs and processing times $p = (2, 1, 2, 1, 2)$ is represented in Figure 6.4(a) with a precedence graph $G(p)$ and in Figure 6.4(b) with the associated graph G^{Δ} for the box uncertainty set $\Delta = \Pi_{i \in J}[0, \delta^+_i]$ with $\delta^+ = (2, 1, 2, 1, 1)$. Note that in this case $L^{\Delta}_{ij} = L_{G(p+\delta^+)}(i, j)$ for every $i, j \in J$.

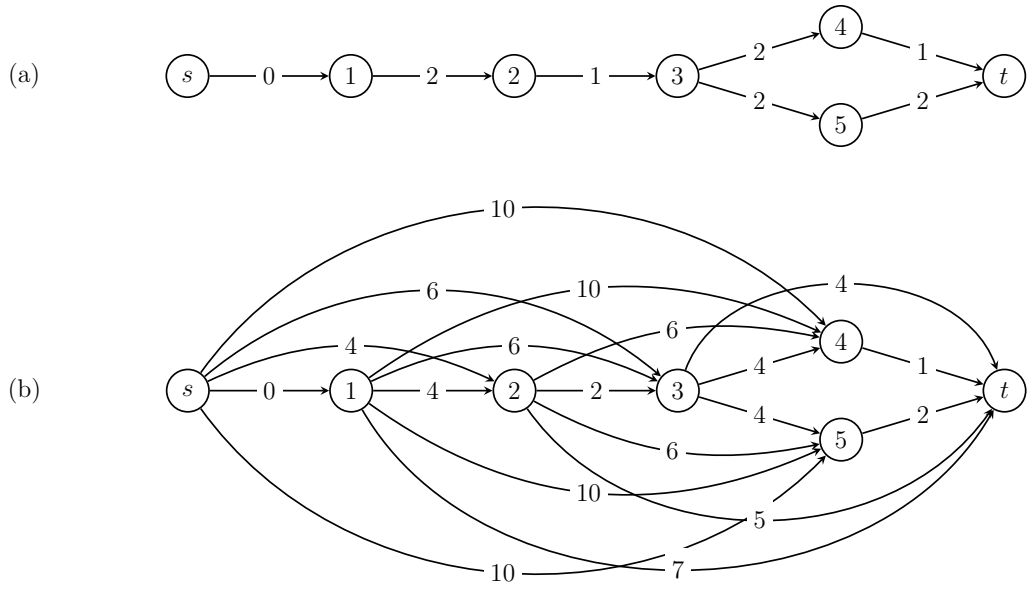


Figure 6.4: (a) A precedence graph $G(p)$ (b) The associated graph G^{Δ} for $\Delta = \Pi_{i\in J}[0, \delta_i^+]$ with $\delta^+ = (2, 1, 2, 1, 1)$.

We prove the following result.

Theorem 6.1. *Let $H \subseteq J$.*
(a) Let x be a schedule of $G(p)$. The set H is anchored w.r.t. schedule x if and only if $x_j - x_i \geq L^{\Delta}_{ij}$ for every $i, j \in H \cup \{s\}$.
(b) The set H is anchored w.r.t. a baseline schedule with makespan at most M if and only if $L_{G^{\Delta}[H]}(s, t) \leq M$.

Proof. Note first that equivalence (a) follows from Lemma 6.1. Indeed, the set H is anchored w.r.t. x if and only if for every $\delta \in \Delta$, we have $x_j - x_i \geq L_{G(p+\delta)}(i, j)$ for every $i, j \in H \cup \{s\}$, which is equivalent to $x_j - x_i \geq \max_{\delta\in\Delta} L_{G(p+\delta)}(i, j) = L^{\Delta}_{ij}$.

Let us now prove equivalence (b). Assume H is anchored w.r.t. a baseline schedule x^0. As a consequence of (a), it holds that $x^0_j - x^0_i \geq L^{\Delta}_{ij}$ for every $i, j \in H \cup \{s\}$. Also $x^0_t - x^0_i \geq L_{G(p)}(i, t)$ since x^0 is a schedule of $G(p)$. Hence $x^0_{H\cup\{s,t\}}$ is a schedule of $G^{\Delta}[H]$ and $x^0_t \leq M$, thus $L_{G^{\Delta}[H]}(s, t) \leq M$.
Conversely, assume $L_{G^{\Delta}[H]}(s, t) \leq M$. Let $\widetilde{G}$ be the (multi)graph obtained from $G(p)$ by adding all the arcs of $G^{\Delta}[H]$. The claim is that $L_{\widetilde{G}}(s, t) \leq M$. Let P be an s–t path in $\widetilde{G}$. Let $h_1, \ldots, h_q$ be the vertices of $H \cap P$, indexed in the order they appear in the path. Let $h_0 = s$ and $h_{q+1} = t$ for the simplicity of notation. Let P_k be the subpath of P from h_k to h_{k+1}. Then for $k < q$, $\ell_{\widetilde{G}}(P_k) \leq L^{\Delta}_{h_k h_{k+1}}$:

indeed either P_k is the arc $(h_k h_{k+1})$ from $G^\Delta[H]$ of length $L^\Delta_{h_k h_{k+1}}$, or P_k is a path in $G(p)$, hence since $\Delta \subseteq \mathbb{R}^J_+$, an upper bound on the length of P_k is $L^\Delta_{h_k h_{k+1}}$. Also $\ell_{\widetilde{G}}(P_q) \leq L_{G(p)}(h_q, t)$. Therefore, the length of P in $\widetilde{G}$ is at most the length of the path $(s, h_1, \ldots, h_q, t)$ in $G^\Delta[H]$. It follows that $L_{\widetilde{G}}(s,t) \leq L_{G^\Delta[H]}(s,t) \leq M$. As a consequence from the claim, there exists a schedule x^0 of $\widetilde{G}$ such that $x^0_t \leq M$, hence x^0 is a baseline schedule. Also $x^0_j - x^0_i \geq L^\Delta_{ij}$ for every $i, j \in H \cup \{s\}$ by definition of $\widetilde{G}$, which implies with (a) that H is anchored w.r.t. x^0. □

Referring to the example shown in Figure 6.4, consider the subset of jobs $H = \{1, 5\}$. The longest s–t path in $G^\Delta[H]$ is $(s, 1, 5, t)$ and has length 12. From Theorem 6.1, for the considered uncertainty set Δ the subset $H = \{1, 5\}$ can then be anchored w.r.t. a baseline schedule if and only if the deadline M is at least 12.

Note that the graph G^Δ is dense since it is the transitive closure of the precedence graph. Moreover, the computation of the arc-weights L^Δ_{ij} might already be a hard problem, depending on the definition of Δ. Indeed computing a weight L^Δ_{ij} amounts to solving a maximization problem on Δ. More details are provided in the complexity analysis presented in Section 6.3.1.

6.2.2 A Layered Graph for Budgeted Uncertainty

Let $\widehat{\delta} \in \mathbb{R}^J_+$, $\Gamma \in \{1, \ldots, n\}$ and let Δ be the budgeted uncertainty set $\Delta = \{(\widehat{\delta}_i u_i)_{i \in J} : \ u_i \in \{0,1\}^J, \ \sum_{i \in J} u_i \leq \Gamma\}$. The objective of this section is to provide a dedicated graph model for budgeted uncertainty. The *layered graph* G^{lay} is defined as follows. It contains $\Gamma + 1$ copies of the precedence graph $G(p)$, called *layers*, indexed by $\gamma \in \{0, \ldots, \Gamma\}$. The layer Γ is called *upper layer*, the layer 0 is the *lowest layer*. The vertices in layer γ are denoted by $s^\gamma, 1^\gamma, \ldots, n^\gamma, t^\gamma$. The arcs of each copy of $G(p)$ are called *horizontal arcs*. Additionally, for each arc (i, j) of $G(p)$, for each $\gamma \in \{0, \ldots, \Gamma - 1\}$, the layered graph contains a *transversal arc* $(i^{\gamma+1}, j^\gamma)$ with weight $p_i + \widehat{\delta}_i$. Given $H \subseteq J$, the *layered graph* $G^{\text{lay}}(H)$ *associated with* H is defined from the layered graph G^{lay} by adding, for each job $i \in H$, for each $\gamma \in \{0, \ldots, \Gamma - 1\}$, a *vertical arc* (i^γ, i^Γ) with weight 0.

Referring to the instance presented in Figure 6.4, the corresponding layered graph $G^{\text{lay}}(H)$ for $H = \{1, 5\}$, $\widehat{\delta} = \delta^+ = (2, 1, 2, 1, 1)$ and $\Gamma = 2$ is represented in Figure 6.5, with copies of jobs in H appearing in gray.

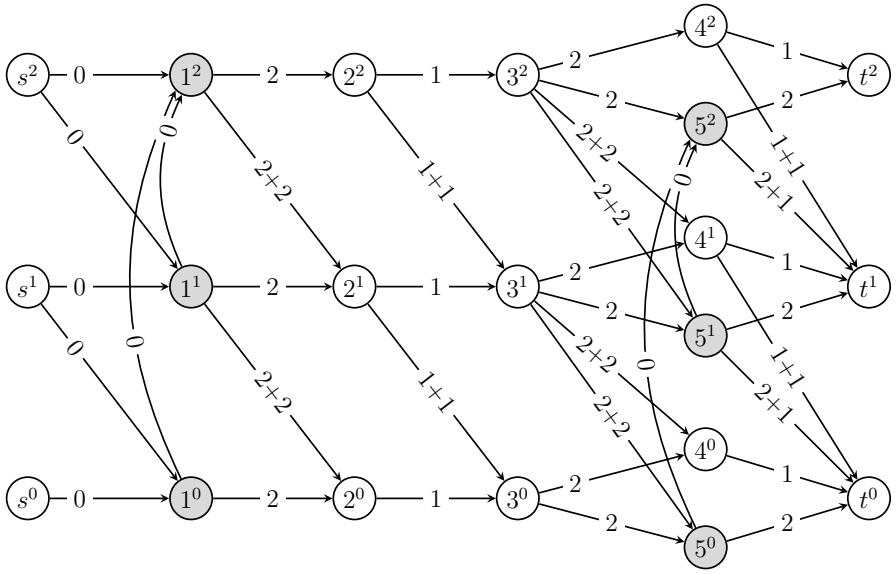


Figure 6.5: The layered graph $G^{\text{lay}}(H)$ associated with precedence graph from Figure 6.4, for $H = \{1, 5\}$ and $\Gamma = 2$.

A key property of the layered graph is stated in Lemma 6.2.

Lemma 6.2. *Let $i, j \in \overline{J}$. In the layered graph G^{lay}, the longest path from i^Γ to a copy of j has length L^Δ_{ij}.*

Proof. First, let us prove that by construction of the layered graph, the length of any path P from i^Γ to a copy of j is upper-bounded by L^Δ_{ij}. For every job k on the path P, let δ^*_k be equal to $\widehat{\delta}_k$ if P goes through a transversal arc outgoing from a copy of k, and $\delta^*_k = 0$ otherwise. Then the length of P in the layered graph is equal to the length of the corresponding path in $G(p + \delta^*)$. Also, path P uses at most Γ transversal arcs, hence $\delta^* \in \Delta$ and the length of P is at most L^Δ_{ij}. Let us now show that the equality holds by exhibiting for every i, j a path P of the layered graph of length L^Δ_{ij}. The value L^Δ_{ij} is equal to $\ell_{G(p+\delta^*)}(Q)$ for some $\delta^* \in \Delta$ and some path Q in the precedence graph. Let P be the path in the layered graph obtained by starting at vertex i^Γ and following the arcs of Q: path P follows a horizontal arc if the corresponding job is not disrupted in δ^*, or a transversal arc if the corresponding job is disrupted in δ^*. Then the length of P is equal to L^Δ_{ij}. □

We note that the layered graph G^{lay} can be interpreted as the state graph corresponding to the dynamic programming algorithm from (Minoux, 2007b).

Theorem 6.2. *Let Δ be a budgeted uncertainty set. Let $H \subseteq J$.*
(a) Let z be a schedule of $G(p)$. The set H is anchored w.r.t. schedule z if and only if there exists a schedule x of $G^{\text{lay}}(H)$ such that $x_i^\Gamma = z_i$ for every $i \in \overline{J}$.
(b) A set H is anchored w.r.t. a baseline schedule with makespan at most M if and only if $L_{G^{\text{lay}}(H)}(s^\Gamma, t^\Gamma) \leq M$.

Proof. (a) Assume the set H is anchored w.r.t. z, or equivalently with Theorem 6.1, that $z_j - z_i \geq L^\Delta_{ij}$ for every $i, j \in H \cup \{s\}$. Let us prove that $z_j - z_i \geq L_{G^{\text{lay}}(H)}(i^\Gamma, j^\Gamma)$ for every $i, j \in \overline{J}$: indeed by Lemma 6.1 it is a sufficient condition for the existence of a schedule x of $G^{\text{lay}}(H)$ such that $x_i^\Gamma = z_i$ for every $i \in \overline{J}$. Let P be an $i^\Gamma - j^\Gamma$ path in $G^{\text{lay}}(H)$, and let $h_1, \ldots, h_q$ be the jobs of H corresponding to vertical arcs of P. Let $h_0 = i$ and $h_{q+1} = j$ for the ease of notation. Consider P_k, $k \in \{0, \ldots, q\}$ the subpath of P going from h_k^Γ to a copy of h_{k+1}. By definition of the h_k's the path P_k uses no vertical arc: it is a path in G^{lay}. By Lemma 6.2, it comes $\ell_{G^{\text{lay}}(H)}(P_k) \leq L^\Delta_{h_k h_{k+1}}$. By assumption on z it holds that $L^\Delta_{h_k h_{k+1}} \leq z_{h_{k+1}} - z_{h_k}$, hence $\ell_{G^{\text{lay}}(H)}(P_k) \leq z_{h_{k+1}} - z_{h_k}$. Summing up the inequality over $k \in \{0, \ldots, q\}$ we obtain $\ell_{G^{\text{lay}}(H)}(P) \leq z_{h_{q+1}} - z_{h_0} = z_j - z_i$. This holds for every $i^\Gamma - j^\Gamma$ path P, therefore $z_j - z_i \geq L_{G^{\text{lay}}(H)}(i^\Gamma, j^\Gamma)$.
Conversely, assume the existence of such a schedule x. Let us prove that $z_j - z_i \geq L^\Delta_{ij}$ for every $i, j \in H \cup \{s\}$. Let $i, j \in H \cup \{s\}$, and consider $\gamma \in \{0, \ldots, \Gamma\}$ such that there is a path from i^Γ to j^γ which is a longest path between i^Γ and a copy of j in G^{lay}. Then $z_j - z_i = x_j^\Gamma - x_i^\Gamma = x_j^\Gamma - x_j^\gamma + x_j^\gamma - x_i^\Gamma$, where $x_j^\Gamma - x_j^\gamma \geq 0$ because of the vertical arc (j^γ, j^Γ), and $x_j^\gamma - x_i^\Gamma \geq L^\Delta_{ij}$ by definition of γ and with Lemma 6.2. Thus $z_j - z_i \geq L^\Delta_{ij}$ and with Theorem 6.1, the set H is anchored w.r.t. z.

Equivalence (b) can then be proven as follows. From (a), a subset H is anchored w.r.t. a baseline schedule if and only if there exists a schedule of $G^{\text{lay}}(H)$ whose restriction to the upper layer has makespan at most M. The existence of such a schedule is equivalent to $L_{G^{\text{lay}}(H)}(s^\Gamma, t^\Gamma) \leq M$. □

Referring to the layered graph represented in Figure 6.5, the longest $s^2 - t^2$ path is $(s^2, 1^2, 2^1, 3^1, 5^0, 5^2, t^2)$, with length 11. From Theorem 6.2, for uncertainty budget $\Gamma = 2$, the subset $H = \{1, 5\}$ can be anchored w.r.t. a baseline schedule if and only if the deadline M is at least 11.

Let us now derive a compact mixed integer programming (MIP) formulation for AnchRobPSP under budgeted uncertainty, using the layered graph. A real variable $x_j^\gamma \geq 0$ is associated with every vertex j^γ of the layered graph; a binary variable $h_j \in \{0, 1\}$ is associated with every job $j \in J$, so that $h_j = 1$ if j is anchored, and $h_j = 0$ otherwise. Let $\mathcal{A}$ denote the arc-set of the precedence graph G. Let $D_j = L_{G(p+\widehat{\delta})}(s, j) - L_{G(p)}(s, j)$ for every $j \in J$. Consider the following program.

$$
\begin{aligned}
\text{(Lay)} \qquad \max \quad & \sum_{i \in J} a_i h_i \\
\text{s.t.} \quad & x_j^\gamma - x_i^\gamma \geq p_i && \forall (i,j) \in \mathcal{A},\ \forall \gamma \in \{0,\dots,\Gamma\} && (1)\\
& x_j^\gamma - x_i^{\gamma+1} \geq p_i + \widehat{\delta}_i && \forall (i,j) \in \mathcal{A},\ \forall \gamma \in \{0,\dots,\Gamma-1\} && (2)\\
& x_j^\Gamma - x_j^\gamma \geq -D_j(1-h_j) && \forall j \in J,\ \forall \gamma \in \{0,\dots,\Gamma-1\} && (3)\\
& x_t^\Gamma \leq M && && (4)\\
& x_j^\gamma \geq 0 && \forall j \in \overline{J},\ \forall \gamma \in \{0,\dots,\Gamma\} &&\\
& h_j \in \{0,1\} && \forall j \in J &&
\end{aligned}
$$

Theorem 6.3. *Program* (Lay) *is a valid MIP formulation for AnchRobPSP under uncertainty budget* Γ.

Proof. With Theorem 6.2, AnchRobPSP is equivalent to maximizing $a(H)$ subject to $H \subseteq J$, x schedule of $G^{\text{lay}}(H)$, $x_t^\Gamma \leq M$. Let us show that there exists a feasible pair (x, H) of the problem with objective value $a(H)$ if and only if there exists a feasible solution of program (Lay) with objective value equal to $a(H)$. First note that, given a feasible solution (x, h) of the MIP, it holds that x is a schedule of $G^{\text{lay}}(H)$ where $H = \{i \in J\colon\ h_i = 1\}$. Indeed constraints (1) and (2) correspond to horizontal and transversal arcs respectively, and constraints (3) correspond to vertical arcs for jobs $i \in H$. Conversely, consider $H \subseteq J$ such that there exists a schedule x of $G^{\text{lay}}(H)$, $x_t^\Gamma \leq M$. Let x' be the earliest schedule of $G^{\text{lay}}(H)$. Note that $x'^\Gamma_t \leq M$. Let h be the indicator vector of H. Then (x', h) is a feasible solution of the MIP with objective value $a(H)$. Indeed x' clearly satisfies constraints (1)-(2)-(4), and constraint (3) when $h_j = 1$. It remains to prove that x' also satisfies constraint (3) when $h_j = 0$. We have $x'^\gamma_j = L_{G^{\text{lay}}(H)}(s^\Gamma, j^\gamma) \leq L_{G^{\text{lay}}(J)}(s^\Gamma, j^\gamma) \leq L_{G(p+\widehat{\delta})}(s, j)$. Indeed in $G^{\text{lay}}(J)$ the longest $s^\Gamma - j^\gamma$ path uses alternatively vertical and transversal arcs, hence its length is the length of the corresponding $s-j$ path in $G(p+\widehat{\delta})$. Also $x'^\Gamma_j \geq L_{G(p)}(s, j)$, hence $x'^\gamma_j - x'^\Gamma_j \leq L_{G(p+\widehat{\delta})}(s, j) - L_{G(p)}(s, j) = D_j$. □

Note that the values D_j can be computed in polynomial time in a preprocessing step.

Theorem 6.3 thus provides a compact MIP formulation for AnchRobPSP under budgeted uncertainty, with a linear number of binary variables, a polynomial number of continuous variables and a polynomial number of constraints. Note also that program (Lay) can readily be modified to tackle A-AnchRobPSP, by setting $\min x_t^\Gamma$ in the objective and adding constraint $\sum_{i \in J} a_i h_i \geq A$ instead of (4). It

is important to emphasize that a compact formulation is not a general rule for robust 2-stage problems but rather an exception. For static-robust problems such formulations were key results of the work of (Soyster, 1973) and (Bertsimas and Sim, 2004). Both AnchRobPSP and A-AnchRobPSP for budgeted uncertainty can thus be readily solved with an off-the-shelf MIP solver.

6.3 Complexity of the Anchor-Robust Project Scheduling Problem

To study the complexity of AnchRobPSP, we consider two distinct complexity issues, depending on whether the inputs of the problem contain $G(p)$ and Δ, or directly the graph G^{Δ} with the arc-weights L^{Δ}_{ij}. On the one hand, if the values L^{Δ}_{ij} are part of the input, the decision version of AnchRobPSP is in NP, since checking whether $L_{G^{\Delta}[H]}(s,t) \leq M$ can be done in polynomial time. On the other hand, if the values L^{Δ}_{ij} are not part of the input, we show that for some uncertainty sets it is co-NP-complete to decide if a set H is anchored, and thus AnchRobPSP is co-NP-hard. Let us start with this latter point.

6.3.1 AnchRobPSP is at least as hard as computing L^{Δ}_{ij}

Let $\mathcal{D}$ be a family of uncertainty sets. Let Worst-case Longest Path WLP($\mathcal{D}$) be the following problem: given a precedence graph $G(p)$, a job $j \in J$, $\Delta \in \mathcal{D}$, $L \geq 0$, is it true that for every $\delta \in \Delta$, the longest $s-j$ path in $G(p+\delta)$ has length at most L? Note that this problem is to decide whether $L^{\Delta}_{sj} \leq L$. Let also Anch($\mathcal{D}$) be the decision version of AnchRobPSP over the family of uncertainty sets $\mathcal{D}$, that is: given a precedence graph $G(p)$, $\Delta \in \mathcal{D}$, $M \geq 0$, $a \in \mathbb{R}^J_+$, $A \geq 0$, is there $H \subseteq J$ such that $a(H) \geq A$ and $L_{G^{\Delta}[H]}(s,t) \leq M$?

Lemma 6.3. *There exists a polynomial reduction from* WLP($\mathcal{D}$) *to* Anch($\mathcal{D}$).

Proof. Let $\mathcal{I}_{WLP} = (G(p), j, \Delta, L)$ be an instance of WLP($\mathcal{D}$). An associated instance $\mathcal{I}_A = (G'(p'), \Delta', M', a', A')$ of Anch($\mathcal{D}$) problem is built as follows. Let $G' = G$, and $p'_i = p_i$ if job i is a predecessor of j, and $p'_i = 0$ otherwise (including $p'_j = 0$). Let $M' = L$ and $\Delta' = \Delta$. Finally let $a'_i = 1$ for every $i \neq j$, and $a'_j = A' = n$, where n is the number of jobs in $G(p)$.

For instance $\mathcal{I}_A$, the anchoring weights are such that a subset H of jobs satisfies $a'(H) \geq A'$ if and only if it contains job j. If $\mathcal{I}_{WLP}$ has a 'yes' answer, i.e., if $L^{\Delta}_{sj} \leq L$, then $L_{G'^{\Delta'}[\{j\}]}(s,t) = L^{\Delta}_{sj} + L_{G'(p')}(j,t) \leq L = M'$. Thus the singleton $\{j\}$ yields a 'yes' answer for $\mathcal{I}_A$. Conversely, if H is anchored and $a'(H) \geq A'$,

then $j \in H$. The value L^{Δ}_{sj} is the arc-weight of the arc (s, j) of $G'^{\Delta'}[H]$, hence $L^{\Delta}_{sj} \leq L_{G'^{\Delta'}[H]}(s, t) \leq M' = L$. □

Note that the reduction can be modified so that anchoring weights are all equal to one in $\mathcal{I}_A$. Indeed it suffices to replace job j by n duplicates of j, all with the same predecessors and successors as j. The reduction then applies similarly.

Let us fix notation for the Knapsack problem in order to show the following theorem. The Knapsack problem is: given a set I of m items, $w \in \mathbb{N}^I$, $v \in \mathbb{N}^I$, $W \in \mathbb{N}$, $V \in \mathbb{N}$, is there a subset $S \subseteq I$ of items such that $w(S) \leq W$ and $v(S) \geq V$.

Theorem 6.4. *AnchRobPSP is co-NP-hard for upper-bounded uncertainty sets, even with unit anchoring weights.*

Proof. We claim that the complement of WLP($\mathcal{D}$) is NP-hard when $\mathcal{D}$ denotes the family of sets Δ defined by $\Delta = \left\{(\widehat{\delta}_i u_i)_{i\in J} \colon\ u_i \in \{0,1\}^J,\ \sum_{i\in J} \alpha_i u_i \leq \beta\right\}$ for some $\alpha \in \mathbb{N}^J$ and $\beta \in \mathbb{N}$. Every Δ of that form is an upper-bounded uncertainty set. From the reduction of Lemma 6.3, the claim will imply that AnchRobPSP is co-NP-hard in this case. Let us now show this claim. The complement of WLP($\mathcal{D}$) is: given $G(p)$, job j, set Δ defined by $\widehat{\delta}$, α, β, and $L \geq 0$, is there $u \in \{0,1\}^J$ such that $\sum_{i\in J} \alpha_i u_i \leq \beta$ and $L_{G(p+\widehat{\delta}u)}(s, j) > L$? The proof is a reduction from Knapsack. Let $\mathcal{I}_{KP} = (I = \{1, \ldots, m\}, w_i, v_i, W, V)$ be an instance of the Knapsack problem. The associated instance $\mathcal{I}_{WLP} = (G(p), j, \widehat{\delta}, \alpha, \beta, L)$ of WLP($\mathcal{D}$) is defined as follows. Let $L = V - 1$, let $G(p)$ be a path $(s, 1, \ldots, m, j, t)$ with arc-weights $p_i = 0$ for every $i \in I$, $p_j = 0$. Let $\widehat{\delta}_i = v_i$, $\alpha_i = w_i$ for every $i \in I$, and $\beta = W$. Given $u \in \{0,1\}^J$, the constraint $\sum_{i\in J} \alpha_i u_i \leq \beta$ then writes $\sum_{i\in I} w_i u_i \leq W$, and $L_{G(p+\widehat{\delta}u)}(s, j) = \sum_{i\in I}(p_i + \widehat{\delta}_i u_i) = \sum_{i\in I} v_i u_i$. Hence there exists $u \in \{0,1\}^J$ that yields a 'yes' answer for the complement of WLP($\mathcal{D}$) on $\mathcal{I}_{WLP}$ if and only if it yields a 'yes' answer for the instance $\mathcal{I}_{KP}$ of the Knapsack problem. □

6.3.2 NP-Completeness for Budgeted Uncertainty

In this section we show that for $\mathcal{D}^\Gamma$ the family of budgeted uncertainty sets, the decision version Anch($\mathcal{D}^\Gamma$) of AnchRobPSP is NP-complete. First note that for every $\Delta \in \mathcal{D}^\Gamma$ the values L^{Δ}_{ij} can be computed in polynomial time by the dynamic programming algorithm from (Minoux, 2007b). Hence Anch($\mathcal{D}^\Gamma$) is in NP.

We establish the following lemma when the precedence graph is a path.

Lemma 6.4. *Let G be an s–t path $(s, 1, 2, \ldots, n, t)$. A subset $\{h_1, \ldots, h_q\} \subseteq J$ (with $h_1 < \cdots < h_q$) is anchored w.r.t. a baseline schedule if and only if $L^{\Delta}_{sh_1} + L^{\Delta}_{h_1h_2} + \cdots + L^{\Delta}_{h_{q-1}h_q} + L_{G(p)}(h_q, t) \leq M$.*

Proof. We show that $L^\Delta_{ij} \leq L^\Delta_{ik} + L^\Delta_{kj}$ for every $i \leq k \leq j$. Indeed if P_{ij} denotes the (unique) $i-j$ path in G, for some $\delta^* \in \Delta$ it holds that $L^\Delta_{ij} = \ell_{G(p+\delta^*)}(P_{ij}) = \ell_{G(p+\delta^*)}(P_{ik}) + \ell_{G(p+\delta^*)}(P_{kj}) \leq L^\Delta_{ik} + L^\Delta_{kj}$. This inequality implies that the longest $s-t$ path in $G^\Delta[H]$ is the path $(s = h_0, h_1, \ldots, h_q, t = h_{q+1})$ going through all jobs of H. The result follows from Theorem 6.1. □

Theorem 6.5. Anch$(\mathcal{D}^\Gamma)$ *is NP-complete, even if* $\Gamma = 1$ *and the precedence graph is a path.*

Proof. The proof is a reduction from Knapsack. The problem Anch$(\mathcal{D}^\Gamma)$ on a path with $\Gamma = 1$ is stated as follows: given a precedence graph $G(p)$ which is an $s-t$ path, $\widehat{\delta} \in \mathbb{R}^J_+$, $M \geq 0$, $a \in \mathbb{R}^J_+$, $A \geq 0$, is there a subset $H \subseteq J$ such that $a(H) \geq A$ and $L_{G^\Delta[H]}(s,t) \leq M$, where Δ is the budgeted uncertainty set defined by $\widehat{\delta}$ and $\Gamma = 1$. Let $\mathcal{I}_{KP} = (I = \{1, \ldots, m\}, w_i, v_i, W, V)$ be an instance of the Knapsack problem. The associated instance $\mathcal{I}_A = (J, p_i, \widehat{\delta}, M, a_i, A)$ of Anch$(\mathcal{D}^\Gamma)$ is as follows:

- the set of jobs is $J = \{0, 1, \ldots, m, m+1\}$, where we assume w.l.o.g. that the items $1, \ldots, m$ are sorted by decreasing weight: $w_1 \geq \cdots \geq w_m$
- the precedence graph is the path $(0, 1, \ldots, m+1)$ and $p_i = 0$ for every $i \in J$
- $\widehat{\delta}_i = w_i$ for every $i \in I$, $\widehat{\delta}_0 = 1 + \max\limits_{i \in I} w_i$, $\widehat{\delta}_{m+1} = 0$ (hence $\widehat{\delta}_0 \geq \widehat{\delta}_1 \geq \cdots \geq \widehat{\delta}_{m+1}$)
- $M = W + \widehat{\delta}_0$
- $a_i = v_i$ for every $i \in I$, $a_0 = 0$, $a_{m+1} = \bar{v}$ where $\bar{v} = 1 + \sum_{i \in I} v_i$
- $A = V + \bar{v}$

The size of $\mathcal{I}_A$ is polynomial in the size of $\mathcal{I}_{KP}$. We prove three claims on the instance $\mathcal{I}_A$.

(i) *Let* $H \subseteq J$ *and* $S = H \cap I$. *Then* $a(H) \geq A \iff v(S) \geq V$ *and* $m+1 \in H$. Indeed $a(H) \geq A$ writes $v(S) + \bar{v}\mathbf{1}_{m+1 \in H} \geq V + \bar{v}$. If $m+1 \notin H$ then the inequality cannot be satisfied since $\bar{v} > v(S)$.

(ii) *For every* $i, j \in J$ *such that* $i < j$, $L^\Delta_{ij} = \widehat{\delta}_i$ *and* $L^\Delta_{si} = \widehat{\delta}_0$. Indeed by definition $L^\Delta_{ij} = \max_{\delta \in \Delta} \sum_{i \leq k < j} \delta_k = \max_{i \leq k < j} \widehat{\delta}_k$ which equals $\widehat{\delta}_i$ if $i \neq s$ and $\widehat{\delta}_0$ otherwise.

(iii) *Let* $H \subseteq J$ *such that* $m+1 \in H$ *and let* $S = H \cap I$. *Then* $L_{G^\Delta[H]}(s,t) \leq M \iff w(S) \leq W$. Let $H = \{h_1, \ldots, h_q\}$ with $h_1 < \cdots < h_q$. By Lemma 6.4 and claim (ii), $L_{G^\Delta[H]}(s,t) = \widehat{\delta}_0 + \widehat{\delta}_{h_1} + \cdots + \widehat{\delta}_{h_{q-1}}$. Note that $S = \{h_1, \ldots, h_{q-1}\}$ since $m+1 \in H$. Thus $L_{G^\Delta[H]}(s,t) \leq M \iff \widehat{\delta}_0 + \widehat{\delta}(S) \leq M \iff w(S) \leq W$.

Let us now prove that the instance $\mathcal{I}_{KP}$ of the Knapsack problem has a 'yes' answer if and only if the instance $\mathcal{I}_A$ of Anch$(\mathcal{D}^\Gamma)$ has a 'yes' answer. If there exists some $S \subseteq I$ such that $w(S) \leq W$ and $v(S) \geq V$, then let $H = S \cup \{m+1\}$: by (i) it comes $a(H) \geq A$, and by (iii) $L_{G^\Delta[H]}(s,t) \leq M$. Conversely assume there

exists $H \subseteq J$ such that $a(H) \geq A$ and $L_{G^\Delta[H]}(s,t) \leq M$, and let $S = H \cap I$. Then by (i) $v(S) \geq V$ and $m+1 \in H$, and by (iii) $w(S) \leq W$. Hence the problem $\textsc{Anch}(\mathcal{D}^\Gamma)$ is NP-complete, even if the precedence graph is a path and $\Gamma = 1$. □

Now consider the case where $\widehat{\delta}$ is uniform, i.e., $\widehat{\delta}_i = \widehat{\delta}_0$ for every $i \in J$ for some $\widehat{\delta}_0 \in \mathbb{R}_+$.

Theorem 6.6. $\textsc{Anch}(\mathcal{D}^\Gamma)$ *is NP-complete, even if* $\Gamma = 1$ *and* $\widehat{\delta}$ *is uniform.*

Proof. Let $\mathcal{I}_{KP} = (I = \{1,\dots,m\}, w_i, v_i, W, V)$ be a Knapsack instance. Let us build an instance $\mathcal{I}_A = (n, G, p, \widehat{\delta}_0, M, a, A)$ of $\textsc{Anch}(\mathcal{D}^\Gamma)$ as follows. An illustration is given in Figure 6.6, where the instance is represented, with processing times on arcs and anchoring weights in brackets.

- Let $n = 3m+1$. Jobs are $j_1,\dots,j_{m+1}$, $\alpha_1,\dots,\alpha_m$, $\beta_1,\dots,\beta_m$.
- The precedence graph is as follows: job j_i, $i \leq m$ has two successors α_i and β_i; job α_i has successor job j_{i+1}; job β_i has successor job j_{i+1}.
- Let processing times be defined by: $p_{j_i} = 1$ for every $i \leq m$; $p_{j_{m+1}} = 0$; $p_{\beta_i} = 1$ for every $i \leq m$; $p_{\alpha_i} = 1 + w_{max} - w_i$ for every $i \leq m$, where $w_{max} = \max_{i \in I} w_i$.
- Let $\widehat{\delta}_0 = w_{max}$.
- Let $M = 2m(1+\widehat{\delta}_0) - \sum_{i\in I} w_i + W$.
- With $v_{tot} = \sum_{i\in I} v_i$, let $a_{j_i} = v_{tot}$ for every $i \leq m+1$, $a_{\alpha_i} = 0$ and $a_{\beta_i} = v_i$ for every $i \leq m$.
- Finally $A = (m+1)v_{tot} + V$.

Given a subset of jobs H, it comes that $a(H) = \sum_{i:j_i\in H} v_{tot} + \sum_{i:\beta_i\in H} v_i$. Hence, setting $S := \{i : \beta_i \in H\}$, we have $a(H) = |\{i : j_i \in H\}| v_{tot} + v(S)$. Hence $a(H) \geq A$ if and only if all j_i's are in H and $v(S) \geq V$.

Consider a subset H of jobs that contains all j_i's, but no α_i. Let us now compute $L_{G^\Delta[H]}(s,t)$ in this instance. Fix $i \leq m$. Let us compute $L_{G^\Delta[H]}(j_i, j_{i+1})$. Case 1. If $\beta_i \in H$, the path through α_i has worst-case length $2 + w_{max} - w_i + \widehat{\delta}_0$, the path through β_i has worst-case length $2 + 2\widehat{\delta}_0$. Hence since $\widehat{\delta}_0 = w_{max}$, $L_{G^\Delta[H]}(j_i, j_{i+1}) = 2 + 2\widehat{\delta}_0$.
Case 2. If $\beta_i \notin H$, the path through α_i has worst-case length $2 + w_{max} - w_i + \widehat{\delta}_0$, the path through β_i has worst-case length $2 + \widehat{\delta}_0$. Hence $L_{G^\Delta[H]}(j_i, j_{i+1}) = 2 + w_{max} - w_i + \widehat{\delta}_0$.

We now prove that $\mathcal{I}_{KP}$ has a 'yes' answer if and only if $\mathcal{I}_A$ has a 'yes' answer. Assume there exists S a solution to the knapsack problem for $\mathcal{I}_{KP}$. Consider the subset of jobs $H = \{j_i, i \leq m+1\} \cup \{\beta_i, i \in S\}$. Then $a(H) \geq A$, as noted previously. Consider now the longest s–t path in $G^\Delta[H]$. We claim that w.l.o.g. such a longest path goes through every job j_i, $i \leq m+1$. Consider some job j_i and let u (resp. v) be a job before j_i (resp. after j_i) in the precedence graph.

Then $L^{\Delta}_{uv} = L_{G(p)}(u,v) + \widehat{\delta}_0 = L_{G(p)}(u,j_i) + L_{G(p)}(j_i,v) + \widehat{\delta}_0$, since every $u-v$ path in $G(p)$ goes through j_i. It comes $L^{\Delta}_{uv} \leq L^{\Delta}_{uj_i} + L^{\Delta}_{j_iv}$. With this claim, $L_{G^{\Delta}[H]}(s,t) = L_{G^{\Delta}[H]}(s,j_1) + \sum_{i\leq m} L_{G^{\Delta}[H]}(j_i,j_{i+1}) + L_{G^{\Delta}[H]}(j_{m+1},t)$, and

$$L_{G^{\Delta}[H]}(s,t) = \sum_{i:\beta_i\in H} 2(1+\widehat{\delta}_0) + \sum_{i:\beta_i\notin H} 2(1+\widehat{\delta}_0) + (w_{max} - w_i - \widehat{\delta}_0)$$

$$L_{G^{\Delta}[H]}(s,t) = 2m(1+\widehat{\delta}_0) - \sum_{\beta_i\notin H} w_i = M + \sum_{i:\beta_i\in H} w_i - W = M + w(S) - W$$

Hence $w(S) \leq W$ implies $L_{G^{\Delta}[H]}(s,t) \leq M$.

Conversely, let H be a solution of AnchRobPSP for instance $\mathcal{I}_A$. The previous remark on the weight of H implies that every j_i is in H, and the subset $S = \{i : \beta_i \in H\}$ satisfies $v(S) \geq V$. Then S is a subset of items for the knapsack problem such that $v(S) \geq V$. Let $H' \subseteq H$ be the set of j_i and β_i jobs of H (α_i jobs are removed). Then $L_{G^{\Delta}[H']}(s,t) \leq L_{G^{\Delta}[H]}(s,t) \leq M$. The previous computation gives $L_{G^{\Delta}[H']}(s,t) = M + w(S) - W$. Hence $w(S) \leq W$, and S gives a 'yes' answer to the knapsack instance $\mathcal{I}_{KP}$. This proves the claimed NP-hardness result. □

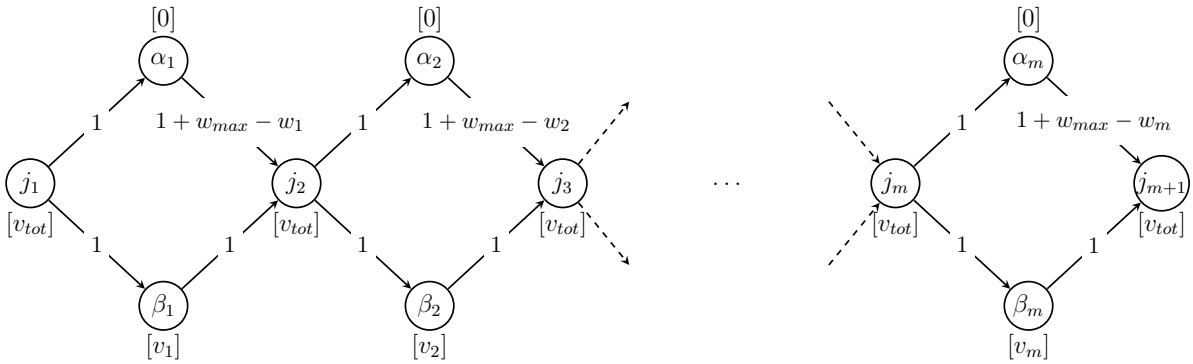


Figure 6.6: Instance of $\textsc{Anch}(\mathcal{D}^{\Gamma})$ used in proof of Theorem 6.6.

6.3.3 Discussion on complexity aspects for given L^{Δ}_{ij}

We finally discuss some aspects of the complexity of the problem, when the values L^{Δ}_{ij} are part of the input of the problem. From Theorem 6.1, the problem AnchRobPSP is equivalent to finding a subset $H \subseteq J$ such that $L_{G^{\Delta}[H]}(s,t) \leq M$, and $a(H)$ is maximized. Equivalently, by considering the set $I = J \setminus H$ of non-anchored jobs, the problem is to find $I \subseteq J$ such that $L_{G^{\Delta}[J\setminus I]}(s,t) \leq M$, and $a(I)$ is minimized. This can be recognized as the min Weight Vertex Blocker to Longest Path Problem (WVBLP), which is a variant of network interdiction problems studied in the literature (Israeli and Wood, 2002; Boros et al., 2006). Given an instance

of a maximization problem on a network, vertex blocker problems look for a subset of vertices to remove, so that the optimum of the maximization problem in the remaining graph falls below a given threshold. Hence AnchRobPSP corresponds to WVBLP on G^Δ, thus raising the question of the complexity of the WVBLP on G^Δ graphs. Many vertex blocker problems have been shown to be NP-complete (Boros et al., 2006), but these results cannot be directly applied to AnchRobPSP, since it is the restriction of WVBLP to very structured instances, namely the G^Δ graphs. Recall that the G^Δ are transitive closures of directed acyclic graphs, and their arc-weights are values L^Δ_{ij} and $L_{G(p)}(j,t)$. Theorem 6.5 provides an NP-hard case of the problem WVBLP on G^Δ instances. Note that the reduction involves non-unit anchoring weights. The complexity of AnchRobPSP when the values L^Δ_{ij} are part of the input, and with unit anchoring weights, is still an open question.

6.4 Algorithms for special cases of AnchRobPSP

In this section we provide an algorithm (Algorithm $\mathbf{A}^+$) to compute a feasible solution for AnchRobPSP under upper-bounded uncertainty, and show that AnchRobPSP under box uncertainty is a polynomial case where Algorithm $\mathbf{A}^+$ is exact. For budgeted uncertainty, polynomial algorithms are exhibited for special cases, and a dynamic programming approach is proposed for series-parallel precedence graphs.

6.4.1 Algorithm $\mathbf{A}^+$ for upper-bounded uncertainty

In this section the set Δ is an upper-bounded uncertainty set, i.e., there exists $\delta^+ \in \mathbb{R}^J_+$ such that $\delta^+_i \geq \delta_i$ for every $i \in J$ for every $\delta \in \Delta$. Let $\underline{x}$ denote the earliest schedule of $G(p+\delta^+)$. A first remark is that if the makespan of $\underline{x}$ is at most M, then $(\underline{x}, J)$ is an optimal solution of AnchRobPSP. Indeed $\underline{x}$ is then both a baseline schedule and a feasible second-stage schedule for every $\delta \in \Delta$, since δ^+ is an upper bound on Δ. However, in general the makespan of $\underline{x}$ is larger than M and then not all jobs can be anchored. Let $\overline{x}$ denote the latest schedule of $G(p)$ such that $\overline{x}_t = M$. Consider the following algorithm.

Algorithm $\mathbf{A}^+$

Input: $G(p)$, δ^+, M.

Output: a solution (x, H) of AnchRobPSP.

Compute $\underline{x}_i = L_{G(p+\delta^+)}(s, i)$ for every $i \in \overline{J}$;

Compute $\overline{x}_i = M - L_{G(p)}(i, t)$ for every $i \in \overline{J}$;

Let $x_i = \min\{\underline{x}_i, \overline{x}_i\}$ for every $i \in \overline{J}$;

Let $H = \{i \in J\colon\ \underline{x}_i \leq \overline{x}_i\}$;

Return (x, H).

Let us first prove that Algorithm $\mathbf{A}^+$ provides a feasible solution for AnchRobPSP for any upper-bounded uncertainty set.

