Decomposition-based approaches for a class of two-stage robust binary optimization problems

Ayşe N Arslan, Boris Detienne

Decomposition-based approaches for a class of two-stage robust binary optimization problems

Ayşe N. Arslan[1] and Boris Detienne[2,3]

[1]Univ Rennes, INSA Rennes, CNRS, IRMAR - UMR 6625, F-35000 Rennes, France
[2]Univ Bordeaux, CNRS, IMB, UMR 5251, F-33400 Talence, France
[3]Inria Bordeaux-Sud-Ouest

Abstract

In this paper, we study a class of two-stage robust binary optimization problems with objective uncertainty where recourse decisions are restricted to be mixed-binary. For these problems, we present a deterministic equivalent formulation through the convexification of the recourse feasible region. We then explore this formulation under the lens of a relaxation, showing that the specific relaxation we propose can be solved using the branch-and-price algorithm. We present conditions under which this relaxation is exact, and describe alternative exact solution methods when this is not the case. Despite the two-stage nature of the problem, we provide NP-completeness results based on our reformulations. Finally, we present various applications in which the methodology we propose can be applied. We compare our exact methodology to those approximate methods recently proposed in the literature under the name $K-$adaptability. Our computational results show that our methodology is able to produce better solutions in less computational time compared to the $K-$adaptability approach, as well as to solve bigger instances than those previously managed in the literature.

1 Introduction and literature review

Robust optimization is an approach to handling uncertainty in optimization where the probability distributions are replaced with uncertainty sets. In robust optimization, constraints are imposed for all realizations whereas the objective function is evaluated for the worst-case realization within the uncertainty set. As such, in applications where the effects of uncertainty can be catastrophic, robust optimization presents itself as a viable modeling approach. Further, robust optimization models with polyhedral or convex uncertainty sets lead to deterministic equivalent formulations that are often in the same complexity class as their deterministic counterparts. For these reasons, robust optimization has enjoyed and continues to enjoy a growing attention from the research community. Advances in static robust optimization are presented in [3],[5] and [16].

On the other hand, robust optimization can sometimes be "over-conservative", especially when uncertainty does not have a row-independent structure. To remedy this problem, one might consider introducing recourse (or adjustability/adaptability) after the realization of uncertainty. Further, some applications may naturally present with a set of "wait-and-see" decisions that are taken after the realization of uncertainty. This is the case, for instance, where strategic design decisions are undertaken by evaluating the future operational conditions of a system under uncertainty. In robust optimization with recourse, first-stage decisions are evaluated by taking the possibility to "recover" a feasible solution after the realization of

uncertainty into account. The difficulty of these problems have long been established in the literature even in the simple case of two-stage adjustable robust optimization with linear programming problems in both stages, and a polyhedral uncertainty set [4].

Example 1.1. Consider the static robust optimization problem

$$\max_{x\in\{0,1\},\boldsymbol{y}\in\{0,1\}^3} \min_{\xi\in[0,1]} \quad (3-2.5\xi)y_1+(-1+4\xi)y_2+(4-6\xi)y_3$$
$$\text{s.t.} \quad y_1+y_2+y_3\leq x$$

Its optimal solution is $x=y_1=1$, $y_2=y_3=0$, with objective value 0.5. Consider now the possibility to take decisions $\boldsymbol{y}$ after the realization of uncertainty. We write the adjustable robust optimization problem

$$\max_{x\in\{0,1\}} \min_{\xi\in[0,1]} \max_{\boldsymbol{y}\in\{0,1\}^3} (3-2.5\xi)y_1+(-1+4\xi)y_2+(4-6\xi)y_3$$
$$\text{s.t.} \quad y_1+y_2+y_3\leq x$$

In Figure 1, we present the functions $(3-2.5\xi)$, $(-1+4\xi)$ and $(4-6\xi)$. The maximum of these three functions is to be minimized over $\xi\in[0,1]$ when $x=1$. This is achieved at $\xi=0.61$ and yields a value of 1.46. On the other hand when $x=0$ the second-stage value is trivially 0. The optimal solution to the two stage problem is therefore $x=1$ with value 1.46.

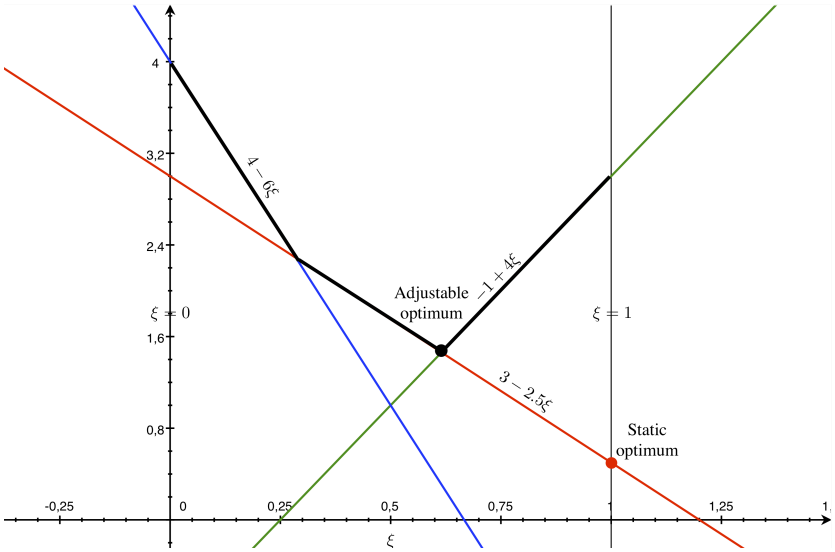


Figure 1: Graphical representation of Example 1.1. The horizontal axis represents the adversarial decision, while the adversarial objective function is over the vertical axis. The colored affine functions each represent a second-stage decision that restrains this value. The bold black line shows the value of the recourse function with respect to the decision of the adversary.

Example 1.1 underlines the value of incorporating wait-and-see decisions into robust optimization. It also demonstrates why doing so is not straightforward. Indeed, for each first-stage solution, evaluating its second-stage cost requires the solution of a bilevel optimization problem where the optimal solution of the outer level is not necessarily an extreme point of the corresponding feasible region.

Existing methodologies in robust adjustable optimization can be categorized as exact or approximate. Exact approaches do not pose any assumptions on the set of possible recourse actions aside from constraints

imposed by the definition of the problem, whereas approximate approaches restrict possible recourse actions to either functions of the uncertain parameters or a preselected subset. On the other hand, all of the exact approaches in the literature consider two-stage models whereas approximations may extend to multiple stages.

Most approximate solution methods for adjustable robust optimization restrict the set of recourse solutions to more or less simple functions of uncertain parameters. These are referred to, in general, as "decision rules". In [4], authors study the complexity of the adjustable robust optimization problem with continuous recourse and propose a linear decision rule that they coin "affine-adjustability". In affine adjustability, continuous recourse decisions are expressed as affine functions of uncertain parameters where the parameters of this affine function are to be optimized. The authors prove that, when the recourse matrix is fixed, if the uncertainty set is tractable then the affinely adjustable robust optimization is tractable. More elaborate decision rule schemes have been explored in the context of robust optimization as well as stochastic and distributionally robust optimization. Some examples of this literature include but are not limited to, [13] where deflected and segregated linear decision rules are considered, [14] where authors propose affine adjustability defined over an extended uncertainty set (extended affine adjustability), and [18] where extended uncertainty sets are again used with piecewise linear decision rules (termed bi-deflected linear decision rules). In [23], authors apply linear decisions rules both in the primal and dual spaces to evaluate the optimality gap resulting from using linear decision rules. In [17] this idea is further generalized to be applied in an extended probability space, encompassing piecewise linear, segregated, and nonlinear decision rules in the original space by a choice of the lifting operator while providing an a posteriori measure of the optimality gap resulting from using decision rules.

Although linear decision rules have desirable properties both from a theoretical and numerical perspective, their application is limited to adaptive problems with continuous recourse. Decision rules have therefore been extended to be able to incorporate binary and integer recourse decisions. In [33], authors develop a conservative approximation for multistage robust MILPs presented in the context of information discovery in multistage stochastic programming. They partition the uncertainty set into hyperrectangles and restrict the continuous and binary recourse decisions to piecewise affine and constant functions of the uncertain parameter over each hyperrectangle, respectively. The resulting conservative approximation can be formulated as an MILP; see also [19]. In [8], authors propose a piecewise constant decision rule for binary recourse variables used in conjunction with a scenario generation scheme resulting in an endogenous design of the decision rule. In [9], piecewise constant functions are again used to describe linearly parameterized binary decision rules, the reformulated problem is mapped to an extended probability space to obtain a deterministic equivalent formulation through linear programming duality. Although the generic reformulation does not scale polynomially in problem data, instances where this is the case are presented.

Another line of recent research in the approximate solution methods literature is based on the idea of restricting the recourse to a preselected set of policies termed "$K-$adaptability" or "finite adaptability". The $K-$adaptability problem consists of selecting a first-stage solution along with K recourse solutions at the first stage. In the second stage, after the realization of uncertainty, the recourse problem reduces to selecting the best solution among these K solutions. As such, this approach is a restriction of the original two-stage problem where the flexibility of actions that can be taken at the second stage is reduced. We remark however that this method becomes exact for binary problems with only objective function uncertainty when K is sufficiently large. The authors of [20] extend the idea of finite adaptability considered in [6], for two-stage robust optimization with pure binary recourse under objective function and right-hand-side uncertainty. They propose a direct solution as a mixed integer program after reformulation

for K fixed. In [30], the concept of K-Adaptability is extended to problems with mixed-integer first- and second-stage feasible regions as well as uncertainty affected technology and recourse matrices. The authors propose a semi-infinite disjunctive programming formulation that imposes that at least one of the K recourse solutions be feasible for every realization of uncertainty. They then propose a branch-and-bound algorithm combining ideas from semi-infinite and disjunctive programming. In [11], authors study K-adaptability for combinatorial optimization problems in the special case where there are no first-stage decisions. For the same type of problems, [12] proposes faster algorithms, when $K = 2$ and the uncertainty set is budgeted. While in these papers, a partition of the uncertainty set and an assignment of K recourse policies to subsets defined by this partition is sought concurrently, it is also possible to define the finite adaptability problem for a given partition of the uncertainty set. This partition is then iteratively improved using the information from the solution obtained. This idea is explored in [7] and [28] in the context of multi-stage adjustable robust mixed-integer optimization.

Most of the exact approaches developed in the literature concern two-stage adjustable robust optimization with continuous recourse and right-hand-side uncertainty. In this case an epigraph formulation can be defined through Benders' type cuts obtained based on the linear programming dual where the subproblem used to identify violated cuts is bilinear. Decomposition-based approaches have been used in [21],[10], and [36] in the context of the well-known unit commitment problem under demand/wind uncertainty. They are presented in a generic framework based on Kelley's cutting plane algorithm in [32]. In [36], authors additionally propose cutting planes expressed in the space of primal variables. They later develop this idea in [35], in a more general context and coin their methodology "constraint-and-column generation". This approach consists of adding a recourse variable and all the associated recourse constraints to the master problem for an identified uncertainty realization that is violated by the current restricted master. The subproblem in this case is a bilevel programming problem. This idea is further explored in [2], which presents a mixed integer programming reformulation for the subproblem based on a Farkas system. Their reformulation is valid in the case where the uncertainty set can be represented as a projection of a binary set, the most important example being the Bertsimas-Sim budgeted uncertainty set. Finally, in [1], authors consider a network design problem under demand uncertainty where the flow decisions on certain arcs can be delayed until after the realization of uncertainty. As such, the recourse problem is a network flow problem with right-hand-side uncertainty. The authors give a projection of the recourse polyhedron to the space of the first-stage variables through an exponential family of inequalities. They propose a cutting-plane algorithm for the solution of this model and show that the separation problem is NP-Hard except for some special cases.

There are few studies that consider exact approaches for two-stage adjustable robust optimization with integer recourse in the literature. In [34], the ideas presented in [35] are extended to the mixed-integer recourse case where the authors propose a nested constraint-and-column generation scheme. Unfortunately, approaches based on on-the-fly generation of uncertainty realizations are no longer finitely convergent when the uncertainty interferes in the objective function or the recourse matrix, as optimal solutions are not necessarily extreme points of the uncertainty set (see Figure 1). Further, their application seems to be restricted to the case where the optimal solution can be defined by adding a very small number of cuts. While this paper was in preparation, [22] proposed an oracle-based solution method based on a relaxation of the combined first- and second-stage feasible regions for two-stage robust binary optimization problems with objective uncertainty. Their method is therefore based on generating columns in the combined space and branching on the first-stage variables, with limited numerical results showing the potential of their approach.

As highlighted by the above review of the existing literature in the domain, there is a need for further re-

search into exact solution methodologies for two-stage robust optimization problems with integer recourse. In this paper, we consider two-stage robust optimization problems of the form

$$\min_{\boldsymbol{x}\in\mathcal{X}} \ \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} + \max_{\boldsymbol{\xi}\in\Xi} \min_{\boldsymbol{y}\in\mathcal{Y}(\boldsymbol{x})} \ (\boldsymbol{f}+\boldsymbol{Q}\boldsymbol{\xi})^{\mathrm{T}}\boldsymbol{y} \tag{1}$$

where $\mathcal{X} \subseteq \mathbb{R}^N_+$, $\mathcal{Y} \subseteq \mathbb{R}^M_+$ are bounded mixed binary sets, and $\Xi \subseteq \mathbb{R}^S$ is a polyhedral set with $\boldsymbol{c}, \boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{f}, \boldsymbol{Q}, \boldsymbol{y}$ of conforming dimensions. We assume throughout the paper that $\boldsymbol{x} = (x_1, \ldots, x_{N_1}, \ldots, x_N)^\top$, denotes $\boldsymbol{x}_1 = (x_1, \ldots, x_{N_1})^\top$ and $\boldsymbol{x}_1 \subseteq \{0,1\}^{N_1}$, and consider $\mathcal{Y}(\boldsymbol{x}) = \{\boldsymbol{y} \in \mathcal{Y} \mid \boldsymbol{H}\boldsymbol{y} \leq \boldsymbol{d} - \boldsymbol{T}\boldsymbol{x}_1\}$. Therefore, the problems we consider in this paper are two-stage robust mixed-binary optimization problems with objective uncertainty.

Remark 1.1. *We remark that both $\mathcal{X}$ and $\mathcal{Y}$ are mixed binary sets. However, the linking constraints in $\mathcal{Y}(\boldsymbol{x})$ involve only binary variables from the first-stage feasibility set.*

The remainder of this paper is organized as follows: in Section 2 we present a single-stage equivalent reformulation of (1) through convexification of the recourse feasible region, $\mathcal{Y}(\boldsymbol{x})$, and propose a computationally attractive representation of $\mathrm{conv}(\mathcal{Y}(\boldsymbol{x}))$. The reformulations we obtain lead to deterministic equivalent models that can be solved through either column generation or column-and-cut generation. We additionally present alternative extended formulations for (1) under certain special cases. In Section 3, we present related complexity results. In Section 4, we illustrate two different applications of our methodology, and present a numerical evaluation of the proposed column generation approaches, comparing them to the direct solution of extended and K-Adaptability formulations. We conclude with future research directions and insights in Section 5.

2 Methodological development

In this section, we present the main results underlying our solution approach. We first present a result that allows us to write (1) as an equivalent deterministic problem. This equivalent formulation is based on the convexification of the recourse feasible region $\mathcal{Y}(\boldsymbol{x})$ for a given first-stage solution $\boldsymbol{x} \in \mathcal{X}$.

