ARCHIVIO ISTITUZIONALE
DELLA RICERCA

Alma Mater Studiorum Università di Bologna
Archivio istituzionale della ricerca

Adjustable Robust Optimization with Discrete Uncertainty

This is the final peer-reviewed author's accepted manuscript (postprint) of the following publication:

Published Version:
Lefebvre, H., Malaguti, E., Monaci, M. (2024). Adjustable Robust Optimization with Discrete Uncertainty. INFORMS JOURNAL ON COMPUTING, 36(1), 78-96 [10.1287/ijoc.2022.0086].

Availability:
This version is available at: https://hdl.handle.net/11585/967605 since: 2024-04-15

Published:

DOI: http://doi.org/10.1287/ijoc.2022.0086

Terms of use:

Some rights reserved. The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (https://cris.unibo.it/).
When citing, please refer to the published version.

(Article begins on next page)

11 November 2026

The final version of this paper was published on the
INFORMS Journal on Computing 36(1):78-96.
https://doi.org/10.1287/ijoc.2022.0086

Adjustable robust optimization with discrete uncertainty

Henri Lefebvre
Department of Mathematics, Trier University, Universitätsring 15, 54296 Trier, Germany. henri.lefebvre@uni-trier.de

Enrico Malaguti
Dipartimento di Ingegneria dell'Energia Elettrica e dell'Informazione "Guglielmo Marconi", Università di Bologna, Viale del Risorgimento, 2, Bologna, 40136, BO, Italy. enrico.malaguti@unibo.it

Michele Monaci
Dipartimento di Ingegneria dell'Energia Elettrica e dell'Informazione "Guglielmo Marconi", Università di Bologna, Viale del Risorgimento, 2, Bologna, 40136, BO, Italy. michele.monaci@unibo.it

In this paper, we study Adjustable Robust Optimization (ARO) problems with discrete uncertainty. Under a very general modeling framework, we show that such two-stage robust problems can be exactly reformulated as ARO problems with objective uncertainty only. This reformulation is valid with and without the fixed recourse assumption and is not limited to continuous wait-and-see decision variables, unlike most of the existing literature. Additionally, we extend an enumerative algorithm akin to a branch-and-cut scheme for which we study the asymptotic convergence. We discuss how to apply the reformulation on two variants of well known optimization problems: a Facility Location Problem in which uncertainty may affect the capacity values and a Multiple Knapsack Problem with uncertain weights, and we report extensive computational results demonstrating the effectiveness of the approach.

Key words: two-stage robust optimization; discrete uncertainty; reformulation; branch-and-cut; computational experiments

1. Introduction

Classical optimization problems usually model a single-stage decision flow where every decision must be taken here and now. In practice, however, every *here-and-now* decision is a call for future *wait-and-see* decisions to be made, which will depend on the former and, potentially, on future uncertain events which cannot be observed immediately.

In this paper, we consider an optimization problem of the form

$$\min \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}^\top \boldsymbol{y} \tag{1}$$

$$\text{s.t.} \boldsymbol{T}\boldsymbol{x} + \boldsymbol{H}\boldsymbol{y} \leq \boldsymbol{f} \tag{2}$$

$$\boldsymbol{x} \in X \tag{3}$$

$$\boldsymbol{y} \in Y \tag{4}$$

$$\boldsymbol{0} \leq \boldsymbol{y} \leq \boldsymbol{u} \tag{5}$$

where $\boldsymbol{x}$ denotes the vector of here-and-now decisions, $\boldsymbol{y}$ represents wait-and-see decisions that can be taken after uncertainty reveals, and $\boldsymbol{c}, \boldsymbol{d}, \boldsymbol{T}, \boldsymbol{H}, \boldsymbol{f}$ and $\boldsymbol{u}$ are real vectors and matrices of appropriate dimension. The objective function includes a cost for both here-and-now and wait-and-see decisions; the corresponding variables are coupled by means of linking constraints (2). Constraints (3) and (4) impose that $\boldsymbol{x}$ and $\boldsymbol{y}$ belong to some polyhedral set; these constraints may also include integrality requirements on some variables, if any. As it happens in practical applications, we assume wait-and-see decisions to be bounded, see, (5). In addition, we restrict our attention to the case in which X is bounded. Finally, we assume that the deterministic problem is feasible.

ASSUMPTION 1. *Problem* (1)–(5) *admits a feasible solution.*

We consider problems in which uncertainty affects coefficients of matrix $\boldsymbol{H}$. Let us denote by $Y(\hat{\boldsymbol{x}}, \hat{\boldsymbol{H}}) = \{\boldsymbol{y} \in Y : \hat{\boldsymbol{H}}\boldsymbol{y} \leq \boldsymbol{f} - \boldsymbol{T}\hat{\boldsymbol{x}}, \boldsymbol{0} \leq \boldsymbol{y} \leq \boldsymbol{u}\}$ the set of feasible wait-and-see decisions for a given here-and-now decision $\hat{\boldsymbol{x}} \in X$ and random outcome $\hat{\boldsymbol{H}}$ of $\boldsymbol{H}$. Finally, let $\mathcal{Q}(\bullet|\boldsymbol{H})$ be an oracle which is able to foresee the actual future outcome $\hat{\boldsymbol{H}}$ of $\boldsymbol{H}$. Then, the goal is to decide here-and-now decision $\boldsymbol{x}^*$ such that

$$\boldsymbol{x}^* \in \operatorname*{argmin}_{\boldsymbol{x} \in X} \left\{ \boldsymbol{c}^\top \boldsymbol{x} + \mathcal{Q}\left(\min_{\boldsymbol{y} \in Y(\boldsymbol{x}, \boldsymbol{H})} \boldsymbol{d}^\top \boldsymbol{y} \middle| \boldsymbol{H} \right) \right\}. \tag{6}$$

We refer to this class of problems as adjustable problems so as to enlight the two-stage nature of the decision flow. Two main approaches have been proposed in the scientific literature to model the fact that the oracle $\mathcal{Q}$ is unknown in practice. On the one hand, stochastic optimization assumes some probabilistic distribution knowledge about $\boldsymbol{H}$ and replaces oracle $\mathcal{Q}(\bullet|\boldsymbol{H})$ with $\mathbb{E}(\bullet|\boldsymbol{H})$. An interested reader may refer to Birge and Louveaux (2011), Prékopa (2013), and Shapiro et al. (2021) for more details on stochastic optimization. On the other hand, robust optimization only assumes to know a subset of the support of the density function of $\boldsymbol{H}$, say $\mathcal{H}$, and replaces $\mathcal{Q}(\bullet|\boldsymbol{H})$ with $\max\{\bullet : \hat{\boldsymbol{H}} \in \mathcal{H}\}$. This approach is also referred to as *worst-case* optimization as it decides $\boldsymbol{x}^*$ in the least advantageous outcome for $\boldsymbol{H}$; see Soyster (1973), Ben-Tal and Nemirovski (1999), and Bertsimas and Sim (2004) for more details.