Proposition 6.1. *If Δ is upper-bounded by δ^+, Algorithm $\mathbf{A}^+$ returns a feasible solution of AnchRobPSP.*

Proof. Let (x, H) be the output of Algorithm $\mathbf{A}^+$. First we show that x is a baseline schedule. It is clear that it has makespan at most M since $x_t \leq \overline{x}_t \leq M$. Also $\overline{x}$ and $\underline{x}$ are both schedules of $G(p)$, since $\delta^+ \geq 0$. Hence $x = \min\{\underline{x}, \overline{x}\}$ is also a schedule of $G(p)$. The set H is anchored with respect to x: indeed let us consider for every $\delta \in \Delta$ the same second-stage schedule $\underline{x}$. It is a schedule of $G(p + \delta)$ because δ^+ is an upper bound on Δ, and for every $i \in H$, $\underline{x}_i = x_i$. It follows that the pair (x, H) is a feasible solution of the instance $(G(p), \Delta, M, a)$ of AnchRobPSP. □

Note that a feasible solution of AnchRobPSP for a budgeted uncertainty set can thus be obtained by using Algorithm $\mathbf{A}^+$ with $\delta^+ = \widehat{\delta}$, since $\widehat{\delta}$ is an upper bound for Δ. It is also worth mentioning that solutions computed with Algorithm $\mathbf{A}^+$ have a specific structure. Namely, if H is the set returned by $\mathbf{A}^+$ and $i \notin H$, one can check that no job on the longest path from i to t in $G(p)$ belongs to H. We now prove that box uncertainty is a polynomial case, where Algorithm $\mathbf{A}^+$ is exact.

Theorem 6.7. *For box uncertainty, Algorithm $\mathbf{A}^+$ solves AnchRobPSP in polynomial time.*

Proof. Let us show that $\mathbf{A}^+$ returns an optimal solution when Δ is the box $\Pi_{i \in J}[\delta_i^-, \delta_i^+]$. First it is feasible by Proposition 6.1. Let j be a job not anchored by the algorithm, i.e., such that $\underline{x}_j > \overline{x}_j$. Let x^0 be an arbitrary baseline schedule. Then $L_{G(p+\delta^+)}(s, j) + L_{G(p)}(j, t) = \underline{x}_j + M - \overline{x}_j > M$ by assumption on j, and since x^0 is a baseline schedule $M \geq x_t^0 - x_j^0 + x_j^0 - x_s^0 \geq L_{G(p)}(j, t) + x_j^0 - x_s^0$. It

comes $L_{G(p+\delta^+)}(s,j) > x^0_j - x^0_s$. From Lemma 6.1, there exists no schedule x^{δ^+} of $G(p+\delta^+)$ such that $x^{\delta^+}_j = x^0_j$. Since $\delta^+ \in \Delta$, job j cannot be in any anchored set. Since all anchoring weights are non-negative, the solution returned by the algorithm is optimal. □

Note that the proof only requires that Δ is upper-bounded by δ^+ and that $\delta^+ \in \Delta$, which is more general than Δ being the box $\Pi_{i\in J}[\delta^-_i, \delta^+_i]$. The proof of Theorem 6.7 also yields as a corollary that the set H optimal for AnchRobPSP under box uncertainty is unique, and it is the set returned by Algorithm $\mathbf{A}^+$.

Interestingly, for box uncertainty it can be shown that the proactive problem DAPSP of (Bendotti et al., 2017) and AnchRobPSP coincide. The analysis of Algorithm $\mathbf{A}^+$ thus yields a simpler proof of the polynomiality of DAPSP for box uncertainty. Moreover, the algorithm proposed in (Bendotti et al., 2017) has complexity $O(n^3)$, while Algorithm $\mathbf{A}^+$ is linear in the number of arcs in G.

6.4.2 Polynomial Cases for Budgeted Uncertainty

Under budgeted uncertainty, AnchRobPSP is NP-hard as shown in Theorem 6.5. In this section we exhibit polynomial special cases.

6.4.2.1 Path precedence graph and unit anchoring weights.

We first investigate the case where the precedence graph is a path. The reduction from Theorem 6.5 relies on the numerical values of anchoring weights. We show that in the case of unit anchoring weights the problem becomes solvable in polynomial time.

Theorem 6.8. *Let G be an $s-t$ path, and let Δ be a budgeted uncertainty set. If $a_i = 1$ for every $i \in J$, then AnchRobPSP is solvable in polynomial time.*

Proof. Under budgeted uncertainty, the arc-weights of graph G^Δ can be computed in polynomial time. Consider then the problem of finding an $s-t$ path in G^Δ with length at most M, and with a maximum number of arcs. This problem is solvable in polynomial time as a polynomial case of the Resource-Constrained Longest Path Problem (Garey and Johnson, 1979). The claim is that this problem is equivalent to solving the special case of AnchRobPSP. Indeed from Lemma 6.4, there is a one-to-one correspondance between subsets of jobs and $s-t$ paths in G^Δ, and a subset H is feasible for AnchRobPSP if and only if the corresponding path has length at most M in G^Δ. Hence maximizing $a(H) = |H|$ amounts to maximizing the number of arcs in the associated $s-t$ path, under the constraint that the length of the path is at most M. □

Note that the result holds not only for budgeted uncertainty, but also for any uncertainty set for which the arc-weights L_{ij}^{Δ} can be computed in polynomial time.

6.4.2.2 A special case with p_i=0, $\widehat{\delta}_i$=1, Γ=1, a_i=1.

Let the precedence graph G be any directed acyclic graph. Let us consider the special case U-AnchRobPSP where $p_i = 0$ and $a_i = 1$ for every $i \in J$ and Δ is a budgeted uncertainty with $\widehat{\delta}_i = 1$ for every $i \in J$ and budget $\Gamma = 1$. An instance of U-AnchRobPSP is thus formed with the graph G and the deadline M, assumed to be an integer. We will show that U-AnchRobPSP is a polynomial case, by an equivalence with a poset problem. Recall that a partial order $\prec$ can be naturally defined from the precedence graph G by setting $i \prec j$ if there exists an $i-j$ path in G. Let MaxSubposet be the problem of finding, given a poset and an integer M, a max-size subposet in which all chains have size at most M.

Let (G, M) be an instance of U-AnchRobPSP, and let J^* denote the set of jobs that have a predecessor other than s in the precedence graph G. The arc-weights of G^{Δ} are as follows: the arcs from s to a job $i \notin J^*$ have weight 0, all incoming arcs of t have weight 0, all other arcs have weight 1. Consequently, given $H \subseteq J$, the length of an $s-t$ path in $G^{\Delta}[H]$ is equal to the number of vertices of J^* in the path. It follows that $L_{G^{\Delta}[H]}(s,t) \leq M$ if and only if all chains in the subposet $(H \cap J^*, \prec)$ have size at most M. Finally, since G can be any directed acyclic graph, the poset $(J^*, \prec)$ can be any poset. Thus U-AnchRobPSP is equivalent to MaxSubposet.

Theorem 6.9. *U-AnchRobPSP is solvable in polynomial time.*

Proof. Note first that a subposet in which all chains have size at most M is exactly the union of M antichains (some of the antichains being possibly empty). The problem of finding a max-size union of M antichains in a poset can be solved in polynomial time, through a min-cost circulation algorithm (see Theorem 14.8 in Schrijver (2003)). Using this result and the equivalence between U-AnchRobPSP and MaxSubposet, the result follows. □

6.4.3 Dynamic Programming for Budgeted Uncertainty and Series-Parallel Precedence Graphs

This section is devoted to solving AnchRobPSP under budgeted uncertainty for series-parallel precedence graphs. It is assumed that vectors p, $\widehat{\delta}$, and the deadline M have integer values. Series-parallel digraphs are defined recursively as follows. A digraph is *series-parallel* with terminals s and t if one of the three assertions is satisfied:

- Its vertex-set is $\{s,t\}$ and its arc-set is $\{(s,t)\}$;
- (Series composition.) It is formed with two series-parallel digraphs G_1 and G_2, where terminals t_1 and s_2 have been identified;
- (Parallel composition.) It is formed with two series-parallel digraphs G_1 and G_2, where the two pairs of terminals s_1 and s_2, and t_1 and t_2, have been identified.

In the sequel it is assumed that the precedence graph G is series-parallel with terminals s and t. Given two jobs i and j such that $i \prec j$, let J_{ij} denote the subset of jobs that are successors of i and predecessors of j, with $i,j \notin J_{ij}$, i.e., $J_{ij} = \{k \in J\colon\ i \prec k \prec j\}$. Let G_{ij} denote the subgraph of G induced by $J_{ij} \cup \{i,j\}$: it is series-parallel with terminals i and j.

The proposed algorithmic scheme relies on the layered graph from Section 6.2.2. With Theorem 6.2, a subset $H \subseteq J$ is anchored when there exists a schedule x of $G^{\text{lay}}(H)$ such that $x_t^\Gamma \leq M$. Let us now introduce a value function suitable for a dynamic programming approach. Given a pair of jobs i,j such that $i \prec j$, and $b = (b^\gamma)_{\gamma\in\{0,\dots,\Gamma\}}$ and $b' = (b'^\gamma)_{\gamma\in\{0,\dots,\Gamma\}}$ two vectors of $(\Gamma+1)$ integer numbers, the *value function* is

$$\begin{array}{rlll} V_{ij}(b,b') = & \max & a(H) & \\ & \text{s.t.} & H \subseteq J_{ij} & \\ & & x \text{ schedule of } G_{ij}^{\text{lay}}(H) & \\ & & x_i^\gamma = b^\gamma & \forall \gamma \in \{0,\dots,\Gamma\} \\ & & x_j^\gamma = b'^\gamma & \forall \gamma \in \{0,\dots,\Gamma\} \end{array}$$

The last two conditions are called *boundary conditions* in the sequel: they enforce the values of starting times of copies of i and j in schedule x. Let $L = L_{G(p+\widehat{\delta})}(s,t)$. Let $\underline{b}$ and $\underline{b}'$ be the vectors defined by $\underline{b}^\gamma = 0$ for every $\gamma \in \{0,\dots,\Gamma\}$, $\underline{b}'^\Gamma = M$, and $\underline{b}'^\gamma = L$ for every $\gamma \in \{0,\dots,\Gamma-1\}$. The maximum weight of an anchored set is then exactly the value $V_{st}(\underline{b},\underline{b}')$. Indeed note that the condition $x_t^\gamma \leq L$ for every $\gamma \in \{0,\dots,\Gamma-1\}$ can be added w.l.o.g. By dynamic programming, the value function will be computed for vectors b,b' in the set $B = \{b = (b^\gamma)_{\gamma\in\{0,\dots,\Gamma\}}\colon\ b^\gamma \in \{0,\dots,L\}\ \forall \gamma \in \{0,\dots,\Gamma\}\}$. We now show how to compute the value function in the base case, and prove decomposition properties with respect to series and parallel composition.

Lemma 6.5 (Base case.)**.** *Let G_{ij} be the digraph with vertex-set $\{i,j\}$ and arc-set $\{(i,j)\}$. Let $b,b' \in B$. Then $V_{ij}(b,b') = 0$ if $b'^\gamma - b^\gamma \geq p_i$ for every $\gamma \in \{0,\dots,\Gamma\}$ and $b'^\gamma - b^{\gamma+1} \geq p_i + \widehat{\delta}_i$ for every $\gamma \in \{0,\dots,\Gamma-1\}$. Otherwise $V_{ij}(b,b') = -\infty$.*

Proof. Note that $J_{ij} = \varnothing$, hence the value function is 0 if there exists a schedule x of $G_{ij}^{\text{lay}}(\varnothing)$ satisfying the boundary conditions, and $-\infty$ otherwise. Also, the

boundary conditions fully define the schedule. Hence the value function is 0 if and only if b and b' satisfy the constraints of $G_{ij}^{\text{lay}}(\varnothing)$, i.e., $b'^{\gamma} - b^{\gamma} \geq p_i$ for every $\gamma \in \{0, \dots, \Gamma\}$ and $b'^{\gamma} - b^{\gamma+1} \geq p_i + \widehat{\delta}_i$ for every $\gamma \in \{0, \dots, \Gamma - 1\}$. □

Assume G_{ij} is obtained by a parallel composition, w.l.o.g. parallel composition of two series-parallel digraphs G_{ij}^1 and G_{ij}^2 with terminals i and j. Let J_{ij}^1 and J_{ij}^2 denote respectively their sets of inner vertices and V_{ij}^1 and V_{ij}^2 denote their value functions.

Lemma 6.6 (Parallel composition)**.** *For every $b, b' \in B$, the value function satisfies $V_{ij}(b, b') = V_{ij}^1(b, b') + V_{ij}^2(b, b')$.*

Proof. Let $H \subseteq J_{ij}$ and (H^1, H^2) be the partition of H defined by $H^1 = H \cap J_{ij}^1$, $H^2 = H \cap J_{ij}^2$. Given x a schedule of G_{ij}^{lay}, let x^1 (resp. x^2) denote the restriction of x to copies of jobs in G_{ij}^1 (resp. G_{ij}^2). It holds that x is a schedule of $G_{ij}^{\text{lay}}(H)$ such that $x_i = b$ and $x_j = b'$ if and only if x^1 and x^2 are schedules of $G_{ij}^{\text{lay}}(H^1)$ and $G_{ij}^{\text{lay}}(H^2)$ respectively, and they satisfy $x_i^1 = b$ and $x_j^1 = b'$, and $x_i^2 = b$ and $x_j^2 = b'$. Hence the value function $V_{ij}(b, b')$ decomposes over the subgraphs G_{ij}^1 and G_{ij}^2, leading to the desired equality. □

Assume G_{ij} is obtained by series composition of two series-parallel digraphs G_{ik} and G_{kj} for a given $k \in J_{ij}$.

Lemma 6.7 (Series composition)**.** *For every $b, b' \in B$, the value function satisfies*

$$V_{ij}(b, b') = \max \left\{ \max_{b'' \in B} \{V_{ik}(b, b'') + V_{kj}(b'', b')\}\,; \quad a_k + \max_{b'' \in B^{anch}} \{V_{ik}(b, b'') + V_{kj}(b'', b')\} \right\}$$

where $B^{anch} = \{b \in B\colon\ b^{\gamma} \leq b^{\Gamma}\ \forall \gamma \in \{0, \dots, \Gamma\}\}$.

Proof. Let $H \subseteq J_{ij}$ and let $H^1 = H \cap J_{ik}$ and $H^2 = H \cap J_{kj}$. Given x a schedule of G_{ij}^{lay}, let x^1 (resp. x^2) denote the restriction of x to copies of jobs in G_{ik} (resp. G_{kj}). It holds that x is a schedule of $G_{ij}^{\text{lay}}(H)$ such $x_i = b$ and $x_j = b'$ if and only if: x^1 (resp. x^2) is a schedule of $G_{ik}^{\text{lay}}(H^1)$ (resp. $G_{kj}^{\text{lay}}(H^2)$), $x_k^1 = x_k^2 = b''$ for some $b'' \in B$, and either (i) or (ii) is satisfied: (i) $k \notin H$ (ii) $k \in H$, $b'' \in B^{anch}$. Indeed when $k \in H$, both schedules x^1 and x^2 must satisfy the vertical arcs constraints in k, which is equivalent to $b'' \in B^{anch}$. Case (i) leads to the first term and case (ii) leads to the second term of the maximum; we thus obtain the desired equality. □

Theorem 6.10. *For fixed uncertainty budget Γ, AnchRobPSP can be solved in $O(mL^{3\Gamma+3})$, where m is the number of arcs in the precedence graph and $L = L_{G(p+\widehat{\delta})}(s, t)$.*

Proof. Given a series-parallel digraph G, its *binary decomposition tree* is a binary tree whose leaves are attached to arcs of the digraph, and internal nodes represent series or parallel compositions. The binary decomposition tree of G is computable in linear time (Valdes et al., 1982) and it has $2m-1$ nodes, each of them corresponding to a subgraph G_{ij} of G, which is series-parallel with terminals i and j.

Let us now describe the overall algorithm, using the binary decomposition tree of the precedence graph. Nodes of the decomposition tree are considered in a bottom-up fashion. For every subgraph G_{ij} associated with the current node of the tree, compute and store the value $V_{ij}(b, b')$ for every pair of vectors $b, b' \in B$. If G_{ij} is obtained by parallel composition, this can be done in $O(1)$ time (Lemma 6.6). If G_{ij} is obtained by series composition, this can be done in $O(|B|)$ time (Lemma 6.7). If G_{ij} is as arc, this can be done in $O(\Gamma)$ time (Lemma 6.5). Finally return $V_{st}(\underline{b}, \underline{b}')$.

The complete table of the value function has $(2m-1)|B|^2$ entries, and it can be filled in $O(m|B|^2 \max\{|B|, \Gamma\})$. For fixed Γ, since $|B| = O(L^{\Gamma+1})$, the total running time is $O(mL^{3\Gamma+3})$. □

Note that the values of anchoring weights have no impact on the running time of the proposed algorithm. Note also that the dynamic programming scheme can be refined, e.g., by decreasing the size of the set B; however this would not change the final pseudo-polynomiality result.

6.5 Comparison to affine decision rules

In this section, we investigate affine decision rules for AnchRobPSP. Affine decision rules are a classical approximation approach to robust 2-stage problems that originates from control systems. Affine decision rules are reminiscent of linear feedback to reduce the adverse effect of disruptions on the system (Ben-Tal et al., 2004). We present an MIP formulation for solving the problem under affine decision rules, and give some cases where affine decision rules can be proven optimal. Affine decision rules being a state-of-the-art approach for robust 2-stage optimization, they will serve as a reference point for evaluating our approaches in numerical experiments.

6.5.1 Approach and MIP

Assume that second-stage schedule variables x^δ affinely depend on the uncertainty realization δ, that is,

$$x^\delta_j - x^0_j = y_j + \sum_{k \in J} Y_{jk}\delta_k \ \forall j \in J$$

where $y \in \mathbb{R}^J$ and $Y \in \mathbb{R}^{J \times J}$ are a vector and a matrix to be determined. The special case where $y = 0$ is referred to as linear decision rule: the difference between the second-stage solution and the baseline solution linearly depends on δ. Let us denote by $[Y\delta]_j$ the j-th coordinate of vector $Y\delta$.

Under this assumption, the problem AnchRobPSP under affine decision rules, denoted by AnchRob-Aff, is:

(AnchRob-Aff):

$$\begin{array}{lll} \max & \min & \max \sum_{i \in J} a_i h_i \\ x^0_j - x^0_i \geq p_i \ \forall (i,j) \in \mathcal{A} & \delta \in \Delta & x^\delta_j = x^0_j + y_j + [Y\delta]_j \ \forall j \in J \\ x^0_t \leq M & & x^\delta_j - x^\delta_i \geq p_i + \delta_i \ \forall (i,j) \in \mathcal{A} \\ x^0 \in \mathbb{R}^{\overline{J}}_+ & & x^\delta \in \mathbb{R}^J_+ \\ h \in \{0,1\}^J & & |x^\delta_j - x^0_j| \leq D_j(1-h_j) \ \forall j \in J \\ y \in \mathbb{R}^J & & \\ Y \in \mathbb{R}^{J \times J} & & \end{array}$$

Let AnchRob-Lin denote the special case with linear decision rule, i.e., when $y = 0$.

Note that the constraint $|x^\delta_j - x^0_j| \leq D_j(1-h_j) \ \forall j \in J$ correctly enforces that $x^0_j = x^\delta_j$ if $h_j = 1$. We define second-stage schedule as a vector of $\mathbb{R}^J_+$ (and not $\mathbb{R}^{\overline{J}}_+$). Since x^δ is non-negative, one can set $x^\delta_s = 0$.

In the AnchRob-Aff problem the uncertainty set Δ can be convexified without changing the optimal value of the problem, as a consequence of a result from (Ben-Tal et al., 2004).

The main advantage of affine decision rules is that it leads to tractable robust problems (Ben-Tal et al., 2004). In the case of AnchRob-Aff under budgeted uncertainty, AnchRob-Aff can be reformulated as an MIP, given next in Proposition 6.2. The result is based both on the technique from (Ben-Tal et al., 2004) to write the problem under affine rules in a form similar to the static-robust problem; and also on the result of (Bertsimas and Sim, 2004) to reformulate robust problems under budgeted uncertainty.

Proposition 6.2. *The problem AnchRob-Aff admits the following MIP reformulation.*

$$
\begin{array}{lll}
(\mathrm{P_{aff}}) \quad \max & \displaystyle\sum_{i\in J} a_i h_i & \\
\text{s.t.} & h_j \in \{0,1\} & \forall j \in J \\
& x^0 \in \mathbb{R}^{\overline{J}}_+ & \\
& y \in \mathbb{R}^J & \\
& Y \in \mathbb{R}^{J\times J} & \\
& x^0_j - x^0_i \geq p_i & \forall (i,j) \in \mathcal{A} \\
& x^0_t \leq M & \\
& x^0_j + y_j - x^0_i - y_i \geq p_i + \eta_{ij}\Gamma + \sum_{k\in J} \rho_{ijk} & \forall (i,j) \in \mathcal{A} \\
& (Y_{jk} - Y_{ik} - \mathbf{1}_{i=k})\widehat{\delta}_k + \rho_{ijk} + \eta_{ij} \geq 0 & \forall (i,j) \in \mathcal{A},\, \forall k \in J \\
& \rho_{ij} \in \mathbb{R}^J_+ & \forall (i,j) \in \mathcal{A} \\
& \eta_{ij} \geq 0 & \forall (i,j) \in \mathcal{A} \\
& y_i + \eta'_i\Gamma + \sum_{k\in J} \rho'_{ik} \leq D_i(1-h_i) & \forall i \in J \\
& Y_{ik}\widehat{\delta}_k \leq \rho'_{ik} + \eta'_i & \forall i,k \in J \\
& \rho'_i \in \mathbb{R}^J_+ & \forall i \in J \\
& \eta'_i \geq 0 & \forall i \in J \\
& -y_i + \eta''_i\Gamma + \sum_{k\in J} \rho''_{ik} \leq D_i(1-h_i) & \forall i \in J \\
& Y_{ik}\widehat{\delta}_k \geq -\rho''_{ik} - \eta''_i & \forall i,k \in J \\
& \rho''_i \in \mathbb{R}^J_+ & \forall i \in J \\
& \eta''_i \geq 0 & \forall i \in J \\
& x^0_i + y_i \geq \eta'''\Gamma + \sum_{k\in J} \rho'''_k & \forall i \in J \\
& Y_{ik}\widehat{\delta}_k + \rho'''_k + \eta''' \geq 0 & \forall i,k \in J \\
& \rho''' \in \mathbb{R}^J_+ & \\
& \eta''' \geq 0 &
\end{array}
$$

Proof. Using equality $x^\delta = x^0 + y + Y\delta$, the variable x^δ can be eliminated from the recourse problem. The AnchRob-Aff problem can be written as a single maximization problem on variables x^0, h, y, Y satisfying the (infinite) set of constraints:

$$
\begin{array}{lll}
(1): & (x_j^0 + y_j + [Y\delta]_j) - (x_i^0 + y_i + [Y\delta]_i) \geq p_i + \delta_i & \forall \delta \in \Delta,\ \forall (i,j) \in \mathcal{A} \\
(2): & y_j + [Y\delta]_j \leq D_j(1 - h_j) & \forall \delta \in \Delta,\ \forall j \in J \\
(3): & -y_j - [Y\delta]_j \leq D_j(1 - h_j) & \forall \delta \in \Delta,\ \forall j \in J \\
(4): & x_j^0 + y_j + [Y\delta]_j \geq 0 & \forall \delta \in \Delta
\end{array}
$$

Consider first the constraint (1) associated to $(i,j) \in \mathcal{A}$. It holds that $(x_j^0 + y_j + [Y\delta]_j) - (x_i^0 + y_i + [Y\delta]_i) \geq p_i + \delta_i\ \forall \delta \in \Delta$ if and only if:

$$
(1'): \quad x_j^0 + y_j - x_i^0 - y_i + \min_{\delta \in \Delta} \left(\sum_{k \in J} (Y_{jk} - Y_{ik}) \delta_k - \delta_i \right) \geq p_i
$$

Since Δ is the budgeted uncertainty set $\{(\widehat{\delta}_k u_k)_{k \in J} \colon\ u \in [0,1]^J,\ \sum_{k \in J} u_k \leq \Gamma\}$, the minimization problem over Δ in (1') can be dualized, as done by Bertsimas and Sim in (Bertsimas and Sim, 2004). Let $\rho_{ij} \in \mathbb{R}_+^J$ and $\eta_{ij} \in \mathbb{R}_+$ denote the dual variables associated respectively to inequalities $u_k \leq 1\ \forall k \in J$ and $\sum_{k \in J} u_k \leq \Gamma$ appearing in the definition of Δ. Replacing the minimization over Δ by its dual, constraint (1') rewrites as

$$
\begin{array}{c}
x_j^0 + y_j - x_i^0 - y_i - \eta_{ij}\Gamma - \sum_{k \in J} \rho_{ijk} \geq p_i \\
(Y_{jk} - Y_{ik} - \mathbf{1}_{i=k})\widehat{\delta}_k + \rho_{ijk} + \eta_{ij} \geq 0 \\
\rho_{ij} \in \mathbb{R}_+^J \\
\eta_{ij} \in \mathbb{R}_+
\end{array}
$$

The very same proof technique applies to (2)-(3)-(4). The claimed reformulation follows. □

Let ($\mathrm{P}_{\mathrm{lin}}$) denote the special case of ($\mathrm{P}_{\mathrm{aff}}$) when $y = 0$.

Note that ($\mathrm{P}_{\mathrm{aff}}$) has n binary variables and $3n^2 + 4n + 3 + |\mathcal{A}|(n+1)$ continuous variables. The number of variables is independent from budget Γ.

6.5.2 Optimality of decision rules

($\mathrm{P}_{\mathrm{aff}}$) is a priori a heuristic for AnchRobPSP. However we exhibit two cases where AnchRob-Aff and AnchRobPSP coincide.

Proposition 6.3. *Under box uncertainty, AnchRob-Aff and AnchRobPSP coincide.*

Proof. Consider the solution x^+ obtained by Algorithm $\mathbf{A}^+$. By the results from Section 6.4.1, choosing x^+ as second-stage solution for every $\delta \in \Delta$, the set H^+ with maximum weight can be anchored. It is thus sufficient to prove that the collection of second-stage solutions $x^\delta = x^+$ for every $\delta \in \Delta$ can be written as an affine function of δ. It holds that $x^+ - x^0 = y + Y\delta$ with $y = x^+ - x^0$ and $Y = 0$. Hence AnchRob-Aff and AnchRobPSP coincide for box uncertainty. □

We now show that the result does not hold anymore if we restrict to linear decision rule only.

Proposition 6.4. *Under box uncertainty, there are instances where the optimal value of AnchRob-Lin is strictly less than the optimal value of AnchRobPSP.*

Proof. Let us present an example where for the optimal anchored solution (x^0, H), there is no second-stage solutions $(x^\delta)_{\delta\in\Delta}$ that can be written as $x^\delta - x^0 = Y\delta$ for given matrix Y. Consider a set of 5 jobs. The precedence graph contains arcs $(1,3),(2,3),(3,4),(3,5)$. Let $p = (1,1,1,1,2)$. Let Δ be a box uncertainty set with $\widehat{\delta} = (1,1,0,0,0)$. Let $M = 4$. With this deadline M, a baseline schedule x^0 satisfies $x^0_1 = x^0_2 = 0$, $x^0_3 = 1$, $x^0_4 \in [2,3]$ and $x^0_5 = 2$. For unit anchoring weights, the set $H^+ = \{1,2,4\}$ is the optimal anchored set returned by Algorithm $\mathbf{A}^+$. There is a unique baseline schedule associated with H^+ which is $x^0 = (0,0,1,3,2)$. In uncertainty realization δ where job 1 has duration $p_1 + \widehat{\delta}_1 = 2$, the second-stage starting time of job 3 must be $x^\delta_3 = 2$. The same holds if job 2 is disrupted, or if jobs 1 and 2 are both disrupted. Assume the linear dependency $x^\delta_3 = x^0_3 + Y\delta$. By the previous remark, the matrix Y must satisfy $2 = 1 + Y_{31}\widehat{\delta}_1$ (case job 1 disrupted), $2 = 1 + Y_{32}\widehat{\delta}_2$ (case job 2 disrupted), $2 = 1 + Y_{31}\widehat{\delta}_1 + Y_{32}\widehat{\delta}_2$ (case jobs 1 and 2 disrupted). It follows that $Y_{31} = Y_{32} = 1$ and $Y_{31} + Y_{32} = 1$ a contradiction. □

Consider now budgeted uncertainty with $\Gamma = 1$. A similar result was proven for linear programs in (Bertsimas and Goyal, 2012). We give a proof for AnchRobPSP for completeness.

Proposition 6.5. *Under budgeted uncertainty with $\Gamma = 1$, AnchRob-Aff and AnchRobPSP coincide.*

Proof. Since the set Δ can be convexified without changing the solutions of AnchRob-Aff, it is sufficient to prove the result for set $\Delta = \{(\widehat{\delta}_k u_k)_{k\in J} \colon\ u \in \{0,1\}^J,\ \sum_{k\in J} u_k \leq 1\}$ which is formed with the $n+1$ vectors: 0 and $\widehat{\delta}_i\chi^i$ for $i \in J$. Consider an optimal solution (x^0, H) to the AnchRobPSP problem, and a collection of second-stage solutions associated with (x^0, H). Consider the second-stage solution $x^\varnothing$ associated with the uncertainty realization $\delta = 0$. W.l.o.g. we can assume $x^\varnothing = x^0$. Indeed x^0 is trivially a schedule of $G(p)$ that coincides with x^0 on anchored set H. For every $i \in J$, let $x^{(i)}$ denote the second-stage solution for the uncertainty realization $\delta = \widehat{\delta}_i\chi^i$. Consider the matrix Y whose i-th row is $[Y]_i = \frac{1}{\widehat{\delta}_i}(x^{(i)} - x^0)$. It clearly satisfies $x^\delta - x^0 = Y\delta$ for every $\delta \in \Delta$. The result follows. □

An open question is whether the optimal values of AnchRobPSP and AnchRob-Aff always coincide. In the literature, there are relatively few cases where the optimality of affine decision rules has been proven (Housni and Goyal, 2019).

6.6 Numerical results

In this section, we highlight the relevance of AnchRobPSP under box or budgeted uncertainty, based on numerical experiments. We assess the numerical performances of the MIP formulation (Lay) from Section 6.2.2 and heuristic $\mathbf{A}^+$ from Section 6.4.1. In particular we compare them to MIP reformulations under affine or linear decision rules. We study the impact of parameters (budget Γ, deadline M) on solutions of the problem, and show that AnchRobPSP provides a convenient way to control the price of robustness.

6.6.1 General Settings

Instances. For evaluation purpose, different categories of instances are considered, either randomly generated or taken from the literature. Let us first describe the precedence graphs of the instances.

- In **PSP instances**, the precedence graphs are taken from the PSPLib (Kolisch and Sprecher, 1996), a benchmark for the Resource-Constrained Project Scheduling Problem. We consider 50 instances from the families j120i, with $i \in \{1, \ldots, 5\}$, which are the largest instances available ($n = 120$ jobs).
- In **ER instances**, the precedence graphs are generated as follows. The number of jobs n ranges in $\{30, 60, 100, 200\}$. Precedence graphs are generated randomly with Erdös-Rényi (**ER**) model, i.e., between each pair of jobs $i < j$ the arc (i, j) is added with probability $pr = \frac{10}{n}$. Processing times of jobs are generated uniformly in $\{1, \ldots, 20\}$.
- In **ERC instances**, the precedence graphs are the same as **ER** instances, but with modified processing times so that every job is on a **C**ritical path. Processing times of jobs are increased by repeating the two following steps until every job is on a critical path: (i) find a job i with positive margin $m = L_{G(p)}(s, t) - (L_{G(p)}(s, i) + L_{G(p)}(i, t)) > 0$ (ii) increment p_i of a random value in $\{1, \ldots, m\}$.

For all instances, the uncertainty set Δ is a budgeted uncertainty set for a given budget Γ, and values $\widehat{\delta}_i$ drawn uniformly from $[0, 0.5p_i]$. We considered 50 PSP instances, and we generated 10 ER instances and 10 ERC instances for each value of $n \in \{30, 60, 100, 200\}$, thus resulting in a total of 40 ER instances and 40 ERC instances. Tests were performed with unit anchoring weights. The deadline M is chosen between the minimum makespan $M_{\min} = L_{G(p)}(s, t)$ and the smallest deadline $M_{\max} = L_{G^{\Delta_{[J]}}}(s, t)$ for which all jobs can be anchored. When the deadline M has to be fixed, it is defined arbitrarily as a convex combination of $M_{\min}$ and $M_{\max}$. Note that the value of Γ or M used in experiments will be specified in the sequel when necessary.

Since PSP instances correspond to a reference for another problem, they have limited relevance for our purpose: namely, they feature precedence graphs with small degree. In constrast, in ER instances the expected value of the average degree in the precedence graph is driven by the constant $n \times pr$, which we arbitrarily fixed at 10. Finally ERC instances are motivated by applications to industrial projects, where it can be observed that a lot of paths are critical or almost critical in the baseline instance.

Implementation. Algorithms for AnchRobPSP were implemented with Julia 0.6.2. MIP formulations were solved using JuMP v0.18.1 and Cplex 12.8. Note that all considered MIP formulations, either (Lay) or (P_{aff}), are compact. Numerical experiments were completed on a PC with Intel Core i7-7500U CPU @ 2.70GHz 2.90GHz and 8 Go RAM.

6.6.2 Performance of formulation (Lay) and A^+ heuristic

Formulation (Lay). Let us first give numerical results of formulation (Lay) on the considered instances. We consider unit anchoring weights. In Table 6.1 and Table 6.2 are reported for ER instances and ERC instances respectively, for $n = 30, 60, 100, 200, 400, 600$ and $\Gamma = 1, 2, 3$:

- *opt*: the optimal value of the MIP reformulation;
- time: the computation time, in seconds;
- #nodes: the number of nodes explored by Cplex.

Note first that the number of anchored jobs is smaller for ERC instances then for ER instances, as a consequence from the design of ERC instances.

It can be seen that formulation (Lay) is efficient to solve instances up to 600 jobs: the worst average computation time is 14.54 seconds for ER instances, $n = 600$, $\Gamma = 2$. The number of nodes is small; in particular the problem is solved at root node for all ERC instances with budget $\Gamma \geq 3$. When the uncertainty budget increases, the size of formulation (Lay) is increased. However this does not lead to harder instances, e.g., the number of nodes for ER instances is smaller for $\Gamma = 3$ than for $\Gamma = 2$, and all ERC instances are solved at root node. These results show that formulation (Lay) is efficient to solve AnchRobPSP instances.

		n	*opt*	time (s)	#nodes
ER	$\Gamma = 1$	30	19.8	0.36	0.0
		60	47.8	0.22	0.0
		100	86.1	0.09	28.8
		200	182.1	0.18	82.9
		400	375.6	1.53	2602.1
		600	571.2	2.44	2698.4
	$\Gamma = 2$	30	17.8	0.10	0.0
		60	44.9	0.21	0.0
		100	82.6	0.09	0.0
		200	177.7	0.16	20.4
		400	367.9	2.87	1717.9
		600	561.6	14.54	6914.1
	$\Gamma = 3$	30	17.0	0.08	0.0
		60	42.9	0.08	0.0
		100	80.1	0.08	0.0
		200	172.7	0.15	0.0
		400	363.5	0.78	188.1
		600	554.4	7.86	2458.3

Table 6.1: Performance of formulation (Lay) on ER instances.

		n	*opt*	time (s)	#nodes
ERC	$\Gamma = 1$	30	11.6	0.07	0.0
		60	23.3	0.06	1.6
		100	42.6	0.10	20.2
		200	83.8	0.28	573.8
		400	169.8	1.36	3944.4
		600	271.7	10.57	5024.4
	$\Gamma = 2$	30	11.0	0.03	0.0
		60	22.1	0.29	0.0
		100	37.0	0.04	0.0
		200	71.8	0.05	0.0
		400	153.1	0.12	13.5
		600	239.7	0.24	65.7
	$\Gamma = 3$	30	10.8	0.02	0.0
		60	22.1	0.03	0.0
		100	36.0	0.04	0.0
		200	70.1	0.04	0.0
		400	150.3	0.08	0.0
		600	233.0	0.14	0.0

Table 6.2: Performance of formulation (Lay) on ERC instances.

Algorithm $\mathbf{A}^+$. Let us now evaluate Algorithm $\mathbf{A}^+$ and compare solutions to the optimal solution of AnchRobPSP computed with formulation (Lay). In Table 6.3 and Table 6.4 are reported

– $val(\mathbf{A}^+)$: the average number of anchored jobs $|H^+|$ returned by $\mathbf{A}^+$;
– gapToOpt: the average value $\frac{opt - val(\mathbf{A}^+)}{n}$.

Algorithm $\mathbf{A}^+$ runs in less than 0.01 seconds for all instances up to $n = 600$ jobs. Regarding the quality of $\mathbf{A}^+$ solution, it can be seen that the gap to optimal number of anchored jobs is small (at most 13.1% for ER instances with $n = 100$ jobs). The gap decreases when the budget increases, since $\mathbf{A}^+$ returns an optimal solution for $\Gamma = n$.

Algorithm $\mathbf{A}^+$ is thus a very fast heuristic, that outputs good quality solutions on our benchmark, especially for the largest instances and large budget.

		n	$val(\mathbf{A}^+)$	gapToOpt
ER	$\Gamma = 1$	30	16.6	10.66%
		60	40.5	12.16%
		100	73.0	13.10%
		200	161.5	10.30%
		400	347.3	7.07%
		600	532.2	6.50%
	$\Gamma = 2$	30	16.6	4.00%
		60	40.5	7.33%
		100	73.0	9.60%
		200	161.5	8.10%
		400	347.3	5.15%
		600	532.2	4.90%
	$\Gamma = 3$	30	16.6	1.33%
		60	40.5	4.00%
		100	73.0	7.10%
		200	161.5	5.60%
		400	347.3	4.05%
		600	532.2	3.70%

Table 6.3: Performance of Algorithm $\mathbf{A}^+$ on ER instances.

		n	$val(\mathbf{A}^+)$	gapToOpt
ERC	$\Gamma = 1$	30	10.8	2.66%
		60	22.1	2.00%
		100	35.6	7.00%
		200	69.8	7.00%
		400	149.1	5.17%
		600	230.2	6.91%
	$\Gamma = 2$	30	10.8	0.66%
		60	22.1	0%
		100	35.6	1.40%
		200	69.8	1.00%
		400	149.1	1.00%
		600	230.2	1.58%
	$\Gamma = 3$	30	10.8	0%
		60	22.1	0%
		100	35.6	0.40%
		200	69.8	0.15%
		400	149.1	0.30%
		600	230.2	0.46%

Table 6.4: Performance of Algorithm $\mathbf{A}^+$ on ERC instances.

Comparison with affine decision rules. Let us now evaluate the MIP formulation obtained for the problem under affine decision rules. The MIP reformulations (P_{aff}) and (P_{lin}) are solved for all instances with $n = 30, 60, 100$. The time limit is set to 600 seconds. In Tables 6.5,6.6 are reported:

– solved: the number of instances solved to optimality, out of 10;
– time: the average computation time on solved instances, in seconds;
– unsolved: the number of instances not solved to optimality but where a feasible solution is found;
– gap: the average optimality gap returned by Cplex on unsolved instances;
– noSol: the number of instances where no feasible solution is found.

Recall that all considered instances were solved in less than 1 second by formulation (Lay). By contrast, solving the MIP formulation (P_{aff}) (resp. (P_{lin})) is harder: while all $n = 30$ instances are solved to optimality, for $n = 100$ instances only 2 instances out of 60 are solved to optimality by (P_{aff}) (resp. 4 out of 60 by (P_{lin})). A difficulty encountered by the solver is to find a feasible solution: e.g., for ERC instances with $n = 100$, in the affine case, an incumbent is found only for 2 instances out of 30.

			Affine decision rule					Linear decision rule				
	n	Γ	solved	time	unsolved	gap	noSol	solved	time	unsolved	gap	noSol
ER	30	1	10	11.45	0	-	-	10	4.58	0	-	-
	30	2	10	4.39	0	-	-	10	3.75	0	-	-
	30	3	10	3.39	0	-	-	10	2.89	0	-	-
	60	1	9	191.41	1	12.44%	-	8	117.36	2	2642.88%	-
	60	2	10	169.47	0	-	-	9	222.08	1	28.62%	-
	60	3	9	103.59	1	Inf	-	9	72.38	1	27.58%	-
	100	1	-	-	4	156.52%	6	-	-	10	Inf	-
	100	2	-	-	6	234.46%	4	1	559.84	6	Inf	3
	100	3	2	603.02	5	345.14%	3	-	-	10	Inf	-

Table 6.5: Affine and linear decision rules for ER instances

			Affine decision rule					Linear decision rule				
	n	Γ	solved	time	unsolved	gap	noSol	solved	time	unsolved	gap	noSol
ERC	30	1	10	17.25	0	-	-	10	8.17	0	-	-
	30	2	10	4.98	0	-	-	10	3.16	0	-	-
	30	3	10	2.75	0	-	-	10	1.78	0	-	-
	60	1	8	292.49	2	23.3%	-	8	171.15	2	54.02%	-
	60	2	9	201.87	1	Inf	-	9	118.31	1	Inf	-
	60	3	10	186.93	0	-	-	10	72.96	0	-	-
	100	1	-	-	1	Inf	9	-	-	6	Inf	4
	100	2	-	-	1	Inf	9	1	564.91	6	Inf	3
	100	3	-	-	0	-	10	2	387.86	6	Inf	2

Table 6.6: Affine and linear decision rules for ERC instances

Linear decision rules lead to an MIP formulation with slightly less constraints and variables than affine decision rule. The performance of (P_{lin}) is thus slightly better than (P_{aff}). Namely, there are less instances with no feasible solution. However the overall performance is comparable, and instances with $n = 100$ jobs are hardly solved.

Decision rules are a priori heuristics. However on all tested instances solved to optimality, we observed that both affine and linear decision rules give optimal solutions to AnchRobPSP. This corresponds to empirical results from the literature (Housni and Goyal, 2019) that affine decision rules very often give optimal solutions. Though there is a priori no theoretical guarantee that affine decision rules are optimal for $\Gamma \geq 2$.

As a conclusion, state-of-the-art affine decision rules lead to MIP formulation that are a priori approximations to the problem. By contrast the formulation (Lay) we proposed is an exact MIP formulation. The computation time of (Lay) is an

order of magnitude lower than the computation time for affine or linear decision rules. Hence formulation (Lay) clearly outperforms formulations under decision rules for AnchRobPSP.

6.6.3 Impact of parameters on AnchRobPSP solutions

Finally let us evaluate the impact of parameters Γ and M on optimal solutions of the problem.

6.6.3.1 Impact of the uncertainty budget Γ

Consider the case where the decision maker is given the deadline M, and has to decide the value of the uncertainty budget Γ. It is clear that the higher the budget Γ, the less jobs can be anchored, i.e., the optimal value of AnchRobPSP is non-increasing with respect to Γ. We now provide experimental results to quantify the impact of Γ on the optimum.

Let us first compare the maximum number of anchored jobs $optAnch(\Gamma)$ for different values of Γ. The value of M is chosen at $\frac{3}{4}M_{\min} + \frac{1}{4}M_{\max}$. Other convex combinations have been tested and lead to similar results. Note that if $M = M_{\max}$ then $optAnch(\Gamma) = n$ for every $\Gamma \in \{1, \ldots, n\}$. We test small values of Γ, namely $\Gamma \in \{1, 2, 3\}$, and values that are proportional to the number of jobs, namely, $\Gamma \in \{\lceil 5\%n \rceil, 10\%n, 20\%n, 100\%n\}$. Note that this latter case $\Gamma = 100\%n = n$ corresponds to box uncertainty. Results are reported in Tables 6.7, 6.8.

		Budget Γ						
		1	2	3	$\lceil 5\%n \rceil$	$10\%n$	$20\%n$	$100\%n$
PSP	j1201	106.5	102.5	100.2	98.6	98.6	98.6	98.6
	j1202	101.0	99.0	99.0	99.0	99.0	99.0	99.0
	j1203	103.8	100.1	98.5	98.1	98.1	98.1	98.1
	j1204	97.0	94.0	94.0	94.0	94.0	94.0	94.0
	j1205	107.0	101.0	99.0	99.0	99.0	99.0	99.0

Table 6.7: Value $optAnch(\Gamma)$ for different values of budget Γ on PSP instances ($n = 120$), for fixed M.