Proposition 2.1. *Problem* (1) *is equivalent to*

$$\min_{\boldsymbol{x}\in\mathcal{X}, \boldsymbol{y}\in\mathrm{conv}(\mathcal{Y}(\boldsymbol{x}))} \ \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} + \max_{\boldsymbol{\xi}\in\Xi} \ (\boldsymbol{f}+\boldsymbol{Q}\boldsymbol{\xi})^{\mathrm{T}}\boldsymbol{y} \tag{2}$$

Proof. For a given first-stage solution $\boldsymbol{x} \in \mathcal{X}$, and an uncertainty realization $\boldsymbol{\xi} \in \Xi$, we have that

$$\min_{\boldsymbol{y}\in\mathcal{Y}(\boldsymbol{x})} \ (\boldsymbol{f}+\boldsymbol{Q}\boldsymbol{\xi})^{\mathrm{T}}\boldsymbol{y} = \min_{\boldsymbol{y}\in\mathrm{conv}(\mathcal{Y}(\boldsymbol{x}))} \ (\boldsymbol{f}+\boldsymbol{Q}\boldsymbol{\xi})^{\mathrm{T}}\boldsymbol{y} \tag{3}$$

as the inner minimization problem is a (linear) mixed integer programming problem. This observation allows for exchanging the order of optimization in $\max_{\boldsymbol{\xi}\in\Xi} \min_{\boldsymbol{y}\in\mathrm{conv}(\mathcal{Y}(\boldsymbol{x}))} \ (\boldsymbol{f}+\boldsymbol{Q}\boldsymbol{\xi})^{\mathrm{T}}\boldsymbol{y}$ using the well-known minimax theorem [25], as $(\boldsymbol{f}+\boldsymbol{Q}\boldsymbol{\xi})^{\mathrm{T}}\boldsymbol{y}$ is convex in $\boldsymbol{y}$ and concave in $\boldsymbol{\xi}$, and the sets Ξ and $\mathrm{conv}(\mathcal{Y}(\boldsymbol{x}))$ are convex by definition. □

The inner maximization problem in (2) is linear in $\boldsymbol{\xi}$, as Ξ is a convex set, and therefore can be reformulated using the linear programming dual, yielding a single-stage equivalent formulation of (1). However,

expressing the conditions $\boldsymbol{y} \in \text{conv}(\mathcal{Y}(\boldsymbol{x}))$ for $\boldsymbol{x} \in \mathcal{X}$ remains a challenge as, in general, $\text{conv}(\mathcal{Y}(\boldsymbol{x}))$ cannot be expressed in a compact form and is dependent on $\boldsymbol{x}$. In principal, given $\boldsymbol{x} \in \mathcal{X}$, we may express $\text{conv}(\mathcal{Y}(\boldsymbol{x}))$ using its Dantzig-Wolfe reformulation, *i.e.*, as a convex combination of its extreme points. We then additionally need to impose a relationship between $\boldsymbol{x}$ and $\text{conv}\,\mathcal{Y}(\boldsymbol{x})$. In other words, a recourse solution from $\text{conv}(\mathcal{Y}(\boldsymbol{x}))$ can be selected only if the first-stage solution $\boldsymbol{x}$ is selected. In Figure 2, we illustrate the resulting deterministic equivalent feasible region with two possible first-stage solutions. As should be clear from this figure, the feasible region is made up of two disjunctions, each describing a recourse polyhedron based on the first-stage solution selected.

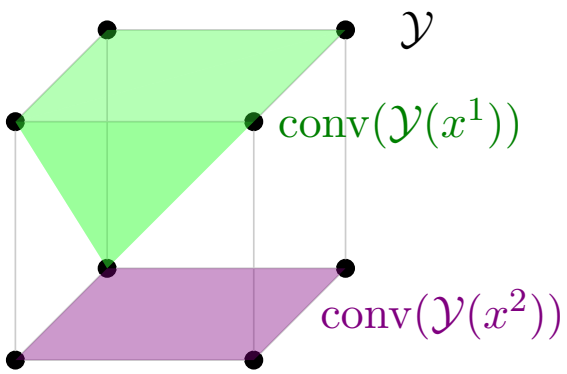


Figure 2: The feasible region of problem (1) after Dantzig-Wolfe reformulation.

2.1 A computationally convenient relaxation

From an algorithmic point of view, expressing the feasible region of (2) in terms of disjunctions is not very useful as it requires an enumeration and encoding of all first-stage extreme point solutions (at least their projection on to the $\{0,1\}^{N_1}$ space). We next explore a relaxation of this formulation that is amenable to algorithmic development.

We define, for $\boldsymbol{x} \in \mathcal{X}$, the set

$$\bar{\mathcal{Y}}(\boldsymbol{x}) = \text{conv}(\mathcal{Y}) \cap \mathcal{Y}(\boldsymbol{x}) = \{\boldsymbol{y} \in \text{conv}(\mathcal{Y}) \mid \boldsymbol{H}\boldsymbol{y} \leq \boldsymbol{d} - \boldsymbol{T}\boldsymbol{x}_1\}\,.$$

Figure 3 illustrates the sets $\text{conv}(\mathcal{Y}(\boldsymbol{x}))$ and $\bar{\mathcal{Y}}(\boldsymbol{x})$ side by side on an example with three binary variables. As the illustration suggests, $\text{conv}(\mathcal{Y}(\boldsymbol{x}))$ is a subset of $\bar{\mathcal{Y}}(\boldsymbol{x})$. We formalize this result in the following proposition.

Proposition 2.2. *For $\boldsymbol{x} \in \mathcal{X}$, we have that* $\text{conv}(\mathcal{Y}(\boldsymbol{x})) \subseteq \bar{\mathcal{Y}}(\boldsymbol{x})$.

Proof. Let $\boldsymbol{y} \in \text{conv}(\mathcal{Y}(\boldsymbol{x}))$, we show that $\boldsymbol{y} \in \bar{\mathcal{Y}}(\boldsymbol{x})$. Firstly, as $\text{conv}(\mathcal{Y}(\boldsymbol{x})) \subseteq \text{conv}(\mathcal{Y})$, then it trivially holds that $\boldsymbol{y} \in \text{conv}(\mathcal{Y})$. Further, we have that $\boldsymbol{H}\boldsymbol{y} \leq \boldsymbol{d} - \boldsymbol{T}\boldsymbol{x}_1$ since $\boldsymbol{y}$ is a convex combination of points that satisfy this constraint. The result follows. □

Proposition 2.2 directly leads to a relaxation of problem (1) and its deterministic equivalent reformulation (2). We have:

$$\min_{\boldsymbol{x}\in\mathcal{X},\boldsymbol{y}\in\text{conv}(\mathcal{Y}(\boldsymbol{x}))} \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} + \max_{\boldsymbol{\xi}\in\Xi} (\boldsymbol{f} + \boldsymbol{Q}\boldsymbol{\xi})^{\mathrm{T}}\boldsymbol{y} \geq \min_{\boldsymbol{x}\in\mathcal{X},\boldsymbol{y}\in\bar{\mathcal{Y}}(\boldsymbol{x})} \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} + \max_{\boldsymbol{\xi}\in\Xi} (\boldsymbol{f} + \boldsymbol{Q}\boldsymbol{\xi})^{\mathrm{T}}\boldsymbol{y}.$$

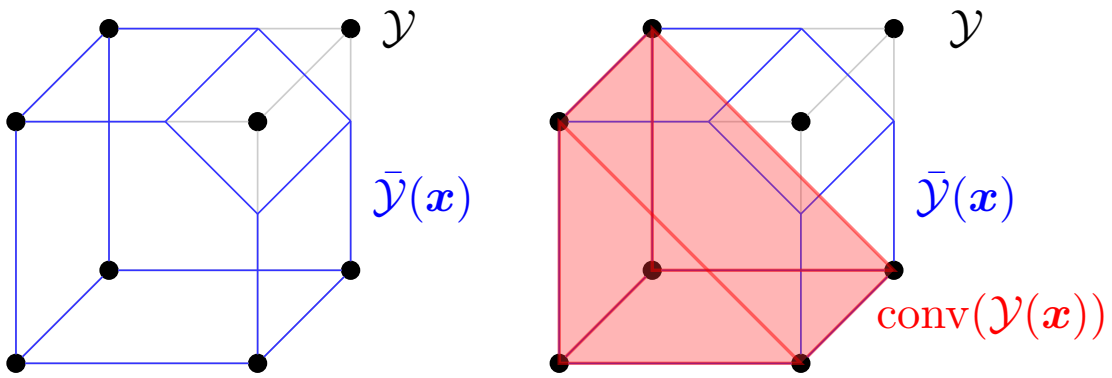


Figure 3: Sets $\text{conv}(\mathcal{Y}(\boldsymbol{x}))$ and $\bar{\mathcal{Y}}(\boldsymbol{x})$ illustrated on an example.

By replacing $\text{conv}(\mathcal{Y}(\boldsymbol{x}))$ with $\bar{\mathcal{Y}}(\boldsymbol{x}) = \text{conv}(\mathcal{Y}) \cap \mathcal{Y}(\boldsymbol{x})$ in (2), we no longer perform the convexification operation on a set that is dependent on $\boldsymbol{x} \in \mathcal{X}$. We may therefore express the conditions $\boldsymbol{y} \in \bar{\mathcal{Y}}(\boldsymbol{x})$ by directly imposing the linking constraints $\boldsymbol{H}\boldsymbol{y} \leq \boldsymbol{d} - \boldsymbol{T}\boldsymbol{x}_1$ on $\text{conv}(\mathcal{Y})$.

To this end, let $\bar{\boldsymbol{y}}^j$ for $j \in \mathcal{L} = \{1, \ldots, L\}$ be the extreme point solutions of $\text{conv}(\mathcal{Y})$ and denote the n-dimensional simplex $\left\{\alpha \in [0,1]^n \middle| \sum_{j=1}^n \alpha^j = 1\right\}$ for $n \in \mathbb{N}$ by Δ^n. Using this notation, we have that

$$\bar{\mathcal{Y}}(\boldsymbol{x}) = \left\{ \sum_{j \in \mathcal{L}} \alpha^j \bar{\boldsymbol{y}}^j \mid \boldsymbol{H} \sum_{j \in \mathcal{L}} \alpha^j \bar{\boldsymbol{y}}^j \leq \boldsymbol{d} - \boldsymbol{T}\boldsymbol{x}_1, \alpha \in \Delta^L \right\}.$$

We may therefore write the relaxation of (1) as:

$$(R): \min \quad \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} + \max_{\boldsymbol{\xi} \in \Xi} \ (\boldsymbol{f} + \boldsymbol{Q}\boldsymbol{\xi})^{\mathrm{T}} \sum_{j \in \mathcal{L}} \alpha^j \bar{\boldsymbol{y}}^j \tag{4}$$

$$\text{s.t.} \quad \boldsymbol{H} \sum_{j \in \mathcal{L}} \alpha^j \bar{\boldsymbol{y}}^j \leq \boldsymbol{d} - \boldsymbol{T}\boldsymbol{x}_1 \tag{5}$$

$$\boldsymbol{x} \in \mathcal{X}, \boldsymbol{\alpha} \in \Delta^L. \tag{6}$$

The computational advantage of relaxation (4)-(6) is clear. After reformulating the inner maximization problem in $\boldsymbol{\xi}$, this equivalent formulation can be solved using a branch-and-price algorithm, adding the columns $\bar{\boldsymbol{y}}^j \in \mathcal{Y}$ to the master problem as needed and branching when solutions $\boldsymbol{x}$ are fractional. We remark that unlike in a typical branch-and-price framework, here the variables $\boldsymbol{\alpha}$ are continuous, *i.e.*, no integrality restrictions are imposed on the reformulated variables $\boldsymbol{y}$. This is by definition of (1) where one can choose a different recourse solution $\boldsymbol{y}$ for each realization of uncertainty. For instance, for the problem of Example 1.1 the optimal recourse value is found as a convex combination of solutions $y_1 = 1$ and $y_2 = 1$.

Below, we present two examples with illustrations of sets $\text{conv}(\mathcal{Y}(\boldsymbol{x}))$ and $\bar{\mathcal{Y}}(\boldsymbol{x})$. Our first example is a case where the two sets are equivalent whereas the second is not.

Example 2.1. Consider the two-stage robust optimization problem with objective uncertainty

$$\min_{\boldsymbol{x} \in \{0,1\}^2} 2x_1 + 4x_2 + \max_{\boldsymbol{\xi} \in \Xi} \ \min_{\boldsymbol{y} \in \{0,1\}^2} \boldsymbol{\xi}^{\mathrm{T}}\boldsymbol{y}$$

$$\text{s.t.} \quad y_i \leq x_i \qquad \forall i = 1, 2.$$

Let $\mathcal{L} = \{1, \ldots, 4\}$ and $\bar{\boldsymbol{y}}^1 = (1, 0)$, $\bar{\boldsymbol{y}}^2 = (0, 1)$, $\bar{\boldsymbol{y}}^3 = (0, 0)$, and $\bar{\boldsymbol{y}}^4 = (1, 1)$. Associating $\alpha^j \geq 0$ with $\bar{\boldsymbol{y}}^j$ for $j \in \mathcal{L}$, it is easy to confirm that $\bar{\mathcal{Y}}(\boldsymbol{x}) = \left\{\sum_{j\in\mathcal{L}} \alpha^j \bar{\boldsymbol{y}}^j \middle| \alpha^1 + \alpha^4 \leq x_1, \alpha^2 + \alpha^4 \leq x_2, \boldsymbol{\alpha} \in \Delta^4\right\}$ and that $\text{conv}(\mathcal{Y}(\boldsymbol{x})) = \left\{\sum_{j\in\mathcal{L}} \bar{\boldsymbol{y}}^j \alpha^j \middle| \sum_{j\in\mathcal{L} \mid \bar{y}_i^j \leq x_i, \forall i=1,2} \alpha^j = 1, \boldsymbol{\alpha} \in \Delta^4\right\}$. It can easily be verified that $\text{conv}(\mathcal{Y}(\boldsymbol{x})) = \bar{\mathcal{Y}}(\boldsymbol{x})$ for $\boldsymbol{x} \in \mathcal{X}$.

Example 2.2. Consider the two-stage robust optimization problem with objective uncertainty

$$\min_{\boldsymbol{x}\in\{0,1\}^2, \boldsymbol{e}^{\mathrm{T}}\boldsymbol{x}\leq 1} 2x_1 + 4x_2 + \max_{\boldsymbol{\xi}\in\Xi} \min_{\boldsymbol{y}\in\{0,1\}^2, \boldsymbol{e}^{\mathrm{T}}\boldsymbol{y}\leq 1} \boldsymbol{\xi}^{\mathrm{T}}\boldsymbol{y}$$
$$\text{s.t.} \quad x_1 + 2x_2 + y_1 + y_2 \geq 1.9.$$

Let $\mathcal{L} = \{1, \ldots, 3\}$ and $\bar{\boldsymbol{y}}^1 = (1, 0)$, $\bar{\boldsymbol{y}}^2 = (0, 1)$ and $\bar{\boldsymbol{y}}^3 = (0, 0)$. Associating $\alpha^j \geq 0$ with $\bar{\boldsymbol{y}}^j$ for $j \in \mathcal{L}$, we have that $\bar{\mathcal{Y}}(\boldsymbol{x}) = \left\{\sum_{j\in\mathcal{L}} \alpha^j \bar{\boldsymbol{y}}^j \middle| \alpha^1 + \alpha^2 \geq 1.9 - x_1 + 2x_2, \boldsymbol{\alpha} \in \Delta^3\right\}$. We further observe that, $\bar{\mathcal{Y}}(\boldsymbol{x}) = \left\{\sum_{j\in\mathcal{L}} \alpha^j \bar{\boldsymbol{y}}^j \middle| \alpha^1 + \alpha^2 \geq 0.9, \boldsymbol{\alpha} \in \Delta^3\right\} \neq \text{conv}(\mathcal{Y}(\boldsymbol{x})) = \left\{\sum_{j\in\mathcal{L}} \alpha^j \bar{\boldsymbol{y}}^j \middle| \sum_{j\in\mathcal{L} | \bar{y}_1^j + \bar{y}_2^j \geq 0.9} \alpha^j = 1, \alpha \in \Delta^3\right\} = \left\{\sum_{j\in\mathcal{L}} \alpha^j \bar{\boldsymbol{y}}^j \middle| \alpha^1 + \alpha^2 = 1, \alpha \in \Delta^3\right\}$ for $\boldsymbol{x} = (1, 0)$. More specifically, in $\bar{\mathcal{Y}}(1, 0)$, one can use the extreme point $\bar{\boldsymbol{y}}^3 = (0, 0) \notin \mathcal{Y}(1, 0)$ with $\alpha^3 \leq 0.1$, whereas $\alpha^3 = 0$ in $\text{conv}(\mathcal{Y}(1, 0))$. We remark that by adding the inequality $\alpha_3 \leq 1 - x_1 + x_2$ to $\bar{\mathcal{Y}}(\boldsymbol{x})$ it can be verified that $\bar{\mathcal{Y}}(\boldsymbol{x}) = \text{conv}(\mathcal{Y}(\boldsymbol{x}))$ for $\boldsymbol{x} \in \mathcal{X}$.