Related literature There is a wide literature on min-max and min-max-min problems with continuous and discrete uncertainty sets, see, e.g., the recent surveys Buchheim and Kurtz (2018) and Gorissen et al. (2015).

Already in the case of min-max optimization and objective discrete uncertainty, it has been shown by Buchheim and Kurtz (2018) that the problem is NP-hard as soon as the here-and-now decisions are binary and the uncertainty set contains exactly two scenarios. Since min-max-min problems are a generalization of min-max optimization, this complexity result trivially extends and several approaches have been designed to approximately or exactly address the general min-max-min case.

Assuming that the convex hull of the uncertainty set is known, one considerably simplifies the problem by replacing Ξ with $\text{conv}(\Xi)$, leading to a relaxation of the original problem. In such cases, typical approaches designed for continuous uncertainty sets may be employed.

In Ben-Tal et al. (2004), based on the consideration that min-max-min problems are equivalently written as min-max problems over the set of general functions $\boldsymbol{y} : \Xi \to \mathbb{R}_+^{n_Y}$, referred to as *decision rules*, the authors proposed a conservative approximation for cases in which the wait-and-see decisions are continuous by restricting such functions to be *affine decision rules* (ADR) of the form $\boldsymbol{y} : \boldsymbol{\xi} \mapsto \bar{\boldsymbol{y}} + \boldsymbol{A}\boldsymbol{\xi}$. The authors show that ADRs lead to tractable problems for cases with fixed recourses (i.e., with a known $\boldsymbol{H}$ and uncertain $\boldsymbol{f}$) and continuous second-stage decisions. Unfortunately, this approach cannot be extended to the case where the wait-and-see decisions are integer as it would imply that $\boldsymbol{y}(\boldsymbol{\xi})$ be integer for all $\boldsymbol{\xi} \in \text{conv}(\Xi)$.

For the case where the second stage is mixed-integer, Bertsimas and Georghiou (2017) have developed linearly parametrized binary decision rules which were shown to provide high quality solutions. However, such an approach fails at guaranteeing optimality and has been shown to lead to mixed-integer formulations whose size grows exponentially with respect to the input data.

Finite adaptability was introduced in Bertsimas and Caramanis (2010) and constitutes another major conservative approximation for general min-max-min problems. It consists in selecting a fixed number of recourse decisions, sometimes referred to as *contingency plans*, before the uncertainty reveals and to select, in the second-stage, one recourse decision among the pre-selected plans. This approximation has been studied, e.g., in Hanasusanto

et al. (2015), Postek and den Hertog (2016), Bertsimas and Dunning (2016), Subramanyam et al. (2019), Romeijnders and Postek (2021) and Malaguti et al. (2022).

Regarding exact approaches, most of the literature addresses cases with right-hand side uncertainty and continuous second stage or objective uncertainty. In the latter case, Kämmerling and Kurtz (2020) introduced a branch-and-cut algorithm for the case where the here-and-now decisions are all binary. The same restriction of objective uncertainty is found in Arslan and Detienne (2022), where the authors proposed a deterministic reformulation of the problem in case the uncertainty set is conv(Ξ) and all linking constraints are linear and do not include continuous here-and-now variables. The resulting reformulation is then solved by using a branch-and-price algorithm. The extension to linking constraints defined by convex functions and possibly including continuous here-and-now variables has been presented in Detienne et al. (2021). A column-and-constraint generation algorithm was introduced in Zeng and Zhao (2012) for the case where the second stage is continuous. Later, this approach was extended in Zhao and Zeng (2013) to address problems with mixed-integer wait-and-see decisions. Recently, Subramanyam (2022) considered adjustable problems with binary uncertainty and introduced an exact method based on a reformulation of the problem and Lagrangian relaxation, thus obtaining a problem in which uncertainty appears in the objective function only. This paper explicitly addresses and computationally evaluates the case where uncertainty affects the right-hand side coefficients only. However, by using a suitable reformulation, this setting can be generalized to the case where uncertainty affects the coefficients of the constraint matrix, i.e., the setting of the present paper.

Paper contribution In this paper, we present an alternative exact approach for problems where the wait-and-see decisions are mixed-integer and uncertainty affects the problem constraints, and we provide the following main contributions:

- In Section 2, we introduce our modeling framework and discuss its generality. We show that virtually any linearly-constrained Adjustable Robust Problem with discrete uncertainty can fit our framework. Among others, our model includes problems with and without the fixed recourse assumption, and allows for both mixed-integer first- and second-stage decisions.
- In Section 3, we present our main theoretical results. We prove the general validity of a non-trivial reformulation of ARO problems with discrete uncertainty as objective-uncertain ARO problems. This reformulation is based on polyhedral results and Lagrangian duality.

• In Section 4, we extend a recent algorithm (Kämmerling and Kurtz 2020) from the ARO literature to the mixed-integer case while it was originally designed for objective-uncertain problems with binary first-stage decisions only and continuous uncertainty set. We also explicitly discuss the behaviour of the algorithm in case of ARO problems which are infeasible (i.e., $\forall x \in X, \exists \hat{H} \in \mathcal{H}$ such that $Y(\boldsymbol{x}, \hat{H}) = \emptyset$), or which do not have complete recourse (i.e., $\exists \hat{x} \in X$ and $\exists \hat{H} \in \mathcal{H}$ such that $Y(\hat{x}, \hat{H}) = \emptyset$).

• In Section 5, we report computational results for the considered class of hard optimization problems, showing that our approach is able to solve medium-size instances within a reasonable computing time.

2. Problem modeling

In the following, we will denote by n_X and n_Y the number of here-and-now and wait-and-see decisions, respectively, and by m_Y the number of linking constraints (2).

2.1. Uncertainty model

Our modelling of uncertainty is based on the following assumption. We will show in the next section that this assumption can be done without loss of generality.