		Budget Γ						
	n	1	2	3	$\lceil 5\%n \rceil$	$10\%n$	$20\%n$	$100\%n$
ER	30	19.8	17.8	17.0	17.8	17.0	16.6	16.6
	60	47.8	44.9	42.9	42.9	41.1	40.5	40.5
	100	86.1	82.6	80.1	75.5	73.1	73.0	73.0
	200	182.1	177.7	172.7	162.3	161.5	161.5	161.5
ERC	30	11.6	11.0	10.8	11.0	10.8	10.8	10.8
	60	23.3	22.1	22.1	22.1	22.1	22.1	22.1
	100	42.6	37.0	36.0	35.6	35.6	35.6	35.6
	200	83.8	71.8	70.1	69.8	69.8	69.8	69.8

Table 6.8: Value $optAnch(\Gamma)$ for different values of budget Γ on ER and ERC instances, for fixed M.

Numerical experiments show that the range between $optAnch(1)$ and $optAnch(n)$ remains small. Also, the maximum number of anchored jobs for $\Gamma = n$ is already attained for small values of Γ: on all considered instances, $optAnch(20\%n)$ is equal to $optAnch(n)$. An interpretation is that uncertainty sets with small budget contain already a large enough variety of disruptions for AnchRobPSP.

A related question is whether an optimal solution computed for some budget Γ (e.g., $\Gamma = 1$) will resist to more than Γ disruptions. Given an optimal solution $(x^{\text{opt}}, H^{\text{opt}})$ computed for budget $\Gamma = 1$, we simulate second-stage instances where Γ^{Simu} disruptions occur, with $\Gamma^{\text{Simu}} > \Gamma$, then check whether the schedule $x_H^{\text{opt}} = \left(x_i^{\text{opt}}\right)_{i \in H^{\text{opt}}}$ can be maintained. We run 1000 simulations and return the percentage of simulations where the answer is yes. Recall that it is easy to check whether the schedule x_H^{opt} can be maintained, with the condition from Lemma 6.1. Numerical results can be found in Table 6.9.

		Number of disruptions Γ^{Simu}			
	n	2	3	5	10
PSP	120	99.8%	99.5%	98.5%	93.9%
ER	30	96.8%	91.1%	74.8%	38.6%
	60	99.0%	96.8%	91.4%	70.0%
	100	99.3%	98.4%	95.1%	81.8%
	200	99.9%	99.6%	98.6%	94.0%
ERC	30	98.9%	96.8%	91.1%	76.0%
	60	99.9%	99.5%	98.2%	92.9%
	100	99.8%	99.1%	97.9%	90.2%
	200	99.8%	99.5%	98.4%	93.2%

Table 6.9: Percentage of simulations where x_H^{opt} can be maintained after Γ^{Simu} disruptions, over 1000 simulations.

We note that the percentage remains over 90% for all instance sets with up to $\Gamma^{\text{Simu}} = 3$ disruptions. Hence the starting times of anchored jobs x_H^{opt} produced by AnchRobPSP for $\Gamma = 1$ is likely to be maintained after more than Γ disruptions.

At first glance budgeted uncertainty sets with $\Gamma = 1$ may seem too optimistic, because realizations with only one disruption can be considered as relatively favorable. However the results from Table 6.9 show that the obtained solution resists well in practice to more than one disruption. Moreover Tables 6.7, 6.8 show that increasing the budget only decrease moderately the value of *optAnch*. Also *optAnch*(n) can be easily evaluated with $\mathbf{A}^+$. Hence $\Gamma = 1$ can be regarded as a good candidate for choosing the uncertainty budget for AnchRobPSP.

6.6.3.2 Trade-off between Makespan and Anchor-Robustness

In this section, the uncertainty set is considered as fixed, we set $\Gamma = 1$ for experiments. We give insights on how AnchRobPSP can be used to achieve the trade-off between makespan and number of anchored jobs.

The price of (anchor-)robustness. We first evaluate numerically the price of (anchor-)robustness. Recall that for $\alpha \in [0, 100]$, M_α is defined as the minimum worst-case makespan over the pairs (x^0, H) feasible for AnchRobPSP, and with $|H| \geq \alpha\%n$. Then the price of anchor-robustness is $PoAR_{\alpha\%} = \frac{M_\alpha}{M_{\min}}$.

Numerical results are provided in Table 6.10. They illustrate the classical fact that adjustable-robust and static-robust problems yields very different outputs in terms of price of robustness: e.g., for ER instances with $n = 30$ jobs, opting for a static-robust problem causes an increase of +28% of the makespan w.r.t. $M_{\min}$, vs. only +4.7% for the adjustable-robust problem. In between are solutions of AnchRobPSP for different values of the anchoring target α. Note that it is possible to obtain solutions with anchored jobs without increasing the price of robustness, e.g., for PSP instances, it is possible to get 50% of jobs anchored within the adjustable-robust makespan.

		Anchoring target α						
	n	0 (adjustable)	10	20	50	80	90	100 (static)
PSP	120	1.049	1.049	1.049	1.049	1.060	1.119	1.313
ER	30	1.047	1.047	1.047	1.051	1.149	1.202	1.280
	60	1.044	1.044	1.044	1.044	1.089	1.164	1.280
	100	1.037	1.037	1.037	1.037	1.052	1.129	1.293
	200	1.037	1.037	1.037	1.037	1.038	1.076	1.274
ERC	30	1.149	1.149	1.154	1.192	1.258	1.295	1.366
	60	1.167	1.167	1.171	1.197	1.266	1.304	1.379
	100	1.169	1.169	1.171	1.189	1.257	1.296	1.388
	200	1.208	1.208	1.208	1.222	1.276	1.313	1.410

Table 6.10: Price of anchor-robustness $PoAR_{\alpha\%}$ for various values of the anchoring target α, for $\Gamma = 1$.

A biobjective perspective on makespan and anchor-robustness. When solving AnchRobPSP, it might be in practice that the deadline M is dictated by exogeneous factors and given to the decision maker. If so, the value of M is fixed regardless of its impact on the anchoring criterion. Our claim is that information on the impact of M on the anchoring criterion may be of great interest for the decision maker, who could consider asking for a revision of the value of M.

By solving AnchRobPSP for various values of M we obtain a Pareto front, where Pareto-optimal solutions of AnchRobPSP are represented in the solution space for the two criteria. It is clear that the optimal value of AnchRobPSP is non-decreasing w.r.t. M. An example is presented in Figure 6.7 on an ERC instance with $n = 60$ jobs. While static-robust problem only provides one point of the Pareto front, AnchRobPSP provides a wider variety of solutions, among which the decision maker may pick its preferred one.

Examples of Gantt charts of four Pareto-optimal solutions are provided in Figure 6.8. In every Gantt chart, jobs of H are in black. Solution S_1 has makespan $M_{\min}$, hence only a few jobs can be anchored (and since it is an ERC instance, only jobs with no predecessors can be anchored). Solution S_4 is a static-robust schedule. Solutions S_2 and S_3 are other possible options. Note that the position of S_3 in the Pareto front indicates that any solution with more anchored jobs than S_3 has a makespan at least 9 units greater than the makespan of S_3.

This approach gives a new way to fix the deadline M, by choosing an anchoring target α, and setting M to the smallest value for which there exists a Pareto-optimal solution with at least $\alpha\%$ anchored jobs.

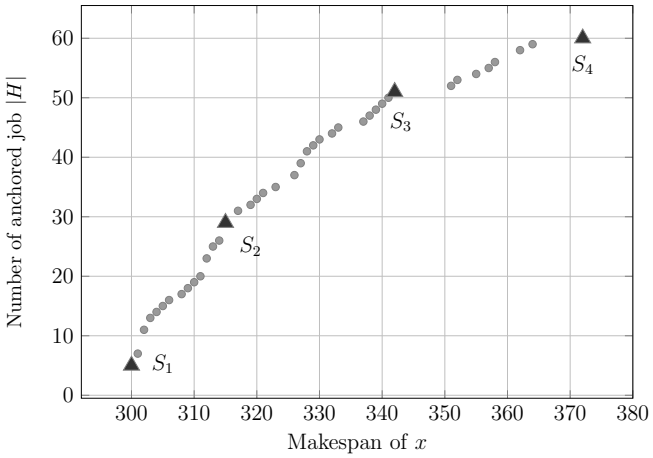


Figure 6.7: Pareto front makespan/anchoring criterion for an ERC instance with $n = 60$.

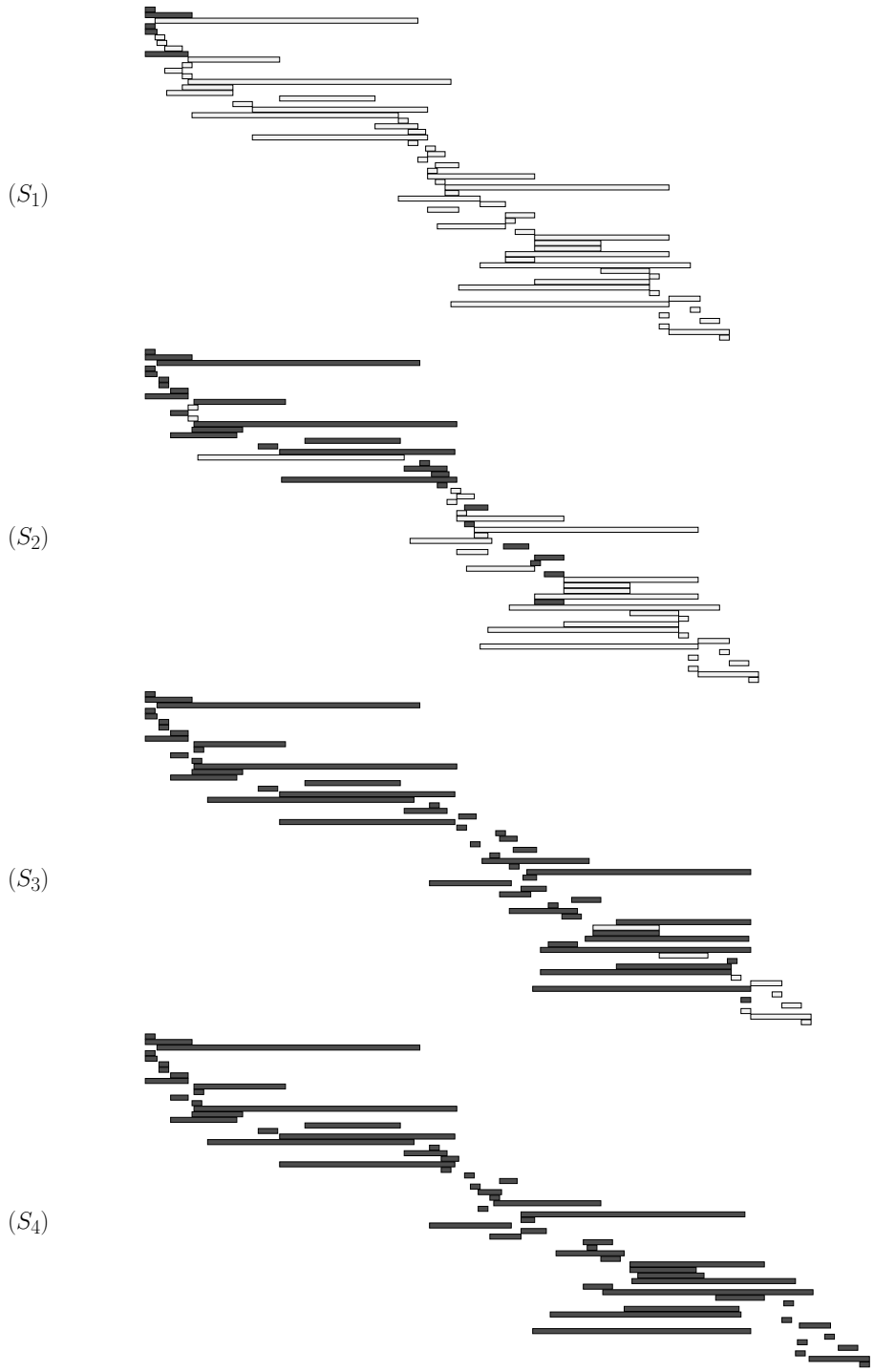


Figure 6.8: Gantt charts of four solutions from the Pareto front. Anchored jobs are in black.

Conclusion

Anchor-robustness was proposed as a concept standing between static-robust and adjustable-robust approaches, to achieve a trade-off between guarantee on starting times and cost of a solution, here corresponding to the schedule makespan. It led to the definition of the AnchRobPSP problem.

In the present work we studied the structure of AnchRobPSP by exploiting both the structure of the underlying project scheduling problem, and the properties of box or budgeted uncertainty. Based on these structural properties, we analyzed the computational complexity of the problem. AnchRobPSP was proven NP-hard for budgeted uncertainty, while the corresponding adjustable-robust problem is solvable in polynomial time. This increase in complexity stems from the binary nature of deciding whether a job is anchored or not, under the constraints imposed by the uncertainty set and the deadline. We then designed a compact MIP reformulation and algorithms. We emphasize that the proposed algorithmic tools, such as the layered graph, are dedicated to the problem. The associated formulation (Lay) appears to outperform state-of-the-art generic techniques, such as decision rules. These results are good promises for the implementability of anchor-robustness as a concept, through efficient algoritmic approaches for AnchRobPSP.

In the line of the complexity analysis of AnchRobPSP, an open question is the complexity of the variant where the values L_{ij}^{Δ} are part of the input (or computable in polynomial time) and with unit anchoring weights. Another perspective is to further investigate mixed-integer programming techniques for the problem. This is done in Part IV of the thesis, where new MIP formulations for AnchRobPSP are proposed.

Chapter 7

The Anchor-Robust RCPSP: exact and heuristic approaches

In this chapter, we study anchor-robustness for the Resource-Constrained Project Scheduling Problem (RCPSP). The RCPSP is to find a minimum makespan schedule for the jobs of a project under precedence constraints and scarce resource availability. By contrast with PERT Scheduling, the RCPSP is a classical problem that is NP-hard and computationally challenging to solve in practice (see, e.g., (Artigues et al., 2008) for a survey). The RCPSP is ubiquitous in applications, where it often appears with uncertain parameters as it can be expected from any real-life problem. We focus on uncertainty in the processing times of the jobs and propose an anchor-robust optimization approach, in the spirit of the general concepts exposed in Chapter 2.

As pointed out in (Hazır and Ulusoy, 2020), research on robust optimization for the RCPSP is rather scarce, compared to stochastic project scheduling. In (Artigues et al., 2013) the authors studied a robust problem under a min regret criterion. In (Bruni et al., 2017) the authors introduced the Adjustable-Robust RCPSP, derived from the general concept of adjustable robustness from (Ben-Tal et al., 2004). Sequencing decisions (corresponding to resource allocation decisions) are fixed before uncertainty is known; by contrast the starting times of jobs adjust to the uncertainty realization. The goal is to minimize the worst-case makespan. As noted already in Chapter 6, adjustable-robustness outputs a worst-case makespan, but it gives no schedule to decide in advance since starting times depend on the uncertainty realization. However, finding a precomputed baseline schedule is often required in applications to prepare the implementation of the schedule. This motivates the study of an approach to compute a baseline solution with bounded makespan, along with a subset of anchored decisions.

The main contributions of this chapter, corresponding to (Pass-Lanneau et al.,

2020), are the following. Anchored solutions are defined as a new concept in two-stage robust optimization for the RCPSP. The Anchor-Robust RCPSP is defined, to find a baseline schedule with bounded makespan, sequencing decisions, and a maximum number of anchored jobs. The connection with Adjustable-Robust RCPSP is pointed out. We show that these two problems can benefit from each other, in order to find both a worst-case makespan and a baseline schedule. We extend algorithmic tools from Chapter 6 and obtain a graph model and compact MIP reformulations for adjustable-robustness and anchor-robustness in a unified way. Heuristic algorithms are also proposed based on the same graph model. We investigate numerical performance of the proposed compact reformulation for the Adjustable-Robust RCPSP on instances based on the PSPLib. We point out that the overhead computational effort for solving the Adjustable-Robust RCPSP, in comparison with the deterministic RCPSP, is low. We follow on by investigating the efficiency of the proposed heuristic approaches to solve larger instances. We then provide numerical results for the Anchor-Robust RCPSP, for both MIP reformulation and heuristics. We highlight the good properties of anchor-robust solutions for finding a trade-off between makespan and guarantee of starting times.

7.1 Preliminaries

Consider an instance of the RCPSP $\mathcal{I} = ((\overline{J}, \mathcal{A}, p), (r_{ik})_{i\in J, k\in\mathcal{R}}, (R_k)_{k\in\mathcal{R}})$. It is formed with a precedence graph $(\overline{J}, \mathcal{A}, p)$, with $p \in \mathbb{R}_+^J$ the processing times of jobs, and resource requirements $(r_{ik})_{i\in J,k\in\mathcal{R}}$ and resource capacity $(R_k)_{k\in\mathcal{R}}$). Given instance $\mathcal{I}$, the RCPSP is to find a schedule $x \in \mathbb{R}_+^{\overline{J}}$ satisfying precedence constraints of $(\overline{J}, \mathcal{A}, p)$ and resource constraints, so that the makespan x_t is minimized.

Consider processing times uncertainty. Processing times have nominal values $p \in \mathbb{R}_+^J$ and their real value may be some $p + \delta$, where deviation δ is assumed to be lying in uncertainty set $\Delta \subseteq \mathbb{R}_+^J$. In this chapter we consider budgeted uncertainty. The budgeted uncertainty set associated with vector $\widehat{\delta} \in \mathbb{R}_+^J$ and integer Γ is $\Delta = \{\delta = (\widehat{\delta}_i u_i)_{i\in J} :\ u \in \{0,1\}^J,\ \sum_{i\in J} u_i \leq \Gamma\}$. If $\Gamma = 0$, set Δ is reduced to the singleton $\{0\}$, and robust approaches reduce to the deterministic RCPSP. Note that Δ is defined as a discrete set, and not a polytope; a result from (Ben-Tal et al., 2004) applicable to all robust approaches we consider, is that Δ can be convexified without changing the solutions. If $\Gamma = |J|$, the convex hull of set Δ is the *box uncertainty set* $\Pi_{i\in J}[0, \widehat{\delta}_i]$, also known as interval uncertainty set.

Let $\mathcal{I} = ((\overline{J}, \mathcal{A}, p), (r_{ik}), (R_k))$ denote the nominal RCPSP instance. Let also $\mathcal{I}^\delta = ((\overline{J}, \mathcal{A}, p+\delta), (r_{ik}), (R_k))$ be the instance where processing times are affected by deviation $\delta \in \Delta$. A first attempt to account for uncertainty is to define the Static-Robust RCPSP: given nominal instance $\mathcal{I}$ and set Δ, the problem is to find

a schedule x^+ that is feasible for $\mathcal{I}^\delta$ for every $\delta \in \Delta$, so as to minimize makespan x_t^+. The static-robust schedule x^+ is likely to have a very large makespan. Note in this case that changing the budget Γ has no impact on the price of robustness, as formalized in the following observation.

Observation 7.1. *For every budget $\Gamma \in \{1, \dots, |J|\}$, solutions of the Static-Robust RCPSP for budgeted uncertainty set $\Delta = \{(\widehat{\delta}_i u_i)_{i\in J} :\ u \in \{0,1\}^J,\ \sum_{i\in J} u_i \leq \Gamma\}$ are the same as solutions of the Static-Robust RCPSP for box uncertainty set $\Pi_{i\in J}[0, \widehat{\delta}_i]$.*

Proof. Let $\Gamma < |J|$. Clearly a static-robust solution for box uncertainty is also static-robust for Δ since $\Delta \subseteq \Pi_{i\in J}[0, \widehat{\delta}_i]$. Conversely, assume that x is feasible for uncertainty set Δ. Let $(i,j) \in \mathcal{A}$. It holds that $x_j - x_i \geq p_i + \delta_i$ for every $\delta \in \Delta$. In particular for every budget $\Gamma \geq 1$, the uncertainty realization where $\delta_i = \widehat{\delta}_i$ is in Δ. Hence $x_j - x_i \geq p_i + \widehat{\delta}_i$, and x is a schedule of $(\overline{J}, \mathcal{A}, p + \widehat{\delta})$. Since $\widehat{\delta}$ is an upper-bound on the box uncertainty set, schedule x is a static-robust solution for box uncertainty. □

We mention that this result is not specific to the RCPSP problem, but from the uncertainty being on the right-hand side of constraints. This observation motivates the study of 2-stage robust approaches, where the schedule could be adapted to uncertainty realization.

7.2 Anchor-Robust approach for the RCPSP

In this section we propose the concept of anchor-robustness for the RCPSP, based on the general concepts of Chapter 2. We first recall the flow formulation for the RCPSP, which serves as a basis for the proposed anchor-robust approach. Anchored sets are then formally defined, along with the Anchor-Robust RCPSP. Finally we highlight the connection with the Adjustable-Robust RCPSP from the literature.

7.2.1 Flow formulation for the RCPSP

Let us first present the flow formulation for the RCPSP from the literature (Artigues et al., 2003). Let $J^2 = \{(i,j), i,j \in \overline{J}, i \neq j\}$. With a schedule x satisfying resource constraints, one can associate a so-called *resource flow* f. The resource flow is a collection $(f^k)_{k\in\mathcal{R}}$ of $s{-}t$ flows in the graph $(\overline{J}, J^2)$. Flow f^k has value R_k, and the quantity flowing through every job $i \in J$ equals the resource requirement

r_{ik}. Intuitively, the flow f^k represents how units of resource k are passed from a job to another. If $f^k_{ij} > 0$ for some $k \in \mathcal{R}$, i.e., if job j uses some units of resource k after completion of job i, then an extra precedence constraint $x_j - x_i \geq p_i$ must be satisfied by schedule x. To represent such extra precedence constraints, consider vector $\sigma \in \{0,1\}^{J^2}$. Let $\mathcal{A}^\sigma = \{(i,j) \in J^2 \colon \ \sigma_{ij} = 1\}$. Vector $\sigma \in \{0,1\}^{J^2}$ is a *sequencing decision* (also called sufficient selection in the literature) associated with resource flow f if: (i) σ contains all original precedence constraints, i.e., $\mathcal{A} \subseteq \mathcal{A}^\sigma$; (ii) σ contains all arcs corresponding to non-zero flow values, i.e., $\{(i,j) \in J^2 \colon \ \exists\ k \in \mathcal{R},\ f^k_{ij} > 0\} \subseteq \mathcal{A}^\sigma$. Using the resource flow, the following formulation for the RCPSP was proposed in (Artigues et al., 2003):

$$
\begin{array}{lll}
\min & x_t & \\
\text{s.t.} & x_j - x_i \geq p_i - \overline{M}(1-\sigma_{ij}) & \forall i,j \in J^2 \quad (1)\\
& \sigma_{ij} = 1 & \forall (i,j) \in \mathcal{A} \quad (2)\\
& f^k_{ij} \leq \min\{r_{ik}, r_{jk}\}\sigma_{ij} & \forall i,j \in J^2,\ k \in \mathcal{R} \quad (3)\\
& \sum_{j:(i,j)\in J^2} f^k_{ij} = \tilde{r}_{ik} & \forall k \in \mathcal{R},\ \forall i \in J \cup \{s\} \quad (4)\\
& \sum_{j:(j,i)\in J^2} f^k_{ji} = \tilde{r}_{ik} & \forall k \in \mathcal{R},\ \forall i \in J \cup \{t\} \quad (5)\\
& f^k_{ij} \geq 0 & \forall i,j \in J^2, k \in \mathcal{R} \quad (6)\\
& \sigma_{ij} \in \{0,1\} & \forall i,j \in J^2 \quad (7)\\
& x_j \geq 0 & \forall j \in \overline{J} \quad (8)
\end{array}
$$

where $\tilde{r}_{ik} = R_k$ if $i = s$ or $i = t$, and $\tilde{r}_{ik} = r_{ik}$ otherwise, and $\overline{M}$ is an upper bound on the optimal value. Constraint (1) imposes that vector $x \in \mathbb{R}^{\overline{J}}_+$ is a schedule of $(\overline{J}, \mathcal{A}^\sigma, p)$, $\overline{M}$ being a bigM value. Remark that for $\sigma \in \{0,1\}^{J^2}$ and non-zero processing times, the existence of a schedule x of $(\overline{J}, \mathcal{A}^\sigma, p)$ implies the absence of circuits in $\mathcal{A}^\sigma$. Constraint (2) imposes that $\mathcal{A} \subseteq \mathcal{A}^\sigma$. Constraints (4)–(5) define every $(f^k_{ij})_{i,j\in J^2}$ to be a flow in the complete graph $(\overline{J}, J^2)$ and the quantity of resource k going out of s (resp. through $i \in J$) to be equal to R_k (resp. r_{ik}). Constraint (3) imposes that if a non-zero flow goes from i to j then there must be a precedence relation from i to j in σ.

An example is provided in Figure 7.1 on an instance with 5 jobs. There is one resource, with availability $R = 2$. On the left the precedence graph is represented, with precedence arcs as black arrows. The resource consumption of jobs is given into brackets in green. A feasible resource flow is represented with dotted green arrows, and each arrow corresponds to a unit of flow. Hence the sequencing decision is defined by $\mathcal{A}^\sigma = \{(s,1),(1,2),(2,t),\ (s,4),(4,2),(2,5),(5,t)\}$. On the right, a feasible schedule of $(\overline{J}, \mathcal{A}^\sigma, p)$ for $p_i = 1$ for every $i \in J$.

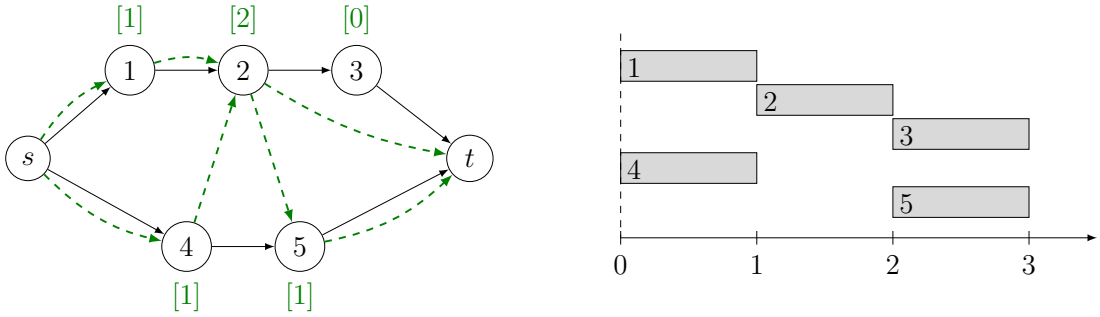


Figure 7.1: Example of resource flow for an instance of 5 jobs, and an associated feasible schedule.

Let $\mathfrak{S}$ denote the set of all sequencing decisions, i.e.,

$$\mathfrak{S} = \{\sigma \in \{0,1\}^{J^2} : \ \exists f \text{ such that } (\sigma, f) \text{ satisfies (2)–(7)}\}.$$

Importantly, flow f does not interfere directly with schedule x but only implies that σ is an element of $\mathfrak{S}$. Hence a solution of the RCPSP can be represented as a pair (x, σ) with $\sigma \in \mathfrak{S}$ and x a schedule of $(\overline{J}, \mathcal{A}^\sigma, p)$.

The flow formulation is known to have a poor linear relaxation bound, but it is compact and its size is independent from the time horizon (Koné et al., 2013). In the sequel, it will be shown that the flow formulation can be used as the underlying structure of robust formulations.

7.2.2 Anchored sets

We consider the following approach for 2-stage decisions. A baseline solution (x, σ) is chosen in first stage, feasible for the instance $\mathcal{I}$. Then, the real instance may be $\mathcal{I}^\delta$: the schedule can be changed in second stage into a new schedule x^δ of $(\overline{J}, \mathcal{A}^\sigma, p+\delta)$, with sequencing decision σ unchanged. Note that it corresponds to the idea of restricted reoptimization proposed in Section 2.2.4 of Chapter 2 to handle reoptimization of NP-hard problems. The decision maker may also want to guarantee starting times, so that the starting times of some jobs are the same in x and x^δ: this is done through the definition of anchored jobs.

Definition 7.1. *Let (z, σ) be a solution of the RCPSP instance $\mathcal{I}$. Let $\overline{H} \subseteq \overline{J}$. The set $\overline{H}$ is* anchored *w.r.t. schedule z and sequencing decision σ if for every $\delta \in \Delta$, there exists a schedule z^δ of $(\overline{J}, \mathcal{A}^\sigma, p+\delta)$ such that $z_i = z_i^\delta$ for every $i \in \overline{H}$.*

We say equivalently that $\overline{H}$ is an anchored set or a subset of anchored jobs. An anchored set $\overline{H}$ corresponds to jobs whose starting times are guaranteed in the baseline schedule z: indeed for every realization in the uncertainty set, it is possible to repair the baseline schedule z into a feasible schedule z^δ of the new

instance $(\overline{J}, \mathcal{A}^\sigma, p+\delta)$ without changing the starting times of anchored jobs. We mention that contrary to Chapter 6, the anchored set is in $\overline{J}$. This allows for a guarantee of the worst-case makespan by setting the final job t in the anchored set.

For illustrative purpose, consider again the instance and the sequencing decision of Figure 7.1. The corresponding graph $(\overline{J}, \mathcal{A}^\sigma)$ is represented in Figure 7.2 at the top. Consider the schedule $z = (0, 0, 1.5, 3, 0.5, 3, 5)$ represented at the bottom of Figure 7.2. Each job has $p_i = 1$ and $\widehat{\delta}_i = 1$. The uncertainty budget is $\Gamma = 1$. Then the set $\overline{H} = \{1, 3, 5, t\}$ is anchored w.r.t. z and σ. Indeed, for every uncertainty realization, it is possible to modify the starting times of jobs 2 and 4 and recover a feasible schedule of $(\overline{J}, \mathcal{A}^\sigma, p + \delta)$. For example if job 2 is disrupted, it is sufficient to left-shift jobs 2 and 4 of one time unit.

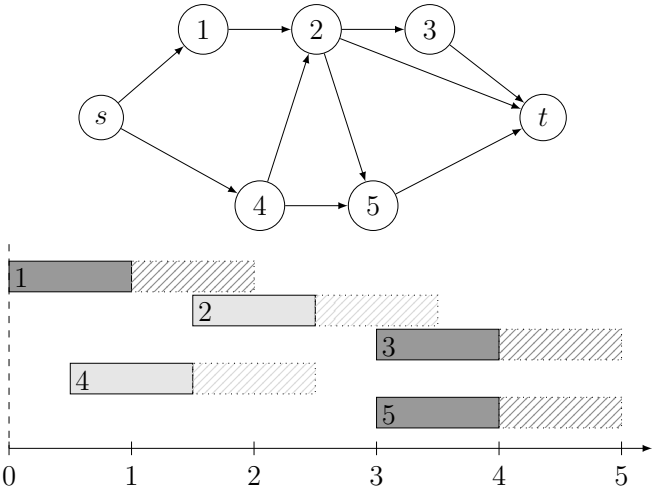


Figure 7.2: Graph $(\overline{J}, \mathcal{A}^\sigma)$ (top) and schedule z such that $\overline{H} = \{1, 3, 5, t\}$ is anchored w.r.t. z and σ when $\Gamma = 1$ (bottom).

In the sequel, we will consider *anchored solutions* given as triplets $(z, \sigma, \overline{H})$, formed with a solution (z, σ) of $\mathcal{I}$ with baseline schedule z and sequencing decision σ, and subset of jobs $\overline{H}$ anchored w.r.t. z and σ.

Static-robust solutions correspond exactly to solutions with anchored set $\overline{H} = \overline{J}$. Indeed if $\overline{J}$ is anchored w.r.t. z and σ, then z is a schedule of $(\overline{J}, \mathcal{A}^\sigma, p+\delta)$ for every $\delta \in \Delta$. Then (z, σ) is a solution of $\mathcal{I}^\delta$ for every $\delta \in \Delta$: it is a static-robust solution.

7.2.3 The Anchor-Robust RCPSP

Let us now define the optimization problem Anchor-Robust RCPSP:
Given RCPSP instance $\mathcal{I} = ((\overline{J}, \mathcal{A}, p), (r_{ik})_{i\in J, k\in\mathcal{R}}, (R_k)_{k\in\mathcal{R}})$ and a deadline $M \geq 0$, find a sequencing decision $\sigma \in \mathfrak{S}$, a baseline schedule z of $(\overline{J}, \mathcal{A}^\sigma, p)$ with makespan at most M, and a subset of jobs $H \subseteq J$, such that $H \cup \{t\}$ is anchored w.r.t. z and σ, and the number of anchored jobs $|H|$ is maximized.

The Anchor-Robust RCPSP can be seen as a robust 2-stage problem, and written as the following mathematical program:

$$\begin{array}{lll} \max & \min & \max \quad |H| \\ \sigma \in \mathfrak{S} & \delta \in \Delta & z^\delta \text{ schedule of } (\overline{J}, \mathcal{A}^\sigma, p+\delta) \\ z \text{ schedule of } (\overline{J}, \mathcal{A}^\sigma, p) & & z_i^\delta = z_i \ \forall i \in H \\ z_t \leq M & & z_t^\delta = z_t \\ H \subseteq J & & \end{array}$$

The inner min/max problem has finite value $|H|$ if the set $H \cup \{t\}$ is anchored w.r.t. z and σ; otherwise it has infinite value. The Anchor-Robust RCPSP is thus to find an anchored solution $(z, \sigma, H \cup \{t\})$ where the makespan of z is bounded by deadline M, and the size of H is maximized. Note that the final job t is forced to be in the anchored set $H \cup \{t\}$. This offers a guarantee on the worst-case makespan, in the sense that $z_t^\delta \leq M$ for every $\delta \in \Delta$.

The connection with static-robustness is as follows. There exists a solution with anchored set $\overline{H} = \overline{J}$ if and only if $M \geq M_{stat}$, where M_{stat} is the optimal value of the Static-Robust RCPSP. If $M < M_{stat}$ a solution of the Anchor-Robust RCPSP only has a subset of the jobs that are anchored, but of maximum size.

In the Anchor-Robust RCPSP, the schedule in second stage is subject to precedence constraints only since the sequencing decision $\sigma \in \mathfrak{S}$ is fixed. This is why algorithmic tools from Chapter 6 can be used.

The Anchor-Robust RCPSP provides a baseline solution where decisions are guaranteed in two ways. First, some starting times, i.e., starting times of jobs in H, are not modified in second stage by definition of anchored sets. Second, the sequencing decision σ is a first-stage decision and it is not to be modified in second stage. We emphasize that fixing sequencing decision in advance can be of practical interest. In applications, the sequencing decision may correspond, e.g., to the sequence of operations performed by workers with specific skills or equipment. They would prefer not to revise the order in which operations are performed; by contrast, it may be acceptable to adapt some starting times. From a computational point of view, sequencing decisions in the second stage would lead to very difficult optimization problems. Namely, if the sequencing decision is

in second stage, even deciding if the starting time of a job can be guaranteed, is NP-hard. This corresponds to the NP-hardness results obtained in Chapter 5.

7.2.4 Connection with the Adjustable-Robust RCPSP

An adjustable-robust approach for the RCPSP was proposed in (Bruni et al., 2017). First stage decision is the sequencing decision $\sigma \in \mathfrak{S}$; schedule x is decided in second stage. The Adjustable-Robust RCPSP writes as the following mathematical program:

$$\begin{array}{llll} \min & \max & \min & x_t^\delta \\ \sigma \in \mathfrak{S} & \delta \in \Delta & \text{s.t.} & x^\delta \text{ schedule of } (\overline{J}, \mathcal{A}^\sigma, p{+}\delta) \end{array}$$

Let $\mathcal{Q}_\Gamma(\sigma)$ denote the inner max-min problem, i.e., the worst-case makespan for sequencing decision $\sigma \in \mathfrak{S}$. The Adjustable-Robust RCPSP is then to minimize $\mathcal{Q}_\Gamma(\sigma)$ for $\sigma \in \mathfrak{S}$.

Contrary to the anchor-robust approach, the Adjustable-Robust RCPSP does not include the computation of a baseline schedule. However the Adjustable-Robust RCPSP is related to anchored sets by the following observation.

Observation 7.2. *Let $\sigma \in \mathfrak{S}$. The worst-case makespan $\mathcal{Q}_\Gamma(\sigma)$ is equal to the minimum makespan of z, where $(z, \sigma, \{t\})$ is an anchored solution.*

Proof. By definition $\mathcal{Q}_\Gamma(\sigma)$ is the minimum value M such that for every $\delta \in \Delta$, there exists a schedule x^δ of $(\overline{J}, \mathcal{A}^\sigma, p{+}\delta)$ with makespan at most M. Equivalently, it is the minimum M such that z is a schedule of $(\overline{J}, \mathcal{A}^\sigma, p)$ with $z_t = M$, and $\{t\}$ is anchored w.r.t. z and σ, by definition of anchored solutions. □

A consequence of Observation 7.2 is that there exists a solution of the Anchor-Robust RCPSP if and only if $M \geq M_{adj}$, where M_{adj} is the optimal value of the Adjustable-Robust RCPSP. The singleton $\{t\}$ can be anchored if and only if $M \geq M_{adj}$. Note that even with deadline $M = M_{adj}$, there exists solutions with anchored set $H \cup \{t\}$, $H \neq \varnothing$, as illustrated in Section 7.6. That is, it is possible to guarantee a non-trivial subset of starting times without increasing the worst-case makespan M_{adj}.

The Adjustable-Robust RCPSP and the Anchor-Robust RCPSP can benefit from each other in the following way. First the Adjustable-Robust RCPSP can be solved to determine a robust worst-case makespan M_{adj}. Then the Anchor-Robust RCPSP can be solved with deadline M_{adj} to determine schedule z^* satisfying this deadline, sequencing decision σ^*, along with an anchored set H^*.

Let us review the approaches proposed in the literature to solve the Adjustable-Robust RCPSP. In (Bruni et al., 2017) the problem was introduced and a Benders decomposition was proposed. The computation of $\mathcal{Q}_\Gamma(\sigma)$ for fixed σ appears as subproblem. The authors then focused on budgeted uncertainty and proved that $\mathcal{Q}_\Gamma(\sigma)$ can be computed in polynomial time by dynamic programming in that case. Note that it is the same problem solved in (Minoux, 2007a). The authors proposed enhancements to the Benders decomposition algorithm, evaluated on instances built upon the PSPLib. In (Bruni et al., 2018) the same authors investigated two new methods: a primal and a dual one. The dual method has the same subproblem as in (Bruni et al., 2017) but other cuts associated with dual information are added on the fly. The primal method is a column-and-constraint generation scheme inspired from (Zeng and Zhao, 2013). Computational comparison of the two decomposition approaches of (Bruni et al., 2018) and the one of (Bruni et al., 2017) is reported. In (Bold and Goerigk, 2020) a compact reformulation for the Adjustable-Robust RCPSP is proposed, independently from the present work.

7.3 Graph model and compact MIP reformulations

In this section, we extend the layered graph from Chapter 6 to characterize anchored sets and solutions of the Anchor-Robust RCPSP. Compact MIP reformulations are deduced for the Anchor-Robust RCPSP and for the Adjustable-Robust RCPSP. Heuristics based on the graph model are also proposed.

7.3.1 Layered graph

Let us propose a graph model based on the layered graph. Let Δ be the budgeted uncertainty set defined by deviation $\widehat{\delta} \in \mathbb{R}^J_+$ and budget Γ, and let nominal processing times be $p \in \mathbb{R}^J_+$.

Let $(\overline{J}, \mathcal{A}, p)$ be a precedence graph. For $\overline{H} \subseteq \overline{J}$, consider the *layered graph* $G^{\text{lay}}(\mathcal{A}, \overline{H})$. There is no difference with the definition of Chapter 6, except that there are additional vertical arcs between copies of t if $t \in \overline{H}$. Let us recall the construction of $G^{\text{lay}}(\mathcal{A}, \overline{H})$. It contains $\Gamma + 1$ *layers*, indexed from 0 to Γ. Each layer γ contains a copy of the set of jobs, denoted by i^γ, $i \in \overline{J}$. The arc-set of the layered graph contains three types of arcs. *Horizontal arcs* are arcs (i^γ, j^γ) for every γ and $(i, j) \in \mathcal{A}$, with arc-weights p_i. *Transversal arcs* are arcs $(i^{\gamma+1}, j^\gamma)$ for

every $\gamma < \Gamma$ and $(i,j) \in \mathcal{A}$, with arc-weights $p_i + \widehat{\delta}_i$. *Vertical arcs* are arcs (j^γ, j^Γ) for every $j \in \overline{H}$, $\gamma < \Gamma$, with arc-weights 0.

Note that the layered graph has all information from Δ, since the number of layers depends on the uncertainty budget Γ, and deviations $\widehat{\delta}$ appear on arc-weights of transversal arcs. Note also that if the graph $(\overline{J}, \mathcal{A})$ is acyclic, then $G^{\text{lay}}(\mathcal{A}, \overline{H})$ is acyclic. In Chapter 6 the set $\overline{H}$ was defined as a subset of J and not $\overline{J}$, but all results extend to the case of a set $\overline{H}$ containing s or t.

Let us now derive a characterization of anchored sets for the RCPSP. It follows directly from Definition 7.1 and Theorem 6.2 that

Theorem 7.1. *Let (z, σ) be a solution of the RCPSP instance $\mathcal{I}$. Let $\overline{H} \subseteq \overline{J}$. The set $\overline{H}$ is anchored w.r.t. sequencing decision σ and schedule z if and only if there exists x a schedule of $G^{\text{lay}}(\mathcal{A}^\sigma, \overline{H})$ such that $x_i^\Gamma = z_i$ for every $i \in \overline{J}$.*

Combining Observation 7.2 and Theorem 7.1, it follows that

Corollary 7.1. *For every $\sigma \in \mathfrak{S}$, the worst-case makespan $\mathcal{Q}_\Gamma(\sigma)$ is equal to the minimum value of x_t^Γ for x a schedule of $G^{\text{lay}}(\mathcal{A}^\sigma, \{t\})$.*

Hence the layered graph can be used to compute the worst-case makespan $\mathcal{Q}_\Gamma(\sigma)$, since the minimum value of x_t^Γ for x a schedule of $G^{\text{lay}}(\mathcal{A}^\sigma, \{t\})$ is equal to the length of the longest $s^\Gamma - t^\Gamma$ path in $G^{\text{lay}}(\mathcal{A}^\sigma, \{t\})$. Recall that $G^{\text{lay}}(\mathcal{A}^\sigma, \{t\})$ is acyclic when $(\overline{J}, \mathcal{A}^\sigma)$ is acyclic. Such longest path can be computed in polynomial time by dynamic programming.

7.3.2 Compact MIP reformulations

Let us now introduce new MIP reformulation for the Anchor-Robust and Adjustable-Robust RCPSP. The formulations involve the following decision variables:
– variables f and σ as in the flow formulation for the RCPSP;
– continuous variables $x_j^\gamma \geq 0$ for every $j \in \overline{J}$, $\gamma \in \{0, \ldots, \Gamma\}$;
– for the Anchor-Robust RCPSP, binary variables $h \in \{0,1\}^J$.