We now present an equivalent deterministic MILP formulation of (R), that allows both a theoretical characterization of the complexity of the problem, and development of solution algorithms. To do so, we first dualize the inner maximization problem in (4)

$$\max_{\boldsymbol{\xi}\in\Xi} \ (\boldsymbol{f} + \boldsymbol{Q}\boldsymbol{\xi})^{\mathrm{T}} \sum_{j\in\mathcal{L}} \alpha^j \bar{\boldsymbol{y}}^j \tag{7}$$

which is a linear programming problem as we assume that Ξ is a polyhedral set. We write, without loss of generality, that $\Xi = \left\{\boldsymbol{\xi} \in \mathbb{R}^S \middle| \boldsymbol{A}\boldsymbol{\xi} \leq \boldsymbol{b}\right\}$ with $\boldsymbol{A} \in \mathbb{R}^{S'\times S}$ and $\boldsymbol{b} \in \mathbb{R}^{S'}$. Let $\boldsymbol{u} \in \mathbb{R}_+^{S'}$ be the dual variables associated with the constraints of the uncertainty set. We then have that

$$\max_{\boldsymbol{\xi}\in\Xi} \ (\boldsymbol{f} + \boldsymbol{Q}\boldsymbol{\xi})^{\mathrm{T}} \sum_{j\in\mathcal{L}} \alpha^j \bar{\boldsymbol{y}}^j = \boldsymbol{f}^{\mathrm{T}} \sum_{j\in\mathcal{L}} \alpha^j \bar{\boldsymbol{y}}^j + \min_{\boldsymbol{u}\in\mathbb{R}_+^{S'}} \ \boldsymbol{u}^{\mathrm{T}}\boldsymbol{b} \tag{8}$$
$$\text{s.t.} \quad \boldsymbol{A}^{\mathrm{T}}\boldsymbol{u} = \boldsymbol{Q}^{\mathrm{T}} \sum_{j\in\mathcal{L}} \alpha^j \bar{\boldsymbol{y}}^j. \tag{9}$$

Therefore the deterministic equivalent of the formulation (4)-(6) is expressed as:

$$\min \quad \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} + \boldsymbol{f}^{\mathrm{T}} \sum_{j\in\mathcal{L}} \alpha^j \bar{\boldsymbol{y}}^j + \boldsymbol{u}^{\mathrm{T}}\boldsymbol{b} \tag{10}$$
$$\text{s.t.} \quad \boldsymbol{H} \sum_{j\in\mathcal{L}} \alpha^j \bar{\boldsymbol{y}}_1^j \leq \boldsymbol{d} - \boldsymbol{T}\boldsymbol{x}_1 \tag{11}$$
$$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{u} = \boldsymbol{Q}^{\mathrm{T}} \sum_{j\in\mathcal{L}} \alpha^j \bar{\boldsymbol{y}}^j \tag{12}$$
$$\sum_{j\in\mathcal{L}} \alpha^j = 1 \tag{13}$$

$$\boldsymbol{x} \in \mathcal{X}, \boldsymbol{\alpha} \in \mathbb{R}^L_+, \boldsymbol{u} \in \mathbb{R}^{S'}_+. \tag{14}$$

We remark at this point that although the starting point of the idea presented in [22] and the approach we propose is the same, we cultivate the two-stage structure of the problem generating columns in the $\mathcal{Y}$ space whereas their approach generates columns in the $\mathcal{X} \times \mathcal{Y}$ space.

2.2 Exact formulations based on relaxation (R)

Although additional developments are in general necessary for imposing $\boldsymbol{y} \in \text{conv}(\mathcal{Y}(\boldsymbol{x}))$, there are some important practical cases in which the relaxation proposed in Proposition 2.2 is exact. To present them, we pose the following assumption.

Assumption 2.1. *We assume that* $\mathcal{Y}(\boldsymbol{x}) = \{\boldsymbol{y} \in \mathcal{Y} \mid \boldsymbol{H}\boldsymbol{y}_1 \leq \boldsymbol{d} - \boldsymbol{T}\boldsymbol{x}_1\}$ *with* $\boldsymbol{y}_1 \subseteq \{0,1\}^{N_1}$.

Let $\boldsymbol{x}^i$ for $i \in \mathcal{K} = \{1, \ldots, K\}$ be the set of extreme point solutions of $\text{conv}(\mathcal{X})$. Further let $\bar{\boldsymbol{y}}^j$ for $j \in \mathcal{L} = \{1, \ldots, L\}$ be the set of extreme point solutions of $\text{conv}(\mathcal{Y})$ as before, and define, $\mathcal{L}_i = \{j \in \mathcal{L} \mid \boldsymbol{H}\bar{\boldsymbol{y}}^j_1 \leq \boldsymbol{d} - \boldsymbol{T}\boldsymbol{x}^i_1\}$ for $i \in \mathcal{K}$. We first present an intermediary result that allows the characterization of $\text{conv}(\mathcal{Y}(\boldsymbol{x}))$ in terms of the extreme points of the set $\text{conv}(\mathcal{Y})$. This result generalizes the construction of convex hulls presented in Examples 2.1 and 2.2.

Proposition 2.3. $\text{conv}(\mathcal{Y}(\boldsymbol{x}^i)) = \left\{\sum_{j\in\mathcal{L}_i} \alpha^j \bar{\boldsymbol{y}}^j \middle| \boldsymbol{\alpha} \in \Delta^{|\mathcal{L}_i|}\right\}$ *for* $i \in \mathcal{K}$.

Proof. Given $i \in \mathcal{K}$, let $\text{ext}(\mathcal{Y}(\boldsymbol{x}^i))$ denote the set of extreme points of $\text{conv}(\mathcal{Y}(\boldsymbol{x}^i))$.

Let $\boldsymbol{y} \in \text{ext}(\mathcal{Y}(\boldsymbol{x}^i)) \subseteq \mathcal{Y}(\boldsymbol{x}^i) = \left\{\boldsymbol{y} \in \mathcal{Y} \;\middle|\; \boldsymbol{H}\boldsymbol{y}_1 \leq \boldsymbol{d} - \boldsymbol{T}\boldsymbol{x}^i_1\right\}$. Since $\boldsymbol{y} \in \mathcal{Y}$, there exists $\boldsymbol{\alpha} \in \Delta^{|\mathcal{L}|}$ such that $\boldsymbol{y} = \sum_{j\in\mathcal{L}} \alpha^j \bar{\boldsymbol{y}}^j$. Further, for all $r \in \mathcal{L}$ such that $\alpha^r > 0$, we must have that $\bar{\boldsymbol{y}}^r_1 = \boldsymbol{y}_1$, as otherwise there exists a row index ℓ such that $\sum_{j\in\mathcal{L}} \alpha^j \left(\bar{\boldsymbol{y}}^j_1\right)_\ell \in]0,1[$. It follows that $\boldsymbol{H}\bar{\boldsymbol{y}}^r_1 \leq \boldsymbol{d} - \boldsymbol{T}\boldsymbol{x}^i_1$ for all $r \in \mathcal{L}$ such that $\alpha^r > 0$, and therefore $r \in \mathcal{L}_i$ following the definition of $\mathcal{L}_i$. This shows that $\text{ext}(\mathcal{Y}(\boldsymbol{x}^i)) \subseteq \left\{\sum_{j\in\mathcal{L}_i} \alpha^j \bar{\boldsymbol{y}}^j \middle| \boldsymbol{\alpha} \in \Delta^{|\mathcal{L}_i|}\right\}$, and as a consequence that $\text{conv}(\mathcal{Y}(\boldsymbol{x}^i)) \subseteq \left\{\sum_{j\in\mathcal{L}_i} \alpha^j \bar{\boldsymbol{y}}^j \middle| \boldsymbol{\alpha} \in \Delta^{|\mathcal{L}_i|}\right\}$.

Conversely, take any solution $\bar{\boldsymbol{y}}^j$ for $j \in \mathcal{L}_i$. By definition of the set $\mathcal{L}_i$, we have that $\bar{\boldsymbol{y}}^j \in \mathcal{Y}$, and that $\boldsymbol{H}\bar{\boldsymbol{y}}^j_1 \leq \boldsymbol{d} - \boldsymbol{T}\boldsymbol{x}^i_1$. It follows that $\bar{\boldsymbol{y}}^j \in \mathcal{Y}(\boldsymbol{x}^i)$ and therefore can be expressed as convex combination of points $\boldsymbol{y} \in \text{ext}(\mathcal{Y}(\boldsymbol{x}^i))$. Therefore $\left\{\sum_{j\in\mathcal{L}_i} \alpha^j \bar{\boldsymbol{y}}^j \middle| \boldsymbol{\alpha} \in \Delta^{|\mathcal{L}_i|}\right\} \subseteq \text{conv}(\mathcal{Y}(\boldsymbol{x}^i))$, proving the result. □

Remark 2.1. *The reason for allowing only binary variables in the linking constraints should now be clear. Indeed, Proposition 2.3 no longer holds if continuous recourse variables are allowed to enter the linking constraints, as the extreme points of* $\mathcal{Y}(\boldsymbol{x}^i)$ *are no longer characterized by the set* $\mathcal{L}_i$.

We next present a result that characterizes a sufficient condition for the equivalence between $\text{conv}(\mathcal{Y}(\boldsymbol{x}))$ and $\bar{\mathcal{Y}}(\boldsymbol{x})$ for $\boldsymbol{x} \in \mathcal{X}$.

Proposition 2.4. *If* $\boldsymbol{H} = I$, $\boldsymbol{T} = -I$ *and* $\boldsymbol{d} = 0$, *then* $\bar{\mathcal{Y}}(\boldsymbol{x}) = \text{conv}(\mathcal{Y}(\boldsymbol{x}))$ *for* $\boldsymbol{x} \in \mathcal{X}$.

Proof. Under the given assumptions, $\mathcal{Y}(\boldsymbol{x}) = \{\boldsymbol{y} \in \mathcal{Y} \mid \boldsymbol{y}_1 \leq \boldsymbol{x}_1\}$. Assume now that there exists a solution $\boldsymbol{y}$ such that $\boldsymbol{y} = \sum_{j\in\mathcal{L}} \alpha^j \bar{\boldsymbol{y}}^j \in \bar{\mathcal{Y}}(\boldsymbol{x})$ and $\boldsymbol{y} \notin \text{conv}(\mathcal{Y}(\boldsymbol{x}))$. Then, we must have that $\sum_{j\in\mathcal{L}} \alpha^j = 1$ and $\sum_{j\in\mathcal{L}|\boldsymbol{y}^j_1\leq\boldsymbol{x}_1} \alpha^j < 1$ by using the characterization of Proposition 2.3. This implies the existence of a linking

constraint, say constraint i, and an index $k \in \mathcal{L}$ such that $\bar{y}_i^k > x_i$ and $\alpha_k > 0$. Since $x_i \in \{0,1\}$ and $\bar{y}_i^k \in \{0,1\}$, we must have that $x_i = 0$ and $\bar{y}_i^k = 1$. We may therefore write,

$$y_i = \sum_{j \in \mathcal{L}} \bar{y}_i^j \alpha^j \geq \alpha_k \bar{y}_i^k = \alpha_k > 0 = x_i$$

Thus $\boldsymbol{y} \notin \mathcal{Y}(\boldsymbol{x})$, which contradicts the assumption that $\boldsymbol{y} \in \bar{\mathcal{Y}}(\boldsymbol{x})$. Therefore, we have proved $\bar{\mathcal{Y}}(\boldsymbol{x}) \subseteq \text{conv}(\mathcal{Y}(\boldsymbol{x}))$. We additionally have that $\text{conv}(\mathcal{Y}(\boldsymbol{x})) \subseteq \bar{\mathcal{Y}}(\boldsymbol{x})$ by Proposition 2.2. As a result $\bar{\mathcal{Y}}(\boldsymbol{x}) = \text{conv}(\mathcal{Y}(\boldsymbol{x}))$. □

Corollary 2.1. *If $\boldsymbol{H} = I$, $\boldsymbol{T} = I$ and $\boldsymbol{d} = \mathbf{1}$, then $\bar{\mathcal{Y}}(\boldsymbol{x}) = \text{conv}(\mathcal{Y}(\boldsymbol{x}))$ for $\boldsymbol{x} \in \mathcal{X}$.*

As a result of Proposition 2.4, we may solve problems of form (1) that satisfy its assumptions by the exact relaxation (10)-(14). As the number of extreme points of the set $\mathcal{Y}$ are in general prohibitively large, we propose to solve this model using the branch-and-price algorithm generating columns from the set $\mathcal{Y}$.

Remark 2.2. *Let, $\boldsymbol{\pi}$, $\boldsymbol{\mu}$, and λ be the dual variables associated with the constraints* (11),(12)*, and* (13)*, respectively. Then the pricing problem takes the form:*

$$\min_{\boldsymbol{y} \in \mathcal{Y}} \quad -\lambda + \left(\boldsymbol{f} - \boldsymbol{H}^{\mathrm{T}}\boldsymbol{\pi} + \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{\mu}\right)^{\mathrm{T}} \boldsymbol{y}$$

We remark that the pricing problem is free of the first-stage variables $\boldsymbol{x}$. Therefore, it is possible that unsuitable columns will be generated throughout the branch-and-price algorithm. However, the utilization of these columns is prohibited by the presence of the linking constraints in the master problem.

We next turn our attention to the case where the assumptions of Proposition 2.4 are not verified. We first generalize the inequality we have introduced in Example 2.2 to no-good cut type inequalities in the following proposition. To this end, let, for $\boldsymbol{x} \in \mathcal{X}$, $\mathcal{I}(\boldsymbol{x}) = \{i \in \mathcal{N} \mid x_i = 1\}$ and $\mathcal{L}(\boldsymbol{x}) = \{j \in \mathcal{L} \mid \boldsymbol{H}\bar{\boldsymbol{y}}_1^j \leq \boldsymbol{d} - \boldsymbol{T}\sum_{i \in \mathcal{I}(\boldsymbol{x})} \boldsymbol{e}_i\}$.

Proposition 2.5. *Let $\mathcal{N} = \{1, \ldots, N_1\}$ and $\mathcal{I} \subseteq \mathcal{N}$ and define $\mathcal{L}(\mathcal{I}) = \left\{ j \in \mathcal{L} \;\middle|\; \boldsymbol{H}\bar{\boldsymbol{y}}_1^j \leq \boldsymbol{d} - \boldsymbol{T}\sum_{i \in \mathcal{I}} \boldsymbol{e}_i \right\}$ where $\boldsymbol{e}_i$ is the i^{th} unit vector of conforming dimensions. The inequalities*

$$\sum_{j \in \mathcal{L} \setminus \mathcal{L}(\mathcal{I})} \alpha^j \leq |\mathcal{I}| - \sum_{i \in \mathcal{I}} x_i + \sum_{i \in \mathcal{N} \setminus \mathcal{I}} x_i \qquad \forall \mathcal{I} \subseteq \mathcal{N} \tag{15}$$

are valid for $\text{conv}(\mathcal{Y}(\boldsymbol{x})) = \left\{ \sum_{j \in \mathcal{L}(\boldsymbol{x})} \alpha^j \bar{\boldsymbol{y}}^j \mid \boldsymbol{\alpha} \in \Delta^{|\mathcal{L}(\boldsymbol{x})|} \right\}$

Proof. Given $\boldsymbol{x} \in \mathcal{X}$, we show that, if $\boldsymbol{\alpha} \in \Delta^{|\mathcal{L}|}$ and $\sum_{j \in \mathcal{L}} \alpha^j \bar{\boldsymbol{y}}^j \in \text{conv}(\mathcal{Y}(\boldsymbol{x}))$ then it satisfies inequalities (15). In this case, by definition of $\text{conv}(\mathcal{Y}(\boldsymbol{x}))$, we must have that $\alpha^j = 0$ for all $j \in \mathcal{L} \setminus \mathcal{L}(\boldsymbol{x})$. Therefore, if $\mathcal{I}(\boldsymbol{x}) = \mathcal{I}$ then we have that $0 \leq 0$. Otherwise, we have that either $\sum_{i \in \mathcal{I}} x_i < |\mathcal{I}|$ or $\sum_{i \in \mathcal{N} \setminus \mathcal{I}} x_i > 0$. Therefore, we have on the left-hand-side $\sum_{j \in \mathcal{L} \setminus \mathcal{L}(\mathcal{I})} \alpha^j$ which is not greater than 1, and on the right-hand-side $|\mathcal{I}| - \sum_{i \in \mathcal{I}} x_i + \sum_{i \in \mathcal{N} \setminus \mathcal{I}} x_i > 0$, which is greater than 1 by integrality of the terms involved, for all $\mathcal{I} \subseteq \mathcal{N}$, $\mathcal{I} \neq \mathcal{I}(\boldsymbol{x})$. □

Proposition 2.6. *Let $\boldsymbol{\alpha} \in \Delta^{|\mathcal{L}|}$ such that $\sum_{j \in \mathcal{L}} \alpha^j \bar{\boldsymbol{y}}^j \in \bar{\mathcal{Y}}(\boldsymbol{x})$ and $\alpha^j > 0$ for some $j \in \mathcal{L} \setminus \mathcal{L}(\boldsymbol{x})$, then there exists an inequality of form* (15) *that is violated.*