ASSUMPTION 2. *For all $i = 1, ..., m_Y$ and all $j = 1, ..., n_Y$, let $\underline{h}_{ij}$ and $\bar{h}_{ij}$ be two real numbers such that $\underline{h}_{ij} \leq \bar{h}_{ij}$. The set $\mathcal{H}$ of possible outcomes for $\boldsymbol{H}$ is such that $\mathcal{H} \subseteq \{\hat{\boldsymbol{H}} : (\hat{h}_{ij} = \underline{h}_{ij} \vee \hat{h}_{ij} = \bar{h}_{ij}), i = 1, ..., m_Y, j = 1, ..., n_Y\}$.*

To ease the presentation, we introduce a binary set $\Xi \subseteq \{0,1\}^{m_Y \times n_Y}$ used to encode the combinatorial aspects of $\mathcal{H}$ as follows: for all $\boldsymbol{\xi} \in \Xi$ and any (i,j), $\xi_{ij} = 0$ if $\hat{h}_{ij} = \underline{h}_{ij}$ and $\xi_{ij} = 1$ if $\hat{h}_{ij} = \bar{h}_{ij}$. For a given $\hat{\boldsymbol{x}} \in X$ and $\hat{\boldsymbol{H}} \in \mathcal{H}$, with a small abuse of notation we denote set $Y(\hat{\boldsymbol{x}}, \hat{\boldsymbol{H}})$ as $Y(\hat{\boldsymbol{x}}, \hat{\boldsymbol{\xi}})$ for an appropriate $\hat{\boldsymbol{\xi}} \in \Xi$. Using this notation, set $Y(\hat{\boldsymbol{x}}, \hat{\boldsymbol{\xi}})$ includes all those elements fulfilling constraints (4), (5), and

$$\sum_{j=1}^{n_X} t_{ij}\, \hat{x}_j + \sum_{j=1}^{n_Y} \left(\underline{h}_{ij} y_j + (\bar{h}_{ij} - \underline{h}_{ij})\, \hat{\xi}_{ij}\, y_j \right) \leq f_i \qquad i = 1, ..., m_Y \tag{7}$$

Accordingly, the class of problems which we are addressing can be reformulated as follows:

$$\min_{\boldsymbol{x} \in X} \left\{ \boldsymbol{c}^\top \boldsymbol{x} + \max_{\boldsymbol{\xi} \in \Xi} \min_{\boldsymbol{y} \in Y(\boldsymbol{x}, \boldsymbol{\xi})} \boldsymbol{d}^\top \boldsymbol{y} \right\}. \tag{8}$$

In the robust optimization terminology, matrix $\boldsymbol{H}$ is often called the *recourse* matrix. The class of problems we consider in this paper can therefore be summarized as *adjustable robust problems with mixed-integer first and second stage, uncertain recourse matrix and binary uncertainty.*

2.2. Expressiveness of our model

In this section, we show that Assumption 2 can be done without loss of generality. In addition, we discuss how a large variety of uncertain problems can be cast in our framework without introducing further assumptions.

More than two possible outcomes Assume that a generic coefficient, say h_{ij}, has more than two possible outcomes. Let $R > 2$ be this number and denote by $\hat{h}_{ij}^1, ..., \hat{h}_{ij}^R$ the possible values, sorted by increasing order. In the i-th constraint, we replace variable y_j by R additional variables $y_{ij}^1, ..., y_{ij}^R$, the r-th associated with an uncertain coefficient h_{ij}^r having $\underline{h}_{ij}^r = 0$ and $\bar{h}_{ij}^r = \hat{h}_{ij}^r$. In order to impose that exactly one value is selected by the uncertainty for coefficient h_{ij}, the following constraint

$$\sum_{r=1}^{R} \xi_{ij}^r = 1 \tag{9}$$

has to be added to the definition of Ξ. Finally, the relationship between the original y_j and the additional y_{ij}^r variables can be enforced by imposing the following constraints

$$y_j = y_{ij}^r \quad r = 1, ..., R \tag{10}$$

Constraints (10) can simply be added to the definition of set Y.

"$\geq$"-inequalities and equalities In our definition of the problem, all linking constraints (2) are assumed to be written in the $\leq$ form. Since we make no assumption on the sign of coefficients of $\hat{H}$, any $\geq$ inequality can be rewritten in $\leq$ form.

As to equations, we assume they are replaced by a pair of $\leq$ inequalities before the reformulation is applied. Let i_1 and i_2 be the indices of the inequalities derived from a given equation and notice that each variable y_j has, in these constraints, the same coefficients but with complementary signs: $\bar{h}_{i_1 j} = -\underline{h}_{i_2 j}$ and $\underline{h}_{i_1 j} = -\bar{h}_{i_2 j}$. Hence, consistency in the realization of the same parameter can be enforced by the following constraint

$$\xi_{i_1 j} + \xi_{i_2 j} = 1 \tag{11}$$

More in general, our framework allows modelling of situations in which the realization of a pair of coefficients of $\hat{H}$ is correlated; for example, if the first coefficient takes its smallest value, then the second assumes its smallest value as well, and vice-versa. These situations can be handled by adding suitable constraints to the definition of Ξ.

Right-hand side uncertainty Though our uncertainty model assumes recourse matrix uncertainty, it is clear that right-hand side uncertainty is comprised within our framework. In other words, it can be used for modelling contexts with and without the fixed recourse assumption. To see this, consider one constraint whose right-hand side is uncertain, i.e., assume that f_i is replaced by $\underline{f}_i + (\bar{f}_i - \underline{f}_i)\xi'_i$ in (7) with $\underline{f}_i \leq \bar{f}_i$ for some (unknown) binary ξ'_i. Then, clearly, adding one decision variable y_{n_Y+1} fixed to 1 which multiplies ξ'_i turns our problem in the desired form.

Objective uncertainty Finally, it is well known (see, e.g., Bertsimas and Sim (2004)) that assuming full knowledge of the objective function is without loss of generality. Indeed, uncertainty in vector $\boldsymbol{d}$ in (1) can be easily handled by introducing an additional (uncertain) constraint that defines the objective function value. Overall, this shows that our modelling approach defines a completely general setting.

3. Theoretical development

This section presents the main theoretical development of our work. We start by reformulating problems of type (8) as adjustable robust problems with objective uncertainty; then, we consider a relaxation introduced in Kämmerling and Kurtz (2020) for this class of problems, extend to our setting an algorithm for the solution of the relaxation and discuss its computational complexity. Finally, we show how to deal with problems for which feasibility is unknown. In Section 4, we will discuss how the relaxation can be embedded in a branch-and-bound algorithm to close the optimality gap.

3.1. Reformulation

The reformulation introduced in this section is based on the complete recourse assumption, i.e., that $Y(\boldsymbol{x},\boldsymbol{\xi}) \neq \emptyset$ for all $\boldsymbol{x} \in X$ and $\boldsymbol{\xi} \in \Xi$. Although this is a mild assumption, quite common in robust optimization, we will show in Section 3.4 that our solution method does not require this assumption to hold.