Theorem 7.2. *The Anchor-Robust RCPSP under budgeted uncertainty admits the following compact MIP reformulation (F_{anch}).*

$$
\begin{array}{lll}
(F_{\text{anch}}) \quad \max & \sum_{i\in J} h_i & \\
\text{s.t.} & x_j^\gamma - x_i^\gamma \geq p_i - M(1-\sigma_{ij}) & \forall i,j \in J^2, \forall \gamma \leq \Gamma \quad \text{(a)} \\
& x_j^\gamma - x_i^{\gamma+1} \geq p_i + \widehat{\delta}_i - M(1-\sigma_{ij}) & \forall i,j \in J^2, \forall \gamma < \Gamma \quad \text{(b)} \\
& x_t^\Gamma - x_t^\gamma \geq 0 & \forall \gamma < \Gamma \quad \text{(c)} \\
& x_j^\Gamma - x_j^\gamma \geq -M(1-h_j) & \forall j \in J, \forall \gamma < \Gamma \quad \text{(d)} \\
& x_t^\Gamma \leq M & \text{(e)} \\
& \sigma,\ f \text{ satisfy (2)–(5)} & \\
& f_{ij}^k \geq 0 & \forall i,j \in J^2, k \in \mathcal{R} \quad (6) \\
& \sigma_{ij} \in \{0,1\} & \forall i,j \in J^2 \quad (7) \\
& x_j^\gamma \geq 0 & \forall j \in \overline{J}, \forall \gamma \leq \Gamma \quad (8) \\
& h_j \in \{0,1\} & \forall j \in J \quad (9)
\end{array}
$$

Proof. Constraints (2)–(7) ensure that $\sigma \in \mathfrak{S}$. By Theorem 7.1, the Anchor-Robust RCPSP is to find a set H, a sequencing decision σ, and a schedule x of $G^{\text{lay}}(\mathcal{A}^\sigma, H \cup \{t\})$ such that $x_t^\Gamma \leq M$. Indeed if x is a schedule of $G^{\text{lay}}(\mathcal{A}^\sigma, H \cup \{t\})$, then x^Γ is a schedule of $(\overline{J}, \mathcal{A}^\sigma, p)$ due to horizontal arcs in layer Γ: hence x^Γ will be the baseline schedule. Let $h \in \{0,1\}^J$ be the incidence vector of set H. It remains to show that (a)–(d) correctly enforce constraints from $G^{\text{lay}}(\mathcal{A}^\sigma, H \cup \{t\})$. For $\sigma_{ij} = 1$, constraints (a) (resp. (b)) enforce horizontal arcs (resp. transversal arcs) constraints. Constraints (c) enforce vertical arc constraints between copies of t. For $h_j = 1$, constraints (d) enforce vertical arc constraints between copies of j. Hence it suffices to check that for $\sigma_{ij} = 0$ (resp. $h_j = 0$) constraints (a)–(b) (resp. constraints (d)) are valid for any schedule x. Note first that the deadline constraint (e) and vertical arc constraints (c) imply $x_t^\gamma \leq M$ for every $\gamma \leq \Gamma$. Thus $x_i^{\gamma+1} + p_i + \widehat{\delta}_i \leq M$ for every $\gamma < \Gamma$, and $x_i^\gamma + p_i \leq M$ for every $\gamma \leq \Gamma$. Hence $x_i^\gamma + p_i - x_j^\gamma \leq x_i^\gamma + p_i \leq M$, and constraint (a) is valid if $\sigma_{ij} = 0$. Similarly $x_i^{\gamma+1} + p_i + \widehat{\delta}_i - x_j^\gamma \leq x_i^{\gamma+1} + p_i + \widehat{\delta}_i \leq M$, and constraint (b) is valid if $\sigma_{ij} = 0$. Finally $x_j^\gamma - x_j^\Gamma \leq M$, and constraint (d) is valid if $h_j = 0$. □

Note that deadline M is used as a common bigM value in constraints (a), (b) and (d).

A similar result holds for the Adjustable-Robust RCPSP.

Theorem 7.3. *The Adjustable-Robust RCPSP under budgeted uncertainty admits the following compact MIP reformulation (F_{adj}), where $\overline{M}$ is an upper-bound on the optimal value.*

$$
\begin{array}{lll}
(F_{\text{adj}}) \quad \min & x_t^\Gamma & \\
\text{s.t.} & x_j^\gamma - x_i^\gamma \geq p_i - \overline{M}(1-\sigma_{ij}) & \forall i,j \in J^2, \forall \gamma \leq \Gamma \quad \text{(i)} \\
& x_j^\gamma - x_i^{\gamma+1} \geq p_i + \widehat{\delta}_i - \overline{M}(1-\sigma_{ij}) & \forall i,j \in J^2, \forall \gamma < \Gamma \quad \text{(ii)} \\
& x_t^\Gamma - x_t^\gamma \geq 0 & \forall \gamma < \Gamma \quad \text{(iii)} \\
& \sigma,\ f \text{ satisfy (2)-(5)} & \\
& f_{ij}^k \geq 0 & \forall i,j \in J^2, k \in \mathcal{R} \quad (6) \\
& \sigma_{ij} \in \{0,1\} & \forall i,j \in J^2 \quad (7) \\
& x_j^\gamma \geq 0 & \forall j \in \overline{J}, \forall \gamma \leq \Gamma \quad (8)
\end{array}
$$

Proof. Constraints (2)–(7) model that $\sigma \in \mathfrak{S}$. By Corollary 7.1 the worst-case makespan $\mathcal{Q}_\Gamma(\sigma)$ is the minimum value of x_t^Γ for x a schedule of $G^{\text{lay}}(\mathcal{A}^\sigma, \{t\})$. It suffices to show that (i)-(ii)-(iii) correctly enforce constraints from $G^{\text{lay}}(\mathcal{A}^\sigma, \{t\})$. The proof is very similar as the one of Theorem 7.2. For $\sigma_{ij} = 1$, constraints (i) (resp. (ii)) enforce horizontal arcs (resp. transversal arcs) constraints. Constraints (iii) enforce vertical arc constraints between copies of t. Hence it suffices to check that for $\sigma_{ij} = 0$ constraints (i)–(ii) are valid for any optimal schedule x. The value $\overline{M}$ is an upper bound hence $x_t^\Gamma \leq \overline{M}$ can be assumed w.l.o.g. Vertical arc constraints (iii) then imply $x_t^\gamma \leq \overline{M}$ for every $\gamma \leq \Gamma$. Thus $x_i^{\gamma+1} + p_i + \widehat{\delta}_i \leq \overline{M}$ for every $\gamma < \Gamma$, and $x_i^\gamma + p_i \leq \overline{M}$ for every $\gamma \leq \Gamma$. Hence $x_i^\gamma + p_i - x_j^\gamma \leq x_i^\gamma + p_i \leq \overline{M}$, and constraint (i) can be imposed w.l.o.g. if $\sigma_{ij} = 0$. Similarly $x_i^{\gamma+1} + p_i + \widehat{\delta}_i - x_j^\gamma \leq x_i^{\gamma+1} + p_i + \widehat{\delta}_i \leq \overline{M}$, and constraint (b) can be imposed w.l.o.g. if $\sigma_{ij} = 0$. □

Note that $\overline{M}$ can be set to $\overline{M} = \sum_{i \in J} p_i + \widehat{\delta}_i$.

The same compact reformulation for the Adjustable-Robust RCPSP was obtained independently in (Bold and Goerigk, 2020).

We emphasize that obtaining such compact reformulations is rather an exception than a general rule for robust 2-stage problems. Hence the Anchor-Robust RCPSP and the Adjustable-Robust RCPSP can be solved directly by MIP solving. Besides this formulation, the literature for the Adjustable-Robust RCPSP contains only decomposition methods based on exponential formulations. Even if such reformulations have reasonable (polynomial) size, they become difficult to solve for off-the-shelf MIP solvers when the size of the instance increases. This motivates the design of heuristic algorithms.

7.3.3 Heuristic algorithms for the Adjustable-Robust RCPSP

Let us propose a framework for designing heuristics for the Adjustable-Robust RCPSP. Consider a heuristic algorithm $\mathcal{H}$ to solve the (deterministic) RCPSP; it

is assumed that when applied to instance $\mathcal{I}$, algorithm $\mathcal{H}$ outputs a sequencing decision $\mathcal{H}(\mathcal{I}) = \widehat{\sigma} \in \mathfrak{S}$. The heuristic algorithm $\mathcal{H}$ can be used to design the following heuristic $Adj_\Gamma(\mathcal{H})$ for the Adjustable-Robust RCPSP:

Algorithm 3: Heuristic $Adj_\Gamma(\mathcal{H})$ for the Adjustable-Robust RCPSP

Data: Instance $\mathcal{I}$, set Δ with budget Γ, algorithm $\mathcal{H}$
Let $\widehat{\sigma} := \mathcal{H}(\mathcal{I})$;
Let $\mathcal{Q}_\Gamma(\widehat{\sigma})$:= longest $s^\Gamma - t^\Gamma$ path in $G^{\text{lay}}(\mathcal{A}^{\widehat{\sigma}}, \{t\})$;
return solution $\widehat{\sigma}$ with value $\mathcal{Q}_\Gamma(\widehat{\sigma})$;

Algorithm $Adj_\Gamma(\mathcal{H})$ has complexity $C_\mathcal{H} + O((|J| + |\mathcal{A}|)\Gamma)$ where $C_\mathcal{H}$ is the complexity of algorithm $\mathcal{H}$. Indeed the computation of $\mathcal{Q}_\Gamma(\widehat{\sigma})$ can be done by dynamic programming since $G^{\text{lay}}(\mathcal{A}^{\widehat{\sigma}}, \{t\})$ is acyclic. The vertex-set (resp. arc-set) of $G^{\text{lay}}(\mathcal{A}^{\widehat{\sigma}}, \{t\})$ has size $O(|J|\Gamma)$ (resp. $O(|\mathcal{A}|\Gamma)$) leading to longest path computation complexity $O((|J| + |\mathcal{A}|)\Gamma)$.

Let us now present a class of heuristic algorithms for the RCPSP to instantiate algorithm $\mathcal{H}$. They are based on Parallel Schedule Generation Scheme (Parallel SGS) and priority rules. We refer to (Kolisch and Hartmann, 1999) for details and references on heuristic algorithms for the RCPSP. In Parallel SGS, a feasible schedule is built incrementally by time incrementation. At current time, the set of eligible jobs is formed with all jobs whose predecessors in $(\overline{J}, \mathcal{A}, p)$ are scheduled and completed, and such that at current time there is enough resources available to start the job. A job is selected for the eligible set, and scheduled. If the eligible set is empty, time is incremented to the next date where a job completes.

To select a job from the eligible set, a common method is to use *priority rules*. We consider static priority rules that are computed before Parallel SGS is executed. In that case, every job is given a priority $\pi(i)$. When a job is to be selected from the eligible set, the job with highest priority $\pi(i)$ is selected. In the sequel, a total of 7 priority rules are considered, a trivial one and 6 priority rules from the literature:

- Trivial rule (ID): $\pi(i)$ is the index of job i
- Shortest Processing Time (SPT): $\pi(i) = -p_i$
- Most Total Successors (MTS): $\pi(i)$ is the number of successors in the transitive closure of $(\overline{J}, \mathcal{A})$
- Latest Finish Time (LFT): $\pi(i) = \bar{x}_i + p_i$ where $\bar{x}$ is the latest schedule of $(\overline{J}, \mathcal{A}, p)$ with minimum makespan
- Latest Starting Time (LST): $\pi(i) = \bar{x}_i$
- Minimum Slack (MSLK): $\pi(i) = -(\bar{x}_i - \underline{x}_i)$, where $\underline{x}$ is the earliest schedule of $(\overline{J}, \mathcal{A}, p)$
- Greatest Rank Positional Weight (GRPW): $\pi(i) = p_i + \sum_{j:(i,j)\in\mathcal{A}} p_j$

Importantly, ParallelSGS with priority rule π can easily be executed so that it

outputs a resource flow, and thus a sequencing decision $\widehat{\sigma}^{\pi} \in \mathfrak{S}$. This was done, e.g., in (Artigues et al., 2003). Let $\mathcal{H}^{\pi}$ denote the algorithm corresponding to Parallel SGS with priority π. Depending on the instance, it is not always the same priority rule that yields the best $\mathcal{Q}_{\Gamma}(\widehat{\sigma}^{\pi})$ value. Consider the following heuristic, denoted by BestRule:

Algorithm 4: Heuristic BestRule for the Adjustable-Robust RCPSP

Data: Instance $\mathcal{I}$, set Δ with budget Γ
for *priority rule* π *in* {*ID, SPT, MTS, LFT, LST, MSLK, GRPW*} **do**
  Let $\widehat{\sigma}^{\pi}$, $\mathcal{Q}_{\Gamma}(\widehat{\sigma}^{\pi})$ be the output of $Adj_{\Gamma}(\mathcal{H}^{\pi})$;
end
Let $\pi^* := \arg\min_{\pi} \mathcal{Q}_{\Gamma}(\widehat{\sigma}^{\pi})$;
return solution $\widehat{\sigma}^{\pi^*}$ with value $\mathcal{Q}_{\Gamma}(\widehat{\sigma}^{\pi^*})$;

Note that the selected sequencing decision is the one that gives the best worst-case makespan value for the considered uncertainty set. Namely, the output of the BestRule heuristic depends on the uncertainty budget Γ.

7.3.4 Heuristic for the Anchor-Robust RCPSP

Due to the connection between Adjustable-Robust RCPSP and Anchor-Robust RCPSP, it can be expected that solving Anchor-Robust RCPSP will be computationally challenging, and heuristics will be needed to solve medium-size instances. Note that in the MIP reformulations, the formulation for Anchor-Robust RCPSP features the same variables as that of the Adjustable-Robust RCPSP, plus additional binary variables $h \in \{0,1\}^J$. Let us now propose a MIP-based heuristic for solving the Anchor-Robust RCPSP.

Consider the case where the deadline M is computed by solving the Adjustable-Robust RCPSP in a first phase, i.e., $M = \mathcal{Q}_{\Gamma}(\widehat{\sigma})$ for some sequencing decision $\widehat{\sigma} \in \mathfrak{S}$. This can be done through the MIP formulation or a heuristic algorithm such as BestRule.

Note first that the knowledge of $\widehat{\sigma}$ readily gives a feasible solution for the Anchor-Robust RCPSP with deadline $M = \mathcal{Q}_{\Gamma}(\widehat{\sigma})$. Indeed with x the earliest schedule of $G^{\text{lay}}(\mathcal{A}^{\sigma}, \{t\})$, it holds that $(x^{\Gamma}, \widehat{\sigma}, \{t\})$ is an anchored solution with baseline schedule x^{Γ} respecting deadline $x^{\Gamma}_t \leq M$. In general, the optimal solution of the Anchor-Robust RCPSP $(z^*, \sigma^*, H^* \cup \{t\})$ can be with $\sigma^* \neq \widehat{\sigma}$. Consider the following heuristic, which is to solve the Anchor-Robust RCPSP while enforcing $\sigma^* = \widehat{\sigma}$:

Algorithm 5: Heuristic FixedSequence for the Anchor-Robust RCPSP

Data: Instance $\mathcal{I}$, set Δ with budget Γ, sequencing decision $\widehat{\sigma}$, deadline $M = \mathcal{Q}_\Gamma(\widehat{\sigma})$

Let $(x^*, H^*) := \arg\max |H|$
s.t. $H \subseteq J$, x schedule of $G^{\text{lay}}(\mathcal{A}^{\widehat{\sigma}}, H)$ with $x_t^\Gamma \leq M$;

return solution $(x^{*\Gamma}, \widehat{\sigma}, H^*)$;

The maximization step can be done by adapting the MIP from Theorem 7.2 to fixed sequencing decision $\widehat{\sigma}$. That is, (x^*, H^*) is an optimal solution to the MIP formulation

$$
\begin{array}{lll}
\max & \sum_{i \in J} h_i & \\
\text{s.t.} & x_j^\gamma - x_i^\gamma \geq p_i & \forall (i,j) \in \mathcal{A}^{\widehat{\sigma}},\ \forall \gamma \leq \Gamma \\
& x_j^\gamma - x_i^{\gamma+1} \geq p_i + \widehat{\delta}_i & \forall (i,j) \in \mathcal{A}^{\widehat{\sigma}},\ \forall \gamma < \Gamma \\
& x_t^\Gamma - x_t^\gamma \geq 0 & \forall \gamma < \Gamma \\
& x_j^\Gamma - x_j^\gamma \geq -M(1 - h_j) & \forall j \in J,\ \forall \gamma < \Gamma \\
& x_t^\Gamma \leq M & \\
& x_j^\gamma \geq 0 & \forall j \in \overline{J},\ \forall \gamma \leq \Gamma \\
& h_j \in \{0,1\} & \forall j \in J
\end{array}
$$

This is an instance of AnchRobPSP. While the problem is still NP-hard, it is easier to solve than the Anchor-Robust RCPSP because there are no sequencing decision variables. In the case where $\widehat{\sigma}$ is obtained by the BestRule heuristic, the FixedSequence heuristic is referred to as the BestRuleSequence heuristic.

In this section we proposed exact and heuristic approaches to both the Adjustable-Robust and the Anchor-Robust RCPSP. In Figure 7.3 these approaches are summarized. For each of them, we indicate the main tools involved in italic, and the reference to the corresponding section for computational results.

	Adjustable-Robust RCPSP *minimize worst-case makespan*	Anchor-Robust RCPSP *maximize the number of anchored jobs under deadline constraint*
Exact	MIP reformulation (F_{adj}) *flow formulation, layered graph* Section 7.4	MIP reformulation (F_{anch}) *flow formulation, layered graph* Section 7.6
Heuristic	BestRule (BR) *ParallelSGS, layered graph* Section 7.5	FixedSequence/BestRuleSequence (BRS) *BestRule, MIP for AnchRobPSP* Section 7.7

Figure 7.3: Proposed exact and heuristic approaches to Adjustable-Robust and Anchor-Robust RCPSP, and associated section for numerical results.

7.4 Computational results: MIP for Adjustable-Robust RCPSP

In this section numerical performances of the compact reformulation obtained in Section 7.3.2 for the Adjustable-Robust RCPSP are discussed.

7.4.1 Instances and settings

The instances are built upon RCPSP instances from the PSPLib. The number of jobs is $n \in \{30, 60, 90, 120\}$. The instances with $n = 30$ jobs are the same as (Bruni et al., 2017, 2018). For fixed n, there are 480 RCPSP instances. There are 4 resources types ($|\mathcal{R}| = 4$). The instances differ through three parameters:
– the network complexity NC $\in \{1.5, 1.8, 2.1\}$ corresponding to the average degree of jobs in the precedence graph;
– the resource factor RF $\in \{0.25, 0.50, 0.75, 1\}$ indicating the number of resources used by a job;
– the resource strength RS $\in \{0.20, 0.50, 0.70, 1\}$ quantifying the size of resource conflicts.

Regarding uncertainty, deviation is defined by $\widehat{\delta} = 0.5p$ and the uncertainty budget is $\Gamma \in \{3, 5, 7\}$. This leads to 1440 instances for each value of n. The case of $\Gamma = 0$, corresponding to deterministic RCPSP, will also be considered.

The MIP formulation (F_{adj}) has been implemented using Julia 0.6.2, with JuMP 0.18.5. It is solved with CPLEX 12.8 on a PC under Windows 10 with Intel Core i7-7500U CPU 2.90GHz and 8 Go RAM. The upper bound $\overline{M}$ is set to $\sum_{i \in J}(p_i + \widehat{\delta}_i)$. The time limit is set to 1200 seconds.

In the sequel we report averaged results. Detailed computational results can be found in the Appendix, page 235.

7.4.2 Performance of (F_{adj}) and impact of parameters on small instances

Let us first consider instances with $n = 30$ jobs, as these small instances were considered in the literature (Bruni et al., 2017, 2018). In this section computational results are presented and the impact of parameters is studied to identify the hardest instances to solve with the considered formulation.

7.4.2.1 Impact of the budget

Let us first present computational results and analyze the impact of uncertainty budget Γ. In Table 7.1 the following results are reported for $\Gamma = 3, 5, 7$:

– #solved: the number of instances solved to optimality within time limit;
– time: the computation time averaged on instances solved to optimality, in seconds;
– #unsolved: the number of instances not solved to optimality within time limit;
– gap: the final gap averaged on instances not solved to optimality.

Γ	#solved	time(s)	#unsolved	gap
3	345	39.53	135	19.26%
5	344	44.65	136	19.01%
7	337	50.98	143	19.17%
all	1026	45.01	414	19.15%

Table 7.1: Performance of (F_{adj}) depending on the budget Γ for $n = 30$ jobs.

Note first that direct implementation of the MIP reformulation allows us to solve 1026 instances over the total number of 1440 instances. The impact of the uncertainty budget appears to be limited: when Γ is increased, the number of solved instances and average computation time smoothly deteriorate.

The performance of (F_{adj}) can be compared to state-of-the-art methods from the literature, that are decomposition methods of (Bruni et al., 2017, 2018). The authors compare three methods; the Primal Method from (Bruni et al., 2018) results to be the more efficient of the three. The Primal Method solves 767 instances out of 1440 within the time limit of 1200 seconds. The average time for solved instances is 113,13 seconds. The average final gap for unsolved instances is 13,48%. Consequently, the MIP reformulation appears to be competitive with state-of-the-art decomposition methods. We acknowledge that we did not re-implement and run the decomposition algorithms from (Bruni et al., 2017, 2018). The comparison in terms of computation time of our compact reformulation and decomposition approaches is thus limited in significance. However the proposed approach is practically attractive since the implementation of a compact MIP formulation is much easier than the implementation of decomposition methods, where convergence issues may arise.

7.4.2.2 Impact of PSPLib parameters

Let us now comment on the impact of benchmark parameters NC, RF, and RS. In Table 7.2, we report for each value of parameter NC, RF and RS, the same entries as for Table 7.1 for all budgets $\Gamma = 3, 5, 7$.

		#solved	time(s)	#unsolved	gap
NC	1.5	328	42.83	152	18.78%
NC	1.8	343	36.82	137	20.00%
NC	2.1	355	54.92	125	18.65%
RF	0.25	360	6.49	0	-
RF	0.5	276	68.46	84	13.46%
RF	0.75	218	67.38	142	20.29%
RF	1	172	59.61	188	20.83%
RS	0.2	107	88.05	253	27.99%
RS	0.5	229	90.84	131	5.66%
RS	0.7	330	42.11	30	3.48%
RS	1	360	5.71	0	-

Table 7.2: Performance of (F_{adj}) depending on PSPLib parameters for $n = 30$ jobs.

The results show that parameter NC has a limited impact on the performance of the compact reformulation in terms of the number of instances solved, time and gap. By contrast, resource parameters RF and RS have an important impact, the hardest instances being for high RF and low RS. It corresponds to the instances where jobs use resources of different types (high RF) but instances are not highly disjunctive (low RS). Note that all instances with RF $= 0.25$ or RS $= 1.0$ are solved optimally; instances with RS ≥ 0.5 are solved with small final gap, 5.66% on average.

7.4.2.3 Overhead computational price of adjustable robustness

In this section, we assess numerically the overhead computational effort that is necessary to solve the Adjustable-Robust RCPSP, in comparison with the deterministic RCPSP. Our claim is that (F_{adj}) inherits from the weakness of the flow formulation for the RCPSP.

Recall that the deterministic RCPSP corresponds to the case $\Gamma = 0$, and (F_{adj}) coincides with the flow formulation for the RCPSP in that case. The 480 instances of the deterministic RCPSP are solved with the MIP formulation (F_{adj}) and $\Gamma = 0$.

In Figure 7.4, the percentage of solved instances is represented depending on computation time, for the flow formulation relative to the deterministic RCPSP, and for the formulation (F_{adj}) with budget $\Gamma = 3, 5, 7$.

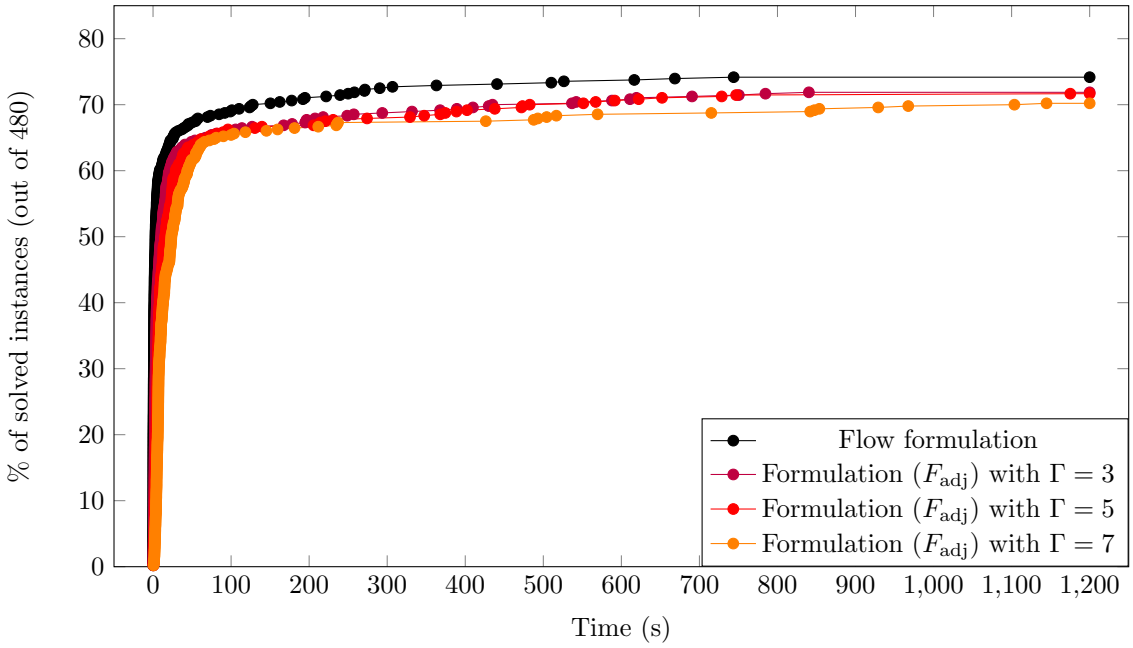


Figure 7.4: Percentage of solved instances over time, for budgets $\Gamma = 0, 3, 5, 7$.

It can be observed that the proportion of solved instances is quite similar for $\Gamma = 3, 5, 7$ or for the deterministic RCPSP. For all of them, 60% of instances are solved within 100 seconds. Then a plateau is observed, since remaining instances are hard. The overhead computation effort is represented by the offset of robust curves w.r.t. the deterministic one.

7.4.3 Performance of (F_{adj}) on larger instances

In this section we provide insights on the scalability of the MIP when the number of jobs n is increased. The analysis from Section 7.4.2 led to the identification of parameters with the most influence on the performance of the MIP. In particular instances with RF ≤ 0.5 and RS $= 1$ are the most efficiently solved for $n = 30$ jobs. The MIP is solved for instances with such parameters and a larger number of jobs ($n = 30, 60, 90$).

In Table 7.3 are reported the same entries as in Table 7.1 and a new one:
– #noSol: the number of instances where no feasible solution was found by MIP within the time limit.

n	Γ	#solved	time(s)	#unsolved	gap	#noSol
30	3	60	2.04	0	-	0
30	5	60	3.69	0	-	0
30	7	60	5.11	0	-	0
60	3	60	57.76	0	-	0
60	5	60	79.08	0	-	0
60	7	60	104.14	0	-	0
90	3	59	272.36	1	20.26%	0
90	5	54	424.80	6	25.09%	0
90	7	42	471.12	13	40.46%	5

Table 7.3: Performance of (F_{adj}) for the 60 instances with RF ≤ 0.5 and RS $= 1$.

While instances with $n = 30$ are very efficiently solved, instances with $n = 90$ with the same parameters are not all solved within the time limit of 1200 seconds. Note that for five instances with $n = 90$ and $\Gamma = 7$, the MIP solver does not find a feasible solution. For such instances, no optimality gap is available. The same behavior appears on all tested instances with $n = 120$ jobs, for which results are unreported. The results show that even for instance classes that are very easy for small size ($n = 30$), the MIP fails for medium-size instances.

7.5 Computational results: Heuristics for Adjustable-Robust RCPSP

In this section, numerical performance of heuristics for the Adjustable-Robust RCPSP is investigated.

7.5.1 Quality evaluation of priority rules

Let us first evaluate priority rules on instances with $n = 30$ jobs. A first question is the relevance of priority rules that were designed for the deterministic RCPSP; a second one is the relevance of applying the BestRule heuristic instead of only one priority rule. Let us denote by $\mathcal{Q}^*_\Gamma$ the optimal value of the Adjustable-Robust RCPSP for budget Γ. Let us denote by $\mathcal{Q}_\Gamma(\mathcal{H})$ and $\mathcal{Q}_\Gamma(BR)$ the value of heuristic $Adj_\Gamma(\mathcal{H})$ and BestRule heuristic respectively.

7.5.1.1 Correlation in quality between deterministic and robust setting

Let us address the first question: does a good heuristic $\mathcal{H}$ for the RCPSP yield a good heuristic $Adj_\Gamma(\mathcal{H})$ for the Adjustable-Robust RCPSP?

Consider the instances with $n = 30$ jobs where the optimal values $\mathcal{Q}^*_\Gamma$, $\Gamma = 0, 3, 5, 7$ have been found by solving the MIP. The question is to assess numerically the correlation between the gaps $\text{Gap}_0(\mathcal{H}) = \frac{\mathcal{Q}_0(\mathcal{H}) - \mathcal{Q}^*_0}{\mathcal{Q}_0(\mathcal{H})}$ and $\text{Gap}_\Gamma(\mathcal{H}) = \frac{\mathcal{Q}_\Gamma(\mathcal{H}) - \mathcal{Q}^*_\Gamma}{\mathcal{Q}_\Gamma(\mathcal{H})}$, i.e., the quality of $\mathcal{H}$ for the RCPSP and the quality of $Adj_\Gamma(\mathcal{H})$ for the Adjustable-Robust RCPSP.

For budgets $\Gamma = 3, 5, 7$ we consider the data points formed with the gaps $\text{Gap}_0(\mathcal{H})$ and $\text{Gap}_\Gamma(\mathcal{H})$ on all solved instances, for each heuristic $\mathcal{H}$ based on one of the 7 priority rules presented in Section 7.3.3. In Table 7.4 the Pearson correlation coefficient between these gaps, computed with Julia function `cor()`, is reported for budgets $\Gamma = 3, 5, 7$. (For each value of Γ, the number of points is thus seven times the number of instances solved for both budget Γ and budget 0.) In Figure 7.5 we represent the data points in the 2D space with first axis Gap_0 and second axis Gap_Γ.

Γ	#points	Correl. Coeff. between Gap_Γ and Gap_0
3	2408	0.9882
5	2387	0.9797
7	2345	0.9838

Table 7.4: Correlation coefficient between optimality gaps in deterministic and robust cases for instances with $n = 30$ jobs.

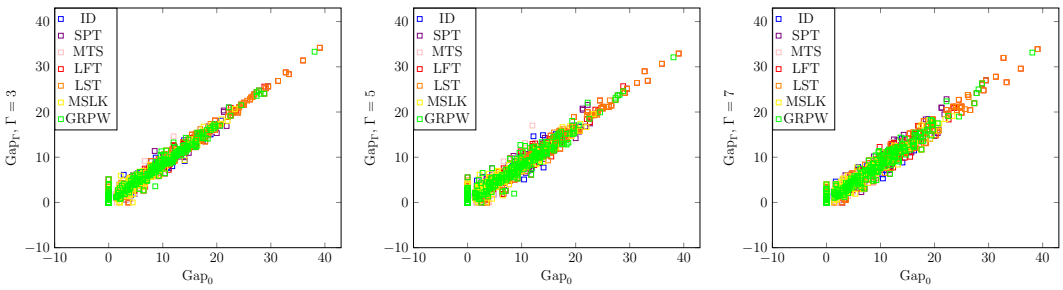


Figure 7.5: Heuristic solutions plotted in $(\text{Gap}_0, \text{Gap}_\Gamma)$ space, for $\Gamma = 3, 5, 7$, for instances with $n = 30$ jobs.

In Table 7.4 correlation coefficients are around 0.98, which indicates an important correlation between Gap_0 and Gap_Γ. Indeed uncorrelated data would lead to a coefficient correlation of 0; affinely related data would lead to a correlation

coefficient of 1. This result justifies the use of priority rules that were studied in the literature for the minimization of the makespan in the deterministic RCPSP setting.

7.5.1.2 Relevance of BestRule

We now investigate the second question, which is the relevance of choosing the best priority rule, rather than only one.
Given an instance and heuristic algorithm $\mathcal{H}$, let $\triangle$toBR denote the gap of $\mathcal{H}$ to BestRule solution, defined as $\frac{\mathcal{Q}_\Gamma(\mathcal{H})-\mathcal{Q}_\Gamma(BR)}{\mathcal{Q}_\Gamma(BR)}$. In Table 7.5 are reported for each priority rule based algorithm $\mathcal{H}$:
– #isBR: the number of instances (over 1440) for which the priority rule gives the best value, i.e., $\mathcal{Q}_\Gamma(\mathcal{H}) = \mathcal{Q}_\Gamma(BR)$;
– $\triangle$toBR avg: the average value of $\triangle$toBR on all instances;
– $\triangle$toBR max: the maximum value of $\triangle$toBR on all instances.

Priority rule	#isBR	$\triangle$toBR avg	$\triangle$toBR max
ID	703	+2.71%	+37.00%
SPT	540	+4.85%	+31.57%
MTS	959	+1.17%	+13.40%
LFT	381	+8.38%	+50.00%
LST	398	+8.32%	+50.00%
MSLK	866	+2.48%	+29.46%
GRPW	446	+6.27%	+48.27%

Table 7.5: Quality of priority rules w.r.t. BestRule on the 1440 instances with $n = 30$ jobs.

It can be observed that every priority rule is the best priority rule for some instances (at least 381 out of 1440 for LFT). Priority rules MTS and MSLK have the best results in terms of #isBR and $\triangle$toBR avg values. However, the makespan given by MTS is up to +13.4% more than the BestRule makespan for some instances, as shown by $\triangle$toBR max. The BestRule heuristic thus has a better performance in average on the benchmark.

7.5.2 Performance of BestRule heuristic

Let us now present insights on the performance of the BestRule heuristic, in terms of computation time and solution quality.
First the BestRule heuristic is evaluated on all instances, i.e., with $n = 30, 60, 90, 120$ jobs. Let BR denote the value of BestRule heuristic. Since some of these instances are difficult to solve through the MIP formulation, the linear bound Rel (i.e., the

optimal value of the continuous relaxation of the MIP) is used as a lower bound for the problem.

In Table 7.6 are given, for each value of $n = 30, 60, 90, 120$ and $\Gamma = 3, 5, 7$:
– BR time: the computation time for BestRule heuristic;
– Gap: the gap $\frac{BR-Rel}{BR}$ of BestRule to the linear bound Rel;
– Rel time: the computation time for the linear bound Rel.

n	Γ	BR time (s)	Gap $\frac{BR-Rel}{BR}$	Rel time (s)
30	3	0.11	10.33%	0.15
30	5	0.11	10.05%	0.17
30	7	0.12	10.37%	0.21
60	3	0.21	9.54%	0.58
60	5	0.27	9.24%	0.73
60	7	0.28	9.15%	0.94
90	3	0.40	9.18%	1.59
90	5	0.32	9.03%	1.87
90	7	0.48	8.97%	2.42
120	3	0.47	21.62%	4.86
120	5	0.67	21.00%	5.33
120	7	0.73	20.66%	6.50

Table 7.6: Performance of BestRule heuristic for all instances.

The computation time of the BR heuristic is below 1 second, making it applicable to large instances with up to 120 jobs. By contrast, the time for solving the continuous relaxation of the MIP with Cplex grows faster, reaching more than 5 seconds for the instances with 120 jobs. The results provide a good signal for the scalability of the BestRule heuristic, in comparison with methods based on MIP solving.

The gap between BR and the linear bound Rel is around 10% for instances with $n \leq 90$ jobs, but goes to more than 20% for $n = 120$ jobs. This gap is affected both by the quality of the BR heuristic and the weakness of the lower bound; in the sequel BestRule is compared to the optimal value of the problem, when known.

Let us now compare BestRule to solutions of (F_{adj}). We distinguish between instances solved to optimality, and instances not solved to optimality where only a lower bound is known. Let us denote by OPT the optimal value, MIPSol the value of the best solution and LB the best lower bound found by Cplex when solving (F_{adj}). The gap returned by Cplex is thus $\frac{MIPSol-LB}{MIPSol}$.

Results are reported in Table 7.7 for all $n = 30$ instances, and in Table 7.8 for instances with $n = 30, 60, 90$ and RF ≤ 0.5, RS $= 1$ for which MIP results were previously analyzed. In both tables, we report for budget $\Gamma = 3, 5, 7$:
– for all instances:
– BR gap: the gap $\frac{BR-LB}{BR}$ of BestRule to lower bound LB;
– MIP gap: the gap $\frac{MIPSol-LB}{MIPSol}$ obtained by MIP solving;
– for instances solved optimally by MIP:
– #instances: the number of such instances;
– BR gap: the gap $\frac{BR-OPT}{BR}$ of BestRule to optimal value OPT;
– #(BR=OPT): the number of instances where BestRule heuristic returns an optimal solution.
– for instances not solved optimally by MIP:
– BR gap: the gap $\frac{BR-LB}{BR}$ of BestRule to lower bound LB;
– MIP gap: the gap $\frac{MIPSol-LB}{MIPSol}$ obtained by MIP solving;
– #(BR<MIPSol) : the number of instances where BestRule heuristic returns a solution that is better than the best incumbent found by MIP;
– #noMIPSol : the number of instances where no incumbent was found by MIP.

		all instances		instances solved by MIP			instances not solved by MIP				
n	Γ	BR gap	MIP gap	#inst	BR gap	#(BR=OPT)	#inst	BR gap	MIP gap	#(BR<MIPSol)	#noMIPSol
30	3	7.42%	5.41%	341	1.49 %	201	139	21.97 %	18.71 %	17	0
30	5	7.36%	5.38%	342	1.49 %	196	138	21.89 %	18.74 %	23	0
30	7	7.54%	5.71%	335	1.42 %	199	145	21.69 %	18.90 %	27	0

Table 7.7: Performance of BestRule heuristic for instances with $n = 30$ jobs.

		all instances		instances solved by MIP			instances not solved by MIP				
n	Γ	BR gap	MIP gap	#inst	BR gap	#(BR=OPT)	#inst	BR gap	MIP gap	#(BR<MIPSol)	#noMIPSol
30	3	0.14%	0.0%	60	0.14 %	55	0	-	-	-	0
30	5	0.16%	0.0%	60	0.16 %	55	0	-	-	-	0
30	7	0.08%	0.0%	60	0.08 %	56	0	-	-	-	0
60	3	0.06%	0.0%	60	0.06 %	57	0	-	-	-	0
60	5	0.08%	0.0%	60	0.08 %	55	0	-	-	-	0
60	7	0.12%	0.0%	60	0.12 %	50	0	-	-	-	0
90	3	0.08%	0.33%	59	0.07 %	51	1	0.54 %	20.26 %	1	0
90	5	0.19%	2.50%	54	0.17 %	45	6	0.39 %	25.09 %	6	0
90	7	0.27%	9.56%	42	0.23 %	33	13	0.39 %	40.46 %	13	5

Table 7.8: Performance of BestRule heuristic for instances with RF ≤ 0.5, RS $= 1$ for $n = 30, 60, 90$ jobs.

In Table 7.7 it can be observed that the BR gap is at most 7.54% on average; BR gap is around 2% larger than the MIP gap. For instances where OPT is known, BR gap is even smaller (at most 1.49%). Notably, BestRule outputs an optimal solution on a large subset of instances: 596 out of 1440. For instances where OPT is not known, the BR gap is also close to MIP gap. On 67 instances, BestRule outputs a solution which is better than the best incumbent found by MIP solving after 1200 seconds computation.

In Table 7.8, similar results are obtained for instances with $n = 30, 60$ jobs, where OPT is known for all instances. The BR gap is very small (at most 0.14% on average), and BestRule gives an optimal solution for 328 instances out of 360. For $n = 90$ jobs, not all instances are solved by MIP. The BR gap is still very small for instances solved (at most 0.23% on average), and BestRule gives an optimal solution for 130 instances out of 180. For instances where OPT is not known, the BestRule heuristic is very efficient (BR gap $\leq 1\%$), while MIP gap is large (more than 20%). The BestRule is better than the best MIP solution on all these instances. Note also that for $\Gamma = 7$, there are 5 instances where no MIP solution was known, while BestRule does provide a solution. The good performance of BestRule on unsolved instances results into a very good performance on average on all instances: e.g., for $\Gamma = 7$, BR gap is only 0.27% versus a MIP gap of 9.56%.

7.6 Computational results: MIP for Anchor-Robust RCPSP

In this section, the compact MIP reformulation (F_{anch}) for Anchor-Robust RCPSP is evaluated. The impact of deadline M on the number of anchored jobs is also discussed.

7.6.1 Instances and settings

Let us describe the settings of numerical experiments.
The Anchor-Robust RCPSP instances are defined as follows. The instances of the RCPSP and uncertainty sets are the same as in Section 7.4. The only additional input of the Anchor-Robust RCPSP is the deadline M. As proposed in Section 7.2, the deadline M can be found by solving the Adjustable-Robust RCPSP. Since the Adjustable-Robust RCPSP is difficult to solve even for some instances with $n = 30$ jobs, we consider in the sequel that deadline M is set to the value $\widehat{M} = \mathcal{Q}_\Gamma(BR)$ of BestRule heuristic. The BestRule heuristic is thus applied on each instance to determine $\widehat{M}$. The solution found by BestRule heuristic is a sequencing decision $\widehat{\sigma}$ with worst-case makespan $\mathcal{Q}_\Gamma(\widehat{\sigma}) = \widehat{M}$. The sequencing decision $\widehat{\sigma} \in \mathfrak{S}$ is fed to (F_{anch}) as a warm-start. Note that this gives a feasible solution to the MIP solver, with $h_i = 0$ for every $i \in J$.

Computational settings are the same as in Section 7.4. The time limit for MIP solving is set to 1200 seconds. The computation times reported in this section do not include the computation time of BestRule heuristic, which is negligible w.r.t. MIP solving.

7.6.2 Performance of (F_{anch}) on small instances

Let us first assess the performance of the compact reformulation (F_{anch}) for instances with $n = 30$ jobs, and analyze the influence of parameters.

In Table 7.9 and Table 7.10 are reported, for each value of $\Gamma = 3, 5, 7$ and for each value of parameter NC, RF and RS respectively:
– Anch: the average number of anchored jobs in the best solution found by (F_{anch});
– #solved: the number of instances solved to optimality within time limit;
– time: the computation time averaged on instances solved to optimality, in seconds;
– #unsolved: the number of instances not solved to optimality within time limit;
– gap: the final gap averaged on instances not solved to optimality. It equals $\frac{UB-Anch}{Anch}$ where UB is the best upper bound found by MIP;

– Gap_n: the final gap $\frac{UB-Anch}{n}$ expressed as a percentage of the number of jobs.

Γ	Anch	#solved	time (s)	#unsolved	gap	Gap_n
3	16.09	307	32.76	173	244.86%	53.78%
5	17.92	321	25.42	159	261.78%	55.10%
7	23.01	372	34.69	108	237.08%	56.08%
all	19.01	1000	31.12	440	249.06%	54.82%

Table 7.9: Performance of (F_{anch}) for $n = 30$ jobs, w.r.t. budget Γ.

		Anch	#solved	time (s)	#unsolved	gap	Gap_n
NC	1.5	20.63	340	28.97	140	223.59%	53.53%
NC	1.8	18.76	332	27.09	148	272.88%	56.24%
NC	2.1	17.62	328	37.44	152	249.34%	54.63%
RF	0.25	21.58	359	1.83	1	25.82%	16.35%
RF	0.5	19.88	261	32.11	99	126.86%	48.33%
RF	0.75	17.82	199	55.69	161	243.28%	51.99%
RF	1	16.74	181	60.8	179	323.1%	61.17%
RS	0.2	12.67	98	36.53	262	356.03%	68.84%
RS	0.5	20.12	232	74.92	128	106.01%	38.43%
RS	0.7	21.25	310	24.66	50	54.78%	23.31%
RS	1	21.98	360	6.99	0	-	-

Table 7.10: Performance of (F_{anch}) for $n = 30$, w.r.t. PSPLib parameters.

Let us first comment on the optimal value Anch of the problem. As reported in Table 7.9, on average around 19 jobs out of 30 can be anchored in a baseline schedule within the makespan $\widehat{M}$. This highlights that a large proportion of jobs can be anchored, while respecting the worst-case makespan $\widehat{M}$ obtained in first place. The impact of the choice for deadline $\widehat{M}$ is discussed in Section 7.6.4.

Let us now comment on performance of the compact reformulation. First (F_{anch}) allows for solving 1000 instances out of 1440 to optimality. For instances that are not solved to optimality, the value Anch can be very small, which results into a very large gap (264% on average). Recall that Anch appears as denominator in the gap formula. The Gap_n value is more suitable for interpretation. The average Gap_n value is 54.82%, thus showing the poor performance of (F_{anch}) on unsolved instances.

Table 7.9 shows that the uncertainty budget has an impact on the number of instances solved: by contrast with the Adjustable-Robust RCPSP, for high budget $\Gamma = 7$, more instances are solved to optimality. This is interpreted in Section 7.6.3.

Regarding the impact of PSPLib parameters reported in Table 7.10, it appears that the same instance subsets are hard for both the Anchor-Robust RCPSP and the Adjustable-Robust RCPSP. Namely, all instances with RF = 0.25 except one are solved to optimality, and all instances with RS = 1 are solved to optimality. By contrast parameter NC has no significant impact on performance.

7.6.3 Performance of (F_{anch}) on larger instances

Let us now examine the scalability of the MIP formulation (F_{anch}) when the number of jobs n is increased. Similarly to Section 7.4.3, we restrict our attention to instances with RF ≤ 0.5 and RS $= 1$ that are efficiently solved for $n = 30$ jobs. The MIP formulation (F_{anch}) is solved for instances with RF ≤ 0.5, RS $= 1$ and $n = 30, 60, 90$ jobs. In Table 7.11 are reported the same entries as in Table 7.9.