Proof. Let, in this case, $\mathcal{I} = \mathcal{I}(\boldsymbol{x})$. We have on the left-hand-side $\sum_{j \in \mathcal{L} \setminus \mathcal{L}(\boldsymbol{x})} \alpha^j > 0$ and on the right-hand-side $|\mathcal{I}(\boldsymbol{x})| - \sum_{i \in \mathcal{I}(\boldsymbol{x})} x_i + \sum_{i \notin \mathcal{I}(\boldsymbol{x})} x_i = 0$. Therefore, inequality (15) with $\mathcal{I} = \mathcal{I}(\boldsymbol{x})$ is violated. □

Following Propositions 2.5 and 2.6, we may write an equivalent formulation for (2) as follows:

$$\min \quad \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} + \max_{\boldsymbol{\xi}\in\Xi} \; \boldsymbol{\xi}^{\mathrm{T}}\boldsymbol{Q}\sum_{j\in\mathcal{L}}\alpha^j\bar{\boldsymbol{y}}^j \tag{16}$$

$$\text{s.t.} \quad \boldsymbol{H}\sum_{j\in\mathcal{L}}\alpha^j\bar{\boldsymbol{y}}_1^j \leq \boldsymbol{d} - \boldsymbol{T}\boldsymbol{x}_1 \tag{17}$$

$$\sum_{j\in\mathcal{L}\setminus\mathcal{L}(\mathcal{I})}\alpha^j \leq |\mathcal{I}| - \sum_{i\in\mathcal{I}}x_i + \sum_{i\in\mathcal{N}\setminus\mathcal{I}}x_i \qquad \forall\mathcal{I}\subseteq\mathcal{N} \tag{18}$$

$$\boldsymbol{x}\in\mathcal{X}, \boldsymbol{\alpha}\in\Delta^L. \tag{19}$$

As this formulation has an exponential number of variables and constraints, we need to couple column generation with cut generation in its solution (a high-level description is given in Figure 4). We remark that, the methodology we present is different from the methodologies presented under the title constraint-and-column generation in the literature. The columns in the constraint-and-column generation are recourse variables corresponding to violated scenarios identified by solving a bilinear problem (which can be very challenging), while the columns we generate are recourse solutions. Similarly, the constraints in the constraint-and-column generation are the set of linking constraints for each scenario generated (which renders the on-the-fly generation difficult), whereas we propose to generate at most one cut of type (15) for each candidate solution.

We next proceed to identification of violated cuts of type (15) given a candidate incumbent solution $(\boldsymbol{x},\boldsymbol{\alpha})$ with $\boldsymbol{x}_1\in\{0,1\}^{N_1}$. In this case, it suffices to verify whether or not inequality (15) with $\mathcal{I}=\mathcal{I}(\boldsymbol{x})$ is satisfied as the proof of Proposition 2.6 suggests. If it is satisfied then $(\boldsymbol{x},\boldsymbol{\alpha})$ is accepted as an incumbent solution and the current upper bound is updated. Otherwise, the cut $\sum_{j\in\mathcal{L}\setminus\mathcal{L}(\boldsymbol{x})}\alpha^j \leq |\mathcal{I}(\boldsymbol{x})| - \sum_{i\in\mathcal{I}(\boldsymbol{x})}x_i + \sum_{i\in\mathcal{N}\setminus\mathcal{I}(\boldsymbol{x})}x_i$ is added to the current restricted master. We remark that, identifying columns that are not feasible for the current first-stage solution $\boldsymbol{x}$ is not complicated, as it suffices to verify whether or not $\boldsymbol{H}\bar{\boldsymbol{y}}_1^j \leq \boldsymbol{d} - \boldsymbol{T}\sum_{i\in\mathcal{I}(\boldsymbol{x})}\boldsymbol{e}_i$ for current columns in the restricted master.

On the other hand, handling the changes induced by the introduction of cuts (15) in the pricing problem is more challenging. Indeed, the coefficient of α^j in (15) is not linear in the value of the corresponding second-stage solution $\bar{\boldsymbol{y}}^j$. This implies the presence of naturally non-linear constraints in the pricing problem. Those can be linearized with help of the appropriate indicator variables and associated constraints. More precisely, we may introduce the indicator variable $z_{\mathcal{I}(\boldsymbol{x})}$, that determines whether a generated column violates the linking constraints $\boldsymbol{H}\boldsymbol{y}_1 \leq \boldsymbol{d} - \boldsymbol{T}\sum_{i\in\mathcal{I}(\boldsymbol{x})}\boldsymbol{e}_i$. This variable is added to the objective function with the coefficient $\eta_{\mathcal{I}(\boldsymbol{x})}$, which is equal to the value of the dual variable associated with the (already generated) cut (15) with $\mathcal{I}=\mathcal{I}(\boldsymbol{x})$ at optimality of the restricted master program. Finally, its definition is imposed by the addition of constraints $Mz_{\mathcal{I}(\boldsymbol{x})} \geq \boldsymbol{d}^k - \boldsymbol{T}^k\sum_{i\in\mathcal{I}(\boldsymbol{x})}\boldsymbol{e}_i - \boldsymbol{H}^k\bar{\boldsymbol{y}}_1^j$ for each recourse constraint, where M is a sufficiently large constant.

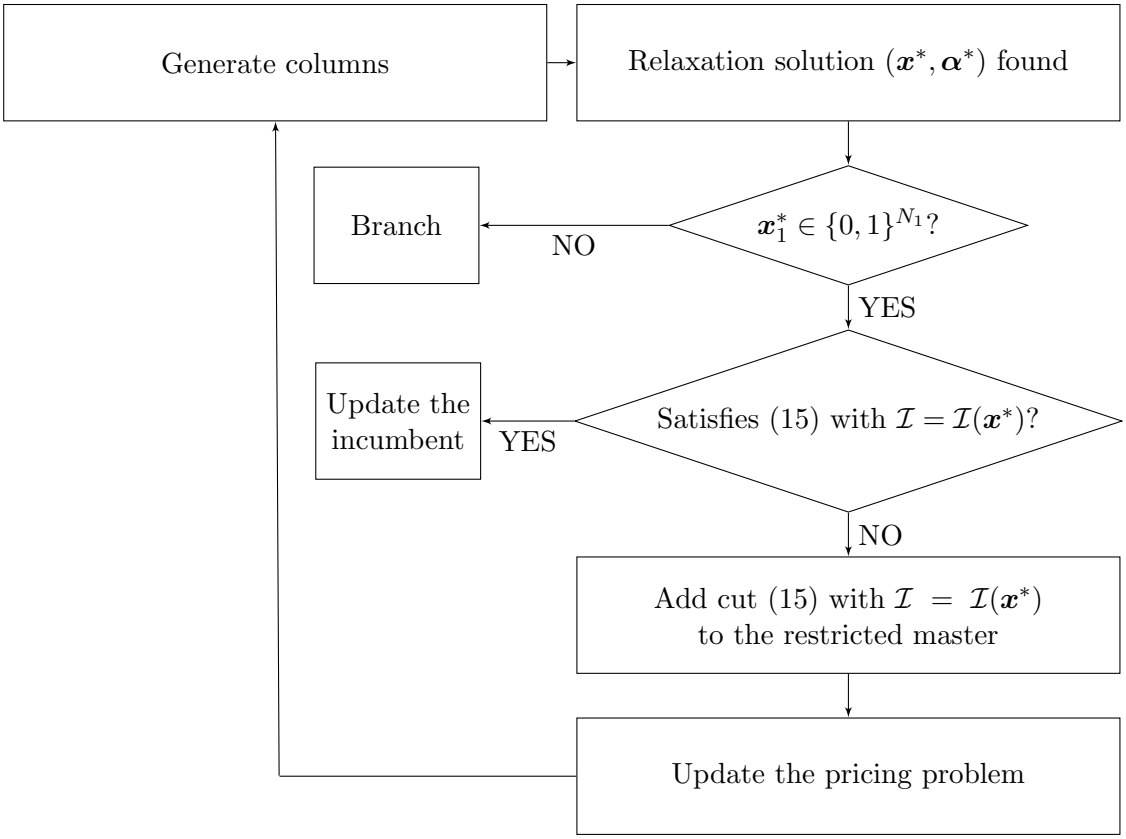


Figure 4: Branch-and-price-and-cut framework. This algorithmic flow illustrates the processing of a node of the branch-and-bound tree.

2.3 An exact deterministic equivalent formulation via lifting

The results of Section 2.2 allow the exact solution of problem (1) when Assumption 2.1 is satisfied. In this section, we propose a reformulation in an extended space that enables us to exploit those results even when problem (1) does not naturally present in a form that satisfy their assumptions. In other words, we present an alternative formulation of the recourse feasible region $\mathcal{Y}$, that allows transforming problem (1) into a problem that satisfies Assumption 2.1.

To this end, let $\boldsymbol{z} \in \{0,1\}^{N_1}$, and consider the reformulation of the recourse feasible region as

$$\mathcal{Y}'(\boldsymbol{x}) = \left\{ \boldsymbol{y} \in \mathcal{Y}, \boldsymbol{z} \in \{0,1\}^{N_1} \middle| \boldsymbol{H}\boldsymbol{y} \leq \boldsymbol{d} - \boldsymbol{T}\boldsymbol{z}, \boldsymbol{z} \leq \boldsymbol{x}_1, \boldsymbol{z} \geq \boldsymbol{x}_1 \right\},$$

essentially incorporating a copy of the first-stage decision variables and the linking constraints into the recourse feasible region. We remark that it suffices to copy those first-stage variables that are involved in the linking constraints. The advantage of this reformulation is that, now the linking constraints are expressed purely in terms of recourse variables and therefore can be incorporated into the subproblem to produce proper columns at each iteration, whereas the linking constraints $\boldsymbol{z} = \boldsymbol{x}$ assure that the relaxation proposed by Proposition 2.2 is exact as variables $\boldsymbol{z}$ are binary.

Let $\mathcal{Y}' = \left\{ \boldsymbol{y} \in \mathcal{Y}, \boldsymbol{z} \in \{0,1\}^{N_1} \middle| \boldsymbol{H}\boldsymbol{y} \leq \boldsymbol{d} - \boldsymbol{T}\boldsymbol{z} \right\}$ and $(\bar{\boldsymbol{y}}, \bar{\boldsymbol{z}})^j$ for $j \in \mathcal{L}' = \{1, \ldots, L'\}$ be the set of extreme

points of $\text{conv}(\mathcal{Y}')$. We may then solve problem (2) writing equivalently as:

$$\min \quad \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} + \boldsymbol{f}^{\mathrm{T}} \sum_{j \in \mathcal{L}'} \alpha^j \bar{\boldsymbol{y}}^j + \boldsymbol{u}^{\mathrm{T}}\boldsymbol{b} \tag{20}$$

$$\text{s.t.} \quad \boldsymbol{x}_1 = \sum_{j \in \mathcal{L}'} \alpha^j \bar{\boldsymbol{z}}^j \tag{21}$$

$$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{u} = \boldsymbol{Q}^{\mathrm{T}} \sum_{j \in \mathcal{L}'} \alpha^j \bar{\boldsymbol{y}}^j \tag{22}$$

$$\sum_{j \in \mathcal{L}'} \alpha^j = 1 \tag{23}$$

$$\boldsymbol{x} \in \mathcal{X}, \boldsymbol{\alpha} \in \mathbb{R}_+^{L'}, \boldsymbol{u} \in \mathbb{R}_+^{T}. \tag{24}$$

Formulation (20)-(24) provides therefore a generic formulation for problem (1). This formulation can be solved using a branch-and-price algorithm generating columns from the set $\mathcal{Y}'$ as needed.

2.4 Alternative formulations in special cases

In this section, we present two alternative formulations of (1) in certain special cases. The first formulation is based on the idea of enumerating the first- and second-stage feasible solutions and creating an uncertainty vector corresponding to each first-stage solution. Writing such a formulation in theory for pure binary sets $\mathcal{X}$ and $\mathcal{Y}$ is always possible but in practice is only viable when $\mathcal{X}$ and $\mathcal{Y}$ are easily enumerable. The second formulation is possible when there is an exact extended reformulation of the recourse problem as a linear program, for instance through a dynamic programming recurrence. Although direct solution of these formulations is extremely time/memory consuming for larger instances, they provide benchmarks for evaluating the computational efficacy of the column generation-based approach proposed above.

2.4.1 Reformulation through enumeration

In this section, we assume that $\mathcal{X}$ and $\mathcal{Y}$ are pure binary sets. Let us denote by $\boldsymbol{x}^i$, for $i \in \mathcal{K} = \{1, \ldots, K\}$, the feasible solutions of $\mathcal{X}$. For $i = 1, \ldots, K$, let $\boldsymbol{y}^{i,j} \in \mathcal{Y}(\boldsymbol{x}^i)$ for $j \in \mathcal{L}_i = 1, \ldots, L_i$ be the feasible solutions of $\mathcal{Y}(\boldsymbol{x}^i)$. We write

$$\max \quad \theta \tag{25}$$

$$\text{s.t.} \quad \theta \leq \theta^i \quad \forall i \in \mathcal{K} \tag{26}$$

$$\theta^i \leq \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}^i + (\boldsymbol{f} + \boldsymbol{Q}\boldsymbol{\xi}^i)^{\mathrm{T}}\boldsymbol{y}^{i,j} \quad \forall i \in \mathcal{K}, j \in \mathcal{L}_i \tag{27}$$

$$\boldsymbol{\xi}^i \in \Xi \quad \forall i \in \mathcal{K}. \tag{28}$$

Proposition 2.7. *Linear program* (25)-(28) *is a formulation of* (1).

Proof. Consider a first-stage feasible solution $\boldsymbol{x}^i \in \mathcal{X}$. We write the inner optimization problem corresponding to this solution:

$$\max_{\boldsymbol{\xi}^i \in \Xi} \min_{\boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x}^i)} (\boldsymbol{f} + \boldsymbol{Q}\boldsymbol{\xi}^i)^{\mathrm{T}}\boldsymbol{y}$$

and its corresponding epigraph formulation by enumerating the feasible solutions of $\mathcal{Y}(\boldsymbol{x}^i)$:

$$\begin{aligned} \max_{\underline{\theta}^i \in \mathbb{R}, \boldsymbol{\xi}^i \in \Xi} \quad & \underline{\theta}^i \\ \text{s.t.} \quad & \underline{\theta}^i \leq (\boldsymbol{f} + \boldsymbol{Q}\boldsymbol{\xi}^i)^{\mathrm{T}} \boldsymbol{y}^{i,j} \quad \forall j \in \mathcal{L}_i \end{aligned}$$

As a result, we may write the objective value of the solution $\boldsymbol{x}^i$ as

$$\begin{aligned} \max_{\theta^i \in \mathbb{R}, \boldsymbol{\xi}^i \in \Xi} \quad & \theta^i \\ \text{s.t.} \quad & \theta^i \leq \boldsymbol{c}^{\mathrm{T}} \boldsymbol{x}^i + (\boldsymbol{f} + \boldsymbol{Q}\boldsymbol{\xi}^i)^{\mathrm{T}} \boldsymbol{y}^{i,j} \quad \forall j \in \mathcal{L}_i. \end{aligned}$$

Problem (1) can then be expressed as

$$\begin{aligned} \min_{i \in \mathcal{K}} \quad & \theta^i \\ \text{s.t.} \quad & \theta^i \leq \boldsymbol{c}^{\mathrm{T}} \boldsymbol{x}^i + (\boldsymbol{f} + \boldsymbol{Q}\boldsymbol{\xi}^i)^{\mathrm{T}} \boldsymbol{y}^{i,j} \quad \forall j \in \mathcal{L}_i \\ & \boldsymbol{\xi}^i \in \Xi \end{aligned}$$

or equivalently,

$$\begin{aligned} \max \quad & \theta \\ \text{s.t.} \quad & \theta \leq \theta^i \quad \forall i \in \mathcal{K} \\ & \theta^i \leq \boldsymbol{c}^{\mathrm{T}} \boldsymbol{x}^i + (\boldsymbol{f} + \boldsymbol{Q}\boldsymbol{\xi}^i)^{\mathrm{T}} \boldsymbol{y}^{i,j} \quad \forall i \in \mathcal{K}, \forall j \in \mathcal{K}_i \\ & \boldsymbol{\xi}^i \in \Xi \quad \forall i \in \mathcal{K} \end{aligned}$$

□

2.4.2 Reformulation through dynamic programming

In this section, we present another reformulation assuming that an extended linear programming formulation for the recourse problem exists. This is, for instance, the case if there exists a dynamic programming equivalent (presumably in a much higher dimensional space) that can be cast as a shortest path problem on an appropriately defined network (see e.g. [24]). As we assume that the linear programming relaxation is integral, in this case $\mathrm{conv}(\mathcal{Y}(\boldsymbol{x})) = \mathcal{Y}(\boldsymbol{x})$ for $\boldsymbol{x} \in \mathcal{X}$. Therefore we may directly write the deterministic equivalent (2) replacing $\mathrm{conv}(\mathcal{Y}(\boldsymbol{x}))$ by its linear programming equivalent. We illustrate this idea below in the case of dynamic programs written as equivalent shortest path formulations on an appropriately defined network.