We first linearize every product involving variables ξ_{ij} and y_j for some (i,j) in constraints (7). An exact reformulation of each product can be obtained by introducing continuous variables $z_{ij} = \xi_{ij} y_j$ and adding the following linear constraints to the definition of set $Y(\hat{\boldsymbol{x}},\hat{\boldsymbol{\xi}})$.

$$z_{ij} \leq u_j \xi_{ij} \qquad i = 1,...,m_Y, j = 1,...,n_Y \tag{12}$$

$$z_{ij} \leq y_j \qquad i = 1, ..., m_Y, j = 1, ..., n_Y \tag{13}$$

$$z_{ij} \geq y_j - (1 - \xi_{ij})u_j \qquad i = 1, ..., m_Y, j = 1, ..., n_Y \tag{14}$$

$$z_{ij} \geq 0 \qquad i = 1, ..., m_Y, j = 1, ..., n_Y \tag{15}$$

Let us introduce, for all $\boldsymbol{x} \in X$ and all $\boldsymbol{\xi} \in \Xi$, set $Z(\boldsymbol{x}, \boldsymbol{\xi})$ as the set of decisions $(\boldsymbol{y}, \boldsymbol{z}) \in \mathbb{R}^{n_Y} \times \mathbb{R}^{m_Y \times n_Y}$ fulfilling constraints (4), (5),(13), (14) and (15) as well as constraints (16) obtained by replacing each bilinear terms in (7).

$$\sum_{j=1}^{n_X} t_{ij} x_j + \sum_{j=1}^{n_Y} \left(\underline{h}_{ij} y_j + (\bar{h}_{ij} - \underline{h}_{ij}) z_{ij} \right) \leq f_i \qquad i = 1, ..., m_Y \tag{16}$$

Remark 1. We enlight that constraints (12) are not part of the definition of $Z(\boldsymbol{x}, \boldsymbol{\xi})$ and that this can be done without changing the optimal objective value of the second stage.

Proof: We show that, given any solution of the second-stage problem violating (12), one can build a solution respecting (12) with the same objective value. Let $\bar{i}, \bar{j}$ be the indices of a violated constraint (12) and note that $z_{\bar{i}\bar{j}} > 0$. Since $z_{\bar{i}\bar{j}} \leq y_{\bar{j}} \leq u_{\bar{j}}$, violation of the constraint implies $\xi_{\bar{i}\bar{j}} = 0$. Therefore (14) reduces to $z_{\bar{i}\bar{j}} \geq y_{\bar{j}} - u_{\bar{j}}$, which remains satisfied by setting $z_{\bar{i}\bar{j}} = 0$. Note that, since $(\bar{h}_{\bar{i}\bar{j}} - \underline{h}_{\bar{i}\bar{j}}) \geq 0$ (see Assumption 2), setting $z_{\bar{i}\bar{j}} = 0$ satisfies constraint (16) as well, as it only reduces its left-hand-side. Finally, notice that $z_{\bar{i}\bar{j}}$ does not appear in the objective function, i.e., the two solutions have the same value.

In turn, it is clear that problem (8) is equivalent to the following adjustable robust problem

$$\min_{\boldsymbol{x} \in X} \left\{ \boldsymbol{c}^\top \boldsymbol{x} + \max_{\boldsymbol{\xi} \in \Xi} \min_{(\boldsymbol{y}, \boldsymbol{z}) \in Z(\boldsymbol{x}, \boldsymbol{\xi})} \boldsymbol{d}^\top \boldsymbol{y} \right\} \tag{17}$$

where the linking constraints between $\boldsymbol{y}$, $\boldsymbol{z}$ and $\boldsymbol{\xi}$ are of simpler kind. In the next theorem, we turn problem (17) into an adjustable robust problem where the uncertainty is confined within the objective function. This result follows from a polyhedral analysis result and Lagrangian duality.

Theorem 1. *Problem* (8) *is equivalently solved by the following problem.*

$$\min_{\boldsymbol{x} \in X} \left\{ \sum_{j=1}^{n_X} c_j x_j + \max_{\boldsymbol{\xi} \in \Xi, \boldsymbol{\lambda} \leq 0} \min_{(\boldsymbol{y}, \boldsymbol{z}) \in Z'(\boldsymbol{x})} \sum_{j=1}^{n_Y} \left(d_j y_j + \sum_{i=1}^{m_Y} \lambda_{ij} \xi_{ij} (z_{ij} - y_j) \right) \right\} \tag{18}$$

where $Z'(\boldsymbol{x})$ *is defined as* $Z(\boldsymbol{x}, \boldsymbol{\xi})$ *after omitting constraints* (14).

Proof: As already observed, one can consider problem (17) instead of (8). Then, by linearity of the objective function, condition "$(\boldsymbol{y},\boldsymbol{z}) \in Z(\boldsymbol{x},\boldsymbol{\xi})$" can equivalently be replaced by "$(\boldsymbol{y},\boldsymbol{z}) \in \text{conv}(Z(\boldsymbol{x},\boldsymbol{\xi}))$". Moreover, it holds that, for all $\boldsymbol{\xi} \in \Xi$, $\text{conv}(Z(\boldsymbol{x},\boldsymbol{\xi})) = \text{conv}(Z'(\boldsymbol{x})) \cap \{(\boldsymbol{y},\boldsymbol{z}) : (14)\}$ (see Theorem 4 in appendix). The inner problem asks to minimize $\sum_{j=1}^{n_Y} d_j y_j$ for $(\boldsymbol{y},\boldsymbol{z}) \in Z'(\boldsymbol{x}) \cap \{(\boldsymbol{y},\boldsymbol{z}) : (14)\}$.} By using a Dantzig-Wolfe reformulation of $\text{conv}(Z'(\boldsymbol{x}))$, one can see this problem as an LP for which strong Lagrangian duality holds, provided that the primal problem is feasible. Since we have assumed that $Y(\boldsymbol{x},\boldsymbol{\xi}) \neq \emptyset$ for all $\boldsymbol{x} \in X$ and $\boldsymbol{\xi} \in \Xi$, strong duality holds. The partial Lagrangian dual is given as follows, where $\boldsymbol{\lambda}$ are the dual variables associated to the interdiction constraints (14).

$$\max_{\boldsymbol{\lambda} \leq 0} \min_{(\boldsymbol{y},\boldsymbol{z}) \in Z'(\boldsymbol{x})} \left\{ \sum_{j=1}^{n_Y} d_j y_j + \sum_{i=1}^{m_Y} \sum_{j=1}^{n_Y} \lambda_{ij}((1-\xi_{ij})u_j + z_{ij} - y_j) \right\} \tag{19}$$

By splitting the terms, we obtain:

$$\max_{\boldsymbol{\lambda} \leq 0} \min_{(\boldsymbol{y},\boldsymbol{z}) \in Z'(\boldsymbol{x})} \left\{ \sum_{j=1}^{n_Y} d_j y_j + \sum_{i=1}^{m_Y} \left(\sum_{j:\xi_{ij}=0} \lambda_{ij}(u_j + z_{ij} - y_j) + \sum_{j:\xi_{ij}=1} \lambda_{ij}(z_{ij} - y_j) \right) \right\} \tag{20}$$

We now argue that $\lambda_{ij} = 0$ is optimal whenever $\xi_{ij} = 0$. In this case, λ_{ij} is multiplied by $z_{ij} + u_j - y_j$, which is nonnegative as $z_{ij} \geq 0$ and $y_j \leq u_j$. Given that $\lambda_{ij} \leq 0$, an optimal choice for the outer maximization problem is $\lambda_{ij} = 0$.