		Anch	#solved	time (s)	#unsolved	gap	Gap_n
30	3	19.78	60	0.25	0	-	
30	5	21.28	60	0.51	0	-	
30	7	24.48	60	0.47	0	-	
60	3	38.11	57	16.79	3	6.13%	
60	5	40.38	59	25.17	1	2.77%	
60	7	44.25	59	6.00	1	4.76%	
90	3	61.68	54	54.33	6	7.46%	
90	5	65.41	59	76.28	1	4.91%	
90	7	70.61	60	76.30	0	-	

Table 7.11: Performance of (F_{anch}) for instances with RF ≤ 0.5 and RS $= 1$.

For $n = 60$ and $n = 90$ jobs, some instances are not solved to optimality within 1200 seconds. Note that for the Adjustable-Robust RCPSP, all instances with $n = 60$ jobs were solved, as reported previously in Table 7.3. Computation time for $n = 90$ instances is lower than that for the Adjustable-Robust RCPSP on the same instances. This is due to the warm-start, which gives a feasible sequencing decision matching the deadline to (F_{anch}). The warm-start prevents (F_{anch}) from finding no feasible solution.

Contrary to the Adjustable-Robust RCPSP, instances for larger Γ are solved more easily. In particular, the number of unsolved instances decreases with Γ. This can be surprising since the size of the MIP formulation increases with Γ. Note that the average number of anchored jobs Anch is higher for $\Gamma = 7$. Hence a large part of anchoring variables can be quickly set to one by the solver, and a small subset of anchoring variables is left for optimization. This yields simpler instances.

7.6.4 Impact of the deadline

Let us finally discuss the impact of the chosen deadline M. We proposed to choose as deadline the output $\widehat{M}$ of the BestRule heuristic, and we observed the number of jobs that can be anchored. The deadline may also be increased, so that more jobs can be anchored. A question is to evaluate the impact of an increase of the deadline w.r.t. the number of jobs that can be anchored.

To that end, the Anchor-Robust RCPSP is solved for deadline $\beta\widehat{M}$, with β a scaling factor ranging in $\{1.0, 1.05, 1.1,\ 1.2, 1.3\}$. Instances with RF = 0.25, RS = 1 are considered, for which the optimal value of the Anchor-Robust RCPSP is efficiently computed. In Table 7.12 are reported, for each value of the deadline $\beta\widehat{M}$, the optimal number of anchored jobs. Each line corresponds to an average over the 30 instances, for budget $\Gamma = 3$. The last line corresponds to the average for $n = 30, 60, 90$, the value being given as a percentage of anchored jobs.

	Scaling factor β applied to deadline				
n	1.0	1.05	1.1	1.2	1.3
30	18.39	21.40	24.26	28.73	30.0
60	39.36	44.60	49.10	55.56	59.9
90	59.83	67.63	74.76	83.73	89.2
all	64.47%	73.6%	81.93%	93.8%	99.64%

Table 7.12: Optimal number of anchored jobs depending on the deadline $\beta\widehat{M}$, for $\Gamma = 3$ and RF = 0.25, RS = 1.

As noted previously, a large proportion of jobs can be anchored even for deadline $\widehat{M}$: around 65% of jobs. A slight increase of the deadline, e.g., an increase of +5%, allows for an increase of +10% of the number of anchored jobs. Finally, for a large increase of the makespan (+30%) on almost all instances all jobs are anchored, meaning that deadline $1.3\widehat{M}$ is more than the makespan of a static-robust solution.

This highlights the interest of using Anchor-Robust RCPSP to find a baseline schedule, with the opportunity of tuning the deadline value. A trade-off can be found between the deadline and the number of anchored jobs.

7.7 Computational results: Heuristic for Anchor-Robust RCPSP

In this section, we evaluate the BestRuleSequence (BRS) heuristic to solve hard instances of the Anchor-Robust RCPSP.

7.7.1 Scalability of BRS heuristic

We first evaluate the performance of BRS heuristic on larger instances. The MIP for BRS is solved under the same computational settings as (F_{anch}) studied in Section 7.6. The fixed sequencing decision is $\widehat{\sigma}$ given by BestRule heuristic, and the deadline is $\widehat{M} = \mathcal{Q}_\Gamma(\widehat{\sigma})$. Recall that the MIP formulation for BRS heuristic does not have any binary variable for the sequencing decision, but only anchoring binary variables and continuous schedule variables.

In Table 7.13 are reported, for each value of $n = 30, 60, 90, 120$ and $\Gamma = 3, 5, 7$:
– AnchBRS: the average number of anchored jobs obtained by BRS;
– AnchBRS(%): AnchBRS expressed as a percentage of n;
– BR time: the average time for BestRule heuristic, in seconds;
– BRS time: the average time for BestRuleSequence heuristic, in seconds;
– Total time: the sum of BR time and BRS time, in seconds.

n	Γ	AnchBRS	AnchBRS(%)	BR time	BRS time	Total time
30	3	9.19	30.63%	0.11	0.01	0.13
30	5	11.47	38.26%	0.11	0.01	0.12
30	7	16.65	55.51%	0.12	0.01	0.13
60	3	13.01	21.69%	0.21	0.01	0.22
60	5	14.75	24.58%	0.27	0.01	0.28
60	7	19.26	32.10%	0.28	0.01	0.30
90	3	16.39	18.22%	0.40	0.01	0.42
90	5	17.72	19.69%	0.32	0.02	0.35
90	7	21.49	23.88%	0.48	0.02	0.51
120	3	12.30	10.25%	0.47	0.01	0.49
120	5	12.74	10.61%	0.67	0.02	0.69
120	7	14.75	12.29%	0.73	0.03	0.76

Table 7.13: Performance of BRS heuristic for all instances.

It can be observed first, that even if BRS is based on solving an MIP, it is solved extremely efficiently with Cplex: the order of magnitude is 10 milliseconds. The program is often solved at presolve or root node. A connection can be made with results for exact solving of the Anchor-Robust RCPSP presented in Section 7.6.

This shows that the computational effort necessary to solve Anchor-Robust RCPSP is mainly due to sequencing decision variables: for fixed sequencing decision the problem is very easily handled through MIP.

Total time (to run BestRule heuristic and find $\widehat{M}$, then run BRS heuristic to find an associated baseline schedule and anchored jobs) is less than a second on all considered instances, proving the scalability of BRS heuristic.

Finally the number of anchored jobs found with BRS is high for small instances: up to 55% for $n = 3$ and $\Gamma = 7$. It drops to 10% for instances with $n = 120$ jobs. This raises the question of evaluating BRS heuristic w.r.t. the optimal value of the Anchor-Robust RCPSP.

7.7.2 Comparison to exact solution of the Anchor-Robust RCPSP

Let us compare the value AnchBRS obtained by BRS heuristic to the optimal value of the Anchor-Robust RCPSP. Let Anch denote the best value found by MIP, and UB the best upper bound found by MIP.

Results are reported in Table 7.7 for all $n = 30$ instances, and in Table 7.8 for instances with $n = 30, 60, 90$ and RF ≤ 0.5, RS $= 1$ for which MIP results were previously analyzed. In both tables, we report for budget $\Gamma = 3, 5, 7$:

– for all instances:
  – BRS Gap$_n$: the gap $\frac{UB-AnchBRS}{n}$ of BestRuleSequence to UB;
  – MIP Gap$_n$: the gap $\frac{UB-MIPSol}{n}$ obtained by MIP solving;

– for instances solved optimally by MIP:
  – #instances: the number of such instances;
  – BRS Gap$_n$: the gap $\frac{OPT-AnchBRS}{n}$ of BestRuleSequence to OPT;
  – #(BRS=OPT): the number of instances where BestRuleSequence heuristic returns an optimal solution.

– for instances not solved optimally by MIP:
  – BRS Gap$_n$: the gap $\frac{UB-AnchBRS}{n}$ of BestRuleSequence to UB;
  – MIP Gap$_n$: the gap $\frac{UB-MIPSol}{n}$ obtained by MIP solving.

		all instances		instances solved by MIP			instances not solved by MIP		
n	Γ	BRS Gap$_n$	MIP Gap$_n$	#inst	BRS Gap$_n$	#(BRS=OPT)	#inst	BRS Gap$_n$	MIP Gap$_n$
30	3	42.39%	19.38%	307	25.38 %	0	173	72.57 %	53.78 %
30	5	39.73%	18.25%	321	21.48 %	48	159	76.57 %	55.10 %
30	7	33.81%	12.61%	372	20.51 %	136	108	79.62 %	56.08 %

Table 7.14: Performance of BRS heuristic for instances with $n = 30$ jobs.

		all instances		instances solved by MIP			instances not solved by MIP		
n	Γ	BRS Gap$_n$	MIP Gap$_n$	#inst	BRS Gap$_n$	#(BRS=OPT)	#inst	BRS Gap$_n$	MIP Gap$_n$
30	3	17.05%	0%	60	17.05 %	0	0	-	-
30	5	7.44%	0%	60	7.44 %	23	0	-	-
30	7	3.05%	0%	60	3.05 %	40	0	-	-
60	3	24.33%	0.16%	57	23.53 %	1	3	39.44 %	3.33 %
60	5	23.19%	0.02%	59	22.99 %	2	1	35.00 %	1.66 %
60	7	19.88%	0.05%	59	19.80 %	8	1	25.00 %	3.33 %
90	3	31.72%	0.42%	54	30.32 %	0	6	44.25 %	4.25 %
90	5	32.98%	0.05%	59	32.54 %	0	1	58.88 %	3.33 %
90	7	29.77%	0%	60	29.77 %	0	0	-	-

Table 7.15: Performance of BRS heuristic for instances with RF ≤ 0.5, RS $= 1$.

Table 7.14 shows that BRS Gap$_n$ is very large, e.g., 42.39% for instances with $n = 30$ jobs and $\Gamma = 3$. The gap is also large for instances solved to optimality, although the BRS heuristic provides an optimal solution to the Anchor-Robust RCPSP for 184 instances out of 1440.

Table 7.15 shows that the BRS heuristic has good performance for small instances with easy RF and RS: namely BRS Gap$_n$ is 3.05% for instances with $n = 30$ and $\Gamma = 7$. For larger instances, BRS Gap$_n$ is large although the MIP Gap$_n$ is very small.

Hence the performance in terms of solution quality of BRS heuristic is limited, except for small, easy instances, and high budget. Note that it implies the following on the Anchor-Robust RCPSP. While the worst-case makespan $\widehat{M}$ may be associated to a sequencing decision $\widehat{\sigma}$, there exists other sequence decisions respecting the deadline $\widehat{M}$, and such that many more jobs can be anchored. Hence, restricting to the sequencing decision $\widehat{\sigma}$ leads to BRS heuristic, which has poor quality but is very efficient in terms of computation time.

Conclusion

In the present work, the concept of anchor-robustness was applied to the RCPSP. The definition of anchored solutions generalizes the definition of solutions of the adjustable-robust approach. It offers a guarantee of starting times, in addition to a guarantee of the worst-case makespan, against realizations of processing times in the uncertainty set. Anchor-robustness bridges the gap between static robustness and adjustable robustness from the literature.

We extended a graph model designed for PERT scheduling and obtained results for the Anchor-Robust RCPSP and the Adjustable-Robust RCPSP, leading to both exact and heuristic approaches. Regarding exact solution approaches, we obtained compact MIP formulations for both problems. For heuristics, we showed how to benefit from efficient heuristics for the RCPSP to design efficient heuristics for Anchor-Robust and Adjustable-Robust RCPSP. Altogether this is a complete toolbox for solving the Anchor-Robust and the Adjustable-Robust RCPSP.

A numerical evaluation of the proposed tools was performed. The compact reformulation for the Adjustable-Robust RCPSP is competitive with decomposition algorithms from the literature. We studied the scalability of MIPs, and identified hard instances. It turns out that these are the same as for the RCPSP, where optimizing the resource sequencing decision is computationally challenging. To address such hard instances, the proposed heuristics are very efficient. The Anchor-Robust RCPSP can also be efficiently solved. In particular, the combination of BestRule and BestRuleSequence heuristics provides both a robust makespan and baseline schedule with anchored jobs.

An important perspective is to improve MIP formulations. As illustrated in numerical experiments, the computational difficulty mainly arises from the binary variables for the sequencing decision. A question is to improve the flow formulation for the RCPSP, e.g., with valid inequalities. This would surely improve the compact reformulations we proposed for the Adjustable-Robust and Anchor-Robust RCPSP. The efficiency of heuristics also raises the question of combining heuristics and MIP solving, beyond the mere warm-start we proposed.

Another perspective is to extend the anchor-robust approach, and include the possibility of sequencing decision revision in second stage. The sequencing decision may be adapted after uncertainty is known, but only partially so that the anchor-robust problem could remain reasonably tractable. The question of finding exact and heuristic approaches for this new problem would be an interesting research perspective.

Industrial use case

Let us now briefly discuss an industrial use case encountered at EDF, on which the anchor-robust approach was used. We first describe the applied problem at EDF. We then discuss how the concepts studied in Chapter 6 and Chapter 7 are well-suited to the industrial application. Finally we describe the decision-aiding tool that was developed at EDF, based on theoretical contributions of the thesis.

Maintenance planning at EDF as a use case

EDF operates a fleet of 56 nuclear units in France, which produce nearly 70% of the national electricity demand. A nuclear unit is regularly placed in maintenance to reload nuclear fuel. During a maintenance outage, a variety of operations are performed on the different circuits of the unit. While some outages are routine, others occur only once in the lifetime of a unit. This is the case for outages when large components must be replaced. An outage where the steam generators are replaced includes very specific operations. These are, e.g., welding operations, and radiographic weld testing to check the quality of weldings.

Maintenance operations are conducted by EDF entity responsible for nuclear units, and executed either by EDF or subcontracting entities. Monitoring the total duration of the outage is an important issue. An outage has a fixed nominal duration, planned in advance. If the outage is longer than its nominal duration, then the nuclear unit will not be back on the grid on time. EDF will thus have to substitute the corresponding production using more expensive units to fullfill the electricity demand. The availability of the nuclear fleet is a major performance criterion for the company. The problem under consideration is, for a given nuclear unit, to schedule maintenance operations of an outage within the given total duration.

Anchor-robustness for industrial planning

Let us now describe how the planning problem can be represented by a project scheduling problem, and how the anchor-robust approach is well-suited to this

use case.

RCPSP representation. The core structure of the problem resembles an instance of the RCPSP problem. Activities are the jobs to be scheduled, under precedence constraints. Resource constraints can be used to model some of the practical constraints. Some workforce constraints simply write as RCPSP constraints: e.g., there is a limited number of welders, hence the number of welding operations that can be done in parallel is bounded. There are also disjunctive constraints on the use of premises. For example, some rooms of the reactor building are small and can accommodate a limited number of workers at a time. Finally there are specific constraints due to weld testing operations. When these are under execution, some premises are made unavailable, to prevent interferences between radiographic sources and staff involved on other activities.

Uncertain activity durations. Uncertainty appears in the activity durations, since activities can be longer than expected. This is exactly processing time uncertainty as studied in previous chapters. Uncertainty representation with an uncertainty set is also well-suited for the industrial case. In particular, budgeted uncertainty appears to be well adapted, with a limited number of parameters – the worst-case activity durations – to be evaluated by practitioners. The uncertainty budget can be tuned in the decision making process, and it is also simple to apprehend. By contrast, a stochastic approach is hardly applicable, since a probability distribution over durations would be difficult to acquire. A default of budgeted uncertainty is that all jobs are considered similar. In practice there are job types that are uncorrelated, hence different budgets could be considered, e.g. one budget for welding operations, one budget for testing operations. In Chapter 8, uncertainty sets with multiple budgets are investigated.

Need for a baseline schedule. In such an industrial context a baseline schedule is highly needed in practice. Schedules are prepared months in advance, due to the project complexity. The course of activities needs to be prepared by teams. Also since some operations are executed by subcontractors, contracts should be established in advance including some schedule information. Moreover, there are some constraints that are not explicitly modelled or written, but checked manually by project managers. These can be implicit linking constraints with other units. For instance, there is some very specialized workforce implied in maintenance operations of all nuclear units. Hence their schedule information must be fixed in advance.

Bounded total duration. The baseline schedule must have bounded makespan, i.e., there is a global deadline for all maintenance operations. It corresponds to the timeslot when the unit is decoupled from the national grid. As explained previously, if the outage extends beyond the deadline, it results into high overhead

costs for EDF to match the electricity demand.

Need for robustness and anchored jobs. A question is the type of robust solutions suitable to such a practical context. An approach producing a baseline schedule is mandatory. The simplest to be thought of would be static-robustness. The static-robust approach can be considered as naturally implemented by planning managers. Margins are added if activities are likely to take longer than expected. Such a static-robust schedule would not satisfy the deadline if deviations of all activities are taken into account. By contrast, an anchor-robust approach produces a baseline satisfying the deadline, where some jobs are anchored, and others are not. In practice the need for guaranteed starting times depends on the activities. Some activities are very difficult or costly to reschedule. For example, there may be human errors if teams do not have enough time for preparation after schedule change. For subcontracted activities, the subcontractor may charge penalties if the schedule changes. By contrast, there are some activities that are easier to adjust. On EDF problem, project managers could identify jobs to anchor as a priority. That is, it was possible to set anchoring weights for an anchor-robust approach.

Discussion on fixed flow. In the presence of resources, we proposed in Chapter 7 fixed flow reoptimization. The tools initially designed for PERT scheduling were extended to the RCPSP. Beyond its computational properties, fixed flow reoptimization matches what is advisable to do in practice. Namely for weld testing, the flow corresponds to an order of operations performed by a weld testing team. It is useful to fix this order in advance, as weld testing is quite heavy to implement. There are many constraints such as interferences with other activities, marking the area before testing, stringent regulation, to name a few. For other resources types, the fixed flow assumption may be less linked to practical reality. For example, for welding operations, it is less difficult to change the order of operations performed by a team of welders.

Tools for EDF maintenance planning problem

Let us now highlight how some contributions of the thesis have been used on this industrial problem. These are mainly tools from Chapter 7, derived from Chapter 5 and Chapter 6. The algorithmic tools have been implemented by EDF optimization engineers in a decision-aiding software as a proof of concept, and applied on industrial data.

Instances of the Anchor-Robust RCPSP are solved according to the following steps:

- Resource flows are computed with ParallelSGS and the priority rules from Section 7.3.3;

- Then the Anchor-Robust RCPSP is solved for fixed flow, similarly to the FixedSequence heuristic of Section 7.3.4. The problem then reduces to the AnchRobPSP.
- The obtained instance of the AnchRobPSP is solved through heuristics, inspired from Algorithm $\mathbf{A}^+$ from Section 6.4.1.

All algorithmic techniques are heuristics, and they do not rely on an MIP solver. By contrast, numerical results from the thesis are mainly for exact approaches. However there are some learnt lessons for the practical application. For the resource flow, the results of Section 7.5 show that the considered heuristics produce good quality solutions. The results of Section 6.6.3 also help in the choice of the uncertainty budget, praising for the use of a small budget. We also emphasized the biobjective nature of the problem, with makespan and number of anchored jobs as criteria. Hence in the decision-aiding software diversified solutions were produced along the Pareto front, so that the decision maker can choose between them.

An additional feature would be to have a tool where a solution can be manually modified by project managers (e.g., moving jobs, and choosing the anchored set). This would raise new algorithmic questions. However we point out that the layered graph is a very convenient structure to re-compute a solution after manual modifications.

The anchor-robust approach, applied to this practical planning problem, helped evaluating the impact of some constraints on schedule robustness. Such a sensitivity analysis would have been difficult to made with only a static robust approach, where the schedule is bound to be feasible when all activities deviate, and thus is very conservative.

Part IV

Polyhedral approaches for the Anchor-Robust Project Scheduling Problem

Preliminaries on polyhedral approaches for AnchRobPSP

In this part, we consider the Anchor-Robust Project Scheduling Problem introduced in Chapter 6, abbreviated as AnchRobPSP. Let us give definitions and notations that will be used in the next two chapters.

Let J be the set of jobs and $\overline{J} = J \cup \{s, t\}$. Let $G = (\overline{J}, \mathcal{A})$ be the precedence graph, with source s and sink t. Let $G(p) = (\overline{J}, \mathcal{A}, p)$ be its arc-weighted version. It is assumed that there is an arc (s, i) (resp. arc (i, t)) for every job $i \in J$ without predecessor (resp. successor) in J. Let $\prec$ denote the partial order on $\overline{J}$ defined by $i \prec j$ if there exists an $i-j$ path in G. Given $i, j \in \overline{J}$, $i \prec j$, let $L_{G(p)}(i, j)$ be the length of the longest $i-j$ path in $G(p)$. In the sequel we will use the shorthand notation $L^0_{ij} = L_{G(p)}(i, j)$. The minimum makespan of a schedule of $G(p)$ is then L^0_{st}. A longest $s-t$ path is a *critical path*. The precedence graph $G(p)$ is *critical* if all $s-t$ paths in $G(p)$ are critical, i.e., they have length L^0_{st}.

The uncertainty set is $\Delta \subseteq \mathbb{R}^J_+$. For uncertainty set Δ, the worst-case longest path value is

$$L^\Delta_{ij} = \max_{\delta \in \Delta} L_{G(p+\delta)}(i, j)$$

for every $i, j \in \overline{J}$, $i \prec j$. Let $\overline{G}$ denote the transitive closure of G. Let G^Δ be the weighted version of $\overline{G}$ with arc-weight L^Δ_{ij} to every arc $i \prec j$, $j \neq t$, and L^0_{jt} for every $j \in J$. For $H \subseteq J$, let $G^\Delta[H]$ be the subgraph of G^Δ induced by $H \cup \{s, t\}$.

In Chapter 6, the AnchRobPSP problem was defined as follows. Given a schedule x of $G(p)$ and uncertainty set Δ, a subset of jobs $H \subseteq J$ is *x-anchored* if for every $\delta \in \Delta$ there exists x^δ schedule of $G(p+\delta)$ such that $x^\delta_i = x_i$ for every $i \in H$ (Definition 6.1). Given a deadline $M \geq 0$, and anchoring weights $w \in \mathbb{R}^J_+$, the Anchor-Robust Project Scheduling Problem AnchRobPSP is to find a pair (x, H) maximizing the anchoring weight $\sum_{i \in H} w_i$, with x a schedule of $G(p)$ with makespan at most M, and $H \subseteq J$ an x-anchored subset of jobs.

The combinatorial decisions of AnchRobPSP correspond to the anchored set H, rather than the baseline schedule. In particular, the objective function depends on H only. The problem can be defined alternatively using the anchored set only.

Formally, a subset $H \subseteq J$ is *anchored* if there exists a schedule x of $G(p)$ with makespan at most M such that H is x-anchored. Theorem 6.1, page 111, shows that H is anchored if and only if the longest $s-t$ path in $G^{\Delta}[H]$ is at most M. Given G^{Δ}, deadline $M \geq 0$, and anchoring weights $w \in \mathbb{R}^J_+$, the AnchRobPSP problem can thus be rewritten as:

$$\text{AnchRobPSP:} \qquad \max w(H) \text{ s.t. } L_{G^{\Delta}[H]}(s,t) \leq M.$$

Importantly, in this part we will assume that L^{Δ}_{ij} values are precomputed. That is, the weighted graph G^{Δ} is part of the input of the problem. In Chapter 8 we present uncertainty sets for which the L^{Δ}_{ij} values can be efficiently computed.

Given $H \subseteq J$, let χ^H denote the incidence vector of H. Let $\mathcal{H} = \{\chi^H : H \text{ anchored set}\}$. Let $\mathcal{Q} = \text{conv}(\mathcal{H})$ be the *anchored set polytope*. AnchRobPSP reduces to finding a max-weight anchored set, i.e., maximizing $\sum_{i \in J} w_i h_i$ for $h \in \mathcal{H}$, or equivalently, for $h \in \mathcal{Q}$. Theorem 6.1 ensures that given an anchored set H, it is easy to retrieve a baseline schedule x for which H is x-anchored. Let $\widetilde{G}_H$ denote the precedence graph obtained from $G(p)$ by introducing additional arcs (i,j), $i \in H \cup \{s\}$, $j \in H$, $i \prec j$ with arc-lengths L^{Δ}_{ij}. Then any schedule of $\widetilde{G}_H$ is a baseline schedule.

Let us give some definitions related to mixed-integer programming formulations for AnchRobPSP. Considered formulations for AnchRobPSP involve binary anchoring variables $h \in \{0,1\}^J$ to indicate if jobs are in the anchored set, and continuous variables, say $x \in \mathbb{R}^q$. A formulation for AnchRobPSP is defined by a polyhedron $\mathcal{P} \subseteq \mathbb{R}^q \times [0,1]^J$ and integrality constraints $h \in \{0,1\}^J$, so that the feasible set of the formulation is $\mathcal{P} \cap (\mathbb{R}^q \times \{0,1\}^J)$. Given $\mathcal{F} \subseteq \mathbb{R}^q \times \mathbb{R}^J$, let $\text{Proj}_h(\mathcal{F}) = \{h \in \mathbb{R}^J : \exists (x,h) \in \mathcal{F}\}$ denote its projection on h variables. A formulation is *valid for AnchRobPSP* if $\text{Proj}_h(\mathcal{P}) \cap \{0,1\}^J = \mathcal{H}$. Given two polyhedra $\mathcal{P}_1$ and $\mathcal{P}_2$, formulation associated with $\mathcal{P}_1$ is *stronger* than formulation associated with $\mathcal{P}_2$ if $\text{Proj}_h(\mathcal{P}_1) \subseteq \text{Proj}_h(\mathcal{P}_2)$. A formulation *yields a polyhedral characterization for AnchRobPSP* if $\text{Proj}_h(\mathcal{P}) = \mathcal{Q} = \text{conv}(\mathcal{H})$. Importantly, if the formulation associated with polyhedron $\mathcal{P}$ yields a polyhedral characterization for a special case of AnchRobPSP, and $\mathcal{P}$ is described by a polynomial number of inequalities, then the special case of AnchRobPSP is polynomial. Indeed AnchRobPSP can be solved by the linear program $\max \sum_{i \in J} w_i h_i$ for $(x,h) \in \mathcal{P}$.

Chapter 8

Dominance-based linear formulation for the Anchor-Robust Project Scheduling Problem

In this chapter, we further investigate exact approaches for the AnchRobPSP problem, and especially compact formulations. To the best of our knowledge, compact formulations have been scarcely encountered for robust 2-stage problems in the literature. In Chapter 6 the AnchRobPSP problem was introduced and it was proven NP-hard even for budgeted uncertainty. For budgeted uncertainty, the compact MIP formulation (Lay) was obtained. It is based on the so-called layered graph and a dedicated analysis of this special case. Formulation (Lay) is inherent to budgeted uncertainty, and thus not applicable to other uncertainty sets.

The main contribution of the chapter, as found in (Bendotti et al., 2020a), are linear formulations for AnchRobPSP that are valid for a variety of uncertainty sets including, but not restricted to, budgeted uncertainty sets. As exposed in the preliminaries, we consider as precomputed the L_{ij}^{Δ} values. We highlight that such a computation can be carried out efficiently for elaborate uncertainty sets, e.g., with several budgets. We exhibit a dominance property among schedules and derive a linear formulation from it. This dominance-based linear formulation, called (Dom), improves over a naive linear formulation. For budgeted uncertainty, (Dom) is also stronger than formulation (Lay) in special cases of interest. A polyhedral study is carried out to highlight how formulation (Dom) captures the combinatorial structure of the problem. We prove that (Dom) yields a complete polyhedral characterization in two special cases. The first one is box uncertainty. The second one is *1-disruption uncertainty*, i.e., budgeted uncertainty with $\Gamma = 1$

and uniform deviation, on critical precedence graphs. Finally numerical experiments give evidence that (Dom) performs well for budgeted uncertainty, even for instances that do not match the polyhedral characterization cases. (Dom) is also capable of solving the problem for uncertainty sets where no MIP formulation was previously investigated, e.g., in the case of several budgets. Such an uncertainty representation was motivated in the description of the industrial case, page 180.

8.1 Preliminaries on uncertainty sets

In this section we present the considered uncertainty sets: budgeted uncertainty sets and special cases, but also generalizations with several budgets.

Note first that some assumptions on Δ can be made without loss of generality. The uncertainty set can be assumed to be convex (Ben-Tal et al., 2004). The uncertainty set can also be assumed to be down-monotone, i.e., if $\delta \in \Delta$ and $\delta' \leq \delta$ then $\delta' \in \Delta$. Indeed it directly follows from the fact that if x^δ is a schedule of $G(p+\delta)$, then it is a schedule of $G(p+\delta')$ for every $\delta' \leq \delta$. In the present work, considered uncertainty sets will be polyhedra, and w.l.o.g. down-monotone.

Let us now define formally uncertainty sets of interest.

Set Δ is a *box uncertainty set* if $\Delta = \{(\delta_i)_{i\in J} \ : \ 0 \leq \delta_i \leq \widehat{\delta_i} \ \forall i \in J\}$ with $\widehat{\delta} \in \mathbb{R}^J_+$, i.e., it is a cartesian product of intervals. Then $\widehat{\delta}$ is a greatest element of Δ in the sense that $\delta \leq \widehat{\delta}$ for every $\delta \in \Delta$. Note that if Δ is any set with greatest element $\widehat{\delta}$, then w.l.o.g. it can be assumed to be down-monotone, and thus equal to the box with greatest element $\widehat{\delta}$.

Set Δ is a *budgeted uncertainty set* if $\Delta = \{(\widehat{\delta_i} u_i)_{i\in J} : u \in [0,1]^J, \ \sum_{i\in J} u_i \leq \Gamma\}$, with *deviation* $\widehat{\delta} \in \mathbb{R}^J_+$ and *uncertainty budget* $\Gamma \in \{1, \ldots, |J|\}$. Box uncertainty is the special case of budgeted uncertainty where $\Gamma = |J|$.

Set Δ is a *1-disruption uncertainty set* if it is a budgeted uncertainty set with unit budget $\Gamma = 1$ and uniform deviation, i.e., $\widehat{\delta_i} = \widehat{\delta_0}$ for every $i \in J$. Extreme points of a 1-disruption uncertainty set represent the situation where one event of fixed – possibly large – deviation $\widehat{\delta_0}$ may happen anywhere in the project. Then the processing time of one job is increased by fixed amount $\widehat{\delta_0}$.

Let us now present more elaborate uncertainty sets, built as unions or intersections of budgeted uncertainty sets.

Set Δ is a *partition-budgeted uncertainty set* if $\Delta = \{(\widehat{\delta_i} u_i)_{i\in J} : u \in [0,1]^J, \ \sum_{i\in J^k} u_i \leq \Gamma^k \ \forall k \in \{1, \ldots, m\}\}$ where $(J^1, ..., J^m)$ is a partition of J and $\Gamma^k \in \{1, \ldots, |J^k|\}$ for every $k \in \{1, \ldots, m\}$. Each group J^k is associated with its own uncertainty budget Γ^k. It holds that $\Delta = \bigcap_{1\leq k\leq m} \Delta^k$ where $\Delta^k = \{(\widehat{\delta_i} u_i)_{i\in J} : u \in [0,1]^J, \ \sum_{i\in J^k} u_i \leq \Gamma^k\}$ for every $k \in \{1, \ldots, m\}$. Distinct uncertainty budgets on

disjoint subsets of the partition are relevant when deviations of jobs from different subsets are uncorrelated. A special case of interest is to consider a partition (J^1, J^2) where jobs of the first subset J^1 are associated with small deviations but large uncertainty budget Γ^1; jobs of the second subset J^2 are associated with large deviations but small uncertainty budget Γ^2.

Set Δ is a *mixed-budgeted uncertainty set* if $\Delta = \bigcup_{1 \leq k \leq m} \Delta^k$ where Δ^k are budgeted uncertainty sets. Consider the following special case, where Δ^1 is defined by deviation $\widehat{\delta}$ and budget Γ^1, and Δ^2 is defined by deviation $\tau\widehat{\delta}$ for a given $\tau \in [0, 1]$, and budget $\Gamma^2 > \Gamma^1$. Then $\Delta = \Delta^1 \cup \Delta^2$ supports two kinds of uncertainty realizations corresponding either to a large number of small deviations (i.e., $\delta \in \Delta^2$) or a small number of large deviations (i.e., $\delta \in \Delta^1$). For uniform deviation, it holds that $\Delta^1 \subsetneq \Delta$ whenever $\tau\Gamma^2 > \Gamma^1$.

In Figure 8.1 are represented examples of such uncertainty sets in $\mathbb{R}^3$. From the left to the right, the uncertainty sets are the following. The first two are budgeted with unit deviation $\widehat{\delta}$, and budget $\Gamma = 2$ and $\Gamma = 1$ respectively. The third set is a partition-budgeted uncertainty set with groups $J^1 = \{1\}, J^2 = \{2, 3\}$ and budgets $\Gamma^1 = \Gamma^2 = 1$. The fourth set is a mixed-budgeted uncertainty set formed as the union of the budgeted uncertainty set with unit deviation $\widehat{\delta}$ and $\Gamma = 1$, and the box with deviation 0.7.

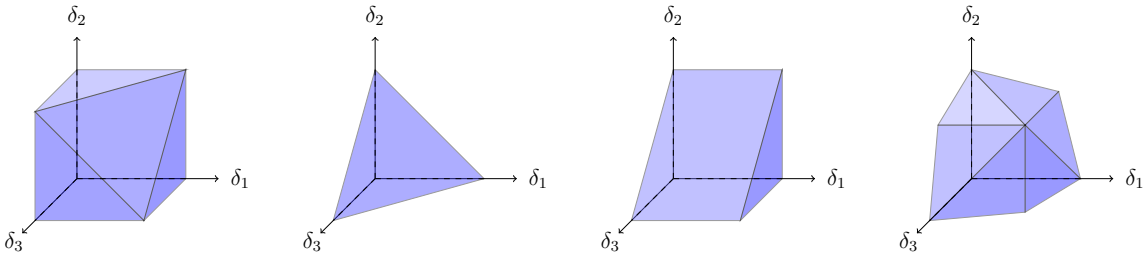


Figure 8.1: Examples of uncertainty sets. From the left to the right: budgeted with $\Gamma = 2$, budgeted with $\Gamma = 1$, partition-budgeted, mixed-budgeted.

All these uncertainty sets are defined implicitly through inequalities or set operations. Note that the uncertainty set may also be given explicitly as the convex hull of a discrete set of points. Given a set S of scenarii, for every $s \in S$ the uncertainty realization is some $\delta^s \in \mathbb{R}^J_+$, and $\Delta = \text{conv}\{\delta^s, s \in S\}$. Any implicitly-defined set Δ can be written under that form by enumerating its extreme points, but then the input size is exponentially increased.

Let us now discuss the assumption on longest paths computation for these uncertainty sets. Computing worst-case longest path values L^{Δ}_{ij} can be an NP-hard problem for polyhedral set Δ defined by inequalities, as shown in Chapter 6.

However, efficient algorithms can be designed for some uncertainty sets. For box uncertainty, the computation is straightforward since $L_{ij}^{\Delta} = L_{G(p+\widehat{\delta})}(i,j)$. For budgeted uncertainty sets, and thus 1-disruption uncertainty sets, the worst-case longest path values L_{ij}^{Δ} can be computed in polynomial time by dynamic programming (Minoux, 2009). The algorithm is linear in $\Gamma|\mathcal{A}|$.

For partition-budgeted uncertainty sets, the L_{ij}^{Δ} values can be computed using a straightforward generalization of the dynamic programming of (Minoux, 2009) which is linear in $(\Pi_{1\leq k\leq m}\Gamma^k)|\mathcal{A}|$. For mixed-budgeted uncertainty sets, the L_{ij}^{Δ} values can be easily obtained since $L_{ij}^{\Delta} = \max_{1\leq k\leq m} L_{ij}^{\Delta^k}$ for every $i,j \in \overline{J}$. For explicitly-defined uncertainty sets, values L_{ij}^{Δ} can be computed in polynomial time in the number of scenarii $|S|$, since $L_{ij}^{\Delta} = \max_{s\in S} L_{G(p+\delta^s)}(i,j)$. This relies on the property that $L_{ij}^{\Delta} = L_{ij}^{\mathrm{conv}(\Delta)}$ for any Δ.

For all considered uncertainty sets, the L_{ij}^{Δ} values can thus be computed in polynomial time. In the sequel, we consider the L_{ij}^{Δ} values as precomputed in a preprocessing step. Thus they will appear as coefficients of constraints in the proposed mixed-integer programming formulations.

8.2 Linear formulations for AnchRobPSP

In this section, we establish linear formulations for AnchRobPSP using L_{ij}^{Δ} values as coefficients. A naive formulation is first given as a benchmark. The main dominance property is proven, and formulation (Dom) is derived from it. Formulation (Dom) is then compared to other formulations.

8.2.1 A naive formulation

Consider schedule continuous variables x_j, $j \in \overline{J}$, and anchoring binary variables $h_j \in \{0,1\}$, $j \in J$. Vector h is the incidence vector of the anchored set H. A formulation for AnchRobPSP requires constraints to enforce that H is an x-anchored set. The characterization of Theorem 6.1 suggests a quadratic constraint

$$x_j - x_i \geq L_{ij}^{\Delta} h_i h_j \qquad \forall i,j \in \overline{J}, i \prec j \tag{1}$$

to represent precedence constraints of the graph $\widetilde{G}_H$. Note that for dummy jobs s and t there is no h_j decision variable but we set $h_s = 1$ and $h_t = 0$ for the ease of notation. For validity, it is sufficient to check that for each $i \prec j$, if $h_i = 0$ or $h_j = 0$ then inequality (1) is valid. Indeed it reduces to $x_j - x_i \geq 0$ which holds for every $i \prec j$ for every schedule x of $G(p)$.

Applying a standard linearization technique, a linear formulation (Std) can be obtained by replacing constraint (1) by the following linear inequality (2)

$$\begin{array}{lll} \text{(Std):} \quad \max & \sum_{i\in J} w_i h_i & \\ \text{s.t.} & x_j - x_i \geq p_i & \forall (i,j) \in \mathcal{A} \\ & x_t \leq M & \\ & x_j - x_i \geq L^{\Delta}_{ij}(h_i + h_j - 1) & \forall i,j \in \overline{J}, i \prec j \\ & x_j \geq 0 & \forall j \in \overline{J} \\ & h_j \in \{0,1\} & \forall j \in J \end{array} \tag{2}$$

It is easy to check that when $h_i = 0$ or $h_j = 0$, (2) induces a valid constraint.

8.2.2 A dominance property

Let $X^{\leq M}$ denote the set of schedules of $G(p)$ with makespan at most M. Let $H \subseteq J$ be a subset of jobs. Let us define the set of all baseline schedules x such that (x, H) is feasible for AnchRobPSP, i.e., $X^{\leq M}(H) = \{x \in X^{\leq M} : \ H \text{ is } x\text{-anchored}\}$. Note that AnchRobPSP problem is to maximize the weight of a set H such that $X^{\leq M}(H) \neq \varnothing$. By Theorem 6.1,

$$X^{\leq M}(H) = \{x \in X^{\leq M} : \quad x_j - x_i \geq L^{\Delta}_{ij} \quad \forall i \in H \cup \{s\}, j \in H, i \prec j\}.$$

Let us now define a set of baseline schedules where the same inequality is imposed, but on pairs i, j with $i \in J \cup \{s\}$:

$$Z^{\leq M}(H) = \{z \in X^{\leq M} : \quad z_j - z_i \geq L^{\Delta}_{ij} \quad \forall i \in J \cup \{s\}, j \in H, i \prec j\}.$$

Theorem 8.1. *Set $Z^{\leq M}(H)$ is dominant, in the sense that $Z^{\leq M}(H) \subseteq X^{\leq M}(H)$, and $X^{\leq M}(H) \neq \varnothing$ implies $Z^{\leq M}(H) \neq \varnothing$.*

Proof. Since $H \subseteq J$, the definition of set $Z^{\leq M}(H)$ contains more constraints than that of set $X^{\leq M}(H)$. Hence $Z^{\leq M}(H) \subseteq X^{\leq M}(H)$. Let us prove $X^{\leq M}(H) \neq \varnothing \implies Z^{\leq M}(H) \neq \varnothing$. Recall that $\widetilde{G}_H$ is the precedence graph obtained from $G(p)$ by introducing additional arcs (i,j), $i \in H \cup \{s\}$, $j \in H$, $i \prec j$ with arc-lengths L^{Δ}_{ij}. Note that $X^{\leq M}(H)$ is exactly the set of schedules of $\widetilde{G}_H$ with makespan at most M. Let z be the earliest schedule of $\widetilde{G}_H$. By assumption, there exists a schedule in $X^{\leq M}(H)$, thus $z_t \leq M$. Let $i \in J \cup \{s\}$ and $j \in H$. Let us show that $z_j - z_i \geq L^{\Delta}_{ij}$ holds, even for $i \notin H \cup \{s\}$. Consider a longest $s{-}i$ path P^*_{si} in $\widetilde{G}_H$, and let $k \in H \cup \{s\}$ be the last vertex of $H \cup \{s\}$ on this path. Then $z_i = L_{\widetilde{G}_H}(s,i) = L_{\widetilde{G}_H}(s,k) + L_{\widetilde{G}_H}(k,i) = z_k + L_{\widetilde{G}_H}(k,i)$. The subpath of P^*_{si} from k to i is a longest $k{-}i$ path in $\widetilde{G}_H$. Since it has no vertex in $H \cup \{s\}$ except k, it uses no additional arc and its length is $L_{\widetilde{G}_H}(k,i) = L^0_{ki}$. It comes $z_k - z_i = -L^0_{ki}$.

Since $z \in X^{\leq M}(H)$, $z_j - z_k \geq L^{\Delta}_{kj}$. Then $z_j - z_i = (z_j - z_k) + (z_k - z_i) \geq L^{\Delta}_{kj} - L^0_{ki}$. Also, for $k \prec i \prec j$, it holds that $L^{\Delta}_{kj} \geq L^0_{ki} + L^{\Delta}_{ij}$: indeed L^{Δ}_{ij} is attained for some $\delta^* \in \Delta \subseteq \mathbb{R}^J_+$, hence $L^{\Delta}_{kj} \geq L_{G(p+\delta^*)}(k,i) + L_{G(p+\delta^*)}(i,j) \geq L^0_{ki} + L^{\Delta}_{ij}$. Finally $z_j - z_i \geq L^{\Delta}_{ij}$, hence $z \in Z^{\leq M}(H)$. □

8.2.3 Dominance-based linear formulation (Dom)

AnchRobPSP problem is to maximize the weight of a set H such that $X^{\leq M}(H) \neq \varnothing$, or equivalently with Theorem 8.1, such that $Z^{\leq M}(H) \neq \varnothing$. We now introduce a new formulation derived from Theorem 8.1, where continuous variables z correspond to a schedule $z \in Z^{\leq M}(H)$.

$$
\begin{array}{lll}
\text{(Dom):} \quad \max & \sum_{i \in J} w_i h_i & \\
\text{s.t.} & z_j - z_i \geq p_i & \forall (i,j) \in \mathcal{A} \\
& z_t \leq M & \\
& z_j - z_i \geq L^0_{ij} + (L^{\Delta}_{ij} - L^0_{ij}) h_j & \forall i,j \in \overline{J},\ i \prec j \\
& z_j \geq 0 & \forall j \in \overline{J} \\
& h_j \in \{0,1\} & \forall j \in J
\end{array}
\tag{3}
$$

Proposition 8.1. *Formulation (Dom) is valid for AnchRobPSP.*

Proof. Let (z,h) be feasible for (Dom), and $H := \{i \in J :\ h_i = 1\}$. Note first that inequality (3) implies $z \in X^{\leq M}(H)$. Hence z is a schedule with makespan at most M and H is z-anchored. Hence $h \in \mathcal{H}$. Conversely, let $h \in \mathcal{H}$ be the incidence vector of an anchored set H. From Theorem 8.1, there exists $z \in Z^{\leq M}(H)$. Such a schedule z satisfies $z_j - z_i \geq L^{\Delta}_{ij}$ for every $i \in J \cup \{s\}$ and $j \in H$. Then z satisfies (3): indeed if $h_j = 1$ inequality (3) corresponds to inequality $z_j - z_i \geq L^{\Delta}_{ij}$; if $h_j = 0$ then (3) amounts to $z_j - z_i \geq L^0_{ij}$, which holds since z is a schedule of $G(p)$. □

Note that precedence constraint $z_j - z_i \geq p_i$ associated with arc $(i,j) \in \mathcal{A}$ is implied by (3) since $i \prec j$ and $L^0_{ij} \geq p_i$.