Let $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ be the network associated with the dynamic programming formulation of $\mathcal{Y}(\boldsymbol{x})$, where $\mathcal{N}$ is the set of nodes and $\mathcal{A}$ is the set of arcs. Let further $\boldsymbol{F}$ be the incidence matrix associated to $\mathcal{G}$, ϕ_a be the flow variable associated with arc $a \in \mathcal{A}$ and $\bar{y}_i^a$ be the value of variable y_i on arc $a \in \mathcal{A}$ for $i = 1, \ldots, M$. Then a deterministic equivalent formulation for (1) can be written as

$$\begin{aligned} \min \quad & \boldsymbol{c}^{\mathrm{T}} \boldsymbol{x} + \boldsymbol{f}^{\mathrm{T}} \boldsymbol{y} + \boldsymbol{u}^{\mathrm{T}} \boldsymbol{b} && (29) \\ \text{s.t.} \quad & \boldsymbol{H}\boldsymbol{y} \leq \boldsymbol{d} - \boldsymbol{T}\boldsymbol{x}_1 && (30) \\ & \boldsymbol{A}^{\mathrm{T}} \boldsymbol{u} = \boldsymbol{Q}^{\mathrm{T}} \boldsymbol{y} && (31) \end{aligned}$$

$$\boldsymbol{F}\boldsymbol{\phi} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \tag{32}$$

$$y_i = \sum_{a \in \mathcal{A}} \bar{y}_i^a \phi_a \quad \forall i \in \mathcal{I} \tag{33}$$

$$\boldsymbol{x} \in \mathcal{X}, \boldsymbol{u} \in \mathbb{R}_+^T, \boldsymbol{y} \in [0,1]^I, \boldsymbol{\phi} \in [0,1]^{\mathcal{A}}. \tag{34}$$

Clearly, the computational tractability of (29)-(34) is dependent on the size of the network $\mathcal{G}$. Unfortunately, in most applications, this size is pseudo-polynomial at best and exponential at worst. However, one might still consider dynamic aggregation/disaggregation (see e.g. [15]) or state-space relaxation techniques (see e.g. [31]) to efficiently solve these models.

3 Complexity results

Despite the three-level structure of problem (1), formulation (10)-(14) reveals that the problem of solving (R) actually lies inside the class NP. Let us define formally the associated decision problem.

PROBLEM: RELAXED TWO STAGE ROBUST MILP (R2SRMILP)
INPUT DATA: Integer η, positive integers $N = p + p', N', M = q + q', M', M'', S, S'$, first-stage data $\boldsymbol{G} \in \mathbb{Z}^{N' \times N}$, $\boldsymbol{g} \in \mathbb{Z}^{N'}$, $\boldsymbol{c} \in \mathbb{Z}^N$, second-stage data $\boldsymbol{E} \in \mathbb{Z}^{M'' \times M}$, $\boldsymbol{e} \in \mathbb{Z}^{M''}$, $\boldsymbol{f} \in \mathbb{Z}^M$, $\boldsymbol{Q} \in \mathbb{Z}^{M \times S}$, linking constraints data $\boldsymbol{H} \in \mathbb{Z}^{M' \times M}$, $\boldsymbol{d} \in \mathbb{Z}^{M'}$, $\boldsymbol{T} \in \mathbb{Z}^{M' \times N}$, uncertainty set data $\boldsymbol{A} \in \mathbb{Z}^{S' \times S}$, $\boldsymbol{b} \in \mathbb{Z}^{S'}$, such that $\mathcal{X} = \{\boldsymbol{x} \in \mathbb{N}^p \times \mathbb{R}_+^{p'} : \boldsymbol{G}\boldsymbol{x} \leq \boldsymbol{g}\}$, $\mathcal{Y} = \{\boldsymbol{y} \in \mathbb{N}^q \times \mathbb{R}_+^{q'} : \boldsymbol{E}\boldsymbol{y} \leq \boldsymbol{e}\}$, and $\Xi = \{\boldsymbol{\xi} \in \mathbb{R}^S : \boldsymbol{A}\boldsymbol{\xi} \leq b\}$ are bounded polyhedral sets. Let $\bar{\mathcal{Y}}(\boldsymbol{x}) = \text{conv}(\mathcal{Y}) \cap \{\boldsymbol{y} : \boldsymbol{H}\boldsymbol{y} \leq \boldsymbol{d} - \boldsymbol{T}\boldsymbol{x}\}$.
QUESTION: Does $\min_{\boldsymbol{x} \in \mathcal{X}, \boldsymbol{y} \in \bar{\mathcal{Y}}(\boldsymbol{x})} \ \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} + \max_{\boldsymbol{\xi} \in \Xi} \ (\boldsymbol{f} + \boldsymbol{Q}\boldsymbol{\xi})^{\mathrm{T}}\boldsymbol{y} \leq \eta$?

Proposition 3.1. *Problem* R2SRMILP *is NP-complete.*

Proof. The problem is trivially NP-hard, since choosing $M = 0$ makes the problem a general MILP. To show that it lies inside NP, let us assume that the answer of R2SRMILP is YES. It follows that there is a solution to (10)-(14), whose cost is not larger than η. Moreover, using Carathéodory's theorem, there exists such a solution $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{u}}, \hat{\boldsymbol{\alpha}})$ where the number Θ of non-zero entries of $\hat{\boldsymbol{\alpha}}$ is bounded above by a polynomial of M' and S. Let $\theta(j)$ be the j^{th} non-zero entry in $\hat{\boldsymbol{\alpha}}$. Then, we can build a certificate C by concatenating the significant elements of $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{u}}, \hat{\boldsymbol{\alpha}})$ and the description of the associated columns: $C = \left(\hat{\boldsymbol{x}}, \hat{\boldsymbol{u}}, \left(\hat{\boldsymbol{\alpha}}^{\theta(j)}\right)_{1 \leq j \leq \Theta}, \left(\hat{\boldsymbol{y}}^{\theta(j)}\right)_{1 \leq j \leq \Theta}\right)$. The description of each column $\hat{\boldsymbol{y}}^{\theta(j)}$ requires at most M integers, so that the size of C is bounded by a polynomial of the input data.

In order to prove the existence of a solution from a certificate C, one could verify that it provides a solution to (10)-(14). However, checking that $\left(\hat{\boldsymbol{y}}^{\theta(j)}\right)_{1 \leq j \leq \Theta}$ is indeed part of the model would require solving a NP-hard problem (checking that each $\hat{\boldsymbol{y}}^{\theta(j)}$ is indeed an extreme point of $\mathcal{Y}$). In order to design a polynomial-time algorithm processing C, we therefore write the equivalent formulation of (10)-(14):

$$\min \left\{ \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} + \boldsymbol{f}^{\mathrm{T}}\boldsymbol{y} + \boldsymbol{u}^{\mathrm{T}}\boldsymbol{b} : H\boldsymbol{y} \leq \boldsymbol{d} - \boldsymbol{T}\boldsymbol{x}, \boldsymbol{A}^{\mathrm{T}}\boldsymbol{u} = \boldsymbol{Q}^{\mathrm{T}}\boldsymbol{y}, \boldsymbol{x} \in \mathcal{X}, \boldsymbol{y} \in \text{conv}(\mathcal{Y}), \boldsymbol{u} \in \mathbb{R}_+^{S'} \right\}. \tag{35}$$

From C, one can check that $\dot{\boldsymbol{y}} = \sum_{j=1}^{\Theta} \hat{\boldsymbol{\alpha}}^{\theta(j)} \hat{\boldsymbol{y}}^{\theta(j)} \in \text{conv}(\mathcal{Y})$ by verifying that $\hat{\boldsymbol{y}}^{\theta(j)} \in \mathcal{Y}$ for all $j \in \{1, \ldots, \Theta\}$ and $\sum_{j=1}^{\Theta} \hat{\boldsymbol{\alpha}}^{\theta(j)} = 1$. This can trivially be done in polynomial time with help of the mathe-

matical programming representation of $\mathcal{Y}$ provided as an input of R2SRMILP. It then suffices to check the cost and the feasibility of $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{u}})$ for the rest of (35). □

Remark 3.1. *This NP-completeness result does not depend on the restrictions over $\boldsymbol{x}_1$ imposed in the definition of* (1)*.*

As a result of Proposition 2.4, problems of form (1) that satisfy its assumptions are equivalent to (4)-(6), *i.e.*, problem R2SRMILP, which leads to the following complexity result.

Corollary 3.1. *If $\boldsymbol{H} = I$, $\boldsymbol{T} = -I$ and $\boldsymbol{d} = 0$, or if $\boldsymbol{H} = I$, $\boldsymbol{T} = I$ and $\boldsymbol{d} = \mathbf{1}$, then solving problem* (1) *is NP-complete.*

Finally, the reformulation of (1) proposed in Section 2.3 allows us to show that this problem lies inside class NP as well.

PROBLEM: TWO STAGE ROBUST MILP (2SRMILP) WITH BINARY FIRST-STAGE VARIABLES
INPUT DATA: Integer η, positive integers $N = N_1 + p + p', N', M = q + q', M', M'', S, S'$, first-stage data $\boldsymbol{G} \in \mathbb{Z}^{N' \times N}$, $\boldsymbol{g} \in \mathbb{Z}^{N'}$, $\boldsymbol{c} \in \mathbb{Z}^{N}$, second-stage data $\boldsymbol{E} \in \mathbb{Z}^{M'' \times M}$, $\boldsymbol{e} \in \mathbb{Z}^{M''}$, $\boldsymbol{f} \in \mathbb{Z}^{M}$, $\boldsymbol{Q} \in \mathbb{Z}^{M \times S}$, linking constraints data $\boldsymbol{H} \in \mathbb{Z}^{M' \times M}$, $\boldsymbol{d} \in \mathbb{Z}^{M'}$, $\boldsymbol{T} \in \mathbb{Z}^{M' \times N}$, uncertainty set data $\boldsymbol{A} \in \mathbb{Z}^{S' \times S}$, $\boldsymbol{b} \in \mathbb{Z}^{S'}$, such that $\mathcal{X} = \{\boldsymbol{x} \in \{0,1\}^{N_1} \times \mathbb{N}^p \times \mathbb{R}_+^{p'} : \boldsymbol{G}\boldsymbol{x} \leq \boldsymbol{g}\}$, $\mathcal{Y} = \{\boldsymbol{y} \in \mathbb{N}^q \times \mathbb{R}_+^{q'} : \boldsymbol{E}\boldsymbol{y} \leq \boldsymbol{e}\}$, and $\Xi = \{\boldsymbol{\xi} \in \mathbb{R}^S : \boldsymbol{A}\boldsymbol{\xi} \leq b\}$ are bounded polyhedral sets. Let $\boldsymbol{x}_1 \in \{0,1\}^{N_1}$ be the subset of binary first-stage variables, and $\mathcal{Y}(\boldsymbol{x}) = \{\boldsymbol{y} \in \mathcal{Y} : \boldsymbol{H}\boldsymbol{y} \leq \boldsymbol{d} - \boldsymbol{T}\boldsymbol{x}_1\}$.
QUESTION: Does $\min_{\boldsymbol{x} \in \mathcal{X}, \boldsymbol{y} \in \mathcal{Y}(\boldsymbol{x})} \ \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} + \max_{\boldsymbol{\xi} \in \Xi} \ (\boldsymbol{f} + \boldsymbol{Q}\boldsymbol{\xi})^{\mathrm{T}}\boldsymbol{y} \leq \eta$?

Theorem 1. *Problem* 2SRMILP *is NP-complete.*

Proof. From any instance Π of 2SRMILP, one can build an equivalent instance Π' satisfying Assumption 2.1, as shown in Section 2.3, whose size is polynomial in the size of Π. Instance Π' is then equivalent to the related instance of R2SRMILP. Thus, solving any instance of 2SRMILP is equivalent to solving an appropriately constructed instance of R2SRMILP, whose size is polynomial in the size of Π. □

Remark 3.2. *This NP-completeness result does not depend on the restrictions over variables $\boldsymbol{y}$. However, the restrictions posed on variables $\boldsymbol{x}_1$ in the definition of* (1) *are necessary.*

4 Numerical results

In this section we demonstrate the application of the methodologies developed in Section 2 in various contexts. A first application is a knapsack variant that gives the possibility to reject/produce or repair items in the second stage. It involves a recourse problem with knapsack-type constraints expressed purely in terms of recourse variables along with linking constraints of type $\boldsymbol{y} \leq \boldsymbol{x}$. In this case, we may apply directly the result of Proposition 2.4 to obtain an equivalent formulation. A second application is centered around a variant of the capital budgeting problem studied previously in K-Adaptability literature (see e.g. [20], [30]). In this case, it turns out that the assumptions of Proposition 2.4 are not verified. As a result, we explore applying the column generation-based approach directly in an extended space.

To obtain optimal integer solutions to our formulations based on models (10)-(14) and (20)-(24), we use the C++ branch-and-price library BaPCod[1]. At each node of the search tree, the linear relaxation

[1] `https://realopt.bordeaux.inria.fr/?page_id=2` (accessed June 2019)

of the problem is solved using column generation. The pricing sub-problems are solved using dynamic programming algorithms, using simple array-based forward label-correcting. At most one column is added to the master program at each iteration. To improve the convergence of the column generation procedure, we use stabilization by automatic smoothing of the dual variables of the master program, as described in [26]. When the optimal solution of the corresponding relaxation does not satisfy the integrality requirements of first-stage variables $\boldsymbol{x}$, one such fractional variable is chosen and two child nodes are created in order to exclude its current value from the search space. In order to reduce the size of the search tree, the branching choices are made with help of the strong branching technique, as described in [29]. The open nodes are processed according to the best first rule. At the root node, and each tenth processed node, a diving heuristic ([29]) is used to derive a feasible solution and try to improve the best known primal bound. The diving heuristic is used only at nodes whose depth is at most ten. The implementations of the branch-and-price and pricing sub-problem solvers are sequential. All mixed integer linear programs, as well as linear programs inside the column generation procedures, are solved using IBM ILOG Cplex 12.9, through the C callable library, using default parameters and four threads.

All our experiments are conducted using a 2 Dodeca-core Haswell Intel Xeon E5-2680 v3 2.5 GHz machine with 128Go RAM running Linux OS. The resources of this machine are strictly partitioned using Slurm Workload Manager[2] to run several tests in parallel. The resources available for each run (algorithm-instance) are set to 4 threads and a 20 Go RAM limit (we remark that our branch-and-price algorithms do not benefit from parallel processing). This virtually creates six independent machines, each running one single instance at a time.

Sections 4.1 and 4.2 introduce the applications in detail, present the details of instances generated, and the results obtained.

4.1 Two-stage robust knapsack problem

Consider a two-stage robust knapsack problem with the set of items $\mathcal{I} = \{1, \ldots, I\}$. Each item has a weight c_i and an uncertain profit $\tilde{p}_i \in [\bar{p}_i - \hat{p}_i, \bar{p}_i]$ for $i \in \mathcal{I}$ where $\bar{p}_i$ is the expected profit and $\hat{p}_i$ is its maximum deviation. In this problem, a first stage decision is to choose a subset of items to produce. After this choice, a profit degradation factor $\boldsymbol{\xi} \in \Xi = \{\boldsymbol{\xi} \in \mathbb{R}^I_+ \mid \sum_{i\in\mathcal{I}} \xi_i \leq \Gamma, 0 \leq \xi_i \leq 1\}$ is revealed. We define $p_i(\boldsymbol{\xi}) = \bar{p}_i - \xi_i \hat{p}_i$ for $i \in \mathcal{I}$. After observing the profit degradation, there are three possible recourse actions:

(i) Produce the item as is, using c_i units of the knapsack capacity, with the degraded profit $\bar{p}_i - \xi_i \hat{p}_i$.

(ii) Repair the item, using an additional t_i units of the knapsack capacity, recovering the original profit $\bar{p}_i$.