The reformulation introduced in Theorem 1 is conceptually simpler than problem (8) as uncertainty interferes within the objective function only. In other words, as shown by the following example, the feasible space does not depend on $\boldsymbol{\xi}$.

EXAMPLE 1. Let us consider a numerical example, in which the here-now-variables have been fixed and the resulting inner optimization problem is as follows:

$$\max_{\boldsymbol{\xi} \in \{0,1\}^2 : \boldsymbol{e}^\top \boldsymbol{\xi} \leq 1} \begin{array}{ll} \min & -y_1 - y_2 \\ \text{s.t.} & (1+\xi_1)y_1 + (1+2\xi_2)y_2 \leq 3 \\ & y_1, y_2 \in \{0,1\} \end{array} \tag{21}$$

It is easily seen (e.g., by enumeration) that the optimal objective value is -1, obtained by choosing $\xi_1 = 0$ and $\xi_2 = 1$. A direct application of Theorem 1 yields the following

reformulation.

$$\max_{\boldsymbol{\xi}\in\{0,1\}^2,\boldsymbol{\lambda}\in\mathbb{R}^2_-:\boldsymbol{e}^\top\boldsymbol{\xi}\leq 1} \begin{array}{ll} \min & -y_1-y_2+\xi_1\lambda_1(z_1-y_1)+\xi_2\lambda_2(z_2-y_2) \\ \text{s.t.} & y_1+z_1+y_2+2z_2\leq 3 \\ & z_1\leq y_1 \\ & z_2\leq y_2 \\ & y_1,y_2\in\{0,1\} \\ & z_1,z_2\geq 0 \end{array} \tag{22}$$

Note that, for any $\boldsymbol{\xi}$, the feasible space does not change. Thus, $(y_1,y_2,z_1,z_2)=(1,1,0,0)$ is always feasible. However, this solution is optimal only when $\xi_1=\xi_2=0$ and the objective is $-y_1-y_2=-2$. Yet, if we consider $\xi_1=0$ and $\xi_2=1$, the objective is now $-y_1-y_2+\lambda_2(z_2-y_2)$. Since $\lambda_2\leq 0$ and $z_2-y_2\leq 0$, the inner minimization problem will "tend to" minimize the distance between z_2 and y_2. With a negative enough value for λ_2, such a penalization will eventually force $y_2=z_2$ since any feasible solution with $y_2\neq z_2$ will be dominated. Observe that, for this example, $\lambda_2=-1$ is enough. Similarly. when $\xi_1=1$ and $\xi_2=0$, one can show that $\lambda_1=-1$ is enough.

Our method moves uncertainty to the objective function by imposing the constraints $z_{ij}=\xi_{ij}y_j$ for each $i=1,\ldots,m_Y$ and $j=1,\ldots,n_Y$, and by relaxing them in a Lagrangian way by using multipliers λ_{ij}. We now discuss a possible way of fixing the multipliers to an optimal value, so as to omit bilinear terms $\lambda_{ij}\xi_{ij}$.

COROLLARY 1. *Let $\lambda^*_{ij}(\boldsymbol{\xi})$ be an optimal solution associated to a given $\boldsymbol{\xi}\in\Xi$ for problem* (19) *and let $\underline{\lambda}_{ij}$ be such that $\underline{\lambda}_{ij}\leq\lambda^*_{ij}(\boldsymbol{\xi})\leq 0$ for all $\boldsymbol{\xi}\in\Xi$. Then, problem* (8) *can be rewritten as follows:*

$$\min_{\boldsymbol{x}\in X}\left\{\sum_{j=1}^{n_X}c_jx_j+\max_{\boldsymbol{\xi}\in\Xi}\min_{(\boldsymbol{y},\boldsymbol{z})\in Z'(\boldsymbol{x})}\sum_{j=1}^{n_Y}\left(d_jy_j+\sum_{i=1}^{m_Y}\underline{\lambda}_{ij}\xi_{ij}(z_{ij}-y_j)\right)\right\}. \tag{23}$$

Proof: By definition of $\lambda^*_{ij}(\boldsymbol{\xi})$, problem (18) is equivalent to

$$\min_{\boldsymbol{x}\in X}\left\{\sum_{j=1}^{n_X}c_jx_j+\max_{\boldsymbol{\xi}\in\Xi}\min_{(\boldsymbol{y},\boldsymbol{z})\in Z'(\boldsymbol{x})}\sum_{j=1}^{n_Y}\left(d_jy_j+\sum_{i=1}^{m_Y}\lambda^*_{ij}(\boldsymbol{\xi})\xi_{ij}(z_{ij}-y_j)\right)\right\}. \tag{24}$$

By optimality, for a given $\boldsymbol{\xi}'\in\Xi$, since $\lambda^*_{ij}(\boldsymbol{\xi}')\leq 0$ and $z_{ij}\leq y_j$, any value $\underline{\lambda}_{ij}\leq\lambda^*_{ij}(\boldsymbol{\xi}')$ is also an optimal solution for $\boldsymbol{\xi}'$. This achieves the proof.

Corollary 1 therefore eliminates the need for variables $\boldsymbol{\lambda}$ by replacing them with a fixed and exact violation penalization in the objective. However, it does not provide a practical value for $\underline{\lambda}_{ij}$ which would not be problem-specific. In the next lemma, we give such a value for a large class of problems.

LEMMA 1. *In Corollary 1, assume that* $d_j \leq 0$ *and* $\underline{h}_{ij} \geq 0$ *(*$i = 1, ..., m_Y$ *and* $j = 1, ..., n_Y$*). Then, one can safely use* $\underline{\lambda}_{ij} = d_j$ *for all* $i = 1, ..., m_Y$ *and* $j = 1, ..., n_Y$*, provided that the integrality requirement on variable* z_{ij} *is added to the definition of* $Z'(\boldsymbol{x})$ *for all* $i = 1, ..., m_Y$ *and all* $j = 1, ..., n_Y$ *such that* y_j *is an integer variable.*

Proof: Note that, for a given $\boldsymbol{\xi} \in \Xi$, the inner maximization term of (18) can be reformulated as follows:

$$\max \left\{ \theta : \begin{array}{ll} \theta \leq \sum_{j=1}^{n_Y} \left(d_j y_j + \sum_{i=1}^{m_Y} \lambda_{ij} \xi_{ij} (z_{ij} - y_j) \right) & \forall (\boldsymbol{y}, \boldsymbol{z}) \in Z'(\boldsymbol{x}) \\ \lambda_{ij} \leq 0 & i = 1, ..., m_Y, j = 1, ..., n_Y \\ \theta \in \mathbb{R} & \end{array} \right\} \tag{25}$$

Let $(\theta^*, \boldsymbol{\lambda}^*)$ be an optimal solution of (25). By strong duality, the following holds:

$$\theta^* = \min_{(\boldsymbol{y}, \boldsymbol{z}) \in Z(\boldsymbol{x}, \boldsymbol{\xi})} \boldsymbol{d}^\top \boldsymbol{y} \tag{26}$$

Our goal is to derive a family of valid inequalities for (25), depending on points in $Z'(\boldsymbol{x})$ but not on the multipliers, that bound θ^* from above.