We now introduce a family of valid inequalities. Let $j \in J$. By inequality (3) with $i = s$, it comes that $z_j \geq L^0_{sj} + (L^{\Delta}_{sj} - L^0_{sj}) h_j$. Since $z \in X^{\leq M}$ it satisfies $z_j + L^0_{jt} \leq M$, thus leading to $M \geq L^0_{sj} + (L^{\Delta}_{sj} - L^0_{sj}) h_j + L^0_{jt}$. Hence the following valid inequality

$$
h_j \leq \left\lfloor \frac{M - (L^0_{sj} + L^0_{jt})}{L^{\Delta}_{sj} - L^0_{sj}} \right\rfloor . \tag{4}
$$

In Section 8.3 these inequalities will be discussed with respect to the polyhedral characterization under box uncertainty.

8.2.4 Comparison between (Dom) and other formulations

In this section it is shown that formulation (Dom) is stronger than (Std), and stronger than (Lay) in some cases of interest.

Consider first the continuous relaxations of (Dom), (Std) and the variant of (Std) with quadratic constraint (1). It turns out that they can easily be compared. Indeed for every $h_i, h_j \in [0,1]$, it holds that $h_j \geq h_i h_j$. Hence the right-hand side of inequality (3) is tighter than the right-hand side of inequality (1). Also for every $h_i, h_j \in [0,1]$, it holds that $(1-h_i)(1-h_j) \geq 0$ or equivalently $h_i h_j \geq h_i + h_j - 1$. Hence the right-hand side of inequality (1) is tighter than the right-hand side of inequality (2).

Let us now investigate the special case of budgeted uncertainty, and compare (Dom) with formulation (Lay), that we now recall. Formulation (Lay) involves anchoring variables $h_j \in \{0,1\}$ for every $j \in J$, and continuous variables x_j^γ for every $\gamma \in \{0,\dots,\Gamma\}$, $j \in \overline{J}$. Formulation (Lay) is based on a so-called *layered graph* $G^{\text{lay}}(h)$ associated with $h \in [0,1]^J$ and built as follows. The layered graph $G^{\text{lay}}(h)$ is formed with $\Gamma+1$ copies of the precedence graph called *layers* indexed from 0 to Γ. Let i^γ denote the copy of job i in layer γ. The layered graph features three types of arcs. Horizontal arcs are copies of arcs of $G(p)$, i.e., arcs (i^γ, j^γ) for $(i,j) \in \mathcal{A}$, with length p_i. Transversal arcs are $(i^{\gamma+1}, j^\gamma)$ for $(i,j) \in \mathcal{A}$, with length $p_i + \widehat{\delta}_i$. Vertical arcs are (i^γ, i^Γ) for $i \in J$, $\gamma < \Gamma$, with length $-D_j(1-h_j)$, where D_j is a bigM value. A valid choice is $D_j = L_{G(p+\widehat{\delta})}(s,j) - L_{G(p)}(s,j)$ for every $j \in J$. It was shown that $h \in \{0,1\}^J$ is the incidence vector of an anchored set if and only if there exists x a schedule of $G^{\text{lay}}(h)$ such that $x_t^\Gamma \leq M$, see page 114. This leads to the formulation

$$
\begin{array}{lll}
\text{(Lay):} \quad \max & \sum_{i\in J} w_i h_i & \\
\text{s.t.} & x_j^\gamma - x_i^\gamma \geq p_i & \forall (i,j) \in \mathcal{A},\ \forall \gamma \in \{0,\dots,\Gamma\} \\
& x_j^\gamma - x_i^{\gamma+1} \geq p_i + \widehat{\delta}_i & \forall (i,j) \in \mathcal{A},\ \forall \gamma \in \{0,\dots,\Gamma-1\} \\
& x_j^\Gamma - x_j^\gamma \geq -D_j(1-h_j) & \forall j \in J,\ \forall \gamma \in \{0,\dots,\Gamma-1\} \\
& x_t^\Gamma \leq M & \\
& x_j^\gamma \geq 0 & \forall j \in \overline{J},\ \forall \gamma \in \{0,\dots,\Gamma\} \\
& h_j \in \{0,1\} & \forall j \in J
\end{array}
$$

Variables x_j^Γ, $j \in \overline{J}$ from layer Γ can be interpreted as a baseline schedule such that $H = \{i \in J :\ h_i = 1\}$ is x^Γ-anchored. Other continuous variables x_j^γ, $j \in \overline{J}$, $\gamma < \Gamma$ can be regarded as additional variables.

Let $\mathcal{P}^{\text{Lay}}$ (resp. $\mathcal{P}^{\text{Dom}}$) denote the polytope of solutions $((x^\gamma)_{\gamma\in\{0,\dots,\Gamma\}}, h)$ (resp. (z,h)) that are feasible for the continuous relaxation of formulation (Lay) (resp.

(Dom)). Consider the following assumption on the D_j values, denoted by (IneqD_j)

$$\text{for every } i \prec j, \quad D_j \geq L^{\Delta}_{ij} - L^0_{ij}. \tag{IneqD_j}$$

Note that assumption (IneqD_j) is that D_j values are "large enough". In formulation (Lay) they play the role of bigM values. We set $D_j = L_{G(p+\widehat{\delta})}(s,j) - L_{G(p)}(s,j)$ because it is a tight value for which (Lay) is valid. The formulation would remain valid for larger values of D_j's. In the sequel, we will show that the (IneqD_j) assumption is satisfied for critical precedence graphs, i.e., precedence graphs where all $s-t$ paths are longest $s-t$ paths. We now prove

Proposition 8.2. *Under assumption* (IneqD_j)*, formulation (Dom) is stronger than formulation (Lay), in the sense that* $\text{Proj}_h(\mathcal{P}^{\text{Dom}}) \subseteq \text{Proj}_h(\mathcal{P}^{\text{Lay}})$.

Proof. Let $h \in \text{Proj}_h(\mathcal{P}^{\text{Dom}})$, i.e., $h \in [0,1]^J$ such that there exists a schedule z satisfying inequalities (3) and $z_t \leq M$. We want to prove the existence of a schedule x of $G^{\text{lay}}(h)$ with makespan $x^\Gamma_t \leq M$. Let us build a graph $\widetilde{G}$ as follows: it contains the graph $G^{\text{lay}}(0)$ (no vertical arcs) to which we add precedence arcs (i,j) for $i \prec j$ with weight $L^0_{ij} + (L^{\Delta}_{ij} - L^0_{ij})h_j$ in the upper layer, so that to enforce constraints (3). Let x be the earliest schedule of $\widetilde{G}$. Since $\widetilde{G}$ has no vertical arc, the longest $s^\Gamma - t^\Gamma$ path in $\widetilde{G}$ is the longest $s^\Gamma - t^\Gamma$ path in its upper layer. Because of the existence of schedule z, the longest $s^\Gamma - t^\Gamma$ path in the upper layer of $\widetilde{G}$ has length at most M.

Let us show that x is a schedule of $G^{\text{lay}}(h)$: it is sufficient to show that vertical arcs constraints are satisfied, i.e., that $x^\Gamma_j - x^\gamma_j \geq -D_j(1-h_j)$ for every $j \in J$, $\gamma \in \{0,\dots,\Gamma\}$. Let us denote by L^γ_{ij} the longest $i-j$ path with uncertainty budget γ (then $L^\Gamma_{ij} = L^{\Delta}_{ij}$). Let $j \in J$, $\gamma \in \{0,\dots,\Gamma\}$. By definition of x, there exists $i^* \in J \cup \{s\}$ such that

$$x^\gamma_j = x^\Gamma_{i^*} + L^{\Gamma-\gamma}_{i^*j}$$

E.g., this equality is satisfied with i^* the last vertex in the upper layer on a longest path from s^Γ to j^γ. Because of arcs in the upper layer, we also have

$$x^\Gamma_j - x^\Gamma_{i^*} \geq L^0_{i^*j} + (L^\Gamma_{i^*j} - L^0_{i^*j})h_j.$$

Combining the two latter (in)equalities, we obtain $x^\Gamma_j - x^\gamma_j = (x^\Gamma_j - x^\Gamma_{i^*}) + (x^\Gamma_{i^*} - x^\gamma_j) \geq L^0_{i^*j} + (L^\Gamma_{i^*j} - L^0_{i^*j})h_j - L^{\Gamma-\gamma}_{i^*j} = (L^\Gamma_{i^*j} - L^0_{i^*j})h_j - (L^{\Gamma-\gamma}_{i^*j} - L^0_{i^*j})$. Let $f(h_j) = (L^\Gamma_{i^*j} - L^0_{i^*j})h_j - (L^{\Gamma-\gamma}_{i^*j} - L^0_{i^*j})$ denote this latter quantity. Let also $g(h_j) = -D_j(1-h_j)$. Both are affine functions of h_j. The inequality $f(h_j) \geq g(h_j)$ writes as $(D_j - L^\Gamma_{i^*j} + L^0_{i^*j})h_j \leq D_j - L^{\Gamma-\gamma}_{i^*j} + L^0_{i^*j}$. The right-hand side can be lower-bounded by $D_j - L^\Gamma_{i^*j} + L^0_{i^*j}$ (attained for $\gamma = 0$). By assumption (IneqD_j), $D_j - L^\Gamma_{i^*j} + L^0_{i^*j} \geq 0$ and the inequality $f(h_j) \geq g(h_j)$ holds for every $h_j \in [0,1]$

and $\gamma \leq \Gamma$. Hence schedule x satisfies vertical arcs constraints. Therefore x is a schedule of $G^{\text{lay}}(h)$ such that $x_t^\Gamma \leq M$, and $h \in \text{Proj}_h(\mathcal{P}^{\text{Lay}})$. □

We now give an example where $\text{Proj}_h(\mathcal{P}^{\text{Dom}})$ is strictly included in $\text{Proj}_h(\mathcal{P}^{\text{Lay}})$. Let $J = \{1, 2, 3\}$ be a set of three jobs, let G be the path $(s, 1, 2, 3, t)$, and let $p_i = 1$ and $\widehat{\delta}_i = 1$ for every $i \in J$. Let also $\Gamma = 1$, and $M = 3$. It is a case where Proposition 8.2 is applicable. Consider $h^* = (1, 0, \frac{1}{2})$. To see that $h \in \text{Proj}_h(\mathcal{P}^{\text{Lay}})$, consider the layered graph $G^{\text{lay}}(h^*)$, represented in Figure 8.2. The vector x defined by $x^1 = (0, 0, 1, 2, 3)$ in layer 1, $x^0 = (0, 0, 2, 3, 4)$ in layer 0 is also represented into brackets on the vertices in Figure 8.2.

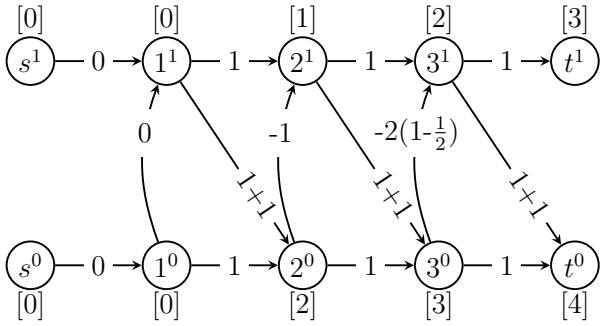


Figure 8.2: Layered graph $G^{\text{lay}}(h^*)$ for $h^* = (1, 0, \frac{1}{2})$, and schedule x into brackets.

It can be checked that vector x is a schedule of $G^{\text{lay}}(h^*)$, hence $(x, h^*) \in \mathcal{P}^{\text{Lay}}$. Assume for a contradiction that there exists $(z, h^*) \in \mathcal{P}^{\text{Dom}}$. Then $z_t - z_3 \geq 1$ and $z_3 - z_s \geq 2 + (3 - 2)h_3^* = 2.5$ and $z_t - z_s \geq 3.5 > M$. Thus $h^* \notin \text{Proj}_h(\mathcal{P}^{\text{Dom}})$, and $\text{Proj}_h(\mathcal{P}^{\text{Dom}}) \subsetneq \text{Proj}_h(\mathcal{P}^{\text{Lay}})$. More precisely, it is a case where (Dom) yields a polyhedral characterization of AnchRobPSP, as shown next in Section 8.3.2.1.

The question whether (Dom) is stronger than (Lay) in general case, even without assumption (IneqD_j), is left open. We mention that on all considered instances in numerical experiments, the linear bound of formulation (Dom) was better than the linear bound of formulation (Lay).

8.3 Polyhedral characterization for special cases

In this section, we provide polyhedral characterizations of AnchRobPSP for two special cases, under box uncertainty and 1-disruption uncertainty.

8.3.1 Box uncertainty

This section is devoted to box uncertainty, which is the special case of budgeted uncertainty where $\Gamma = |J|$, and thus $\widehat{\delta} \in \Delta$. In Section 6.4.1 of Chapter 6, a polynomial algorithm for AnchRobPSP under box uncertainty was provided. This algorithm is as follows: compute $\underline{x}$ the earliest schedule of $G(p+\widehat{\delta})$ and $\overline{x}$ the latest schedule of $G(p)$ such that $\overline{x}_t = M$; let $H^* = \{i \in J \mid \underline{x}_i \leq \overline{x}_i\}$ and $x_i = \min\{\underline{x}_i, \overline{x}_i\}$ for every $i \in \overline{J}$; return (x, H^*).

The main result is that in this case the polytope $\mathcal{Q}$ of anchored sets is characterized by inequalities (4).

Theorem 8.2. *For box uncertainty,*

$$\mathcal{Q} = \{h \in [0,1]^J : \ h \text{ satisfies } (4)\}.$$

Proof. Consider schedules $\underline{x}$ and $\overline{x}$, defined by $\underline{x}_j = L_{G(p+\widehat{\delta})}(s,j)$ and $\overline{x}_j = M - L^0_{jt}$ for every $j \in J$. Note first that for box uncertainty, $L^{\Delta}_{sj} = L_{G(p+\widehat{\delta})}(s,j) = \underline{x}_j$. Inequality (4) writes in this case: $h_j \leq \left\lfloor \frac{M-(L^0_{sj}+L^0_{jt})}{L^{\Delta}_{sj}-L^0_{sj}} \right\rfloor = \left\lfloor \frac{\overline{x}_j - L^0_{sj}}{\underline{x}_j - L^0_{sj}} \right\rfloor$. Equivalently, it implies $h_j \leq 1$ if $j \in H^*$, and $h_j \leq 0$ if $j \notin H^*$. Hence $\{h \in [0,1]^J : h \text{ satisfies } (4)\} = [0,1]^{H^*} \times \{0\}^{J \setminus H^*}$.

Every extreme point of $[0,1]^{H^*} \times \{0\}^{J \setminus H^*}$ is the incidence vector of a set $H \subseteq H^*$. Since any subset of an anchored set is anchored, and H^* is anchored, then $[0,1]^{H^*} \times \{0\}^{J \setminus H^*} \subseteq \mathcal{Q}$. Conversely, if H is an anchored set, its incidence vector χ^H satisfies valid inequalities (4): thus $H \subseteq H^*$. Hence $\mathcal{Q} \subseteq [0,1]^{H^*} \times \{0\}^{J \setminus H^*}$. □

Note that Theorem 8.2 also holds for any Δ that has a greatest element $\widehat{\delta}$. This case is more general than Δ being a box, as mentioned in Section 8.1.

8.3.2 1-disruption uncertainty

In this section, set Δ is a 1-disruption uncertainty set, i.e., a budgeted uncertainty set with $\Gamma = 1$ and $\widehat{\delta}_i = \widehat{\delta}_0$ for every $i \in J$. In Section 8.3.2.1 the polyhedral characterization is shown for the special case with processing times equal to zero. In Section 8.3.2.2 it is extended to any critical precedence graph, with possibly non-zero processing times.

8.3.2.1 The Unitary AnchRobPSP

Consider first the case of zero processing times. W.l.o.g. $\widehat{\delta}_0 = 1$. This is the Unitary AnchRobPSP (U-AnchRob) studied in Section 6.4.2.2, page 124. For unit

anchoring weights, it was identified as a polynomial case by an equivalence with a problem on posets for which a min/max theorem is known (Schrijver, 2003).

An instance is the precedence graph $G = (\overline{J}, \mathcal{A})$, $p = 0$, $\widehat{\delta}_0 = 1$, integer deadline M, and anchoring weights. Indeed deadline M can be assumed integer. If M is not integer, it can be replaced with $\lfloor M \rfloor$ w.l.o.g.: indeed, since p and $\widehat{\delta}$ are integer, for any H the earliest schedule of for which H is anchored is integer-valued with integer makespan.

For (U-AnchRob) our main result is a characterization of the polytope through formulation (Dom). It relies on the polyhedral result of Theorem 8.3. Let $(\overline{J}, \prec)$ be a poset with s (resp. t) a least (resp. greatest) element. Consider the inequalities

$$\begin{array}{lll} z_i - z_s \geq 0 & \forall i \in J & \text{(a)} \\ z_t - z_i \geq 0 & \forall i \in J & \text{(b)} \\ z_t \leq M & & \text{(c)} \\ z_j - z_i \geq h_j & \forall i, j \in J,\ i \prec j & \text{(d)} \\ z_j \geq 0 & \forall j \in \overline{J} & \text{(e)} \\ h_j \leq 1 & \forall j \in J & \text{(f)} \\ h_j \geq 0 & \forall j \in J & \text{(g)} \end{array}$$

and let $\mathcal{P} = \{(z, h) \in \mathbb{R}^{\overline{J}} \times \mathbb{R}^J : \text{ (a)} - \text{(g)}\}$.

Theorem 8.3. *The polytope $\mathcal{P}$ is integer.*

Proof. Consider the polytope $\mathcal{P}$ formed with all pairs $(z, h) \in \mathbb{R}^{\overline{J}}_+ \times \mathbb{R}^J_+$ satisfying the constraints (a)–(g). To prove integrality of $\mathcal{P}$, the main idea is to define an auxiliary extended polyhedron where h variables can be expressed linearly from z variables and additional z' variables. Let us define the auxiliary polyhedron $\mathcal{P}'$ formed with all triplets $(z, z', h) \in \mathbb{R}^{\overline{J}}_+ \times \mathbb{R}^{\overline{J}}_+ \times \mathbb{R}^J$ satisfying the constraints

$$\begin{array}{lll} z_j - z_i \geq 0 & \forall (i,j) \in \mathcal{A} & \text{(i)} \\ z'_j - z'_i \geq 0 & \forall (i,j) \in \mathcal{A} & \text{(ii)} \\ z'_j - z_i \geq 1 & \forall (i,j) \in \mathcal{A} & \text{(iii)} \\ z_j - z'_j \geq -1 & \forall j \in J & \text{(iv)} \\ z_j - z'_j \leq 0 & \forall j \in J & \text{(v)} \\ z_t \leq M & & \text{(vi)} \\ h_j = 1 + z_j - z'_j & \forall j \in J & \text{(vii)} \end{array}$$

Let us prove the following claim. *Claim 1.* $\mathcal{P} = \mathrm{Proj}_{z,h}(\mathcal{P}')$.
First, given $(z, h) \in \mathcal{P}$, let us prove the existence of z' such that $(z, z', h) \in \mathcal{P}'$. Define $z'_j = 1 + z_j - h_j$ for every $j \in J$, and $z'_s = z_s$, $z'_t = 1 + z_t$. Then the (in)equalities (i) to (vii) can be checked for the triplet (z, z', h) as follows.

– (i) and (vi) hold by assumption on z, and (vii) by definition of z';

- (ii): if $j \neq t$ then $z'_j - z'_i = z_j - z_i - h_j + h_i \geq h_i \geq 0$ by (d); if $j = t$ then $z'_j - z'_i = z_t - z_i + h_i \geq h_i \geq 0$ by (b);
- (iii): if $j \neq t$ then $z'_j - z_i = 1 + z_j - h_j - z_i \geq 1$ by (d); if $j = t$, follows from (b);
- (iv) and (v): $z_j - z'_j = -1 + h_j \in [-1, 0]$ since $h_j \in [0, 1]$.

Conversely, let $(z, z', h) \in \mathcal{P}'$. Let us check inequalities (a)–(g) for (z, h).

- (a) holds by sum of (i) along an $s-i$ path;
- (b) holds by sum of (i) along an $i-t$ path;
- (c) is clear;
- (d): for $i \prec j$ we have $z_j - z_i = z_j - z'_j + z'_j - z_k + z_k - z_i$, where k is the last vertex distinct from j on a path from i to j in the precedence graph (possibly $k = i$). Then $(k, j) \in \mathcal{A}$ so $z'_j - z_k \geq 1$ by (iii). Also $z_k - z_i \geq 0$ by summing (i) along the $i-k$ path. Hence $z_j - z_i \geq z_j - z'_j + 1 + 0 = h_j$ by (vii). Hence (d) is satisfied;
- (f) (resp. (g)) comes from (vii) and (iv) (resp. (vii) and (v)).

This completes the proof of Claim 1. $\diamond$

Now we prove: *Claim 2. $\mathcal{P}'$ is integer.*
Let (z, z', h) be an extreme point of $\mathcal{P}'$. It satisfies the n equalities (vii) and it saturates $2(n + 2)$ linearly independent inequalities among (i)–(vi). Thus (z, z') is an extreme point of $\{(z, z') \in \mathbb{R}^{\overline{J}}_+ \times \mathbb{R}^{\overline{J}}_+ : \text{ (i)-(vi)}\}$. The constraint matrix of (i)–(vi) is totally unimodular, and the right-hand side is integer since M is integer. Hence (z, z') is integer and so is (z, z', h). This ends the proof of Claim 2. $\diamond$

By Claim 1, $\mathcal{P} = \mathrm{Proj}_{z,h}(\mathcal{P}')$. It holds that for any extreme point (z, h) of $\mathcal{P}$ there exists z' such that (z, z', h) is an extreme point of $\mathcal{P}'$. By Claim 2, such (z, z', h) is integer, hence (z, h) is integer. This proves the integrality of $\mathcal{P}$. $\square$

Theorem 8.3 implies the following polyhedral characterization

Proposition 8.3. *Formulation (Dom) yields a polyhedral characterization for (U-AnchRob).*

Proof. Let us prove that the linear relaxation of (Dom) correponds to the polytope $\mathcal{P}$ from Theorem 8.3. For (U-AnchRob) for every $i, j \in \overline{J}$ such that $i \prec j$ we have $L^0_{ij} = 0$ since $p = 0$. The worst-case longest paths values are as follows. For every $j \in J$, $L^{\Delta}_{sj} = 0$ if j has no predecessor except s, and $L^{\Delta}_{sj} = 1$ otherwise. For every $i \prec j$ with $i \neq s$, $L^{\Delta}_{ij} = 1$. For a pair $i \prec j$ with $i \neq s$, inequality (3) thus writes $z_j - z_i \geq h_j$, which is inequality (d) or (b) from the definition of $\mathcal{P}$. For a pair $s \prec j$, inequality (3) writes $z_j - z_s \geq 0$ if j has no predecessor in J, and $z_j - z_s \geq h_j$ otherwise. If j has no predecessor in J, it is inequality (a) from the definition of $\mathcal{P}$. Otherwise j has a predecessor $k \in J$ and inequality $z_j - z_s \geq h_j$ is dominated by inequalities $z_k - z_s \geq 0$ and $z_j - z_k \geq h_k$. Hence it is satisfied by any element of $\mathcal{P}$. Thus $\mathcal{P}^{\mathrm{Dom}} = \mathcal{P}$.

By Theorem 8.3, $\mathcal{P}$ is integer, hence $\text{Proj}_h(\mathcal{P})$ is integer. Namely, each extreme point of $\text{Proj}_h(\mathcal{P})$ is the incidence vector of an anchored set. Thus $\text{Proj}_h(\mathcal{P}^{\text{Dom}}) = \text{Proj}_h(\mathcal{P}) = \mathcal{Q}$, and (Dom) yields a polyhedral characterization for the problem. □

Note that Proposition 8.3 implies also the integrality of schedule variables z, which are not required to be integer in general.

As a corollary, a complexity result is that Theorem 8.3 generalizes the polynomial case of (U-AnchRob) proven in Chapter 6 for unit anchoring weights, to any non-negative anchoring weights.

8.3.2.2 Critical precedence graphs

Let us now consider processing times $p \in \mathbb{R}^J_+$ and consider the case of critical precedence graphs. Recall that the precedence graph $G(p)$ is *critical* if the length of all $s-t$ paths is the same.

Properties of critical precedence graphs. A case where $G(p)$ is critical is when processing times are equal to zero, since the length of all $s-t$ paths is zero. Also if G is a path, for any p the precedence graph $G(p)$ is critical since there is a unique $s-t$ path. Let us say that the precedence graph $G(p)$ is *quasi-critical* if every job $i \in J$ belongs to a critical path, i.e., $L^0_{si} + L^0_{it} = L^0_{st}$. Note that if $G(p)$ is critical, a necessary condition is that it is quasi-critical. For non-zero processing times, it can also be proven that for some precedence graphs, if $G(p)$ is quasi-critical then it is critical. This holds for series-parallel precedence graphs.

Proposition 8.4. *If G is series-parallel and $G(p)$ is quasi-critical, then $G(p)$ is critical.*

Proof. If G is a path, then $G(p)$ is critical. If G is obtained by series composition of G^1 and G^2; then any $s-t$ path is formed by an s^1-t^1 path in G^1 and an s^2-t^2 path in G^2. Hence if G^1 and G^2 are critical, it follows that G is critical. If G is obtained by parallel composition of G^1 and G^2, both critical, let $i \in G^1$ and $j \in G^2$. Then every $s-t$ path going through G^1 (resp. G^2) has length $L_{G^1}(s,t) = L_{G^1}(s,i) + L_{G^1}(i,t)$ (resp. $L_{G^2}(s,t) = L_{G^2}(s,j) + L_{G^2}(j,t)$). If G is quasi-critical, $L_G(s,i) + L_G(i,t) = L_G(s,j) + L_G(j,t)$, and it follows that all $s-t$ paths of G have same length. □

This result is used for numerical results in Section 8.4 to generate critical precedence graphs.

Let us now show

Proposition 8.5. *If $G(p)$ is critical, then the assumption* (IneqD_j) *is satisfied for values $D_j = L_{G(p+\widehat{\delta})}(s,j) - L_{G(p)}(s,j)$, for any uncertainty set $\Delta \subseteq \mathbb{R}^J_+$.*

Proof. Let $i \prec j$. By definition of D_j, it holds that $D_j \geq L^{\Delta}_{sj} - L^0_{sj}$. Because $G(p)$ is critical, it holds that $L^0_{sj} = L^0_{si} + L^0_{ij}$. Hence $D_j \geq L^{\Delta}_{sj} - L^0_{si} - L^0_{ij}$. By definition of the L^{Δ}_{ij} values, it holds that $L^{\Delta}_{sj} \geq L^0_{si} + L^{\Delta}_{ij}$. Hence $D_j \geq L^{\Delta}_{ij} - L^0_{ij}$ as claimed. □

Hence by Proposition 8.2, for critical precedence graphs, formulation (Dom) is stronger than formulation (Lay).

Polyhedral characterization for critical precedence graphs. Let us now consider a critical precedence graph $G(p)$, and extend the polyhedral characterization result obtained in Section 8.3.2.1. It is assumed that $M = L^0_{st} + \widehat{\delta}_0 M'$ with M' integer. Indeed M can be tightened to $L^0_{st} + \widehat{\delta}_0 \left\lfloor (M - L^0_{st})/\widehat{\delta}_0 \right\rfloor$ w.l.o.g. Under this assumption we prove that

Theorem 8.4. *For 1-disruption uncertainty and critical precedence graph, formulation (Dom) yields a polyhedral characterization of AnchRobPSP.*

Proof. Let $\mathcal{I}$ denote an instance of AnchRobPSP for 1-disruption uncertainty with deviation $\widehat{\delta}_0$, critical precedence graph $G(p)$, and deadline M. Let z^* be the earliest schedule of $G(p)$, namely $z^*_i = L^0_{si}$ for every $i \in \overline{J}$. Since $G(p)$ is critical, for every $i \prec j$ it holds that $L^0_{si} + L^0_{ij} + L^0_{jt} = L^0_{sj} + L^0_{jt}$, hence $L^0_{ij} = z^*_j - z^*_i$. Also $L^{\Delta}_{ij} = z^*_j - z^*_i + \widehat{\delta}_0$. Inequalities (3) then write $z_j - z_i \geq (z^*_j - z^*_i) + \widehat{\delta}_0 h_j$, or equivalently $\frac{z_j - z^*_j}{\widehat{\delta}_0} - \frac{z_i - z^*_i}{\widehat{\delta}_0} \geq h_j$. Similarly the deadline constraint is equivalent to $\frac{z_t - z^*_t}{\widehat{\delta}_0} \leq \frac{M - z^*_t}{\widehat{\delta}_0}$. Consider a new instance $\mathcal{I}'$ of (U-AnchRob) defined by: the precedence graph G, zero processing times, and $M' = \frac{M - L^0_{st}}{\widehat{\delta}_0}$. Then M' is integer by assumption on M. Solution (z,h) is feasible for (Dom) in instance $\mathcal{I}$ if and only if solution $(\frac{z - z^*}{\widehat{\delta}_0}, h)$ is feasible for (Dom) in instance $\mathcal{I}'$. Hence if (z,h) is an extreme point of $\mathcal{P}^{\text{Dom}}$ then $(\frac{z - z^*}{\widehat{\delta}_0}, h)$ is extreme for the (U-AnchRob) instance. By Proposition 8.3, vector h is integer. Hence the claimed result. □

Theorem 8.4 thus yields a polyhedral characterization of polynomial size for this special case. We mention that this is a polynomial case of AnchRobPSP that was not identified in Chapter 6.

8.4 Numerical results

We investigate the impact of theoretical polyhedral results from Section 8.3 on the performance of formulations (Lay), (Std), (Dom) for various instance classes. In Section 8.4.1, instances and settings are presented. Section 8.4.2 is dedicated to budgeted uncertainty, and Section 8.4.3 to partition-budgeted and mixed-budgeted uncertainty sets.

8.4.1 Instances and settings

Instances are randomly generated as follows. We consider instance classes with a four-field label `F1_F2_F3_F4`:

- Field `F1` concerns precedence graph G
  - `ER`: precedence graphs randomly generated according to Erdos-Renyi model, i.e., arc (i, j) is in G with probability $pr = 10/n$.
  - `SP`: Series-Parallel precedence graphs, inductively generated by drawing randomly series or parallel compositions.
- Field `F2` denotes processing times p
  - `pZero`: $p_i = 0$ for every $i \in J$;
  - `pRand`: p_i is randomly generated in range $[5, 20]$;
  - `pQCri`: p is obtained by applying the following procedure: start from the values generated for class `pRand`; increase the processing time of a randomly selected job until every job is on a critical path. Hence $G(p)$ is quasi-critical.
- Field `F3` denotes deviation $\widehat{\delta}$
  - `dRand`: for instances `pRand` and `pQCri`, $\widehat{\delta}_i$ is randomly generated in $[1, \frac{1}{2}p_i]$ for every $i \in J$; for instances `pZero`, $\widehat{\delta}$ is equal to the values generated for instances `pQCri`;
  - `dUnif`: $\widehat{\delta}_i = \widehat{\delta}_0$ for every $i \in J$. Value $\widehat{\delta}_0$ is randomly selected in the deviation values of instances `dRand`.
- Field `F4` denotes the uncertainty set
  - `Γ1`, `Γ2`, `Γ3` correspond to budgeted uncertainty with deviation defined by `F3` and $\Gamma = 1, 2, 3$ respectively.
  - `Partition`: jobs are partitioned into two subsets J^1 and J^2, every job being in J^1 with probability 0.75. Given the deviation $\widehat{\delta}$ defined by `F3`, the deviation vector of the `Partition` instance is $\lfloor 0.1\widehat{\delta}_i \rfloor$ for every $i \in J^1$, and $\widehat{\delta}_i$ for every $i \in J^2$. Budgets are $\Gamma^1 = 10$ and $\Gamma^2 = 1$.
  - `Mixed`: $\Delta = \Delta^1 \cup \Delta^2$ where Δ^1 has deviation $\widehat{\delta}$ defined by `F3` and $\Gamma^1 = 1$, and Δ^2 has deviation $\tau\widehat{\delta}$, with $\tau = 0.2$ and $\widehat{\delta}$ defined by `F3` and $\Gamma^2 = 10$.

Each label corresponds to a class of 10 instances.

By Proposition 8.4 classes `SP_pQCri` yield critical precedence graphs. Note also that classes `ER_pRand` (resp. `ER_pQcri`) are generated with the same generation scheme as instances ER (resp. ERC) from Chapter 6.

The number of jobs is set to $n = 300$. Anchoring weights are unitary. Deadline M is set to $M^{\frac{1}{2}} = \frac{1}{2}(L_{G(p)}(s,t) + L_{G(p+\widehat{\delta})}(s,t))$, that is, it is halfway between the min makespan of any schedule of $G(p)$ and the min makespan of a robust-static schedule. Unreported results showed that choosing a deadline other than $M^{\frac{1}{2}}$ leads to similar results in terms of the comparison of formulations. For budgeted uncertainty, the budget is $\Gamma \in \{1, 2, 3\}$. The choice of a small uncertainty budget was previously motivated in the literature, see, e.g., (Herroelen and Leus, 2004), and in the numerical results of Chapter 6.

For each instance, formulations (Dom), (Std), and (Lay) have been implemented using Julia 0.6.2, JuMP 0.18.5. Mixed-integer programs are solved with CPLEX 12.8 on a PC under Windows 10 with Intel Core i7-7500U CPU 2.90GHz and 8 Go RAM. The time limit is 300 seconds.

The valid inequalities (4) appear to be added by CPLEX on the fly. Thus they are not hardcoded in formulations.

8.4.2 Impact of instance parameters under budgeted uncertainty

Let us first investigate the case of budgeted uncertainty. Table 8.1 and Table 8.2 present the results for `ER` and `SP` instances respectively. Each table presents the results relative to 8 instance classes: the first 6 instance classes are with $\Gamma = 1$ and with all combinations of processing times and deviations, and the last 2 instance classes are with $\Gamma = 2$ and $\Gamma = 3$. For each instance class, checkmarks in the first three columns indicate if the assumptions of Theorem 8.4 are matched:

– crit.: the precedence graph $G(p)$ is critical;
– unif.: deviation $\widehat{\delta}$ is uniform;
– Γ1: $\Gamma = 1$.

The tables feature:

- opt: average optimal value for instances solved optimally;
- for each formulation (Lay), (Std) and (Dom):
  - #solved: number of instances, out of 10, solved optimally within the time limit;
  - gap: average final gap of unsolved instances;
  - time: average computation time for solved instances in seconds;
  - LPGap: average gap $\frac{b-opt}{opt}$ between integer optimum *opt* and linear bound b;
  - CPXGap: average gap obtained by CPLEX at root node.

The computation times do not include the preprocessing time for computing L_{ij}^{Δ} values. The computation is done by a dynamic programming algorithm linear in $\Gamma|\mathcal{A}|$. Its running time is negligible with respect to MIP computation time: on average 0.153 seconds for `ER` instances and 0.161 seconds for `SP` instances.

Let us now comment on the impact of the instance parameters.

Polyhedral characterization cases. Instance classes `ER_pZero_dUnif_Γ1`, `SP_pZero_dUnif_Γ1`, and `SP_pQCri_ dUnif_Γ1`, correspond to polyhedral characterization cases. As expected, formulation (Dom) solves the problem in less than one second and LPGap = 0%. Formulation (Lay) has non-zero LPGap, but CPLEX adds suitable cuts to close the gap at root node.

Impact of uniform deviation. Consider now the 6 instance classes `F1_F2_dUnif_Γ1`. These are the first three row entries of Table 8.1 and the first three row entries of Table 8.2. On these 6 instance classes, formulations (Dom) and (Lay) still behave well. (Dom) solves all instances in less than one second. In particular, its LPGap is still very small: at most 0.42%. (Lay) also performs well, the LPGap of (Lay) is larger, but CPXGap is comparable for (Dom) and (Lay). We note that uniform deviation has an important impact on the performance of formulations. Consider, e.g., instance classes `SP_pZero_dUnif_Γ1` and `SP_pZero_dRand_Γ1`. For uniform deviation (Dom) is integer ; for non-uniform deviation it has 9.70% CPXGap and solves only 4 instances out of 10 within the time limit. By contrast with (Dom) and (Lay) on these instances, formulation (Std) performs very poorly and solves only 41 instances out of 120 (vs. 107 out of 120 for (Dom)).

Impact of the precedence graph. The impact of the precedence graph being critical is limited, as shown for example by the comparison of instances `ER_pZero` and `ER_pQCri`. Both are efficiently solved, while the precedence graph is critical for the former, and not critical for the latter. Even more, instances with `pRand` appear to be easy instances, while they do not have critical precedence graphs. An interpretation is that for such instances, a large number of jobs are not on critical

paths and thus they can be anchored; note, e.g., that the optimal value is greater for `pRand` instances than for others.

Impact of uncertainty budget. When Γ is increased, the performance of (Lay) deteriorates. It gets even worse than (Std) for `ER` instances, see `ER_pZero_dUnif_Γ3` where (Std) solves 3 instances and (Lay) solves 1 instance. Importantly, the size of formulation (Lay) increases with Γ. For (Dom) and (Std) only the values of the coefficients L_{ij}^{Δ} depend on the budget, and not the size of the formulation.

instance	crit.	unif.	Γ1	opt		#solved	gap	time(s)	LPGap	CPXGap
ER_pZero_dUnif_Γ1	✓	✓	✓	271.90	(Lay)	10	-	1	9.20%	0.90%
					(Std)	1	1.19%	144	5.14%	3.07%
					(Dom)	10	-	<1	0%	0%
ER_pQCri_dUnif_Γ1		✓	✓	274.30	(Lay)	10	-	26	8.39%	0.87%
					(Std)	0	2.05%	-	8.04%	4.29%
					(Dom)	10	-	1	0.42%	0.24%
ER_pRand_dUnif_Γ1		✓	✓	290.40	(Lay)	10	-	<1	3.06%	0.06%
					(Std)	10	-	30	3.02%	1.62%
					(Dom)	10	-	<1	0.10%	0%
ER_pZero_dRand_Γ1	✓		✓	214.77	(Lay)	9	2.84%	21	34.15%	7.93%
					(Std)	9	4.92%	65	28.53%	4.68%
					(Dom)	9	1.81%	11	21.13%	3.28%
ER_pQCri_dRand_Γ1			✓	225.80	(Lay)	8	0.98%	21	28.54%	7.70%
					(Std)	6	2.30%	41	30.25%	5.41%
					(Dom)	10	-	31	18.09%	3.69%
ER_pRand_dRand_Γ1			✓	290.20	(Lay)	10	-	<1	3.00%	0.36%
					(Std)	10	-	<1	3.02%	1.74%
					(Dom)	10	-	<1	0.82%	0.07%
ER_pZero_dUnif_Γ2	✓	✓		255.50	(Lay)	2	1.88%	113	15.11%	9.93%
					(Std)	1	4.20%	237	10.93%	7.58%
					(Dom)	10	-	53	3.91%	2.46%
ER_pZero_dUnif_Γ3	✓	✓		243.50	(Lay)	1	4.41%	111	19.77%	13.69%
					(Std)	3	5.48%	153	15.76%	7.78%
					(Dom)	10	-	105	7.18%	3.51%

Table 8.1: `ER` instances, budgeted uncertainty

instance	crit.	unif.	Γ1	opt		#solved	gap	time(s)	LPGap	CPXGap
SP_pZero_dUnif_Γ1	✓	✓	✓	255.90	(Lay)	10	-	<1	17.09%	0%
					(Std)	0	3.34%	-	8.64%	6.81%
					(Dom)	10	-	<1	0%	0%
SP_pQCri_dUnif_Γ1	✓	✓	✓	255.90	(Lay)	10	-	<1	17.09%	0%
					(Std)	0	6.88%	-	14.82%	8.96%
					(Dom)	10	-	<1	0%	0%
SP_pRand_dUnif_Γ1		✓	✓	262.80	(Lay)	10	-	<1	14.19%	0.42%
					(Std)	0	6.86%	-	13.36%	7.29%
					(Dom)	10	-	<1	0.05%	0%
SP_pZero_dRand_Γ1	✓		✓	247.00	(Lay)	4	3.72%	3	32.46%	16.43%
					(Std)	2	18.60%	11	25.97%	20.07%
					(Dom)	4	9.12%	15	16.42%	9.70%
SP_pQCri_dRand_Γ1	✓		✓	247.00	(Lay)	4	3.52%	3	32.38%	16.72%
					(Std)	2	23.58%	30	35.21%	27.94%
					(Dom)	4	9.83%	14	16.54%	9.90%
SP_pRand_dRand_Γ1			✓	268.60	(Lay)	10	-	3	11.51%	6.97%
					(Std)	1	6.09%	193	11.05%	9.34%
					(Dom)	10	-	16	2.26%	0.92%
SP_pQCri_dUnif_Γ2	✓	✓		246.66	(Lay)	7	0.85%	45	22.65%	10.58%
					(Std)	0	11.80%	-	20.92%	16.63%
					(Dom)	9	0.77%	31	2.44%	1.35%
SP_pQCri_dUnif_Γ3	✓	✓		246.14	(Lay)	5	2.77%	57	26.40%	16.38%
					(Std)	1	17.42%	215	25.47%	20.68%
					(Dom)	7	2.19%	38	3.92%	3.29%

Table 8.2: SP instances, budgeted uncertainty

8.4.3 Beyond budgeted uncertainty

Let us now present results when Δ is an uncertainty set with several budgets.

8.4.3.1 Partition-budgeted uncertainty set

Table 8.3 and Table 8.4 give computational results for ER and SP instances under Partition uncertainty. In this case, the L_{ij}^{Δ} values were computed by a dynamic programming algorithm with complexity linear in $\Gamma^1\Gamma^2|\mathcal{A}|$. The computation was done in 4.364 seconds on average for ER instances, and 4.916 seconds on average for SP instances.

It comes that (Dom) solves all 60 ER instances, and 54 SP instances, which is better than for budgeted uncertainty. For SP instances, the results are comparable to those of budgeted uncertainty. By contrast, ER instances appear to be very easy for this uncertainty set. Hence solving the MIP formulation (Dom) for Partition uncertainty does not seem harder than for budget Γ1. This highlight that (Dom) can be readily used to handle several budget constraints.

instance	crit.	unif.	Γ1	opt		#solved	gap	time(s)	LPGap	CPXGap
ER_pZero_dUnif_Partition	✓	✓		286.50	(Lay)	-	-	-	-	-
					(Std)	10	-	<1	2.17%	0.07%
					(Dom)	10	-	<1	0%	0%
ER_pQCri_dUnif_Partition		✓		284.30	(Lay)	-	-	-	-	-
					(Std)	10	-	1	4.92%	0.22%
					(Dom)	10	-	<1	1.53%	0%
ER_pRand_dUnif_Partition		✓		294.50	(Lay)	-	-	-	-	-
					(Std)	10	-	<1	1.72%	0.05%
					(Dom)	10	-	<1	0.34%	0%
ER_pZero_dRand_Partition	✓			246.10	(Lay)	-	-	-	-	-
					(Std)	10	-	<1	22.03%	0%
					(Dom)	10	-	<1	20.22%	0%
ER_pQCri_dRand_Partition				260.70	(Lay)	-	-	-	-	-
					(Std)	10	-	<1	14.45%	0%
					(Dom)	10	-	<1	12.13%	0%
ER_pRand_dRand_Partition				293.10	(Lay)	-	-	-	-	-
					(Std)	10	-	<1	2.14%	0%
					(Dom)	10	-	<1	0.77%	0%

Table 8.3: ER instances, Partition uncertainty

instance	crit.	unif.	Γ1	opt		#solved	gap	time(s)	LPGap	CPXGap
SP_pZero_dUnif_Partition	✓	✓		274.70	(Lay)	-	-	-	-	-
					(Std)	8	1.36%	14	4.48%	2.01%
					(Dom)	10	-	<1	0%	0%
SP_pQCri_dUnif_Partition	✓	✓		273.44	(Lay)	-	-	-	-	-
					(Std)	4	7.42%	43	10.57%	8.29%
					(Dom)	9	3.22%	14	1.43%	0.84%
SP_pRand_dUnif_Partition		✓		284.90	(Lay)	-	-	-	-	-
					(Std)	8	2.12%	57	4.89%	2.71%
					(Dom)	10	-	<1	0.17%	0%
SP_pZero_dRand_Partition	✓			246.87	(Lay)	-	-	-	-	-
					(Std)	7	11.40%	1	23.12%	4.54%
					(Dom)	8	4.05%	33	19.43%	2.79%
SP_pQCri_dRand_Partition	✓			247.42	(Lay)	-	-	-	-	-
					(Std)	7	13.46%	1	27.36%	5.16%
					(Dom)	7	3.28%	<1	19.33%	2.66%
SP_pRand_dRand_Partition				279.10	(Lay)	-	-	-	-	-
					(Std)	7	4.89%	44	7.10%	5.05%
					(Dom)	10	-	1	1.85%	0.41%

Table 8.4: SP instances, Partition uncertainty

8.4.3.2 Mixed-budgeted uncertainty sets

Table 8.5 and Table 8.6 give computational results for `ER` and `SP` instances under `Mixed` uncertainty. In this case $\Delta = \Delta^1 \cup \Delta^2$ where Δ^1 is the uncertainty set corresponding to instance classes with fourth field Γ`1`. The L_{ij}^{Δ} values were precomputed by the same dynamic programming algorithm as for budgeted uncertainty. This computation was done in 0.152 seconds on average for `ER` instances, and 0.154 seconds on average for `SP` instances.