(iii) Outsource the item, with associated profit $\bar{p}_i - f_i$ where f_i is the cost of outsourcing the item.

We next give a mathematical formulation for this problem. Let, for $i \in \mathcal{I}$, x_i denote the decision to produce item i in the first-stage, and $y_i = 1$, $r_i = 1$ and $y_i = 0$ denote the decisions to produce without repair, repair or outsource item i in the second-stage, respectively. We then write,

$$\min_{\boldsymbol{x}\in\{0,1\}^I} \sum_{i\in\mathcal{I}} (f_i - \bar{p}_i)x_i + \max_{\boldsymbol{\xi}\in\Xi} \min_{\boldsymbol{y},\boldsymbol{r}\in\mathcal{Y}(\boldsymbol{x})} \sum_{i\in\mathcal{I}} (\hat{p}_i(\boldsymbol{\xi}) - f_i)y_i - \hat{p}_i(\boldsymbol{\xi})r_i \tag{36}$$

[2] `https://slurm.schedmd.com/` (accessed June 2019)

where

$$\mathcal{Y}(\boldsymbol{x}) = \left\{ \boldsymbol{y} \in \{0,1\}^I, \boldsymbol{r} \in \{0,1\}^I \;\middle|\; \begin{array}{ll} \sum_{i\in\mathcal{I}} c_i y_i + t_i r_i \leq C & \\ y_i \leq x_i & \forall i \in \mathcal{I} \\ r_i \leq y_i & \forall i \in \mathcal{I} \end{array} \right\}.$$

As the linking constraints $y_i \leq x_i$ for $i \in \mathcal{I}$ conform with the assumption of Proposition 2.4, we can directly write the deterministic equivalent form (10)-(14), generating the columns from the generalized knapsack set:

$$\mathcal{Y} = \left\{ \boldsymbol{y} \in \{0,1\}^I, \boldsymbol{r} \in \{0,1\}^I \;\middle|\; \begin{array}{ll} \sum_{i\in\mathcal{I}} c_i y_i + t_i r_i \leq C & \\ r_i \leq y_i & \forall i \in \mathcal{I} \end{array} \right\}.$$

In this case, the pricing problem takes the form

$$-\lambda + \min_{\boldsymbol{y},\boldsymbol{r}\in\mathcal{Y}} \sum_{i\in\mathcal{I}} (-f_i + \hat{p}_i\pi_i - \mu_i) y_i - \sum_{i\in\mathcal{I}} \hat{p}_i \pi_i r_i$$

which can be solved using an extension of the pseudo-polynomial dynamic programming algorithm for the classical knapsack problem. We additionally test a K-Adaptability approach to this problem, the associated model can be found in Section 6.1 of the Appendix.

4.1.1 Instance generation

For our numerical tests we generate instances inspired by those presented by [27]. These instances are categorized as uncorrelated (UN), weakly correlated (WC), almost strongly correlated (ASC), and strongly correlated (SC). For given number of items I, and parameters $R = 1000$, $H = 100$, $h \in \{40, 80\}$, $\delta \in \{0.1, 0.5, 1\}$, we generate $c_i \in [1, R]$ for $i \in \mathcal{I}$ and $C = \frac{h}{H+1} \sum_{i\in\mathcal{I}} c_i$. The profit $\bar{p}_i$ for $i \in \mathcal{I}$ is then generated based on the category of instances as follows: $\bar{p}_i = [1, R]$ for UN, $\bar{p}_i \in [c_i - \frac{R}{20}, c_i + \frac{R}{20}]$ for WC, $\bar{p}_i \in [c_i + \frac{R}{10} - \frac{R}{1000}, c_i + \frac{R}{10} + \frac{R}{1000}]$ for ASC, and $\bar{p}_i = c_i + \frac{R}{10}$ for SC. Based on $\bar{p}_i$, the maximum degradation factor $\hat{p}_i$ for $i \in \mathcal{I}$ is generated in the interval $[\bar{p}_i(1-\delta)/2, \bar{p}_i(1+\delta)/2]$ and the penalty of rejecting an item f_i for $i \in \mathcal{I}$ is generated in the interval $[1.1\bar{p}_i, 1.5\bar{p}_i]$. Finally, repair capacity t_i for $i \in \mathcal{I}$ is generated depending on the category of instances as follows: $t_i \in [1, c_i]$ for UN, $t_i \in [0.5\hat{p}_i - \frac{R}{40}, 0.5\hat{p}_i + \frac{R}{40}]$ for WC, $t_i \in [0.5\hat{p}_i + \frac{c_i}{10} - \frac{R}{1000}, 0.5\hat{p}_i + \frac{c_i}{10} + \frac{R}{1000}]$ for ASC, and $t_i \in 0.5\hat{p}_i + \frac{c_i}{10}$ for SC.

4.1.2 Results

In this section, we present our results on instances generated as described above, presenting a comparison between the branch-and-price (B&P), an alternative formulation based on dynamic programming (DP-based) and K-Adaptability with $K = 2, 3, 4$ (2-,3-,4-Adapt). We further explore the limits of the branch-and-price method on larger instances.

We solve instances with $I \in \{20, 30, 40, 50, 60, 70, 80\}$ and generate 72 instances for each value of I, 18 in each instance category (UN,WC,ASC,SC), corresponding to each combination of the parameters $h \in \{40, 80\}$, $\delta \in \{0.1, 0.5, 1\}$, and the uncertainty budget $\Gamma \in \{0.1I, 0.15I, 0.2I\}$.

We start by presenting, in Table 1, our results for 20-item instances with all solution methods considered. For these instances the time limit was set to 1 hour. In Table 1, #Conv represents the number of instances for which each method converged within the time limit. Time* represents the solution time (when converged) divided by the solution time of the branch-and-price algorithm. ConvGap(%) represents

the percentage optimality gap reported by each method, that is the gap between the best primal and dual bounds found by the corresponding algorithm. OptGap(%) represents the percentage gap between the best primal solution reported by each method versus the optimal solution of the branch-and-price algorithm. #BestSol represents the number of instances for which each method was able to find the best primal solution, and #NotBestSol represents the number of instances for which this was not the case.

	#Conv	Time*	ConvGap(%)	OptGap(%)	#BestSol	#NotBestSol
B&P	72	1	0	0	72	0
DP-based	35	2611.55	50.92	23.95	41	28
2-Adapt	41	468.26	9.72	0.61	16	56
3-Adapt	8	5019.24	29.52	0.53	17	55
4-Adapt	0	-	40.32	0.66	16	56

Table 1: A summary of 20-item instance results with 5 different solution methods and 1 hour time limit. Solution time ratios are reported as an average over instances solved to convergence before the time limit only. Percentage gaps are reported as an average over 72 instances.

As can be seen in the first column of Table 1, the branch-and-price algorithm converged for all instances considered, while the DP-based model converged for 35, 2−adaptability model converged for 41, the 3−adaptability model converged for 8 instances, respectively. The 4−adaptability model did not converge for any of the instances considered. A comparison of the average solution time for the branch-and-price algorithm to that of the other methods reveals the numerical promise of this approach. It did not only solve the instances considered exactly, but it did so at least two orders of magnitude faster than other methods. On the other hand, although the gaps reported (ConvGap(%)) by $K-$adaptability methods were large, the primal solutions provided by these methods were on average within 1% (OptGap(%)) of the optimal solution provided by the branch-and-price algorithm.

The column #BestSol provides further insight to this observation. It reveals that the optimal solution of the branch-and-price algorithm coincided with that of $K-$adaptability methods for 16 of the instances (17 for 3-Adaptability), which may happen when the number of active columns in an optimal solution is less than or equal to K. This reveals finite adaptability as a good approximation method for finding near-optimal solutions, at least for the problem, and the 20-item instances considered. However, this method suffers from the poor relaxation gap stemming from the linearization of the bilinear terms, slowing down its convergence.

Finally, we remark that although the average gap of the dynamic programming based formulation is very large, this method was able to find the optimal solution for 41 of the 72 instances considered (and was able to prove optimality for 35). The large gap reported is explained by a few instances where this method was not able to find a primal solution and therefore had poor relative gaps. Indeed, if the calculation of this gap is restricted only to those instances where a primal solution is found then the gap for the DP-based method is %5.04. Additionally, for 3 of the instances considered, the memory limitations were reached, hence the discrepancy in the sum of the columns #BestSol and #NotBestSol ($41 + 28 \neq 72$).

In Figure 5, we provide the performance profile for all five methods considered, plotting the number of 20-item instances for which the method converged versus the time (expressed in logarithmic scale). We remark that the 4-Adaptability method was excluded as it did not converge for any of the instances considered. Despite its lack of scalability, the DP-based formulation could be considered as an interesting alternative for solving the problem exactly, since it provides better solutions than 3-Adaptability in a shorter computing time. This plot further confirms the numerical promise of the branch-and-price algorithm. It also

shows that among the other methods considered the most promising is 2−Adaptability.

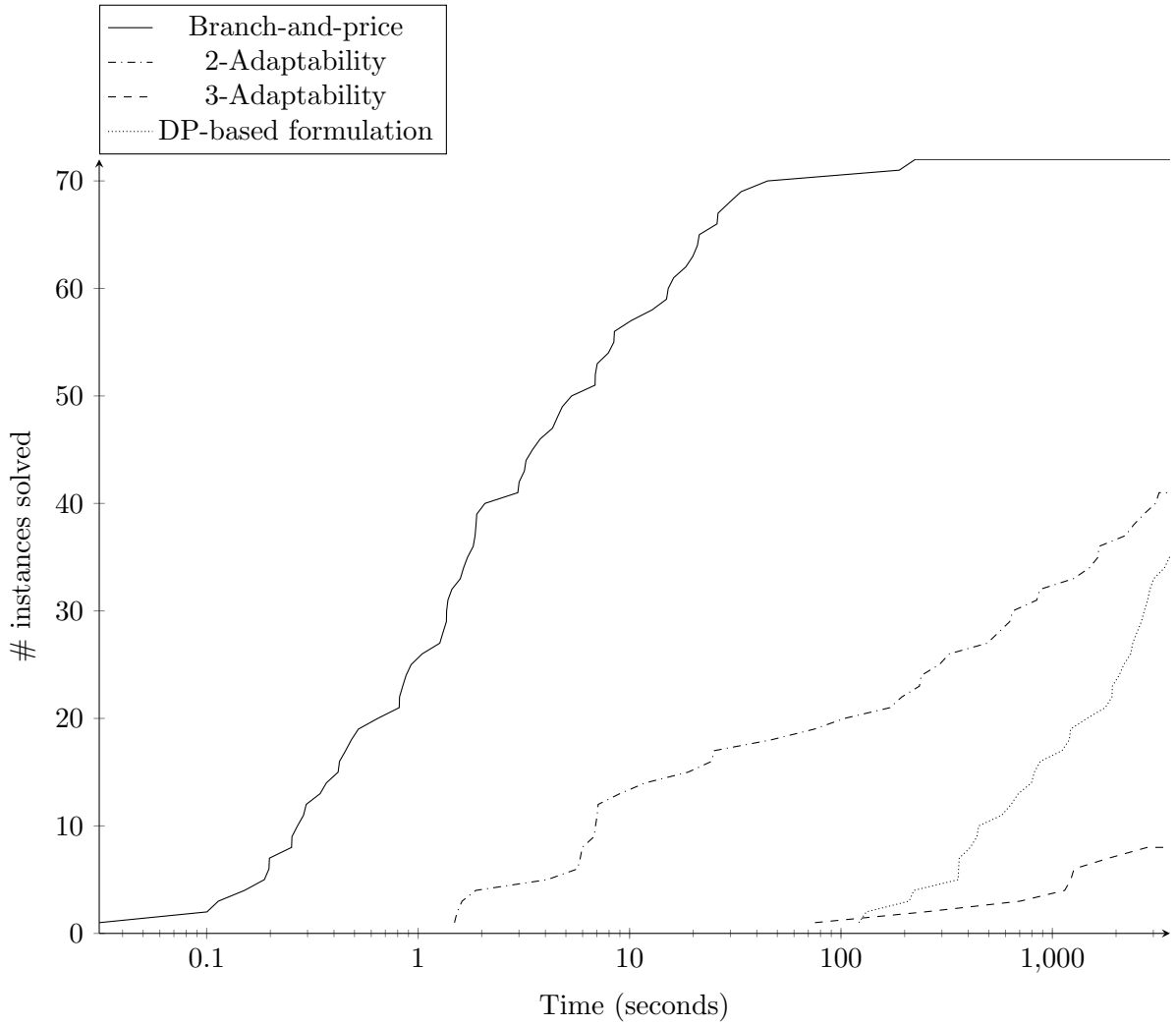


Figure 5: Performance profile comparing four methods for 20-item instances of the Robust Knapsack problem. Results for 4-Adaptability model are not shown since this method does not converge for any of the 20-item instances within the one-hour time limit.

We therefore further compare these two methods on 30-item instances, with a 2-hour time limit. The results are presented in Table 2 where the headers are kept the same as Table 1. The branch-and-price approach converges for 65 of the 72 instances considered whereas the 2-Adaptability method is able to converge for only 12 instances. Interestingly, the primal solutions provided by the 2-Adaptability method are still within %1 of optimality, with this method providing a better solution in 5 of the instances.

We next explore the limits of the column generation-based approach with increasing number of items. In Table 3, we report our results for instances with $I \in \{20, 30, 40, 50, 60, 70, 80\}$, where starting from 40-item instances the time limit was set to 3 hours. We report the number of instances solved to optimality (#Opt), the average optimality gap reported (AvgGap (%)), and the maximum optimality gap reported among the instances considered (MaxGap(%)). The results show that starting from 50-item instances the 3-hour time limit was not sufficient although the average optimality gap was below %5, with the maximum

	#Conv	Time*	ConvGap(%)	OptGap(%)	#BestSol	#NotBestSol
B&P	65	1	3.24	0	67	5
2-Adapt	12	2886.42	21.98	0.63	11	61

Table 2: A summary of 30-item instance results with branch-and-price algorithm and 2-Adaptability, and 2-hour time limit. Solution time ratios are reported as an average over instances solved to convergence before the time limit only. Percentage gaps are reported as an average over 72 instances.

optimality gap being around %11. For those instances solved to optimality, the average solution time by instance category is presented in Table 4.

	$I = 20$	$I = 30$	$I = 40$	$I = 50$	$I = 60$	$I = 70$	$I = 80$
#Opt	72	65	49	35	22	18	18
AvgGap(%)	0	3.24	3.37	3.92	3.43	4.16	4.49
MaxGap(%)	0	6.75	7.8	9.44	11.21	10.97	9.91

Table 3: Summary of results for the branch-and-price algorithm: number of instances solved to optimality within the time limit (#Opt), average (AvgGap) and maximum (MaxGap) optimality gaps for unsolved instances.

	$I = 20$	$I = 30$	$I = 40$	$I = 50$	$I = 60$	$I = 70$	$I = 80$
ASC	8.6	396.0	1715.8	3788.0	1106.4		
SC	9.2	807.6	1556.3	3120.5			
UN	0.3	1.3	3.5	12.4	18.2	46.4	388.1
WC	28.8	453.7	1557.0	3795.6	1453.2		

Table 4: The average branch-and-price solution time for instances solved to optimality.

We further present the number of instances by category solved to optimality by the branch-and-price algorithm within the time limit in Table 5, which gives insight to the difficulty of the generated instances. As observed by [27], introducing correlation between the parameters of the problem renders the solution more difficult. This is also evident from Table 5 where starting from 60-item instances no correlated instances were solved within the time limit. We remark that, we introduce correlation in both stages of the problem. Although the simple dynamic programming algorithm used for solving the pricing problem is not affected by correlation of data, the convergence of the global solution process is.

Finally, we conclude this section with an analysis of the value gained by considering an exact solution method rather than the $K-$adaptability approach. In Table 6, we present the percentage difference between the objective values of the best primal solution found by the branch-and-price algorithm versus the $K-$Adaptability methods at the end of the time limit. This difference is calculated as $2 * \frac{z_{\text{CG}} - z_{\text{K}}}{z_{\text{CG}} + z_{\text{K}}}$, and is reported only for those instances where the optimal branch-and-price solution had at least 2 active columns. Based on these results, the average percentage gap is below %1, whereas the maximum gap is around %2 for the 2-Adaptability method. One can expect the gaps to be smaller for 3- and 4-adaptable solutions when the method is given enough time to converge.