To this end, let $(\boldsymbol{y}', \boldsymbol{z}')$ be any point in $Z'(\boldsymbol{x})$, and define a point $(\hat{\boldsymbol{y}}, \hat{\boldsymbol{z}})$ as follows: for all $j \in \{1, ..., n_Y\}$,

- If $\xi_{ij} = 0$ for every $i = 1, ..., m_Y$, set $\hat{y}_j = y'_j$ and $\hat{z}_{ij} = 0$ for all $i = 1, ..., m_Y$.
- Otherwise, there exists at least one index $i \in \{1, ..., m_Y\}$ such that $\xi_{ij} = 1$.

Let $\bar{i}_j \in \arg\min_{i \in \{1, ..., m_Y\}} \{z'_{ij} : \xi_{ij} = 1\}$ and set $\hat{y}_j = z'_{\bar{i}_j j}$ and $\hat{z}_{ij} = \xi_{ij} \hat{y}_j$ for all $i = 1, ..., m_Y$.

Note that, in both cases, we have $\hat{y}_j \leq y'_j$ for all j. This is straightforward in the first case, and is enforced by (13) in the latter as $\hat{y}_j = z'_{\bar{i}_j j}$. In addition, we have $\hat{z}_{ij} \leq z'_{ij}$ for all i and j. Now, since $\underline{h}_{ij} \geq 0$, $(\hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}) \leq (\boldsymbol{y}', \boldsymbol{z}')$, and $\hat{y}_j = \hat{z}_{ij}$ when $\xi_{ij} = 1$, we can conclude that $(\hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}) \in Z(\boldsymbol{x}, \boldsymbol{\xi})$. From (26) it follows that, given any feasible solution to (25) of value θ, we have:

$$\theta \leq \sum_{j=1}^{n_Y} d_j \hat{y}_j = \sum_{j \in N} d_j \hat{y}_j + \sum_{j \in \{1, ..., n_Y\} \setminus N} d_j \hat{y}_j \tag{27}$$

where N denotes the set of indices of variables for which $\xi_{ij} = 0$ for all $i = 1, ..., n_Y$. Observe that, for each $j \in N$, we have $\hat{y}_j = y'_j$, hence

$$\sum_{j \in N} d_j \hat{y}_j = \sum_{j \in N} d_j y'_j = \sum_{j \in N} \left(d_j y'_j + \sum_{i=1}^{m_Y} d_j \xi_{ij} (z'_{ij} - y'_j) \right). \tag{28}$$

As to the remaining variables, we have

$$\sum_{j \in \{1,...,n_Y\} \setminus N} d_j \hat{y}_j = \sum_{j \in \{1,...,n_Y\} \setminus N} d_j z'_{\bar{i}_j j} = \sum_{j \in \{1,...,n_Y\} \setminus N} \left(d_j y'_j + d_j \xi_{\bar{i}_j j} (z'_{\bar{i}_j j} - y'_j) \right),$$

where the last equality holds as $\xi_{\bar{i}_j j} = 1$. Since, for each i and j, $d_j(z'_{ij} - y'_j) \geq 0$, we have,

$$\sum_{j \in \{1,...,n_Y\} \setminus N} d_j \hat{y}_j \leq \sum_{j \in \{1,...,n_Y\} \setminus N} \left(d_j y'_j + d_j \xi_{\bar{i}_j j} (z'_{\bar{i}_j j} - y'_j) + \sum_{i \neq \bar{i}_j} d_j \xi_{ij} (z'_{ij} - y'_j) \right) =$$

$$\sum_{j \in \{1,...,n_Y\} \setminus N} \left(d_j y'_j + \sum_{i=1}^{m_Y} d_j \xi_{ij} (z'_{ij} - y'_j) \right) \tag{29}$$

By combining (27), (28), and (29), we obtain

$$\theta \leq \sum_{j \in N} \left(d_j y'_j + \sum_{i=1}^{m_Y} d_j \xi_{ij} (z'_{ij} - y'_j) \right) + \sum_{j \in \{1,...,n_Y\} \setminus N} \left(d_j y'_j + \sum_{i=1}^{m_Y} d_j \xi_{ij} (z'_{ij} - y'_j) \right) =$$

$$\sum_{j=1}^{n_Y} \left(d_j y'_j + \sum_{i=1}^{m_Y} d_j \xi_{ij} (z'_{ij} - y'_j) \right) \tag{30}$$

Given the arbitrary choice of point $(\boldsymbol{y}', \boldsymbol{z}') \in Z'(\boldsymbol{x})$, the following family of inequalities are valid for (25):

$$\theta \leq \sum_{j=1}^{n_Y} \left(d_j y_j + \sum_{i=1}^{m_Y} d_j \xi_{ij} (z_{ij} - y_j) \right) \quad \forall (\boldsymbol{y}, \boldsymbol{z}) \in Z'(\boldsymbol{x}) \tag{31}$$

We know from Corollary 1 that all negative enough λ_{ij} values are optimal. Any inequality of (25) would be dominated by these cuts when $\lambda_{ij} \leq d_j$, making $\lambda_{ij} = d_j$ a safe choice.

Example 2. We continue the example based on the problem introduced in Example 1 to discuss the interpretation of Lemma 1. As anticipated, a large enough value for λ_1 and λ_2 is -1. The objective function now becomes $-y_1 - y_2 - \xi_1(z_1 - y_1) - \xi_2(z_2 - y_2)$ which, once re-organized, is equivalently written as follows.

$$-y_1(1 - \xi_1) - \xi_1 z_1 - y_2(1 - \xi_2) - \xi_2 z_2 \tag{32}$$

A very clear interpretation is now at hand: when $\xi_1 = 0$, then the contribution of (y_1, z_1) to the objective is $-y_1$, whereas this term reduces to $-z_1$ if $\xi_1 = 1$. In other words, setting $\xi_1 = 1$ and $\xi_2 = 0$ forces $z_1 = y_1$. More in general, for a given $\boldsymbol{\xi} \in \Xi$, setting $\xi_{ij} = 1$ induces a contribution $\bar{h}_{ij} y_j$ in the left-hand-side of the i-th constraint (16).