First, (Dom) solves optimally 59 `ER` instances and 42 `SP` instances for `Mixed`, in comparison with 59 `ER` instances and 48 `SP` instances for budgeted uncertainty ($\Gamma = 1$). That is, the performance of the formulation is not very sensitive to the change of uncertainty set.

Some conclusions given in Section 8.4.2 also hold for `Mixed` uncertainty. Namely, instances with `dUnif` are easier than instances with `dRand`. It can be related to the influence of uniform deviation on the performance of (Dom), in connection with the polyhedral characterization result.

The optimal number of anchored jobs is often the same for Δ or Δ^1. Namely, the value of opt can be compared between Table 8.1 and Table 8.5 for `ER` instances, and between Table 8.2 and Table 8.6 for `SP` instances. For example for instance classes `SP_pZero_dRand_`Γ`1` and `SP_pZero_dRand_Mixed` the average optimal value is equal to 247.00, hence all instances have the same optimal value for uncertainty set Δ and Δ^1. This means that the uncertainty set can be extended from Δ^1 to $\Delta = \Delta^1 \cup \Delta^2$ without deteriorating the number of anchored jobs.

instance	crit.	unif.	Γ1	opt		#solved	gap	time(s)	LPGap	CPXGap
`ER_pZero_dUnif_Mixed`	✓	✓		271.90	(Lay)	-	-	-	-	-
					(Std)	1	1.17%	286	5.14%	3.06%
					(Dom)	10	-	<1	0%	0%
`ER_pQCri_dUnif_Mixed`		✓		274.20	(Lay)	-	-	-	-	-
					(Std)	0	2.13%	-	8.08%	4.28%
					(Dom)	10	-	1	0.45%	0.24%
`ER_pRand_dUnif_Mixed`		✓		290.40	(Lay)	-	-	-	-	-
					(Std)	10	-	26	3.02%	1.62%
					(Dom)	10	-	<1	0.10%	0.03%
`ER_pZero_dRand_Mixed`	✓			214.77	(Lay)	-	-	-	-	-
					(Std)	9	5.35%	66	28.53%	4.61%
					(Dom)	9	1.83%	13	21.13%	3.22%
`ER_pQCri_dRand_Mixed`				225.80	(Lay)	-	-	-	-	-
					(Std)	7	2.61%	77	30.19%	5.09%
					(Dom)	10	-	22	18.09%	3.49%
`ER_pRand_dRand_Mixed`				290.20	(Lay)	-	-	-	-	-
					(Std)	10	-	<1	3.02%	1.78%
					(Dom)	10	-	<1	0.82%	0.07%

Table 8.5: `ER` instances, `Mixed` uncertainty

instance	crit.	unif.	Γ1	opt		#solved	gap	time(s)	LPGap	CPXGap
SP_pZero_dUnif_Mixed	✓	✓		255.70	(Lay)	-	-	-	-	-
					(Std)	0	3.60%	-	8.72%	6.84%
					(Dom)	10	-	1	0.06%	0%
SP_pQCri_dUnif_Mixed	✓	✓		246.75	(Lay)	-	-	-	-	-
					(Std)	2	24.10%	37	35.77%	24.48%
					(Dom)	4	10.15%	13	16.59%	9.99%
SP_pRand_dUnif_Mixed		✓		262.80	(Lay)	-	-	-	-	-
					(Std)	0	6.74%	-	13.36%	7.64%
					(Dom)	10	-	1	0.05%	0%
SP_pZero_dRand_Mixed	✓			247.00	(Lay)	-	-	-	-	-
					(Std)	2	19.25%	16	26.63%	19.47%
					(Dom)	4	9.43%	13	16.40%	10.19%
SP_pQCri_dRand_Mixed	✓			247.00	(Lay)	-	-	-	-	-
					(Std)	2	23.86%	30	35.52%	25.37%
					(Dom)	4	9.15%	12	16.26%	10.19%
SP_pRand_dRand_Mixed				268.40	(Lay)	-	-	-	-	-
					(Std)	1	6.03%	179	11.00%	9.48%
					(Dom)	10	-	26	2.34%	1.08%

Table 8.6: SP instances, Mixed uncertainty

8.4.4 Conclusion on numerical experiments

In the numerical experiments, we evaluated the performance of formulations for budgeted uncertainty sets, and uncertainty sets obtained by union or intersection of budgeted uncertainty sets. For budgeted uncertainty sets, numerical tests showed that (Dom) outperforms the previously known formulation (Lay), that was dedicated to budgeted uncertainty. An advantage of formulation (Dom) over (Lay) is that the size of (Dom) is independent of the budget Γ, while (Lay) has $O(n\Gamma)$ variables.

We then investigated the impact of the parameters on the performance of (Dom). The influence of uniform deviation and small uncertainty budget is important, while the impact of critical precedence graphs is not significant on the efficiency of the formulation. Interestingly (Dom) is efficient for instances that are not matching the polyhedral characterization case, but where deviation $\widehat{\delta}$ is uniform and Γ is small.

The proposed approach is to precompute the L_{ij}^{Δ} values, then solve the obtained MIP formulation. This allowed us to solve the problem for a variety of new uncertainty sets. The precomputing time remains small (at most 5 seconds for partition-budgeted uncertainty sets) on the considered instances, and the MIP computation time for (Dom) is comparable to that under budgeted uncertainty. While it was expected that (Dom) would outperform standard linearization (Std), the computational interest of applying the dominance is highlighted by the number of solved instances: on a total number of 400 instances, 358 are solved by (Dom),

only 191 by (Std).

Conclusion

In this chapter we investigated a versatile mixed-integer programming approach for the AnchRobPSP problem. This led to a linear formulation that is applicable to any uncertainty set, provided that an algorithm for precomputing the worst-case longest paths values is available. This widens the range of uncertainty sets for which MIP formulations for AnchRobPSP are known. The keypoint for establishing a strong MIP formulation is the analysis of the combinatorial properties of AnchRobPSP, among which a dominance property played a major role. This property allows for a characterization of the anchored sets polytope in interesting special cases. The theoretical positive results for the dominance-based formulation also go together with good numerical performances around the polyhedral characterization case, for both budgeted uncertainty and uncertainty sets with several budgets.

A research direction would be to apply the same dominance idea to the Anchor-Robust RCPSP studied in Chapter 7. Inequalities could be obtained, to strengthen the formulation based on the layered graph. This would require to combine the L_{ij}^{Δ} computation with the sequence variables. It is worth investigating whether dominance may help again in the design of efficient linear formulations, or linear formulations applicable to new uncertainty sets.

Chapter 9

The combinatorial structure of the Anchor-Robust Project Scheduling Problem

In this chapter, we investigate the combinatorial structure of the AnchRobPSP problem defined in Chapter 6. It was mentioned that the combinatorial decisions lie in the choice of the anchored set, rather than in the baseline schedule. We investigate formulations for (AnchRob) with binary anchoring h variables only. A new cover formulation (Cov) is proposed, that is non-compact. Separation algorithms are provided. We then revisit results obtained on mixed-integer formulations (Lay) and (Dom) from Chapter 6 and Chapter 8 respectively. Both (Lay) and (Dom) can be projected out to obtain formulations in h variables only. These projections are obtained explicitly, through new families of inequalities. We propose a unified picture of linear formulations (Lay), (Dom) and (Cov): with both a projected form in h variables, and an extended form with additional continuous variables. The three formulations are compared in the space of h variables. A polyhedral study of the polytope of anchored sets is then undertaken. Facial properties of inequalities from the formulations are exhibited, and new valid inequalities are obtained to strengthen the formulations. Algorithmic perspectives are presented.

9.1 Formulations in anchoring variables

This section is devoted to linear formulations for the (AnchRob) problem in the space of anchoring variables. A new cover formulation is first proposed. Then it is shown how to project mixed-integer formulations from previous chapters, onto the space of h variables. Finally results are given on the comparison of the polytope

of these formulations.

9.1.1 A cover formulation

Let $\mathcal{C}$ denote the set of chains of the poset $(J, \prec)$. Given $C \in \mathcal{C}$, let $j_1, \ldots, j_{|C|}$ denote the jobs of C, ordered for $\prec$. Note that for $C \in \mathcal{C}$, a subset $C' \subseteq C$ is also an element of $\mathcal{C}$. By definition of the poset $(J, \prec)$, chains of $\mathcal{C}$ are in one-to-one correspondence with the set of jobs along an $s-t$ path in the transitive closure $\overline{G}$. Indeed chain C formed with jobs $j_1, \ldots, j_{|C|}$ can be associated with the $s-t$ path $(s, j_1), (j_1, j_2), \ldots, (j_{|C|}, t)$ in $\overline{G}$. Given $C \in \mathcal{C}$, let $\mathcal{A}(C)$ denote the arcs of the associated path from j_1 to $j_{|C|}$, i.e., $\mathcal{A}(C) = \{(j_1, j_2), \ldots, (j_{|C|-1}, j_{|C|})\}$. Let $\mathcal{A}(s, C) = \{s, j_1\} \cup \mathcal{A}(C)$.

Let $\ell^\Delta(C)$ denote the length of C in G^Δ, that is,

$$\ell^\Delta(C) = \sum_{(i,j) \in \mathcal{A}(s,C)} L^\Delta_{ij} + L^0_{j_{|C|}t}.$$

A chain $C \in \mathcal{C}$ is *long* if $\ell^\Delta(C) > M$, *short* otherwise. Let $\mathcal{C}^{>M}$ (resp. $\mathcal{C}^{\leq M}$) denote the set of long chains (resp. short chains). Importantly, function ℓ^Δ is not monotone: it can be that $C' \subseteq C$ and $\ell^\Delta(C') > \ell^\Delta(C)$. That is, a short chain may have a long subchain, as shown in Example 9.1.

Example 9.1. Consider a set of 4 jobs. The precedence arcs are $(1,2)$, $(2,4)$, $(1,3)$, $(3,4)$. Let $p = (1,1,1,1)$. Consider budgeted uncertainty with deviation $\widehat{\delta}$ and budget $\Gamma = 1$. Then $L^\Delta_{12} = 1 + \widehat{\delta}_1$, $L^\Delta_{24} = 1 + \widehat{\delta}_2$ and $L^\Delta_{14} = 2 + \max\{\widehat{\delta}_1, \widehat{\delta}_2, \widehat{\delta}_3\}$. Consider the chain $C = (1,2,4)$ and its subchain $C' = (1,4)$. Then $\ell^\Delta(C) = 0 + 2 + \widehat{\delta}_1 + \widehat{\delta}_2 + 1$ and $\ell^\Delta(C') = 0 + 2 + \max\{\widehat{\delta}_1, \widehat{\delta}_2, \widehat{\delta}_3\} + 1$. E.g. for $\widehat{\delta} = (1,1,3,1)$ it comes that $\ell^\Delta(C) < \ell^\Delta(C')$. ◇

A consequence of the graph model from Chapter 6 is as follows. By Theorem 6.1, a set $H \subseteq J$ is anchored if and only if every chain contained in H is short, or equivalently, $\ell^\Delta(C \cap H) \leq M$ for every $C \in \mathcal{C}$. Hence the AnchRobPSP problem is

$$\begin{array}{lll} \text{AnchRobPSP:} & \max & w(H) \\ & \text{s.t.} & \ell^\Delta(C \cap H) \leq M \quad \forall C \in \mathcal{C} \end{array}$$

Consider the constraint $\ell^\Delta(C \cap H) \leq M$. It can be written as: if $C \in \mathcal{C}^{>M}$, then not all elements of C should be selected in set H. It yields the *long chain cover* inequalities

$$h(C) \leq |C| - 1 \qquad \forall C \in \mathcal{C}^{>M} \tag{LCCov}$$

where $h(C)$ is a shorthand notation for $\sum_{i \in C} h_i$. Consider the following polytope, defined by (LCCov) inequalities

$$\mathcal{P}^{\text{Cov}} = \{h \in [0,1]^J : \ h \text{ satisfies (LCCov)}\}$$

Let (Cov) be the formulation defined by linear inequalities from $\mathcal{P}^{\text{Cov}}$ and integrality constraints $h \in \{0,1\}^J$. It comes that

Proposition 9.1. *The formulation (Cov) is valid for AnchRobPSP.*

It gives a first formulation in h variables for AnchRobPSP. The family of inequalities (LCCov) has exponential size. Let us write the corresponding separation problem.

Separation of (LCCov)	
Input:	G^Δ, $M \geq 0$, $h^* \in [0,1]^J$
Problem:	find $C \in \mathcal{C}$ such that $\ell^\Delta(C) > M$ and $h^*(C) > \lvert C\rvert - 1$, or answer that there is no such chain.

Theorem 9.1. Separation of (LCCov) *is NP-hard.*

Proof. Let us prove the NP-hardness result, by a reduction from the Knapsack problem. Let $\mathcal{I}_{KP}$ be an instance of the Knapsack problem with m items integer weights w_i and integer values v_i, knapsack capacity W and target value V. Assume w.l.o.g. that every item fits in the knapsack: $w_i \leq W$. Let us build an instance $\mathcal{I}_{Sep}$ of Separation of (LCCov) as follows. The construction of the instance is similar to the reduction in proof of Theorem 6.5.

- The set of jobs is $J = \{0, 1, \ldots, m\}$ where the items $1, \ldots, m$ are indexed by increasing value: $v_1 \leq \cdots \leq v_m$; and 0 is an additional job.
- The precedence graph is the path $(s, 0, 1, \ldots, m, t)$ and processing times are zero.
- Deviations are defined by $\widehat{\delta}_i = v_{i+1}$ for every $i \in \{0, 1 \ldots, m-1\}$ (hence $\widehat{\delta}_i \leq \widehat{\delta}_{i+1}$ for every i) and $\widehat{\delta}_m = 0$.
- Budget is $\Gamma = 1$

Arc-weights of G^Δ are then: $L^\Delta_{ij} = \widehat{\delta}_{j-1} = v_j$ for every $i \in J \cup \{s\}$, $j \in J$ and $L^0_{jt} = 0$ for every $j \in J$. The rest of the instance $\mathcal{I}_{Sep}$ is

- $M = V - 1$
- $h^*_i = 1 - \frac{w_i}{W+1}$ for every $i \in \{1, \ldots, m\}$ and $h^*_0 = 0$.

Let $C \subseteq J$ be a subset of jobs, and S the associated subset of items defined by $S = C \setminus \{0\}$. Let us show that S is a subset of items of the knapsack such that $\sum_{i \in S} v_i \geq V$ and $\sum_{i \in S} w_i \leq W$ if and only if the (LCCov) inequality associated with C is violated in $\mathcal{I}_{Sep}$. Note first that $\ell^\Delta(C) = \sum_{i \in S} v_i$. Hence $\ell^\Delta(C) > M$ is

equivalent to $\sum_{i\in S} v_i > V - 1$, or $\sum_{i\in S} v_i \geq V$ since V and the v_i's are integer. Also $\sum_{j\in C}(1 - h^*_j) = (\sum_{i\in S} w_i)/(W + 1)$, hence $h(C) > |C| - 1$ is equivalent to $(\sum_{i\in S} w_i)/(W + 1) < 1$, or $\sum_{i\in S} w_i \leq W$ since W and the w_i's are integer. This proves the claimed NP-hardness result. □

Theorem 9.2. Separation of (LCCov) *admits a pseudo-polynomial algorithm. In an integer point $h \in \{0, 1\}^J$, separation of constraints (LCCov) can be done in polynomial time.*

Proof. Note that the separation problem can be cast as a constrained longest path problem in G^Δ. The problem is to find a path $C \in \mathcal{C}$ such that $\ell^\Delta(C) > M$ and $\sum_{i\in C}(1 - h^*_i)$ is minimized. If the minimum value is less than 1, then an optimal solution C^* yields a violated (LCCov) inequality; otherwise, all (LCCov) inequalities are satisfied at h^*.

This problem can be solved in pseudo-polynomial time. Let us describe the dynamic programming algorithm when all L^Δ_{ij} and L^0_{ij} are integers. Let $\bar{L}$ be the length of the longest $s-t$ path in G^Δ (then $\bar{L} > M$). For every $L \in \{0, \ldots, \bar{L}\}$, for every $j \in J$, let $\lambda_j[L]$ be a label representing the minimum of $\sum_{i\in C'}(1 - h^*_i)$ where $C' \in \mathcal{C}$ has last vertex $j_{|C'|} = j$ and $\sum_{(i,i')\in\mathcal{A}(s,C')} L^\Delta_{ii'} = L$. Then the λ labels satisfy the dynamic programming equations:

$$\begin{array}{ll}
\lambda_s[L] = 0 & \forall L \in \{0, \ldots \bar{L}\} \\
\lambda_j[L] = \min_{i\prec j} \left(\lambda_i[L - L^\Delta_{ij}] + 1 - h^*_j\right) & \forall j \in J, \forall L \in \{0, \ldots, \bar{L}\} \\
\lambda_t[L] = \min_{i\in J} \left(\lambda_i[L - L^0_{it}]\right) & \forall L \in \{0, \ldots \bar{L}\}
\end{array}$$

and the optimal value of the constrained longest path problem is

$$\min_{L\in\{M+1,\ldots,\bar{L}\}} \lambda_t[L].$$

If $h^* \in \{0, 1\}^J$, the inequality $h^*(C) > |C| - 1$ is equivalent to $h^*_j = 1$ for every $j \in C$. Let H be the set with incidence vector h^*. The separation problem can then be solved in polynomial time as follows. Compute the longest $s-t$ path in the subgraph of G^Δ induced by H: if it is greater than M, return a longest path C^*, otherwise all (LCCov) inequalities are satisfied. □

9.1.2 Formulations obtained by projections or extensions

9.1.2.1 Projections

We previously obtained formulations for AnchRobPSP involving two categories of variables: anchoring h variables and additional schedule variables. Formulation (Lay) was introduced in Chapter 6 and formulation (Dom) introduced in Chapter 8. We refer to Chapter 8, pages 192–193 for definitions of both formulations. We now show how to project explicitly those two formulations on the space of h variables. The projections then are valid formulations in h variables only.

Let $\mathcal{P}^{\text{Lay}}$ (resp. $\mathcal{P}^{\text{Dom}}$) denote the polytope of formulation (Lay) (resp. of formulation (Dom)). Recall that they are not in the same variable space, since (Lay) uses continuous variables $(x^\gamma)_{\gamma\in\{0,\dots,\Gamma\}} \in \mathbb{R}^{\overline{J}\times\{0,\dots,\Gamma\}}$ and (Dom) uses continuous variables $z \in \mathbb{R}^{\overline{J}}$.

Theorem 9.3. *Consider the family of inequalities*

$$\sum_{(i,j)\in\mathcal{A}(s,C)} \left(L^0_{ij} + (L^\Delta_{ij} - L^0_{ij})h_j\right) + L^0_{j_{|C|}t} \leq M \quad \forall C \in \mathcal{C} \tag{CDom}$$

The projection of (Dom) formulation onto h variables is defined by inequalities (CDom)*, that is,*

$$\text{Proj}_h(\mathcal{P}^{\text{Dom}}) = \left\{h \in [0,1]^J : h \text{ satisfies (CDom)}\right\}.$$

Proof. Let $h \in [0,1]^J$. Let $G^\Delta(h)$ denote the transitive closure of G, where arc (i,j) is given the weight $L^0_{ij} + (L^\Delta_{ij} - L^0_{ij})h_j$, with $h_t = 0$ for the ease of notation. Then $(z,h) \in \mathcal{P}^{\text{Dom}}$ if and only if z is a schedule of $G^\Delta(h)$ with makespan $z_t \leq M$, by definition of formulation (Dom). The existence of such z is equivalent to $L_{G^\Delta(h)}(s,t) \leq M$; or equivalently, every $s{-}t$ path in $G^\Delta(h)$ has length at most M. The length of path $C \in \mathcal{C}$ in $G^\Delta(h)$ is exactly the left-hand side of the proposed inequality, hence the result. □

Theorem 9.4. *Consider the family of inequalities*

$$\sum_{(i,j)\in\mathcal{A}(s,C)} \left(L^\Delta_{ij} - D_j(1-h_j)\right) + L^0_{j_{|C|}t} \leq M \quad \forall C \in \mathcal{C} \tag{CLay}$$

The projection of (Lay) formulation onto h variables is defined by inequalities (CLay)*, that is,*

$$\text{Proj}_h(\mathcal{P}^{\text{Lay}}) = \left\{h \in [0,1]^J : h \text{ satisfies (CLay)}\right\}.$$

Proof. Let $h \in [0,1]^J$. Let $\ell_{G^{\text{lay}}(h)}(R)$ denote the length of a path R in the layered graph $G^{\text{lay}}(h)$. It holds that $h \in \text{Proj}_h(\mathcal{P}^{\text{Lay}})$ if and only if there exists x a schedule of $G^{\text{lay}}(h)$ such that $x_t^\Gamma \leq M$. The existence of x is equivalent to the longest path condition: $L_{G^{\text{lay}}(h)}(s^\Gamma, t^\Gamma) \leq M$; or equivalently, $\ell_{G^{\text{lay}}(h)}(R) \leq M$ for every path R from s^Γ to t^Γ in the layered graph $G^{\text{lay}}(h)$. Let us now show that it is equivalent to $\sum_{(i,j)\in\mathcal{A}(s,C)} \left(L_{ij}^\Delta - D_j(1-h_j)\right) + L^0_{j_{|C|}t} \leq M$ for every $C \in \mathcal{C}$.

Assume first $\ell_{G^{\text{lay}}(h)}(R) \leq M$ for every path R from s^Γ to t^Γ in $G^{\text{lay}}(h)$. Let $C \in \mathcal{C}$ be an s–t path in the transitive closure of G. Consider an associated s^Γ–t^Γ path R^* in the layered graph built as follows: for every arc $(i,j) \in \mathcal{A}(s,C)$, path R^* contains the subpath of length L_{ij}^Δ going from i^Γ to a copy j^γ of j, and the vertical arc (j^γ, j^Γ); path R^* also contains the subpath of length $L^0_{j_{|C|}t}$ going from ${j_{|C|}}^\Gamma$ to t^Γ. Then R^* is an s^Γ–t^Γ path in $G^{\text{lay}}(h)$, by assumption it has length at most M, hence $\sum_{(i,j)\in\mathcal{A}(s,C)} \left(L_{ij}^\Delta - D_j(1-h_j)\right) + L^0_{j_{|C|}t} \leq M$.

Conversely assume $\sum_{(i,j)\in\mathcal{A}(s,C)} \left(L_{ij}^\Delta - D_j(1-h_j)\right) + L^0_{j_{|C|}t} \leq M$ for every $C \in \mathcal{C}$. Let R be an s^Γ–t^Γ path in $G^{\text{lay}}(h)$. Let C^* be the subset of successive jobs j such that R contains a vertical arc (j^γ, j^Γ); then $C^* \in \mathcal{C}$. For every $(i,j) \in C^*$, let $R^{(i,j)}$ denote the subpath of R from i^Γ to a copy j^γ of job j. Since $R^{(i,j)}$ uses at most Γ transversal arcs, it comes $\ell_{G^{\text{lay}}(h)}(R^{(i,j)}) \leq L_{ij}^\Delta$. Similarly, let $R^{(j_{|C^*|},t)}$ denote the subpath of R between ${j_{|C^*|}}^\Gamma$ and t^Γ, then $\ell_{G^{\text{lay}}(h)}(R^{(j_{|C^*|},t)}) \leq L^0_{j_{|C^*|}t}$. The total length of R is $\ell_{G^{\text{lay}}(h)}(R) = \sum_{(i,j)\in\mathcal{A}(s,C^*)}(\ell_{G^{\text{lay}}(h)}(R^{(i,j)}) - D_j(1-h_j)) + \ell_{G^{\text{lay}}(h)}(R^{(j_{|C^*|},t)})$, thus upper-bounded by $\sum_{(i,j)\in\mathcal{A}(s,C^*)}(L_{ij}^\Delta - D_j(1-h_j)) + L^0_{j_{|C^*|}t}$. By assumption this quantity is at most M since $C^* \in \mathcal{C}$, hence $\ell_{G^{\text{lay}}(h)}(R) \leq M$. This proves the claimed result. □

Both projections are defined by an exponential number of inequalities, since $\mathcal{C}$ has exponential size. Note that by contrast with (LCCov) inequalities, inequalities (CLay) and (CDom) concern all chains, and not only long chains.

Inequalities (CDom) and (CLay) can both be separated in polynomial time, since the corresponding separation problem can be cast as a longest path problem. Formulations (Lay) and (Dom) can be regarded as extended formulations for the polytopes defined by (CLay) and (CDom) respectively. For example for formulation (Dom), the separation of (CDom) inequalities is to compute a longest path in $G^\Delta(h)$. Schedule z corresponds to labels for computing such a longest path by dynamic programming. Note that only a polynomial number of additional schedule variables suffices to obtain a compact valid extended formulation.

9.1.2.2 Extended form of formulation (Cov)

By contrast, the formulation (Cov) was natively obtained in exponential form. The separation problem was solved in pseudo-polynomial time by dynamic pro-

gramming, as shown in Theorem 9.2. The dynamic programming scheme can be used to design an extended formulation for (Cov) that has pseudo-polynomial size. Consider a real decision variable λ_j^L for every $j \in J$, $L \in \{0, \dots, \bar{L}\}$.

Proposition 9.2. *Consider the polytope* $\mathcal{P}^{\text{Ext(Cov)}}$ *defined by inequalities*

$$\begin{array}{lll}
\lambda_j^L \leq \lambda_i^{L-L_{ij}^{\Delta}} + 1 - h_j & \forall i \prec j, j \neq t & \forall L \in \{L_{ij}^{\Delta}, \dots, \bar{L}\} \\
\lambda_t^L \leq \lambda_i^{L-L_{it}^{0}} & \forall i \in J & \forall L \in \{L_{it}^{0}, \dots, \bar{L}\} \\
\lambda_t^L \geq 1 & & \forall L \in \{M+1, \dots, \bar{L}\} \\
\lambda_j^L \geq 0 & \forall j \in J & \forall L \in \{0, \dots, \bar{L}\} \\
h_j \in [0,1] & \forall j \in J &
\end{array}$$

Then $\text{Proj}_h(\mathcal{P}^{\text{Ext(Cov)}}) = \mathcal{P}^{\text{Cov}}$.

Proof. Let $h \in \mathcal{P}^{\text{Cov}}$. Since h satisfies all (LCCov) inequalities, applying the dynamic programming algorithm of Theorem 9.2 we obtain λ labels such that $\min_{L \in \{M+1,\dots,\bar{L}\}} \lambda_t[L] \geq 1$. Then it is clear that the λ labels satisfy all inequalities defining $\mathcal{P}^{\text{Ext(Cov)}}$. Hence $h \in \text{Proj}_h(\mathcal{P}^{\text{Ext(Cov)}})$.

Conversely, let $h \notin \mathcal{P}^{\text{Cov}}$. The output of the dynamic programming algorithm is thus a label $\lambda_t[L^t] < 1$ for some $L^t \in \{M+1, \dots, \bar{L}\}$. Backtracking the algorithm we obtain a path $(s, L^s), \dots, (i, L^i), \dots, (t, L^t)$ such that for every successive i, j, $L^j = L^i + L_{ij}^{\Delta}$, and the sum of $1 - h_j$ values along the path is less than one. Consider the inequalities in the definition of $\mathcal{P}^{\text{Ext(Cov)}}$: summing along this path yields the inequality $\lambda_t^{L^t} \leq \sum_{j \in C}(1 - h_j) < 1$, in contradiction with inequality $\lambda_t^{L^t} \leq 1$. Thus $h \notin \text{Proj}_h(\mathcal{P}^{\text{Ext(Cov)}})$. □

In particular, this implies that the inequalities defining $\mathcal{P}^{\text{Ext(Cov)}}$ and integrality constraints $h \in \{0,1\}^J$ define a valid formulation for AnchRobPSP, with a pseudo-polynomial size.

The obtained results show that we obtained three formulations (Lay), (Dom), (Cov), each of them with an exponential-size projected form in h variables only; and a polynomial or pseudo-polynomial extended form with additional variables. The whole picture of formulations is summarized in Figure 9.1.

Δ		Formulations		
budgeted	(Lay)	Projected form	$\sum_{(i,j)\in\mathcal{A}(s,C)}\left(L^{\Delta}_{ij}-D_j(1-h_j)\right)+L^0_{j_{\mid C\mid}t}\le M \quad \forall C\in\mathcal{C}$ separation: polynomial	(CLay)
		Extended form	x schedule of $G^{\text{lay}}(h)$ such that $x^{\Gamma}_t\le M$ #vars: $(n+2)(\Gamma+1)$	
any with L^{Δ}	(Dom)	Projected form	$\sum_{(i,j)\in\mathcal{A}(s,C)}\left(L^0_{ij}+(L^{\Delta}_{ij}-L^0_{ij})h_j\right)+L^0_{j_{\mid C\mid}t}\le M \quad \forall C\in\mathcal{C}$ separation: polynomial	(CDom)
		Extended form	z schedule of $G^{\Delta}(h)$ such that $z_t\le M$ #vars: $n+2$	
any with L^{Δ}	(Cov)	Projected form	$\sum_{j\in C}h_j\le\mid C\mid-1 \qquad \forall C\in\mathcal{C}^{>M}$ separation: NP-hard	(LCCov)
		Extended form	λ labels of dynamic programming from Thm. 9.2 #vars: $(n+2)(\bar{L}+1)$	

Figure 9.1: Formulations for AnchRobPSP in projected and extended forms.

9.1.3 Comparison of formulations

In this section, we address the comparison of formulations (Lay), (Dom) and (Cov). We compare their associated polytope, and thus their linear bounds.

Formulations (Lay) and (Dom). Recall that it was proven in Chapter 8 that formulation (Dom) is stronger than formulation (Lay), i.e., $\text{Proj}_h(\mathcal{P}^{\text{Dom}})\subseteq\text{Proj}_h(\mathcal{P}^{\text{Lay}})$ under assumption (IneqD_j). This was proven by building a point in the (x,h) space associated with a vector $h\in\text{Proj}_h(\mathcal{P}^{\text{Dom}})$.

Using the explicit description of projections from Theorem 9.3 and Theorem 9.4 this result can now be seen easily. Consider the left-hand sides of (CLay) and (CDom) inequalities. For every $i\prec j$, it holds that $L^{\Delta}_{ij}-D_j(1-h_j)\le L^0_{ij}+(L^{\Delta}_{ij}-L^0_{ij})h_j$: indeed it is equivalent to $(D_j-L^{\Delta}_{ij}+L^0_{ij})h_j\le D_j-L^{\Delta}_{ij}+L^0_{ij}$ which is clearly satisfied for every $h_j\in[0,1]$, since assumption (IneqD_j) ensures $D_j-L^{\Delta}_{ij}+L^0_{ij}\ge 0$ for every $i\prec j$. Hence the inequalities defining the projection of (Dom) are tighter than the inequalities defining the projection of (Lay).

Formulations (Lay) and (Cov). Let us first compare formulation (Cov) to formulation (Lay), in the case of budgeted uncertainty. Let $C\in\mathcal{C}^{>M}$. Let us say that C is *minimal* if for every $C'\subseteq C$, the subchain C' is short.

Proposition 9.3. *Let $h\in\mathcal{P}^{\text{Cov}}$. Then h satisfies* (CLay) *inequality for every $C\in\mathcal{C}^{\le M}$ and for every $C\in\mathcal{C}^{>M}$ that is minimal.*

Proof. Let $h \in \mathcal{P}^{\mathrm{Cov}}$. Note that inequality (CLay) rewrites as

$$\sum_{j \in C} D_j(1 - h_j) \geq \ell^{\Delta}(C) - M$$

The left-hand side is non-negative since $h \in [0,1]^J$. For $C \in \mathcal{C}^{\leq M}$ the right-hand side $\ell^{\Delta}(C) - M$ is non-positive, so the inequality is satisfied by h.

Consider $C \in \mathcal{C}^{>M}$ that is minimal. Let us prove the following claim: *for every* $j \in C$, $\ell^{\Delta}(C) - M \leq D_j$. Fix some $j \in C$. Let us partition the chain C into $C_{\prec j} = \{i \in C\colon\ i \prec j\}$, $\{j\}$ and $C_{\succ j} = \{i \in C\colon\ j \prec i\}$. Arcs from chain C are either in $C_{\prec j} \cup \{j\}$ or $C_{\succ j} \cup \{j\}$. It holds that

$$\ell^{\Delta}(C) - M = (\ell^{\Delta}(C_{\prec j} \cup \{j\}) - L^0_{jt}) + (\ell^{\Delta}(C_{\succ j} \cup \{j\}) - L^{\Delta}_{sj}) - M.$$

Note that the value $L_{G(p+\widehat{\delta})}(s,j)$ is greater than the length of any $s-j$ path in G^{Δ}, namely $L_{G(p+\widehat{\delta})}(s,j) \geq \ell^{\Delta}(C_{\prec j} \cup \{j\}) - L^0_{jt}$. Since $D_j = L_{G(p+\widehat{\delta})}(s,j) - L^0_{sj}$, we have $D_j \geq \ell^{\Delta}(C_{\prec j} \cup \{j\}) - L^0_{jt} - L^0_{sj}$. Hence

$$\ell^{\Delta}(C) - M \leq D_j + L^0_{sj} + \ell^{\Delta}(C_{\succ j} \cup \{j\}) - L^{\Delta}_{sj} - M.$$

Denote by k the first vertex of $C_{\succ j}$. It holds that $L^0_{sj} + L^{\Delta}_{jk} \leq L^{\Delta}_{sk}$ by definition of the L^{Δ} values. Thus $L^0_{sj} + \ell^{\Delta}(C_{\succ j} \cup \{j\}) - L^{\Delta}_{sj} \leq \ell^{\Delta}(C_{\succ j})$. The subchain $C_{\succ j}$ is a subchain of C hence by assumption on C, it is short, i.e., $\ell^{\Delta}(C_{\succ j}) \leq M$. Finally we obtain $\ell^{\Delta}(C) - M \leq D_j$ and the claim is proven.

Let us now prove the main result. It holds that $\sum_{j \in C} D_j(1-h_j) \geq D_{j^*}(\sum_{j \in C}(1-h_j))$ where $D_{j^*} = \min_{j \in C} D_j$. Since $h \in \mathcal{P}^{\mathrm{Cov}}$ and $C \in \mathcal{C}^{>M}$, the (LCCov) inequality gives $\sum_{j \in C}(1 - h_j) \geq 1$. Then $\sum_{j \in C} D_j(1 - h_j) \geq D_{j^*} \geq \ell^{\Delta}(C) - M$ by the claim. This proves that h satisfies the (CLay) inequality associated with chain C. □

An open question is whether $h \in \mathcal{P}^{\mathrm{Cov}}$ also satisfies (CLay) for long chains that are not minimal. If so, formulation (Cov) would be proven stronger than (Lay). Note that for a large enough value of the D_j values, formulation (Cov) would be stronger than (Lay).

Formulations (Dom) and (Cov). Let us now answer the question of the comparison between the formulations (Cov) and (Dom).

Theorem 9.5. *There exists an instance of AnchRobPSP where* $\mathcal{P}^{\mathrm{Cov}}$ *and* $\mathrm{Proj}_h(\mathcal{P}^{\mathrm{Dom}})$ *are not comparable, in the sense that*

$$\mathcal{P}^{\mathrm{Cov}} \setminus \mathrm{Proj}_h(\mathcal{P}^{\mathrm{Dom}}) \neq \varnothing \text{ and } \mathrm{Proj}_h(\mathcal{P}^{\mathrm{Dom}}) \setminus \mathcal{P}^{\mathrm{Cov}} \neq \varnothing.$$

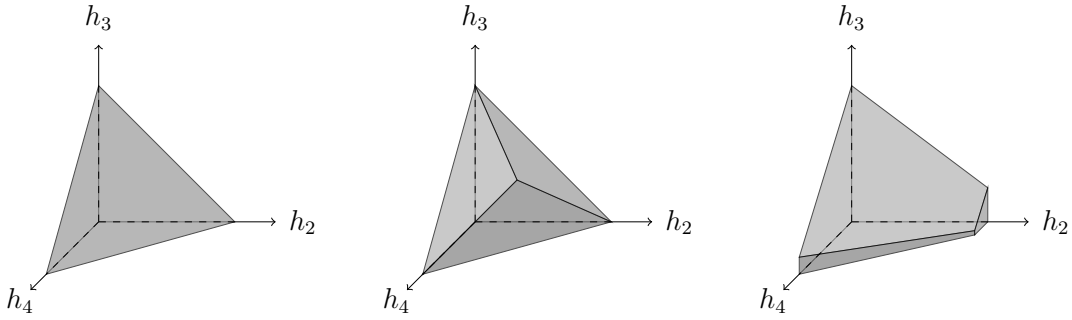


Figure 9.2: Polytopes $\mathcal{Q}$, $\mathcal{P}^{\text{Cov}}$, $\text{Proj}_h(\mathcal{P}^{\text{Dom}})$ on the instance from proof of Theorem 9.5.

Proof. Let us define an instance of AnchRobPSP as follows. Let G be a chain of 4 jobs. Let $p_i = 0$ for every $i \in J$, let $\widehat{\delta} = [6, 8, 7, 0]$ and $\Gamma = 1$ (recall that processing times and $\widehat{\delta}_n$ have no influence on the problem when G is a chain). Let $M = 8$. Let us compute polytopes $\mathcal{Q}$, $\mathcal{P}^{\text{Cov}}$, $\text{Proj}_h(\mathcal{P}^{\text{Dom}})$. They are illustrated in Figure 9.2.

Polytope $\mathcal{Q}$. It can be checked that anchored sets are: the empty set, $\{2\}, \{3\}$, $\{4\}$, and $\{1\}, \{1,2\}, \{1,3\}, \{1,4\}$. In particular job 1 can always be added to the anchored set. Hence $\mathcal{Q} = [0,1] \times \text{conv}\{\chi^{\varnothing}, \chi^{\{2\}}, \chi^{\{3\}}, \chi^{\{4\}}\}$. We have $\mathcal{Q} = \{h \in [0,1]^J\colon\ h_2 + h_3 + h_4 \leq 1\}$. In the space of h_2, h_3, h_4 it yields a tetrahedron.

Polytope $\mathcal{P}^{\text{Cov}}$. Here $\mathcal{P}^{\text{Cov}} = \{h \in [0,1]^J\colon\ h_i + h_j \leq 1\ \forall i, j \in \{2,3,4\},\ h_2 + h_3 + h_4 \leq 2\}$. This last inequality is dominated. Consider the restriction to the space of h_2, h_3, h_4: it is a polytope with 5 extreme points $\chi^{\varnothing}, \chi^{\{2\}}, \chi^{\{3\}}, \chi^{\{4\}}$ and $h^* = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$.

Polytope $\text{Proj}_h(\mathcal{P}^{\text{Dom}})$ is defined by inequalities (CDom) of form $\sum_{(i,j)\in C} L^{\Delta}_{ij} h_j \leq M$ for C a chain in $\mathcal{C}$. It can be checked that the non-dominated (CDom) inequalities are $6h_2 + 8h_4 \leq 8$ and $6h_2 + 8h_3 + 7h_4 \leq 8$. Hence $\text{Proj}_h(\mathcal{P}^{\text{Dom}}) = \{h \in [0,1]^J\colon\ \frac{3}{4}h_2 + h_4 \leq 1,\ \frac{3}{4}h_2 + h_3 + \frac{7}{8}h_4 \leq 1\}$. The restriction to the space of h_2, h_3, h_4 is a polytope with extreme points $\chi^{\varnothing}, \chi^{\{2\}}, \chi^{\{3\}}, \chi^{\{4\}}$ and also (among others) the point $h^\circ = (1, \frac{1}{32}, \frac{1}{4})$.

To prove the theorem, it is sufficient to see that the point $h^* = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$ violates the inequality $\frac{3}{4}h_2 + h_3 + \frac{7}{8}h_4 \leq 1$, and $h^\circ = (1, \frac{1}{32}, \frac{1}{4})$ violates the inequality $h_2 + h_4 \leq 1$. □

Note that in this example, the anchored set polytope has a facet defined by inequality $h_2 + h_3 + h_4 \leq 1$. This inequality is not in family (LCCov) nor (CDom). This motivates the study of the anchored set polytope, to strengthen inequalities (LCCov) or (CDom), identify new valid inequalities, and facets.

9.2 Polyhedral study of the polytope of anchored sets

In this section we study the polytope

$$\mathcal{Q} = \text{conv}\{\chi^H : \ H \text{ anchored set}\}.$$

In Section 9.2.1 simple facets are identified. In Section 9.2.2 polyhedral characterization cases are considered and a minimal description of $\mathcal{Q}$ is given. In Section 9.2.3 and Section 9.2.4 we propose new valid inequalities to strengthen (LCCov) inequalities.

9.2.1 Preliminaries

Let us start with some standard polyhedral results on $\mathcal{Q}$. Let $J^{\text{out}} = \{i \in J : \ L^{\Delta}_{si} + L^{0}_{it} > M\}$. Job i is in J^{out} when the singleton $\{i\}$ is not an anchored set. Note that if $i \in J^{\text{out}}$ then no anchored set contains job i. Then the dimension of $\mathcal{Q}$ is $|J| - |J^{\text{out}}|$. Indeed the polytope $\mathcal{Q}$ is included in $\bigcap_{i \in J^{\text{out}}} \{h_i = 0\}$ hence it has dimension at most $n - |J^{\text{out}}|$. The vectors $\chi^{\{i\}}$ for $i \notin J^{\text{out}}$ are in $\mathcal{Q}$ and linearly independent.

Proposition 9.4. *The inequality $h_i \geq 0$ defines a facet of $\mathcal{Q}$ for every $i \in J$.*

Proof. The vectors $\chi^{\varnothing}$, $\chi^{\{j\}}$ for $j \neq i$ are n vectors of the face $\mathcal{Q} \cap \{h : \ h_i = 0\}$ that are affinely independent. □

Proposition 9.5. *Every facet of $\mathcal{Q}$ distinct from non-negativity can be written as $\alpha^T h \leq \beta$ with $\alpha \in \mathbb{R}^J_+$ and $\beta \in \mathbb{R}_+$.*

Proof. Consider the facet $F = \mathcal{Q} \cap \{\alpha^T h = \beta\}$ with any α, β. If all points in F satisfy $h_i = 0$ for some $i \in J$ then the facet is induced by the non-negativity constraint $h_i \geq 0$. Otherwise, for $i \in J$ there exists $h \in F$ such that $h_i = 1$, and $\alpha_i + \alpha^T(h - \chi^{\{i\}}) = \beta$. Since $h - \chi^{\{i\}} \in \mathcal{Q}$, $\alpha^T(h - \chi^{\{i\}}) \leq \beta$, hence $\alpha_i \geq 0$. Then $\beta \geq 0$ follows. □

Proposition 9.6. *Let $i \in J$. The inequality $h_i \leq 1$ defines a facet if and only if for every $j \neq i$, the pair $\{i, j\}$ can be anchored.*

Proof. If $\{i, j\}$ is not anchored for some $j \neq i$, then the inequality $h_i + h_j \leq 1$ is valid for $\mathcal{Q}$ and it dominates $h_i \leq 1$. Assume $\{i, j\}$ can be anchored for every $j \neq i$. Consider the family of n vectors $\chi^{\{i\}}, \chi^{\{i,j\}}$ for $j \neq i$: they are affinely independent vectors of $\mathcal{Q}$ satisfying $h_i = 1$. Hence $h_i \leq 1$ defines a facet. □

Remark that the inequality $h_i + h_j \leq 1$ used in the necessity proof is of form (LCCov); indeed it writes $h(C) \leq |C| - 1$ with $C = (s, i, j, t)$ (w.l.o.g. $i \prec j$). It holds that $C \in \mathcal{C}^{>M}$ when $\{i, j\}$ is not anchored.

9.2.2 Integrality cases

In this section we revisit integrality cases established in Chapter 8 and exhibit minimal representation for the anchored sets polytope $\mathcal{Q}$ in that cases.