Although the gains reported in Table 6 are rather small, we would expect this number to increase as the number of items increases. In Figure 6, we present side by side a histogram of the number of active columns (*i.e.*, second-stage solutions that are active for the optimal solution of the adversarial problem)

	$I = 20$	$I = 30$	$I = 40$	$I = 50$	$I = 60$	$I = 70$	$I = 80$
ASC	18	14	12	5	1	0	0
SC	18	15	10	6	0	0	0
UN	18	18	18	18	18	18	18
WC	18	18	9	6	3	0	0

Table 5: Number of instances by category solved to optimality by branch-and-price algorithm.

	Average		Max	
	$I = 20$	$I = 30$	$I = 20$	$I = 30$
2-Adapt	0.78	0.75	1.98	2.12
3-Adapt	0.67		4.19	
4-Adapt	0.79		4.81	

Table 6: Percentage profit gain by an exact approach when the number of active columns is at least 2.

in the best solution found by the branch-and-price algorithm, for 20-item instances and all instances, respectively. This comparison sheds more light into the reason why finite adaptability is a very good approximation for smaller instances. Indeed, for 20-item instances, most exact solutions used less than 9 active columns. On the other hand, when all instances are considered, the best primal solution of the branch-and-price algorithm mostly uses between 10 and 40 active columns. In this case, the quality of a finite adaptable solution will mostly depend on the contribution of these columns to the solution, as well as the variability of their profits.

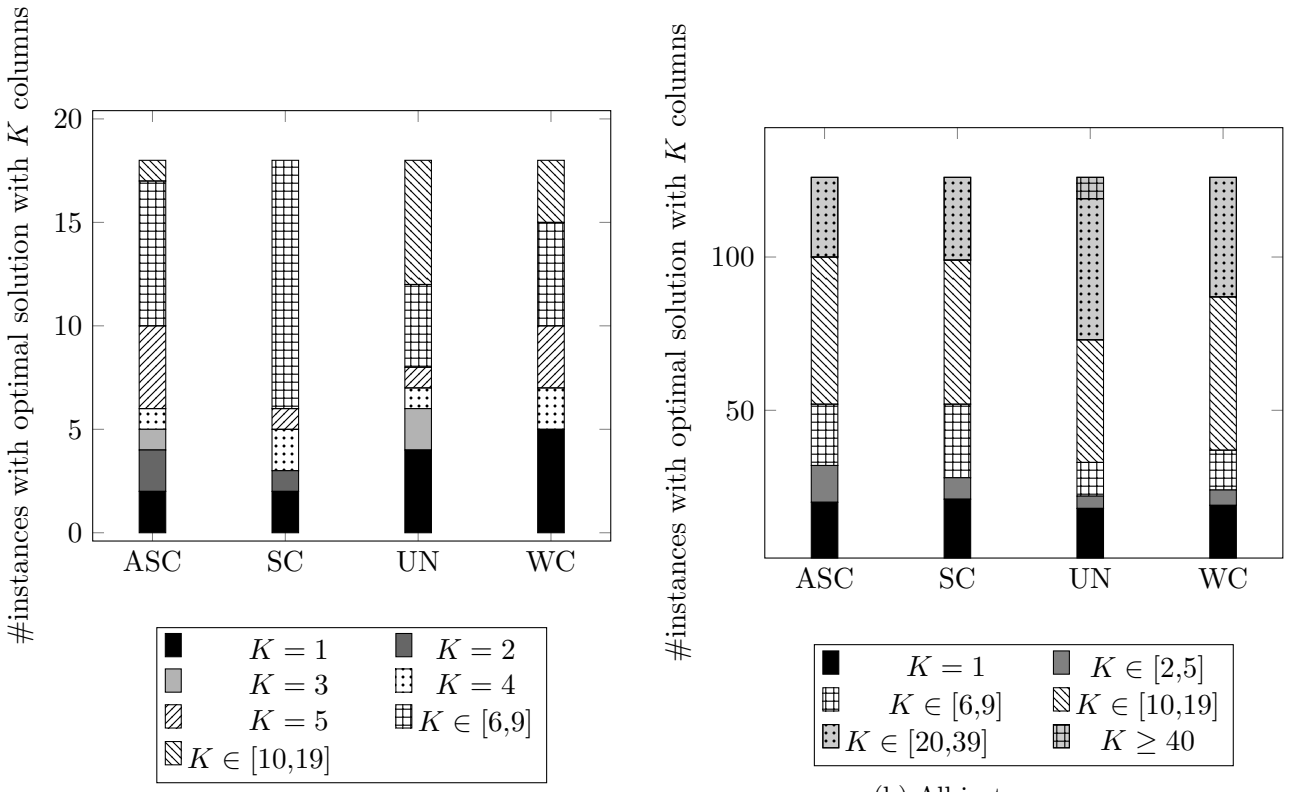
4.2 Capital budgeting problem

Consider an investment planning problem where a company can allocate an investment budget of B to a subset of projects $i \in \mathcal{N} = \{1, \ldots, N\}$ with possible extensions to the budget with loans. Each project $i \in \mathcal{N}$ has cost c_i and uncertain profit $\tilde{p}_i$ which is modeled as a function of uncertain vector $\boldsymbol{\xi} \in \Xi$ of risk factors. The company can invest in a project before or after observing the risk factors $\boldsymbol{\xi} \in \Xi$. We assume that it is also possible to obtain loans before or after the realization of uncertainty, however the interest rate will be higher in the latter case. Here, we suppose that there is one loan option each before and after the realization of uncertainty, for an amount of C_1 and C_2, respectively. To model the uncertainty associated with this problem, we assume M risk factors that reside in the hyper-rectangle $\Xi = [-1, 1]^M$ with $M << N$. We model the project profits as affine functions of these factors, $\tilde{p}_i(\boldsymbol{\xi}) = (1 + Q_i^{\mathrm{T}}\boldsymbol{\xi}/2)\bar{p}_i$. Here $\bar{p}_i$ is the nominal cost of project i and Q_i represents the i^{th} row of the factor loading matrix $Q \in \mathbb{R}^{N \times M}$ as a column vector. Early investments enjoy a first-mover advantage whereas a postponed investment in project i generates only a fraction $f \in [0, 1)$ of the profits $\tilde{p}_i$. An initial formulation of the problem is given as

$$\max_{(\boldsymbol{x}, x_0) \in \mathcal{X}} -\lambda x_0 + \sum_{i \in \mathcal{N}} \bar{p}_i(x_i + f y_i) + \min_{\boldsymbol{\xi} \in \Xi} \max_{(\boldsymbol{y}, y_0) \in \mathcal{Y}(\boldsymbol{x})} \sum_{i \in \mathcal{N}} \left(\sum_{j=1}^{M} \frac{Q_{i,j}\xi_j}{2} \right) \bar{p}_i(x_i + f y_i) - \lambda \mu y_0 \tag{37}$$

with $\mu > 1$, where $\mathcal{X} = \left\{ (\boldsymbol{x}, x_0) \in \{0, 1\}^{N+1} \mid \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \leq B + C_1 x_0 \right\}$ and

$$\mathcal{Y}(\boldsymbol{x}) = \left\{ (\boldsymbol{y}, y_0) \in \{0, 1\}^{N+1} \;\middle|\; \begin{array}{ll} \boldsymbol{c}^{\mathrm{T}}\boldsymbol{y} - C_2 y_0 \leq B + C_1 x_0 - \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} & \\ y_i \leq 1 - x_i & \forall i \in \mathcal{N} \end{array} \right\}.$$



(a) 20-item instances

(b) All instances

Figure 6: Number of active columns (by instance category) in the optimal solution found by the branch-and-price algorithm.

It is easy to verify in this case that $\text{conv}(\mathcal{Y}(\boldsymbol{x})) \neq \bar{\mathcal{Y}}(\boldsymbol{x})$ for $\boldsymbol{x} \in \mathcal{X}$. We alternatively describe how columns can be generated in an extended space to create a relaxation $\bar{\mathcal{Y}}(\boldsymbol{x})$ that is exact. To this end, we begin with reformulating the recourse problem above. We write

$$\mathcal{Y}'(\boldsymbol{x}) = \left\{ (\boldsymbol{y}, y_0) \in \{0,1\}^{N+1} \;\middle|\; \begin{array}{ll} \boldsymbol{c}^{\mathrm{T}}\boldsymbol{y} \leq B + C_1 x_0 + C_2 y_0 & \\ y_i \geq x_i \quad \forall i \in \mathcal{N} & \end{array} \right\}.$$

In the set $\mathcal{Y}'(\boldsymbol{x})$, variable y_i takes value 1 if the corresponding first-stage variable x_i is equal to 1, therefore implicitly counting for the budget already allocated to first-stage investments. With this redefinition of the set $\mathcal{Y}(\boldsymbol{x})$, we also change the objective function to replace variable y_i with the difference $y_i - x_i$ to differentiate between investment decisions in the first and second stage. The problem then becomes

$$\max_{(\boldsymbol{x},x_0)\in\mathcal{X}} -\lambda x_0 + \sum_{i\in\mathcal{N}} \bar{p}_i((1-f)x_i + f y_i) + \min_{\boldsymbol{\xi}\in\Xi} \max_{(\boldsymbol{y},y_0)\in\mathcal{Y}'(\boldsymbol{x})} \sum_{i\in\mathcal{N}} \left(\sum_{j=1}^{M} \frac{Q_{i,j}\xi_j}{2} \right) \bar{p}_i((1-f)x_i + f y_i) - \lambda\mu y_0.$$

We next proceed to extending the recourse feasible region so as to include a copy of the variable x_0. To this end we may write

$$\mathcal{Y}''(\boldsymbol{x}) = \left\{ (\boldsymbol{y}, y_0, z_0) \in \{0,1\}^{N+2} \;\middle|\; \begin{array}{ll} \boldsymbol{c}^{\mathrm{T}}\boldsymbol{y} \leq B + C_1 z_0 + C_2 y_0 & \\ y_i \geq x_i \quad \forall i \in \mathcal{N} & \\ z_0 = x_0 & \end{array} \right\},$$

therefore defining the budget constraint $c^{\mathrm{T}}\boldsymbol{y} \leq B + C_1 z_0 + C_2 y_0$ purely in terms of recourse variables. Let the extreme point solutions of the set $\mathcal{Y}'' = \left\{(\boldsymbol{y}, y_0) \in \{0,1\}^{N+1} \middle| c^{\mathrm{T}}\boldsymbol{y} \leq B + C_1 z_0 + C_2 y_0\right\}$ be denoted by $(\bar{\boldsymbol{y}}, \bar{z}_0, \bar{y}_0)^k$ for $k \in \mathcal{K} = \{1, \ldots, K\}$. When applying the branch-and-price algorithm to this modified problem, the budget constraint $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{y} \leq B + C_1 z_0 + C_2 y_0$, is removed from the master problem while the constraint $x_0 = \sum_{k\in\mathcal{K}} \alpha^k z_0^k$ is added. Under this reformulation the linking constraints are $x_0 = \sum_{k\in\mathcal{K}} \alpha^k z_0^k$ and $\sum_{k\in\mathcal{K}} \alpha^k y_i^k \geq x_i$ for $i \in \mathcal{N}$, and they follow the assumptions of Proposition 2.4. We may therefore obtain an optimal solution of the capital budgeting problem by directly solving this model with the branch-and-price algorithm without the need to add any cuts.

We additionally test a K-Adaptability approach to this problem, the associated model can be found in Section 6.2 of the Appendix.

4.2.1 Instance generation

For our numerical tests, we generate instances inspired by those presented in [20]. For given number of items N, and parameters $R = 100$ and $H = 100$, $h \in \{20, 40, 60, 80\}$, we generate the costs c_i for $i \in \mathcal{N}$ uniformly from the interval $[1, R]$, and we set the nominal profits $\bar{\boldsymbol{p}} = \boldsymbol{c}/5$. The components in each row of Q are generated uniformly from the unit simplex in $\mathbb{R}^M$, which implies that the profits of each project can deviate up to %50 from their nominal values. We set the investment budget to $B = \frac{h}{H+1}\sum_{i\in\mathcal{I}} c_i$ and we assume that postponed investments only generate %80 of the profits, that is $f = 0.8$. The loans C_1 and C_2 are set to %20 of the initial budget B, and the cost parameters are chosen as $\lambda = \frac{0.12}{5}$ and $\mu = 1.2$.

4.2.2 Results

In this section, we present our results on instances generated as described above, presenting a comparison between the branch-and-price algorithm (B&P), and K-Adaptability with $K = 2, 3$ (2-,3-Adapt). We solve instances with $N \in \{10, 20, 30, 40, 50, 100\}$ and generate 60 instances for each value of N, corresponding to 5 replications for each combination of the parameters $h \in \{20, 40, 60, 80\}$ and $M \in \{4, 6, 8\}$. All instances are solved with a 1-hour time limit.

We first present in Tables 7 and 8 the results comparing the number of instances for which each method has converged, and the solution time for each method (when converged) divided by the solution time of the branch-and-price algorithm, respectively. Similar to our results for the robust knapsack instances, our results reveal the column generation-based approach to be very effective for this problem. The exact solution method scales up very well and is able to solve more than %95 of the instances considered. It is also 3 to 4 orders of magnitude faster than the $K-$adaptability method for larger instances starting with 30 items. A performance profile is presented in Figure 7, where the number of instances for which each method has converged is plotted over time presented in logarithmic scale.

	$N = 10$	$N = 20$	$N = 30$	$N = 40$	$N = 50$	$N = 100$
B&P	60	60	58	55	60	57
2-Adapt	60	60	21	15	7	0
3-Adapt	60	31	13	0	0	0

Table 7: Number of instances solved to convergence by each method.

	$N = 10$	$N = 20$	$N = 30$	$N = 40$	$N = 50$	$N = 100$
2-Adapt	2.9	74.1	1025.0	1826.6	22011.4	-
3-Adapt	14.8	2191.0	34573.5	-	-	-

Table 8: Average solution time for $K-$adaptability methods (when converged) divided by the solution time of the extended column generation method.

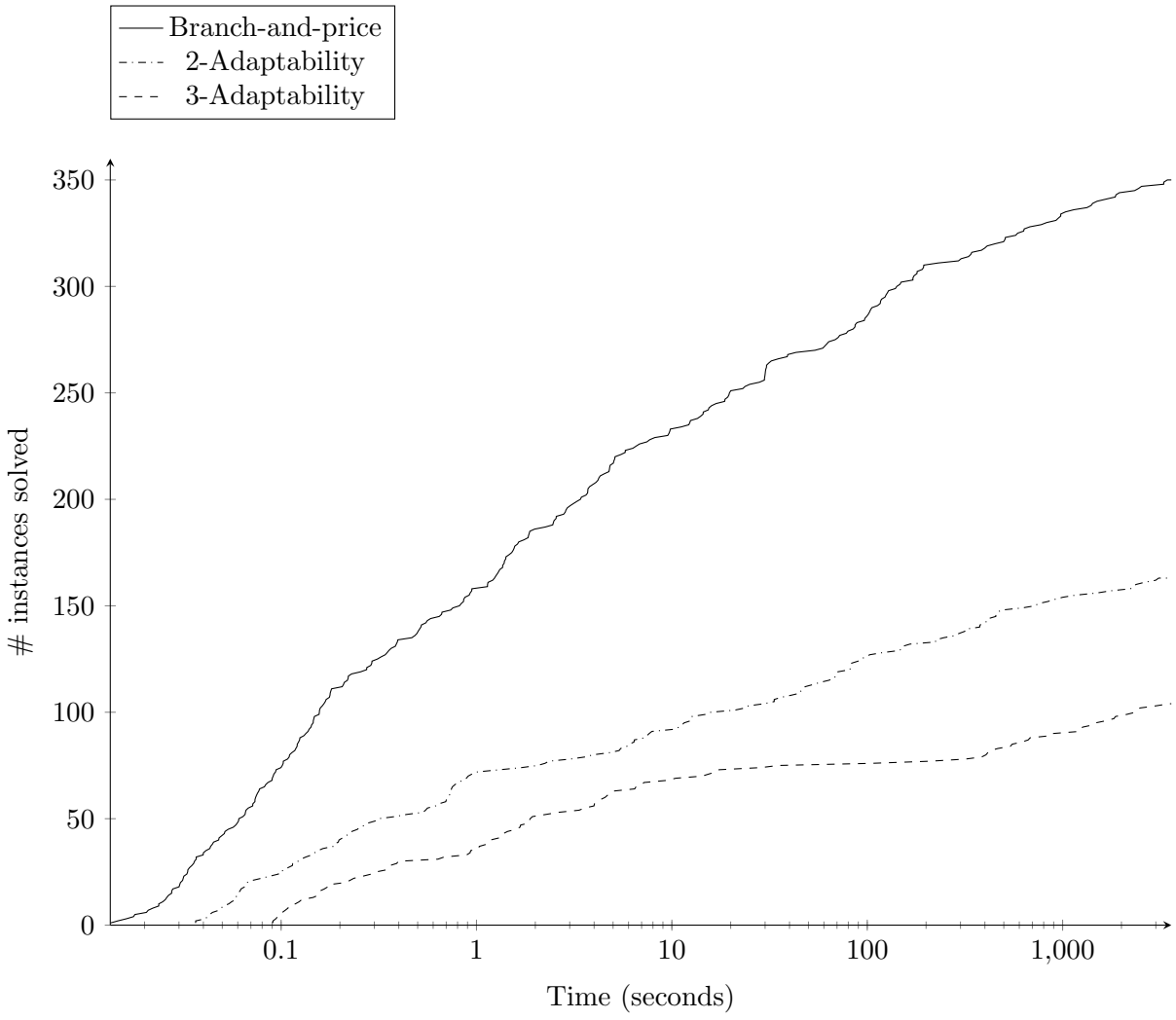


Figure 7: Performance profile comparing all three methods for all instances for the Robust Capital Budgeting problem.