Lemma 1 achieves the ultimate goal of reformulating problem (8) as an adjustable robust optimization problem with objective uncertainty. We emphasize that this result generalizes the work of Fischetti et al. (2019) on bi-level interdiction games in two directions: (i) it extends their results (in particular, Theorems 2 and 3) to two-stage robust problems; and (ii) it provides a dual interpretation of valid cost penalization.

We conclude this section by reminding that, in case Lemma 1 cannot be applied and one is not capable of deriving tight values of the multipliers by exploiting the structure of the problem, one can resort to Corollary 1, i.e., the fact that a sufficiently small lower bound on the multipliers value allows for a valid reformulation. To this aim, one can use the algorithm discussed in Subramanyam (2022), which makes an initial guess on the smallest (i.e., most negative) value of a multiplier and iteratively multiplies this value by a factor of two until validity can be proved. Note that, for a fixed $\hat{\boldsymbol{\xi}} \in \Xi$, a given $\boldsymbol{\lambda} \leq 0$ is valid if $\sum_{j=1}^{n_Y} \sum_{i=1}^{m_Y} \lambda_{ij} \hat{\xi}_{ij} (z^*_{ij} - y^*_j) = 0$ holds, where $(\boldsymbol{y}^*, \boldsymbol{z}^*)$ denotes an optimal wait-and-see decision in case $\hat{\boldsymbol{\xi}}$ realizes. This directly follows from KKT optimality conditions.

3.2. Relaxation

In the previous section, we have reformulated problem (8) so as to obtain an objective-uncertain ARO. This class of problems has been studied, among others, in Kämmerling and Kurtz (2020), Arslan and Detienne (2022) and Detienne et al. (2021); most of these works only consider the case in which here-and-now variables are binary and the uncertainty set is convex. We now make a step further in the analysis of uncertain adjustable robust problems, and consider the more general case with mixed-integer here-and-now decisions, and in which the uncertainty set is binary. In the following theorem, we adapt to our setting a result introduced in Kämmerling and Kurtz (2020).

THEOREM 2. *Let v^* be the optimal objective value of problem* (8) *and let v^*_R be the optimal objective value of the following problem*

$$\max \theta \tag{33}$$

$$\text{s. t. } \theta \leq \sum_{j=1}^{n_X} c_j \hat{x}_j + \sum_{j=1}^{n_Y} \left(d_j \hat{y}_j + \sum_{i=1}^{m_Y} \underline{\lambda}_{ij} \xi_{ij} (\hat{z}_{ij} - \hat{y}_j) \right) \qquad \forall (\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}) \in W \tag{34}$$

$$\boldsymbol{\xi} \in \Xi, \tag{35}$$

where W denotes the set of extreme points of the convex hull of $\{(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) : \boldsymbol{x} \in X, (\boldsymbol{y}, \boldsymbol{z}) \in Z'(\boldsymbol{x})\}$. Then, $v^* \geq v_R^*$. Moreover, if all active constraints of type (34) *are built upon the same here-and-now decision, say $\bar{\boldsymbol{x}}$, then $v^* = v_R^*$ and $\bar{\boldsymbol{x}}$ solves problem* (8)*.*

We let the reader refer to Kämmerling and Kurtz (2020) for the proof of this result. Instead, we emphasize that the identification of active constraints of type (34), which is straightforward in the convex setting considered in Kämmerling and Kurtz (2020), is not trivial in case the uncertainty set is discrete, as discussed in Section 4.2.

Though the relaxation introduced in Theorem 2 consists in a monolithic Mixed Integer Linear Program (MILP), it contains an exponential number of cuts of type (34). For the sake of simplicity, let us introduce the following function:

$$\Pi(\boldsymbol{x}, \boldsymbol{\xi}, \boldsymbol{y}, \boldsymbol{z}) := \sum_{j=1}^{n_X} c_j x_j + \sum_{j=1}^{n_Y} \left(d_j y_j + \sum_{i=1}^{m_Y} \underline{\lambda}_{ij} \xi_{ij} (z_{ij} - y_j) \right) \tag{36}$$

The procedure described in Algorithm 1 is a cut-generation approach for solving problem (33)-(35). The finite convergence of the obtained algorithm is well-established. Algorithm 1 is initialized with one point $(\boldsymbol{x}^0, \boldsymbol{y}^0, \boldsymbol{z}^0)$ such that $\boldsymbol{x}^0 \in X$ and $(\boldsymbol{y}^0, \boldsymbol{z}^0) \in Z'(\boldsymbol{x}^0)$; such a point always exists by Assumption 1.

Algorithm 1 Cut-generation algorithm

Initialize $\hat{W}$ with some point $(\boldsymbol{x}^0, \boldsymbol{y}^0, \boldsymbol{z}^0)$ with $\boldsymbol{x}^0 \in X$ and $(\boldsymbol{y}^0, \boldsymbol{z}^0) \in Z'(\boldsymbol{x}^0)$.

repeat

$(\theta^*, \boldsymbol{\xi}^*) \leftarrow \text{argmax} \left\{ \theta : (\theta, \boldsymbol{\xi}) \in \mathbb{R} \times \Xi \quad \theta \leq \Pi(\hat{\boldsymbol{x}}, \boldsymbol{\xi}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}) \quad \forall (\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}) \in \hat{W} \right\}$

$(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*) \leftarrow \text{argmin} \{\Pi(\boldsymbol{x}, \boldsymbol{\xi}^*, \boldsymbol{y}, \boldsymbol{z}) : \boldsymbol{x} \in X, (\boldsymbol{y}, \boldsymbol{z}) \in Z'(\boldsymbol{x})\}$

$\hat{W} \leftarrow \hat{W} \cup \{(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)\}$

until $\theta^* \geq \Pi(\boldsymbol{x}^*, \boldsymbol{\xi}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$

stop, and $v_R^* = \theta^*$.

3.3. Solving the separation problem

In this subsection, we discuss the complexity of the separation problem of Algorithm 1. This separation problem is recalled here: for a fixed $\bar{\boldsymbol{\xi}} \in \Xi$, this problem reads:

$$\min \sum_{j=1}^{n_X} c_j x_j + \sum_{j=1}^{n_Y} \left(d_j y_j + \sum_{i=1}^{m_Y} \underline{\lambda}_{ij} \bar{\xi}_{ij} (z_{ij} - y_j) \right) \tag{37}$$

$$\text{s.t. } \boldsymbol{x} \in X, (\boldsymbol{y}, \boldsymbol{z}) \in Z'(\boldsymbol{x}) \tag{38}$$

We make the following remark on solving the separation problem.

Remark 2 (Relation with the deterministic problem). The separation problem can be solved by any oracle designed for solving the deterministic problem (1)-(5) with appropriate input values.