Box uncertainty. For box uncertainty, a polyhedral characterization was obtained in Theorem 8.2, page 196, with inequalities

$$h_j \leq \left\lfloor \frac{M - (L^0_{sj} + L^0_{jt})}{L^\Delta_{sj} - L^0_{sj}} \right\rfloor$$

The right-hand side can be rewritten as $\left\lfloor 1 - \frac{L^\Delta_{sj} + L^0_{jt} - M}{L^\Delta_{sj} - L^0_{sj}} \right\rfloor$, hence it is 0 if $j \in J^{\text{out}}$, 1 otherwise. It was mentioned in Chapter 8 that these inequalities can be derived by rounding from (CDom) inequalities. They also are directly in the family (LCCov), since they write $h(C) \leq |C| - 1$ with $C = \{j\}$, and $j \in J^{\text{out}}$ is equivalent to $C \in \mathcal{C}^{>M}$. These inequalities are trivially facets of $\mathcal{Q}$, hence we obtain the following result.

Proposition 9.7. *For box uncertainty, a minimal characterization of $\mathcal{Q}$ is $h_j = 0$ for every $j \in J^{\text{out}}$, $h_j \in [0, 1]$ for every $j \notin J^{\text{out}}$.*

Unitary case. Let us now consider the unitary case U-AnchRobPSP, as defined page 124. For U-AnchRobPSP, a polyhedral characterization was obtained with formulation (Dom) in Proposition 8.3, page 198. Let $M \geq 1$. In the h space the polyhedral characterization corresponds to inequalities (CDom) that are $h(C) \leq M$ for every $C \in \mathcal{C}$.

A question is to identify, among inequalities (CDom), those which are facet-defining for U-AnchRobPSP. Let us now give a minimal representation of $\mathcal{Q}$ in that case.

Proposition 9.8. *For the U-AnchRobPSP, a minimal representation of $\mathcal{Q}$ is given by the inequalities*

$$\begin{array}{ll} h_j \geq 0 & \forall j \in J \\ h_j \leq 1 & \forall j \in J \\ h(C) \leq M & \forall C \in \mathcal{C}\colon\ C \text{ maximal},\ |C| > M. \end{array}$$

Proof. Using Theorem 8.3 for the polyhedral characterization, and Proposition 9.4 and Proposition 9.6, it is sufficient to prove that $h(C) \leq M$ is a facet if and only if the chain C is maximal and $|C| > M$.

Let us prove necessity. Assume C is not maximal. There exists $C' \in \mathcal{C}$ such that $C \subsetneq C'$: then inequality $h(C') \leq M$ dominates inequality $h(C) \leq M$. Assume

$|C| \leq M$. Then $|C| = M$, otherwise no element of $\mathcal{Q}$ satisfies $h(C) = M$. Every element of $\mathcal{Q}$ such that $h(C) = M = |C|$ also satisfies $h_i = 1$ for every $i \in C$. Hence inequality $h(C) \leq M$ is not facet-defining.

Let us now prove sufficiency. Let $\{h\colon\ \alpha^T h = \beta\}$ be a hyperplane containing $\mathcal{Q} \cap \{h\colon\ h(C) = M\}$. Let us first prove that vector α is zero outside of chain C. Let $i \notin C$. Since C is maximal chain, $C \cup \{i\}$ is not a chain, and there exists $j \in C$ such that $i \not\prec j$ and $j \not\prec i$. Let C' be a subset of C such that $|C'| = M$ and $j \in C'$. Let $h^* = \chi^{C' \cup \{i\}}$. The set $C' \cup \{i\}$ does not contain a chain of size $M + 1$ by definition of j and C', hence $h^* \in \mathcal{Q}$. Hence $h^* \in \mathcal{Q} \cap \{h\colon\ h(C) = M\}$ and $\sum_{k \in C'} \alpha_k + \alpha_i = \beta$. Similarly since $\chi^{C'} \in \mathcal{Q} \cap \{h\colon\ h(C) = M\}$, it comes $\sum_{k \in C'} \alpha_k = \beta$. Thus $\alpha_i = 0$. Let us now prove that all non-zero coordinates of vector α are equal. Let $i, j \in C$, $i \neq j$. Let C' be a subset of C such that $i \notin C'$, $j \in C$, and $|C'| = M$. Such a set exists whenever $M < |C|$. Let $C'' = (C' \setminus \{j\}) \cup \{i\}$. It holds that $\chi^{C'}, \chi^{C''} \in \mathcal{Q} \cap \{h\colon\ h(C) = M\}$. Hence $\sum_{k \in C'} \alpha_k = \beta$ and $\sum_{k \in C''} \alpha_k = \sum_{k \in C'} \alpha_k - \alpha_j + \alpha_i = \beta$. Thus $\alpha_i = \alpha_j$. Therefore $h(C) \leq M$ is facet-defining. □

9.2.3 Conflict graph

Let us now introduce new valid inequalities based on a so-called conflict graph. Given an instance of AnchRobPSP, let $G^\perp = (J, E^\perp)$ be an undirected graph defined by $(i, j) \in E^\perp$ if $\{i, j\}$ cannot be anchored, i.e., $L^\Delta_{si} + L^\Delta_{ij} + L^0_{jt} > M$ (w.l.o.g. $i \prec j$). Clearly if $(i, j) \in E^\perp$, in any anchored set at most one job among i and j can be anchored. That is, any anchored set corresponds to a stable set in the conflict graph. Let $STAB(G^\perp)$ denote the stable set polytope of $G^\perp$. It comes that

$$\mathcal{Q} \subseteq STAB(G^\perp)$$

hence all valid inequalities for the stable set polytope can be used for the anchored set polytope. In particular, a classical class of inequalities are *clique inequalities*. Let $K \subseteq J$ be a clique in $G^\perp$. Then the clique inequality

$$h(K) \leq 1$$

is valid for $\mathcal{Q}$. A classical result is that

Proposition 9.9. *Let K be a clique of $G^\perp$. The clique inequality $h(K) \leq 1$ defines a facet of $\mathcal{Q}$ if and only if K is an inclusion-wise maximal clique in $G^\perp$.*

Indeed maximality is necessary to have the clique inequality non-dominated. The face $\mathcal{Q} \cap \{h\colon\ h(K) = 1\}$ contains the n vectors $\chi^{\{i\}}, i \in K, \chi^{\{i,k\}}, i \notin K$ for a fixed $k \in K$. These vectors are affinely independent.

Remark that in the example of Theorem 9.5, the anchored set polytope has a facet $h_2 + h_3 + h_4 \leq 1$ which is a clique inequality.

The conflict graph $G^{\perp}$ is not any graph but it has structural properties implied by the underlying AnchRobPSP instance. Concerning cliques, we obtain that

Proposition 9.10. *For every clique K of the conflict graph, it holds that $K \in \mathcal{C}$.*

Proof. For every edge $(i, j) \in E^{\perp}$ note that $i \prec j$ or $j \prec i$, otherwise $\{i, j\}$ could be anchored. Hence K is a set of jobs that are all pairwise comparable with $\prec$, i.e., $K \in \mathcal{C}$. □

In particular, all cliques can be found by following a path of the precedence graph. The jobs of a clique can be ordered with precedence relation $\prec$.

9.2.4 Rank inequalities

A classical family of valid inequalities is defined using the *rank* of a subset, defined as follows.

Rank. Given $X \subseteq J$, let $r(X)$ be the maximum number of jobs of X that can be anchored, that is

$$r(X) = \max\{|Y| : \ Y \subseteq X, Y \text{ anchored}\}$$

Then clearly, the rank inequality

$$h(X) \leq r(X) \qquad \forall X \subseteq J$$

is a valid inequality for the polytope of anchored sets $\mathcal{Q}$. A first question is the complexity of computing the rank of a subset.

Rank computation	
Input:	G^{Δ}, M, $X \subseteq J$
Problem:	compute $r(X)$.

Observation 9.1. Rank Computation *is as hard as the AnchRobPSP problem with given G^{Δ} and unit anchoring weights, whose complexity is open.*

Note indeed that the optimum of AnchRobPSP for unit anchoring weights is exactly the maximum number of jobs that can be anchored, that is, the rank $r(J)$. Computing the rank of a subset X is exactly solving AnchRobPSP for anchoring weights equal to one on X, and zero on $J \setminus X$.

Chain-rank. Because the computation of the rank inherits from the complexity of the AnchRobPSP problem, we now define another notion which is the *chain-rank*. The chain-rank is defined on chains $C \in \mathcal{C}$ (instead of any subset X for the rank). The chain-rank $\widetilde{r}(C)$ of chain $C \in \mathcal{C}$ is the max-size of a short subchain Y of C, that is,

$$\widetilde{r}(C) = \max\{|Y| \colon\ Y \subseteq C, \ell^{\Delta}(Y) \leq M\}.$$

Let us point out the difference between the rank and the chain-rank of a chain. The rank of chain C is $r(C) = \max\{|Y| \colon\ Y \subseteq C, \ell^{\Delta}(Y \cap C') \leq M\ \forall C' \in \mathcal{C}\}$. The maximum is attained for a set Y that is anchored. For the chain-rank, only Y must be short. The maximum can be attained in some Y that is not necessarily anchored because it may contain a long subchain.

For every $C \in \mathcal{C}$ it holds that $r(C) \leq \widetilde{r}(C)$ for every $C \in \mathcal{C}$. It follows that the chain-rank inequalities are valid

$$h(C) \leq \widetilde{r}(C) \qquad \forall C \in \mathcal{C}$$

and they are weaker than rank inequalities.

Let us now show that chain-rank inequalities subsume some of the inequalities introduced previously.

Proposition 9.11. *Chain-rank inequalities with right-hand side equal to one, are exactly clique inequalities.*

Proof. As noted in Proposition 9.10, a clique K is in $\mathcal{C}$. The chain-rank of a clique is then $\widetilde{r}(K) = 1$. Indeed, under the assumption that $J^{\text{out}} = \varnothing$, $\widetilde{r}(K) \geq 1$. For any pair of vertices $Y = \{i, j\} \subseteq K$, we have $\ell^{\Delta}(Y) > M$ by definition of the conflict graph. Thus the clique inequality $h(K) \leq 1$ corresponds to the chain-rank inequality $h(K) \leq \widetilde{r}(K)$. Conversely if a set C has chain-rank equal to one, it follows that it is a clique in the conflict graph. □

Proposition 9.12. *Inequalities of family (LCCov) are dominated by chain-rank inequalities.*

Indeed for every long chain $C \in \mathcal{C}^{>M}$, the right-hand side of (LCCov) inequality $|C| - 1$ is an upper-bound for $\widetilde{r}(C)$. Hence the (LCCov) inequality $h(C) \leq |C| - 1$ is dominated by the chain-rank inequality $h(C) \leq \widetilde{r}(C)$.

A question is when chain-rank inequalities are facet-defining for $\mathcal{Q}$. Necessary conditions are the following. The set C must be *non-separable*, i.e., there exists no non-trivial partition of C into C_1 and C_2 such that $\widetilde{r}(C_1) + \widetilde{r}(C_2) \leq \widetilde{r}(C)$. The set C must be *$\widetilde{r}$-maximal*, i.e., for every $C' \in \mathcal{C}$ such that $C \subsetneq C'$, it holds that $\widetilde{r}(C) \leq \widetilde{r}(C')$.

Let us now give complexity results for the algorithmic aspects of chain-rank inequalities. First we consider the problem of computing the chain-rank.

Chain-Rank computation	
Input:	G^Δ, M, $C \in \mathcal{C}$
Problem:	compute $\widetilde{r}(C)$.

Proposition 9.13. *Problem* Chain-Rank computation *is solvable in polynomial time.*

Proof. The problem is to find a subset $Y \subseteq C$, or equivalently, an $s-t$ path in G^Δ with vertex-set Y, such that $\ell^\Delta(Y) \leq M$ and $|Y|$ is maximized. It is a longest path problem with objective function being the number of arcs in the path, which is solvable in polynomial time, by dynamic programming. □

This is in contrast with the result for the rank, since the complexity of rank computation is an open question. The polynomiality of Chain-Rank Computation suggests that inequalities on chains such as (LCCov) could be separated (either heuristically or exactly), then reinforced by computing the chain-rank.

9.3 Linear bounds evaluation

To compare formulations (Lay), (Dom) and (Cov) for the problem, a first question is to evaluate the strength of the linear bound of these formulations. For such preliminary computations, the linear bound of (Lay) and (Dom) can be computed in a straightforward manner since the formulations are compact. For (Cov), using Proposition 9.2 the linear bound can be computed solving the linear relaxation of the extended form of (Cov).

Linear bound computations were carried out on the instances `ER_pQcri_dRand` from Chapter 8, with budget $\Gamma = 1, 2, 3$, and $n = 100, 300, 400$. This instance class corresponds to class ERC from Chapter 6. As shown in Chapter 8, it is a class where compact formulations have a large LP gap (see Table 8.1, page 204).

In Table 9.3, for each class of 10 instances, and each formulation (Lay), (Dom), and (Cov), we report:

- timeLP: the LP computation time, in milliseconds;
- optLP the linear bound, i.e., the optimal value of the LP;
- %imprLP for (Dom) and (Cov), the improvement of the linear bound w.r.t. (Lay), expressed as a percentage of optLP for (Lay).

It can be observed that formulations (Dom) and (Cov) both have tighter linear bounds than (Lay). The improvement is around 9% for (Dom), and up to more than 25% for (Cov). The linear bound of (Cov) largely improves over the linear bounds of the other formulations. However the computation time for the linear bound of (Cov) is two orders of magnitude larger than the computation time for continuous relaxations of (Dom) and (Lay). This is due to the size of the extended formulation for (Cov), which is pseudo-polynomial.

			timeLP(ms)	optLP	%imprLP
$n = 100$	$\Gamma = 1$	(Lay)	30.7	96.177	
		(Dom)	52.8	87.376	-9.15%
		(Cov)	1512.0	80.436	-16.36%
	$\Gamma = 2$	(Lay)	37.9	94.146	
		(Dom)	54.7	84.191	-10.57%
		(Cov)	1586.1	71.521	-24.03%
	$\Gamma = 3$	(Lay)	65.6	93.139	
		(Dom)	58.1	83.181	-10.69%
		(Cov)	1488.7	68.616	-26.32%
$n = 300$	$\Gamma = 1$	(Lay)	115.4	288.773	
		(Dom)	614.7	265.610	-8.02%
		(Cov)	20310.8	235.601	-18.41%
	$\Gamma = 2$	(Lay)	175.0	282.487	
		(Dom)	431.1	257.087	-8.99%
		(Cov)	17426.4	209.210	-25.93%
	$\Gamma = 3$	(Lay)	183.7	279.339	
		(Dom)	303.0	253.678	-9.18%
		(Cov)	19120.8	200.924	-28.07%
$n = 400$	$\Gamma = 1$	(Lay)	1085.2	386.030	
		(Dom)	498.4	355.365	-7.94%
		(Cov)	40547.9	321.997	-16.58%
	$\Gamma = 2$	(Lay)	341.6	378.757	
		(Dom)	488.3	344.816	-8.96%
		(Cov)	37364.1	290.498	-23.30%
	$\Gamma = 3$	(Lay)	344.5	375.232	
		(Dom)	538.3	340.700	-9.20%
		(Cov)	35816.1	280.250	-25.31%

Figure 9.3: Linear bounds of formulations (Lay) (Dom) and (Cov) for instances `ER_pQcri_dRand`, with budget $\Gamma = 1, 2, 3$.

The implementation of (Cov) requires separation algorithms for (LCCov) inequalities, in a Branch&Cut algorithm. A straightforward implementation of (LCCov) separation gave somehow poor results, because of the large number of inequalities separated. These tests are unreported.

Theoretical results from this chapter give promising perspectives to reach an efficient implementation of formulation (Cov). The (LCCov) inequalities could be

reinforced using polyhedral results, e.g., chain-rank inequalities. We note that the evaluation of formulations would require a specific work on the instance benchmark. Indeed numerical performance of formulations are likely to depend strongly on the precedence graph, its density, or the numerical values of deviations. Another perspective is to combine formulations (Dom) and (Cov). Indeed as noted in Theorem 9.5 the two formulations are not comparable. A hybrid formulation could benefit from (Dom) to ensure validity with a compact number of variables and constraints, and from the tight linear bound of (Cov).

Conclusion

We studied the AnchRobPSP problem in the space of anchoring variables. These variables are the combinatorial decisions in AnchRobPSP. First a new cover exponential formulation was proposed. For formulations obtained previously, we were able to establish explicit projections onto the space of anchoring variables. This allows for a comparison of the formulations' polytopes, in a common variable space. We highlighted the connection between longest path separation problems, and additional variables appearing in extended formulations. The variety of valid inequalities obtained for the problem motivated a polyhedral study of the anchored set polytope. In special cases with a polyhedral characterization, the list of all facets was obtained. New inequalities were proposed based on conflict graph or rank, their facial properties were considered.

Theoretical questions could be further investigated on the anchored set polytope. Concepts such as the conflict graph or the rank are quite generic; a question is thus to analyze their structure in the case of AnchRobPSP to obtain more dedicated results. Another perspective is to investigate the similarity with the knapsack problem, since anchoring variables must satisfy knapsack constraints along chains, as shown, e.g., by (CDom) inequalities.

A research direction is also to perform numerical experiments to assess the performance of exponential formulations. An efficient implementation of separation algorithms for (Cov) could lead to an efficient Branch&Cut algorithm, benefiting from the tight linear bound of this formulation. Since (Dom) and (Cov) are not comparable, we could design a hybrid formulation, where a compact formulation such as (Dom) is used for validity, and separation of (Cov) is used for reinforcement.

Conclusion and research perspectives

In this thesis we investigated anchored solutions for discrete optimization problems under uncertainty. Anchored solutions appear to be a quite simple yet practically attractive concept, and a new idea in robust optimization. The anchor-robust problem produces a baseline solution with bounded cost, and a subset of anchored decisions. As emphasized throughout the manuscript, it is a middle ground between static and adjustable-robustness, and it gives control over the price of robustness. The anchor-robust problem belongs to the class of robust 2-stage problems, which is computationally challenging. In spite of its conceptual simplicity, the problem can be very hard to solve. We thus thoroughly investigated the tractability of the anchor-robust problem, first by a study of its complexity and combinatorial structure, then by the design of algorithmic approaches based on mixed-integer programming. This led to a dedicated work, first on combinatorial problems such as spanning trees and matchings, then on project scheduling problems.

For problems written as integer programs in binary variables, we studied the complexity of anchor-reoptimization and anchor-robust optimization under cost uncertainty. We investigated formulations through linear programs (via polyhedral characterization) or mixed-integer programs (via reformulations). The focus was on polynomial combinatorial problems, such as matroid bases.

An important research direction is to extend the results to other combinatorial problems, and in particular NP-hard ones. For the anchor-reoptimization problem, a question is to study the polytope of k-red classical problems, for which new facets arise, starting with the k-red matching polytope. For the anchor-robust problem, a question is to extend the applicability range of the anchor-robust problem to any combinatorial problem with a valid formulation. This implies to design algorithmic approaches where an efficient polyhedral characterization of the original problem is not explicitly required. The price of anchor-robustness should also be further investigated, both theoretically and based on numerical experiments.

For project scheduling problems with starting times as continuous variables, the anchoring level led to new rescheduling problems. We extended the polynomiality result to generalized precedence and tolerance. We pointed out the boundary

between polynomial and NP-hard anchored rescheduling problems. The anchor-robust approach was then investigated for project scheduling. It differentiates itself from the project scheduling literature, that contained mainly proactive problems with continuous stability measures, or adjustable-robust approaches producing no baseline. An important part of the thesis is dedicated to anchor-robust project scheduling.

We first proposed the Anchor-Robust Project Scheduling Problem (AnchRobPSP) in the case of precedence constraints. This problem can be seen as the core of anchor-robustness for project scheduling. We studied the combinatorial structure of the problem and proposed two dedicated graph models, one based on the transitive closure of the precedence graph, the other one on a layered graph dedicated to budgeted uncertainty. The complexity of AnchRobPSP was analyzed for various polyhedral uncertainty sets (box, budgeted, any polytope). In particular, we highlighted the connection with the complexity of computing the worst-case value of longest paths in the precedence graph. Polynomial cases were exhibited, depending on the uncertainty set and the precedence graph.

We proposed mixed-integer programming formulations for AnchRobPSP, strengthened by a study of their polyhedral properties. We obtained first a formulation based on the layered graph, valid for budgeted uncertainty. We then provided other formulations that are applicable to more uncertainty sets, based on the transitive closure graph model. A dominance property was a keypoint to obtain a stronger formulation, which is still compact. A non-compact formulation was also proposed. We showed the connection between the different variable spaces, through projections or extensions of our formulations. This leads to a variety of MIP tools for the problem. They appear to be quite efficient in practice, which is a good sign for the practical implementability of anchor-robustness in project scheduling.

For the AnchRobPSP, we observed the very good numerical performance of compact MIP formulations. An open question is whether the AnchRobPSP is strongly NP-hard. In particular, the complexity of the AnchRobPSP under budgeted uncertainty and unit anchoring weights, is open. A related perspective is to further investigate the anchored set polytope.

We extended the anchor-robust approach to the Resource-Constraint Project Scheduling Problem, which is strongly NP-hard. In the design of an anchor-robust problem, the keypoint was to consider restricted reoptimization, and namely, reoptimization with fixed resource flow. This simplifies the recourse problem, as shown by the complexity analysis of anchored rescheduling problems. Most results obtained on the AnchRobPSP problem then carried over to the Anchor-Robust RCPSP, such as the layered graph for budgeted uncertainty. We proposed exact and heuristic approaches, for both the Adjustable-Robust RCPSP and the Anchor-

Robust RCPSP. Numerical experiments showed that for exact approaches, the computational cost for solving the Adjustable-Robust RCPSP or Anchor-Robust RCPSP was not a lot larger than the original RCPSP. The good quality of heuristic solutions was also highlighted. This is a robustness toolbox that can be used in any practical application of the RCPSP. We highlighted the relevance of the Anchor-Robust RCPSP on an industrial use case at EDF.

A research perspective is to further improve the proposed algorithms on hard instances of the Anchor-Robust RCPSP. A combination of the proposed tools can be designed, benefiting from the efficiency of rule-based heuristics for the RCPSP, and mixed-integer programming for anchor-robustness.

A broader perspective is to extend the anchor-robust approach to other scheduling problems. Note that a specific attention should then be given to the definition of anchored decisions, in the design of new anchor-robust problems. For instance the definition we chose for the RCPSP is structurally based on the resource flow and processing time uncertainty. We may want, e.g., to consider a tolerance in the definition of anchored jobs, as done in rescheduling. Resources can also be treated differently, with non-renewable resources, or resource leveling problems where the objective is to minimize the maximum consumption of resources. Uncertainty on resources (e.g., machine breakdowns) is another research direction of practical interest. Finally anchor-robustness could have an applicative interest on other problems such as production planning problems, combining combinatorial decisions and scheduling decisions.

Appendix

Extended computational results for the Adjustable-Robust RCPSP

Let us present complete computational results for solving the Adjustable-Robust RCPSP with MIP formulation (F_{adj}). The computational settings are those described in Chapter 7, page 161. In Tables 9.1, 9.2, 9.3 are reported the results for $\Gamma = 3, 5, 7$ respectively. For each instance class j30x, x $\in \{1, \ldots, 48\}$, we indicate for (F_{adj}):

– the corresponding parameters NC, RF, RS;

– solved: the number of instances solved to optimality (out of 10);

– time: the computation time averaged on instances solved to optimality in seconds;

– unsolved = 10 - solved

– gap: the final gap, averaged on instances not solved to optimality.

The results are presented in aggregated form in Chapter 7, e.g., in Table 7.1.

class	NC	RF	RS	solved	time(s)	unsolved	gap
1	1.5	0.25	0.2	10	7.21	0	-
2	1.5	0.25	0.5	10	4.34	0	-
3	1.5	0.25	0.7	10	2.36	0	-
4	1.5	0.25	1	10	1.43	0	-
5	1.5	0.5	0.2	0	-	10	16.43%
6	1.5	0.5	0.5	9	151.6	1	1.52%
7	1.5	0.5	0.7	10	9.44	0	-
8	1.5	0.5	1	10	2.56	0	-
9	1.5	0.75	0.2	0	-	10	31.38%
10	1.5	0.75	0.5	3	198.88	7	4.94%
11	1.5	0.75	0.7	8	146.67	2	1.76%
12	1.5	0.75	1	10	5.31	0	-
13	1.5	1	0.2	0	-	10	36.13%
14	1.5	1	0.5	1	23.14	9	6.69%
15	1.5	1	0.7	9	39.23	1	6.0%
16	1.5	1	1	10	4.76	0	-
17	1.8	0.25	0.2	10	4.15	0	-
18	1.8	0.25	0.5	10	2.03	0	-
19	1.8	0.25	0.7	10	1.5	0	-
20	1.8	0.25	1	10	0.94	0	-
21	1.8	0.5	0.2	1	587.97	9	10.76%
22	1.8	0.5	0.5	9	61.41	1	1.47%
23	1.8	0.5	0.7	10	14.33	0	-
24	1.8	0.5	1	10	3.69	0	-
25	1.8	0.75	0.2	0	-	10	31.53%
26	1.8	0.75	0.5	7	193.15	3	2.36%
27	1.8	0.75	0.7	10	11.97	0	-
28	1.8	0.75	1	10	4.71	0	-
29	1.8	1	0.2	0	-	10	39.4%
30	1.8	1	0.5	0	-	10	5.74%
31	1.8	1	0.7	8	42.99	2	4.14%
32	1.8	1	1	10	10.92	0	-
33	2.1	0.25	0.2	10	3.2	0	-
34	2.1	0.25	0.5	10	1.71	0	-
35	2.1	0.25	0.7	10	1.36	0	-
36	2.1	0.25	1	10	0.78	0	-
37	2.1	0.5	0.2	6	361.35	4	17.97%
38	2.1	0.5	0.5	10	55.07	0	-
39	2.1	0.5	0.7	10	5.82	0	-
40	2.1	0.5	1	10	2.83	0	-
41	2.1	0.75	0.2	0	-	10	26.36%
42	2.1	0.75	0.5	6	66.56	4	4.55%
43	2.1	0.75	0.7	10	155.76	0	-
44	2.1	0.75	1	10	4.2	0	-
45	2.1	1	0.2	0	-	10	35.24%
46	2.1	1	0.5	2	483.59	8	6.51%
47	2.1	1	0.7	6	81.55	4	4.19%
48	2.1	1	1	10	4.33	0	-

Table 9.1: Performance of the compact reformulation (F_{adj}) for $\Gamma = 3$.

class	NC	RF	RS	solved	time(s)	unsolved	gap
1	1.5	0.25	0.2	10	17.98	0	-
2	1.5	0.25	0.5	10	9.45	0	-
3	1.5	0.25	0.7	10	4.16	0	-
4	1.5	0.25	1	10	2.23	0	-
5	1.5	0.5	0.2	0	-	10	15.83%
6	1.5	0.5	0.5	8	162.29	2	1.62%
7	1.5	0.5	0.7	10	45.57	0	-
8	1.5	0.5	1	10	6.0	0	-
9	1.5	0.75	0.2	0	-	10	29.86%
10	1.5	0.75	0.5	4	184.07	6	5.72%
11	1.5	0.75	0.7	7	51.11	3	1.11%
12	1.5	0.75	1	10	6.61	0	-
13	1.5	1	0.2	0	-	10	36.17%
14	1.5	1	0.5	2	215.34	8	6.29%
15	1.5	1	0.7	9	26.63	1	3.1%
16	1.5	1	1	10	6.21	0	-
17	1.8	0.25	0.2	10	8.97	0	-
18	1.8	0.25	0.5	10	5.2	0	-
19	1.8	0.25	0.7	10	2.99	0	-
20	1.8	0.25	1	10	1.91	0	-
21	1.8	0.5	0.2	1	551.34	9	14.04%
22	1.8	0.5	0.5	9	126.62	1	2.68%
23	1.8	0.5	0.7	10	17.43	0	-
24	1.8	0.5	1	10	5.5	0	-
25	1.8	0.75	0.2	0	-	10	30.75%
26	1.8	0.75	0.5	7	236.67	3	2.65%
27	1.8	0.75	0.7	10	18.03	0	-
28	1.8	0.75	1	10	6.39	0	-
29	1.8	1	0.2	0	-	10	38.34%
30	1.8	1	0.5	0	-	10	6.17%
31	1.8	1	0.7	8	74.08	2	5.4%
32	1.8	1	1	10	10.13	0	-
33	2.1	0.25	0.2	10	5.98	0	-
34	2.1	0.25	0.5	10	3.65	0	-
35	2.1	0.25	0.7	10	2.82	0	-
36	2.1	0.25	1	10	1.54	0	-
37	2.1	0.5	0.2	3	327.01	7	12.39%
38	2.1	0.5	0.5	9	26.93	1	2.38%
39	2.1	0.5	0.7	10	10.6	0	-
40	2.1	0.5	1	10	4.98	0	-
41	2.1	0.75	0.2	0	-	10	26.76%
42	2.1	0.75	0.5	6	72.76	4	5.06%
43	2.1	0.75	0.7	9	69.47	1	1.23%
44	2.1	0.75	1	10	6.67	0	-
45	2.1	1	0.2	0	-	10	33.59%
46	2.1	1	0.5	3	446.55	7	6.81%
47	2.1	1	0.7	7	140.71	3	3.57%
48	2.1	1	1	10	7.15	0	-

Table 9.2: Performance of the compact reformulation (F_{adj}) for $\Gamma = 5$.

class	NC	RF	RS	solved	time(s)	unsolved	gap
1	1.5	0.25	0.2	10	66.63	0	-
2	1.5	0.25	0.5	10	12.67	0	-
3	1.5	0.25	0.7	10	4.66	0	-
4	1.5	0.25	1	10	3.77	0	-
5	1.5	0.5	0.2	0	-	10	17.86%
6	1.5	0.5	0.5	7	186.02	3	3.55%
7	1.5	0.5	0.7	10	25.1	0	-
8	1.5	0.5	1	10	6.53	0	-
9	1.5	0.75	0.2	0	-	10	30.86%
10	1.5	0.75	0.5	3	269.24	7	6.02%
11	1.5	0.75	0.7	7	149.01	3	2.16%
12	1.5	0.75	1	10	10.02	0	-
13	1.5	1	0.2	0	-	10	38.03%
14	1.5	1	0.5	1	45.89	9	7.24%
15	1.5	1	0.7	9	39.59	1	5.17%
16	1.5	1	1	10	9.48	0	-
17	1.8	0.25	0.2	10	16.87	0	-
18	1.8	0.25	0.5	10	5.01	0	-
19	1.8	0.25	0.7	10	3.88	0	-
20	1.8	0.25	1	10	2.39	0	-
21	1.8	0.5	0.2	1	847.24	9	15.06%
22	1.8	0.5	0.5	8	231.58	2	1.9%
23	1.8	0.5	0.7	10	18.29	0	-
24	1.8	0.5	1	10	8.74	0	-
25	1.8	0.75	0.2	0	-	10	32.62%
26	1.8	0.75	0.5	6	68.88	4	2.75%
27	1.8	0.75	0.7	10	23.98	0	-
28	1.8	0.75	1	10	12.48	0	-
29	1.8	1	0.2	0	-	10	39.47%
30	1.8	1	0.5	0	-	10	7.24%
31	1.8	1	0.7	8	47.78	2	8.07%
32	1.8	1	1	10	13.33	0	-
33	2.1	0.25	0.2	10	8.95	0	-
34	2.1	0.25	0.5	10	4.86	0	-
35	2.1	0.25	0.7	10	4.05	0	-
36	2.1	0.25	1	10	2.18	0	-
37	2.1	0.5	0.2	4	624.37	6	14.42%
38	2.1	0.5	0.5	10	44.43	0	-
39	2.1	0.5	0.7	10	18.27	0	-
40	2.1	0.5	1	10	7.03	0	-
41	2.1	0.75	0.2	0	-	10	26.71%
42	2.1	0.75	0.5	6	98.71	4	4.96%
43	2.1	0.75	0.7	8	71.07	2	1.11%
44	2.1	0.75	1	10	9.38	0	-
45	2.1	1	0.2	0	-	10	34.48%
46	2.1	1	0.5	3	589.12	7	7.56%
47	2.1	1	0.7	6	197.8	4	2.67%
48	2.1	1	1	10	8.61	0	-

Table 9.3: Performance of the compact reformulation (F_{adj}) for $\Gamma = 7$.

Bibliography

V. Aggarwal, Y. P. Aneja, and K. P. K. Nair. Minimal spanning tree subject to a side constraint. *Computers and Operations Research*, 9(4):287–296, 1982.

R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

H. Aissi, C. Bazgan, and D. Vanderpooten. Min-max and min-max regret versions of combinatorial optimization problems: A survey. *European Journal of Operational Research*, 197(2):427–438, 2009.

C. Artigues and F. Roubellat. A polynomial activity insertion algorithm in a multi-resource schedule with cumulative constraints and multiple modes. *European Journal of Operational Research*, 127(2):297–316, 2000.

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

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

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

G. Ausiello, V. Bonifaci, and B. Escoffier. Complexity and Approximation in Reoptimization. Preprint. https://hal.archives-ouvertes.fr/hal-00906941, 2008.

J. Ayoub and M. Poss. Decomposition for adjustable robust linear optimization subject to uncertainty polytope. *Computational Management Science*, 13(2): 219–239, 2016.

E. Bampis, B. Escoffier, M. Lampis, and V. T. Paschos. Multistage matchings. In *16th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2018, June 18-20, 2018, Malmö, Sweden*, pages 7:1–7:13, 2018.

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

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

A. Ben-Tal, O. E. Housni, and V. Goyal. A tractable approach for designing piecewise affine policies in two-stage adjustable robust optimization. *Mathematical Programming*, 182(1):57–102, 2020.

P. Bendotti, P. Chrétienne, P. Fouilhoux, and A. Quilliot. Anchored reactive and proactive solutions to the CPM-scheduling problem. *European Journal of Operational Research*, 261(1):67–74, 2017.

P. Bendotti, P. Chrétienne, P. Fouilhoux, and A. Pass-Lanneau. The Anchor-Robust Project Scheduling Problem. Preprint. https://hal.archives-ouvertes.fr/hal-02144834, 2019.

P. Bendotti, P. Chrétienne, P. Fouilhoux, and A. Pass-Lanneau. Dominance-based linear formulation for the Anchor-Robust Project Scheduling Problem. Preprint. https://hal.archives-ouvertes.fr/hal-02938158, 2020a.

P. Bendotti, P. Chrétienne, P. Fouilhoux, and A. Pass-Lanneau. Anchored rescheduling problems under generalized precedence constraints. In M. Baïou, B. Gendron, O. Günlük, and A. R. Mahjoub, editors, *Combinatorial Optimization. ISCO 2020*, volume 12176 of *Lecture Notes in Computer Science*, 2020b.

A. Berger, V. Bonifaci, F. Grandoni, and G. Schäfer. Budgeted matching and budgeted matroid intersection via the gasoline puzzle. *Mathematical Programming*, 128:355–372, 2011.

D. Bertsimas and C. Caramanis. Finite adaptability in multistage linear optimization. *IEEE Transactions on Automatic Control*, 55(12):2751–2766, 2010.

D. Bertsimas and V. Goyal. On the power and limitations of affine policies in two-stage adaptive optimization. *Mathematical Programming*, 134(2):491–531, 2012.

D. Bertsimas and M. Sim. Robust discrete optimization and network flows. *Mathematical Programming*, 98(1-3):49–71, 2003.

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

D. Bienstock and N. Özbay. Computing robust basestock levels. *Discrete Optimization*, 5(2):389–414, 2008.

A. Billionnet, M. Costa, and P. Poirion. 2-stage robust MILP with continuous recourse variables. *Discrete Applied Mathematics*, 170:21–32, 2014.

J. R. Birge and F. Louveaux. *Introduction to Stochastic Programming*. Springer, 2011.

H.-J. Böckenhauer, J. Hromkovič, T. Mömke, and P. Widmayer. On the hardness of reoptimization. In V. Geffert, J. Karhumäki, A. Bertoni, B. Preneel, P. Návrat, and M. Bieliková, editors, *SOFSEM 2008: Theory and Practice of Computer Science*, pages 50–65. Springer, 2008.

M. Bold and M. Goerigk. A compact reformulation of the two-stage robust resource-constrained project scheduling problem. Preprint. https://arxiv.org/abs/2004.06547, 2020.

E. Boros, K. Borys, K. M. Elbassioni, V. Gurvich, and G. Rudolf. Inapproximability bounds for shortest-path network interdiction problems. Technical report, Rutgers University, 2006.

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

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

C. Buchheim and J. Kurtz. Robust combinatorial optimization under convex and discrete cost uncertainty. *EURO Journal on Computational Optimization*, 6(3): 211–238, 2018.

C. Büsing. Recoverable robust shortest path problems. *Networks*, 59(1):181–189, 2012.

C. Büsing, A. M. C. A. Koster, and M. Kutschka. Recoverable robust knapsacks: Γ-scenarios. In J. Pahl, T. Reiners, and S. Voß, editors, *Network Optimization - 5th International Conference, INOC 2011, Hamburg, Germany, June 13-16, 2011. Proceedings*, volume 6701 of *Lecture Notes in Computer Science*, pages 583–588. Springer, 2011.

K. Calhoun, R. Deckro, J. Moore, J. Chrissis, and J. Hove. Planning and replanning in project and production scheduling. *Omega*, 30:155–170, 2002.

P. Chrétienne. Reactive and proactive single-machine scheduling to maintain a maximum number of starting times. *Annals of Operations Research*, pages 1–14, 2018.

G. D'Angelo, G. Di Stefano, A. Navarra, and C. Pinotti. Recoverable robust timetables: An algorithmic approach on trees. *IEEE Transactions on Computers*, 60:433–446, 2011.

S. V. de Vonder, F. Ballestín, E. Demeulemeester, and W. Herroelen. Heuristic procedures for reactive project scheduling. *Computers and Industrial Engineering*, 52(1):11–28, 2007.

F. Deblaere, E. Demeulemeester, and W. Herroelen. Reactive scheduling in the multi-mode RCPSP. *Computers and Operations Research*, 38(1):63–74, 2011.

R. P. Dilworth. A decomposition theorem for partially ordered sets. *Annals of Mathematics*, 51(1):161–166, 1950.

J. R. Edmonds. Matroids and the greedy algorithm. *Mathematical Programming*, 1(1):127–136, 1971.

G. N. Frederickson and R. Solis-Oba. Increasing the weight of minimum spanning trees. *Journal of Algorithms*, 33(2):244–266, 1999.

V. Gabrel, M. Lacroix, C. Murat, and N. Remli. Robust location transportation problems under uncertain demands. *Discrete Applied Mathematics*, 164:100–111, 2014.

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

A. Gupta, K. Talwar, and U. Wieder. Changing bases: Multistage optimization for matroids and matchings. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, pages 563–575, 2014.

H. W. Hamacher and G. Ruhe. On spanning tree problems with multiple objectives. *Annals of Operations Research*, 52:209–230, 1994.

G. A. Hanasusanto, D. Kuhn, and W. Wiesemann. K-adaptability in two-stage robust binary programming. *Operations Research*, 63(4):877–891, 2015.

Ö. Hazır and G. Ulusoy. A classification and review of approaches and methods for modeling uncertainty in projects. *International Journal of Production Economics*, 223:107522, 2020.

W. Herroelen and R. Leus. Project scheduling under uncertainty: Survey and research potentials. *European Journal of Operational Research*, 165:289–306, 2002.

W. Herroelen and R. Leus. The construction of stable project baseline schedules. *European Journal of Operational Research*, 156(3):550–565, 2004.

O. E. Housni and V. Goyal. On the optimality of affine policies for budgeted uncertainty sets. Preprint. https://arxiv.org/abs/1807.00163, 2019.

M. Hradovich, A. Kasperski, and P. Zieliński. Recoverable robust spanning tree problem under interval uncertainty representations. *Journal of Combinatorial Optimization*, 34(2):554–573, 2017.

E. Israeli and R. K. Wood. Shortest-path network interdiction. *Networks*, 40(2): 97–111, 2002.

A. Kasperski and P. Zieliński. Robust discrete optimization under discrete and interval uncertainty: A survey. In M. Doumpos, C. Zopounidis, and E. Grigoroudis, editors, *Robustness Analysis in Decision Aiding, Optimization, and Analytics*, pages 113–143. Springer International Publishing, 2016.

A. Kasperski and P. Zieliński. Robust recoverable and two-stage selection problems. *Discrete Applied Mathematics*, 233:52–64, 2017.

R. Kolisch and S. Hartmann. Heuristic algorithms for the resource-constrained project scheduling problem: Classification and computational analysis. In J. Weglarz, editor, *Project Scheduling: Recent Models, Algorithms and Applications*, pages 147–178. Springer US, Boston, MA, 1999.

R. Kolisch and A. Sprecher. PSPLIB – a project scheduling problem library. *European Journal of Operational Research*, 96:205–216, 1996.

O. Koné, C. Artigues, P. Lopez, and M. Mongeau. 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(1-2):24–47, 2013.

B. Korte and J. Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer Publishing Company, Incorporated, 5th edition, 2012.

P. Kouvelis and G. Yu. *Robust Discrete Optimization and Its Applications*. Springer, 1996.

S. Lendl, B. Peis, and V. Timmermans. Matroid bases with cardinality constraints on the intersection. Preprint. http://arxiv.org/abs/1907.04741, 2019.

C. Liebchen, M. Lübbecke, R. Möhring, and S. Stiller. The concept of recoverable robustness, linear programming recovery, and railway applications. *Robust and Online Large-Scale Optimization*, 5868:1–27, 2009.

M. Minoux. Models and algorithms for robust PERT scheduling with time-dependent task durations. *Vietnam Journal of Mathematics*, 35, 2007a.

M. Minoux. Duality, Robustness, and 2-stage robust LP decision models. Application to Robust PERT Scheduling. Annales du LAMSADE n.7, 2007b.

M. Minoux. Robust linear programming with right-hand-side uncertainty, duality and applications. In *Encyclopedia of Optimization, Second Edition*, pages 3317–3327. 2009.

K. Mulmuley, U. V. Vazirani, and V. V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987.

E. Nasrabadi and J. B. Orlin. Robust optimization with incremental recourse. Preprint. http://arxiv.org/abs/1312.4075, 2013.

C. H. Papadimitriou and M. Yannakakis. The complexity of restricted spanning tree problems. *Journal of the ACM*, 29(2):285–309, 1982.

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

M. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Springer, Boston, MA, 4th edition, 2002.

R. Rahmaniani, T. G. Crainic, M. Gendreau, and W. Rei. The Benders decomposition algorithm: A literature review. *European Journal of Operational Research*, 259(3):801 – 817, 2017.

R. Ravi and M. X. Goemans. The constrained minimum spanning tree problem. In R. Karlsson and A. Lingas, editors, *Algorithm Theory — SWAT'96*, pages 66–75. Springer Berlin Heidelberg, 1996.

H. Sakkout and M. Wallace. Probe backtrack search for minimal perturbation in dynamic scheduling. *Constraints*, 5:359–388, 2000.

B. Schieber, H. Shachnai, G. Tamir, and T. Tamir. A theory and algorithms for combinatorial reoptimization. *Algorithmica*, 80(2):576–607, 2018.

A. Schrijver. *Combinatorial Optimization – Polyhedra and Efficiency*. Springer-Verlag Berlin Heidelberg, 2003.

O. Şeref, R. K. Ahuja, and J. B. Orlin. Incremental network optimization: Theory and algorithms. *Operations Research*, 57(3):586–594, 2009.

S. F. Smith. *Reactive Scheduling Systems*, pages 155–192. Springer US, Boston, MA, 1995.

F. Sourd and O. Spanjaard. A multiobjective branch-and-bound framework: Application to the biobjective spanning tree problem. *INFORMS Journal on Computing*, 20(3):472–484, 2008.

A. L. Soyster. Technical note – convex programming with set-inclusive constraints and applications to inexact linear programming. *Operations Research*, 21(5): 1154–1157, 1973.

A. Subramanyam, C. E. Gounaris, and W. Wiesemann. K-adaptability in two-stage mixed-integer robust optimization. *Mathematical Programming Computation*, 12(2):193–224, 2020.

J. Valdes, R. E. Tarjan, and E. L. Lawler. The recognition of series parallel digraphs. *SIAM Journal on Computing*, 11(2):298–313, 1982.

T. Yi, K. G. Murty, and C. Spera. Matchings in colored bipartite networks. *Discrete Applied Mathematics*, 121(1):261 – 277, 2002.

R. Yuster. Almost exact matchings. *Algorithmica*, 63:39–50, 2012.

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