Next, similar to what was done for the knapsack instances, we report the number of instances based on the number of active columns in the best primal solution reported by the branch-and-price algorithm in Table 9. As can be seen in this table, in all the instances we considered the number of active columns required was smaller than 9. Additionally, for instances with smaller number of items, the percentage of instances that required 3 columns or less was high. Accordingly, one would expect the $K-$adaptability methods to provide very good approximations for the instances considered. The relative gaps between

the primal solutions provided by $K-$adaptability methods and the column generation-based approach are presented in Table 10. This difference is calculated as $2 * \frac{z_{\mathrm{CG}} - z_{\mathrm{K}}}{z_{\mathrm{CG}} + z_{\mathrm{K}}}$, and is reported only for those instances where the optimal branch-and-price solution had at least 3 active columns. Here, average 3-Adaptability gaps are higher for larger instances as the size of the model probably hindered the search for a good primal solution. The results confirm our expectation where the 2-Adaptability gaps are always below %1 on average (except for $N = 10$) although the maximum gap can be higher. Interestingly, the relative gap decreases with increasing number of projects N. Indeed, the order of magnitude of the absolute gap remained the same with increasing N whereas the objective value increased, which therefore resulted in ever smaller relative gaps reported.

	$N = 10$	$N = 20$	$N = 30$	$N = 40$	$N = 50$	$N = 100$
K=1	33	17	16	14	10	10
K=2	18	19	3	4	6	5
K=3	5	8	6	4	0	0
K=4	2	11	13	3	5	1
K=[5,9]	2	5	22	35	39	44

Table 9: Number of instances categorized by the number of active columns in the best primal solution reported by the branch-and-price algorithm.

	$N = 10$	$N = 20$	$N = 30$	$N = 40$	$N = 50$	$N = 100$
Avg 2-Adapt (%)	1.91	0.27	0.38	0.44	0.32	0.12
Max 2-Adapt (%)	6.29	1.36	1.50	2.01	1.20	0.50
Avg 3-Adapt (%)	0.28	0.19	0.61	0.44	0.35	0.14
Max 3-Adapt (%)	0.99	1.97	3.57	1.43	1.09	0.75

Table 10: Average and maximum gaps between the best primal solution provided by the $K-$adaptability method and the branch-and-price algorithm when at least 3 columns were active in an optimal solution.

5 Conclusion

In this paper, we study a class of two-stage robust binary optimization problems with objective uncertainty. We propose for these problems a deterministic equivalent formulation through the convexification of the recourse feasible region. We then explore this formulation through the lens of a relaxation that makes it possible to exploit the problem structure and leads to a solution methodology by the branch-and-price algorithm. We provide sufficient conditions for this relaxation to be exact, and propose alternative modeling approaches when this is not the case. The formulations we propose lead also to complexity results on the problem class we study. More specifically, we show that the two-stage robust binary optimization problems with objective uncertainty are NP-complete. Finally, we present two applications where the methodology we propose can be utilized. We provide numerical results on these problems, comparing our methodology to the approximate solution methods known as $K-$adaptability. Our results show that the methodology we propose is able to find better solutions in solution times that are up to 4 orders of magnitudes smaller than other methods. We also show that our method is able to scale to instance sizes where the approximate solution methods are not able to converge within reasonable solution times.

Our results additionally evoke many research questions to be considered, first-and-foremost being the adaptation of such methods to two-stage robust binary optimization problems with right-hand-side uncertainty. Secondly, our numerical results reveal an interesting observation on finite adaptability methods, that is these methods seem to find near optimal solutions for the problems and instances considered. It is then natural to ask whether this is often the case, or whether instances where finitely adaptable solutions are arbitrarily away from being optimal can be constructed, and if so how. Finally, we believe that there is still a considerable amount of research work to be done in the finite adaptability literature. Although, these methods seem to provide good solutions their poor convergence behavior is a deterring factor in their utilization. Different formulations or solution strategies based on decomposition for these approaches should be considered to see whether this behavior can be improved.

Acknowledgement

Experiments presented in this paper were carried out using the PlaFRIM experimental testbed, supported by Inria, CNRS (LABRI and IMB), Université de Bordeaux, Bordeaux INP and Conseil Régional d'Aquitaine (see `https://www.plafrim.fr/`).

References

[1] Alper Atamtürk and Muhong Zhang. Two-stage robust network flow and design under demand uncertainty. *Operations Research*, 55(4):662–673, 2007.

[2] Josette Ayoub and Michael Poss. Decomposition for adjustable robust linear optimization subject to uncertainty polytope. *Computational Management Science*, 13(2):219–239, 2016.

[3] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust optimization*, volume 28. Princeton University Press, 2009.

[4] Aharon Ben-Tal, Alexander Goryashko, Elana Guslitzer, and Arkadi Nemirovski. Adjustable robust solutions of uncertain linear programs. *Mathematical Programming*, 99(2):351–376, 2004.

[5] Dimitris Bertsimas, David B Brown, and Constantine Caramanis. Theory and applications of robust optimization. *SIAM review*, 53(3):464–501, 2011.

[6] Dimitris Bertsimas and Constantine Caramanis. Finite adaptability in multistage linear optimization. *IEEE Transactions on Automatic Control*, 55(12):2751–2766, 2010.

[7] Dimitris Bertsimas and Iain Dunning. Multistage robust mixed-integer optimization with adaptive partitions. *Operations Research*, 64(4):980–998, 2016.

[8] Dimitris Bertsimas and Angelos Georghiou. Design of near optimal decision rules in multistage adaptive mixed-integer optimization. *Operations Research*, 63(3):610–627, 2015.

[9] Dimitris Bertsimas and Angelos Georghiou. Binary decision rules for multistage adaptive mixed-integer optimization. *Mathematical Programming*, 167(2):395–433, 2018.

[10] Dimitris Bertsimas, Eugene Litvinov, Xu Andy Sun, Jinye Zhao, and Tongxin Zheng. Adaptive robust optimization for the security constrained unit commitment problem. *IEEE Transactions on Power Systems*, 28(1):52–63, 2013.

[11] Christoph Buchheim and Jannis Kurtz. Min–max–min robust combinatorial optimization. *Mathematical Programming*, 163(1-2):1–23, 2017.

[12] André Chassein, Marc Goerigk, Jannis Kurtz, and Michael Poss. Faster algorithms for min-max-min robustness for combinatorial problems with budgeted uncertainty. *European Journal of Operational Research*, 2019.

[13] Xin Chen, Melvyn Sim, Peng Sun, and Jiawei Zhang. A linear decision-based approximation approach to stochastic programming. *Operations Research*, 56(2):344–357, 2008.

[14] Xin Chen and Yuhan Zhang. Uncertain linear programs: Extended affinely adjustable robust counterparts. *Operations Research*, 57(6):1469–1482, 2009.

[15] François Clautiaux, Saïd Hanafi, Rita Macedo, Marie-Émilie Voge, and Cláudio Alves. Iterative aggregation and disaggregation algorithm for pseudo-polynomial network flow models with side constraints. *European Journal of Operational Research*, 258(2):467–477, 2017.

[16] Virginie Gabrel, Cécile Murat, and Aurélie Thiele. Recent advances in robust optimization: An overview. *European journal of operational research*, 235(3):471–483, 2014.

[17] Angelos Georghiou, Wolfram Wiesemann, and Daniel Kuhn. Generalized decision rule approximations for stochastic programming via liftings. *Mathematical Programming*, 152(1-2):301–338, 2015.

[18] Joel Goh and Melvyn Sim. Distributionally robust optimization and its tractable approximations. *Operations research*, 58(4-part-1):902–917, 2010.

[19] Bram L Gorissen, İhsan Yanıkoğlu, and Dick den Hertog. A practical guide to robust optimization. *Omega*, 53:124–137, 2015.

[20] Grani A Hanasusanto, Daniel Kuhn, and Wolfram Wiesemann. K-adaptability in two-stage robust binary programming. *Operations Research*, 63(4):877–891, 2015.

[21] Ruiwei Jiang, Muhong Zhang, Guang Li, and Yongpei Guan. Two-stage network constrained robust unit commitment problem. *European Journal of Operational Research*, 234(3):751–762, 2014.

[22] Nicolas Kämmerling and Jannis Kurtz. Oracle-based algorithms for binary two-stage robust optimization. *arXiv preprint arXiv:1905.05257*, 2019.

[23] Daniel Kuhn, Wolfram Wiesemann, and Angelos Georghiou. Primal and dual linear decision rules in stochastic and robust optimization. *Mathematical Programming*, 130(1):177–209, 2011.

[24] R Kipp Martin, Ronald L Rardin, and Brian A Campbell. Polyhedral characterization of discrete dynamic programming. *Operations research*, 38(1):127–138, 1990.

[25] J v Neumann. Zur theorie der gesellschaftsspiele. *Mathematische annalen*, 100(1):295–320, 1928.

[26] Artur Pessoa, Ruslan Sadykov, Eduardo Uchoa, and François Vanderbeck. Automation and combination of linear-programming based stabilization techniques in column generation. *INFORMS Journal on Computing*, 30(2):339–360, 2018.

[27] David Pisinger. Where are the hard knapsack problems? *Computers & Operations Research*, 32(9):2271–2284, 2005.

[28] Krzysztof Postek and Dick den Hertog. Multistage adjustable robust mixed-integer optimization via iterative splitting of the uncertainty set. *INFORMS Journal on Computing*, 28(3):553–574, 2016.

[29] Ruslan Sadykov, François Vanderbeck, Artur Pessoa, Issam Tahiri, and Eduardo Uchoa. Primal heuristics for branch and price: The assets of diving methods. *INFORMS Journal on Computing*, 31(2):251–267, 2019.

[30] Anirudh Subramanyam, Chrysanthos E Gounaris, and Wolfram Wiesemann. K-adaptability in two-stage mixed-integer robust optimization. *arXiv preprint arXiv:1706.07097*, 2017.

[31] Shunji Tanaka, Shuji Fujikuma, and Mituhiko Araki. An exact algorithm for single-machine scheduling without machine idle time. *Journal of Scheduling*, 12(6):575–593, 2009.

[32] Aurélie Thiele, Tara Terry, and Marina Epelman. Robust linear optimization with recourse. *Rapport technique*, pages 4–37, 2009.

[33] Phebe Vayanos, Daniel Kuhn, and Berç Rustem. Decision rules for information discovery in multi-stage stochastic programming. In *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, pages 7368–7373. IEEE, 2011.

[34] Chaoyue Zhao, Jianhui Wang, Jean-Paul Watson, and Yongpei Guan. Multi-stage robust unit commitment considering wind and demand response uncertainties. *IEEE Transactions on Power Systems*, 28(3):2708–2717, 2013.

[35] Long Zhao and Bo Zeng. An exact algorithm for two-stage robust optimization with mixed integer recourse problems. *submitted, available on Optimization-Online. org*, 2012.

[36] Long Zhao and Bo Zeng. Robust unit commitment problem with demand response and wind energy. In *Power and Energy Society General Meeting, 2012 IEEE*, pages 1–8. IEEE, 2012.

6 Appendix

6.1 Robust knapsack problem K-Adaptability model

We write the mixed integer programming reformulation for the K-Adaptability version of the robust knapsack problem of Section 4.1 following [20]:

$$
\begin{aligned}
\min \quad & \sum_{i\in\mathcal{I}}(f_i-\bar{p}_i)x_i - \sum_{i\in\mathcal{I}} f_i \sum_{k=1}^{K} z_i^k + \Gamma u + \sum_{i\in\mathcal{I}} v_i \\
\text{s.t.} \quad & u + v_i \geq \hat{p}_i \sum_{k=1}^{K}(z_i^k - w_i^k) && \forall i \in \mathcal{I} \\
& \sum_{i\in\mathcal{I}} c_i y_i^k + t_i r_i^k \leq C && \forall k = 1,\dots,K \\
& r_i^k \leq y_i^k && \forall i \in \mathcal{I}, \forall k = 1,\dots,K \\
& y_i^k \leq x_i && \forall i \in \mathcal{I}, \forall k = 1,\dots,K \\
& \sum_{k=1}^{K} \mu^k = 1 \\
& z_i^k \leq \mu^k \qquad w_i^k \leq \mu^k && \forall i \in \mathcal{I}, \forall k = 1,\dots,K \\
& z_i^k \leq y_i^k \qquad w_i^k \leq r_i^k && \forall i \in \mathcal{I}, \forall k = 1,\dots,K \\
& z_i^k \geq \mu^k + y_i^k - 1 \qquad w_i^k \geq \mu^k + r_i^k - 1 && \forall i \in \mathcal{I}, \forall k = 1,\dots,K \\
& \boldsymbol{x} \in \{0,1\}^I, \boldsymbol{y} \in \{0,1\}^{K\times I}, \boldsymbol{r} \in \{0,1\}^{K\times I} \\
& \boldsymbol{z} \in [0,1]^{K\times I}, \boldsymbol{w} \in [0,1]^{K\times I}, \boldsymbol{\mu} \in [0,1]^K \\
& u \in \mathbb{R}_+, \boldsymbol{v} \in \mathbb{R}_+^I
\end{aligned}
$$

where z_i^k and w_i^k are variables introduced for the linearization of bilinear terms $\mu^k y_i^k$ and $\mu^k r_i^k$ resulting from the dualization of the inner minimization problem over K possible recourse solutions.

6.2 Robust capital budgeting $K-$adaptability model

We write the mixed integer programming reformulation for the K-Adaptability version of the robust capital budgeting problem of Section 4.2 following [20]:

$$
\begin{aligned}
\max \quad & -\lambda x_0 + \sum_{i\in\mathcal{N}} \bar{p}_i (x_i + f\sum_{k=1}^{K} z_i^k) - \lambda\mu \sum_{k=1}^{K} z_0^k - \sum_{j=1}^{M}(u_j + v_j) \\
\text{s.t.} \quad & u_j - v_j = \sum_{i\in\mathcal{N}} \frac{Q_{i,j}\bar{p}_i}{2}(x_i + f\sum_{k=1}^{K} z_i^k) && \forall j = 1,\dots,M \\
& \sum_{i\in\mathcal{N}} c_i x_i \leq B + C_1 x_0 \\
& \sum_{i\in\mathcal{N}} c_i (x_i + y_i^k) \leq B + C_1 x_0 + C_2 y_0^k && \forall k = 1,\dots,K
\end{aligned}
$$

$$
\begin{aligned}
& x_i + y_i^k \leq 1 && \forall i \in \mathcal{N}, \forall k = 1, \ldots, K \\
& \sum_{k=1}^{K} \rho^k = 1 && \\
& z_i^k \leq \rho^k && \forall i \in \mathcal{N} \cup \{0\}, \forall k = 1, \ldots, K \\
& z_i^k \leq y_i^k && \forall i \in \mathcal{N} \cup \{0\}, \forall k = 1, \ldots, K \\
& z_i^k \geq \rho^k + y_i^k - 1 && \forall i \in \mathcal{N} \cup \{0\}, \forall k = 1, \ldots, K \\
& \boldsymbol{x} \in \{0,1\}^N, x_0 \in \{0,1\}, \boldsymbol{y} \in \{0,1\}^{K \times N}, \boldsymbol{y}_0 \in \{0,1\}^K && \\
& \boldsymbol{\rho} \in [0,1]^K, \boldsymbol{z} \in [0,1]^{K \times N}, \boldsymbol{u} \in \mathbb{R}_+^M, \boldsymbol{v} \in \mathbb{R}_+^M &&
\end{aligned}
$$

where z_i^k are variables introduced for the linearization of bilinear terms $\rho^k y_i^k$ resulting from the dualization of the inner minimization problem over K possible recourse solutions.