Proof: Reading the proofs of Corollary 1 and Theorem 1 in the reversed order, the following holds.

$$\min_{\boldsymbol{x} \in X, (\boldsymbol{y}, \boldsymbol{z}) \in Z'(\boldsymbol{x})} \Pi(\boldsymbol{x}, \bar{\boldsymbol{\xi}}, \boldsymbol{y}, \boldsymbol{z}) \tag{39}$$

$$= \min_{\boldsymbol{x} \in X} \left\{ \boldsymbol{c}^\top \boldsymbol{x} + \min_{(\boldsymbol{y}, \boldsymbol{z}) \in Z'(\boldsymbol{x})} \boldsymbol{d}^\top \boldsymbol{y} + \sum_{j=1}^{n_Y} \sum_{i=1}^{m_Y} \underline{\lambda}_{ij} \bar{\xi}_{ij} (z_{ij} - y_j) \right\} \tag{40}$$

$$= \min_{\boldsymbol{x} \in X} \left\{ \boldsymbol{c}^\top \boldsymbol{x} + \min_{(\boldsymbol{y}, \boldsymbol{z}) \in \text{conv}(Z'(\boldsymbol{x}))} \boldsymbol{d}^\top \boldsymbol{y} + \sum_{j=1}^{n_Y} \sum_{i=1}^{m_Y} \underline{\lambda}_{ij} \bar{\xi}_{ij} (z_{ij} - y_j) \right\} \tag{41}$$

$$= \min_{\boldsymbol{x} \in X} \left\{ \boldsymbol{c}^\top \boldsymbol{x} + \max_{\boldsymbol{\lambda} \leq \boldsymbol{0}} \min_{(\boldsymbol{y}, \boldsymbol{z}) \in \text{conv}(Z'(\boldsymbol{x}))} \boldsymbol{d}^\top \boldsymbol{y} + \sum_{j=1}^{n_Y} \sum_{i=1}^{m_Y} \lambda_{ij} ((1 - \bar{\xi}_{ij}) u_j + z_{ij} - y_j) \right\} \tag{42}$$

$$= \min_{\boldsymbol{x} \in X} \left\{ \boldsymbol{c}^\top \boldsymbol{x} + \min_{(\boldsymbol{y}, \boldsymbol{z}) \in Z(\boldsymbol{x}, \bar{\boldsymbol{\xi}})} \boldsymbol{d}^\top \boldsymbol{y} \right\} \tag{43}$$

$$= \min_{\boldsymbol{x} \in X, \boldsymbol{y} \in Y(\boldsymbol{x}, \bar{\boldsymbol{\xi}})} \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}^\top \boldsymbol{y} \tag{44}$$

Now, given an optimal solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ of the deterministic problem arising when $\boldsymbol{\xi} = \bar{\boldsymbol{\xi}}$, an optimal solution to the separation problem is given by $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{z}^*)$, where $\boldsymbol{z}$ is defined as $z^*_{ij} = \bar{\xi}_{ij} y^*_j$.

Remark 3. The separation problem and the deterministic problem (1)-(5) belong to the same complexity class.

Proof: We already have that any instance of the separation problem can be solved using an oracle for the deterministic problem. We also show that any instance of the deterministic

problem can be solved by any oracle solving the separation problem. Indeed, using $\bar{\boldsymbol{\xi}} = \mathbf{0}$, problem (37)-(38) reduces to the deterministic problem since variable $\boldsymbol{z}$ becomes useless and can be removed from the model. Indeed, $\boldsymbol{z}$ comes in $\leq$-inequality constraints with positive coefficients and have no contribution to the objective function. Thus, solutions such that $\boldsymbol{z} > \mathbf{0}$ are dominated by those solutions with $\boldsymbol{z} = \mathbf{0}$.

3.4. Dealing with infeasibility

In this section we discuss the behaviour of our approach when (i) problem (8) is feasible but the complete recourse assumption is not satisfied; and (ii) feasibility of problem (8) is unknown.

In case (i), the proposed relaxation is valid, provided that at least one here-and-now decision with a feasible recourse decision exists, for any given scenario. This is ensured by feasibility of problem (8). Since the separation procedure *first* decides the scenario and *then* computes the corresponding here-and-now and recourse decisions, it cannot happen a here-and-now decision without recourse decision be selected.

As to case (ii), the only feasibility check which has to be added at each iteration of Algorithm 1 is that, given the current scenario $\boldsymbol{\xi}^* \in \Xi$ returned by the maximization, there indeed exists one $\boldsymbol{x} \in X$ such that $Y(\boldsymbol{x}, \boldsymbol{\xi}^*) \neq \emptyset$. This condition is met if and only if the following optimization problem (which is feasible as the deterministic problem was assumed feasible) has an optimal objective value of zero:

$$\min \sum_{i=1}^{m_Y} \sum_{j=1}^{n_Y} \underline{\lambda}_{ij} \xi^*_{ij} (z_{ij} - y_j) \tag{45}$$

$$\text{s. t. } \boldsymbol{x} \in X, (\boldsymbol{y}, \boldsymbol{z}) \in Z'(\boldsymbol{x}). \tag{46}$$

If this is not the case, we exhibit a scenario $\boldsymbol{\xi}^*$ for which $Y(\boldsymbol{x}, \boldsymbol{\xi}^*) = \emptyset$ for each $\boldsymbol{x} \in X$, and hence Algorithm 1 is stopped.

3.5. Relation to Subramanyam (2022)

An alternative approach to the class of problems addressed in this paper where uncertainty is moved from the constraints to the objective function appears in Subramanyam (2022). As already mentioned, this approach explicitly considers problems in which uncertainty affects the right-hand side coefficients, each described by means of a binary uncertainty variable. Then, each variable ξ_i is replaced by a deterministic variable v_i, and constraint

$\|v-\xi\|_2^2=0$ is relaxed in a Lagrangian way by means of a single multiplier. This would lead to a problem where uncertainty affects the objective function only.

However, the method proposed in Subramanyam (2022) can handle problems in which uncertainty affects the constraint matrix as well. In this case, each uncertainty coefficient of the constraint matrix is modelled by introducing uncertainty variable ξ_{ij}, thus originating bilinear terms. These product terms can be linearized by using McCormick inequalities (provided that variables are bounded), thus resorting to a problem in which uncertainty variables appear in the right-hand side only and the previous scheme can be used.

We now show that they are not equivalent because, for a problem in the general form (8), they result in different reformulations. Consider the second-stage optimization problem for a fixed $\hat{\boldsymbol{x}} \in X$ and $\hat{\boldsymbol{\xi}} \in \Xi$ after having introduced variables z_{ij} to linearize every products $\xi_{ij}y_j$. To lighten our notation, let us assume that $u_j = 1$ $(j = 1, ..., n_Y)$. We have that $\boldsymbol{y} \in Y(\hat{\boldsymbol{x}}, \hat{\boldsymbol{\xi}})$ if, and only if, $\boldsymbol{y} \in Y$ and there exists $\boldsymbol{z}$ such that

$$\sum_{j=1}^{n_Y} \underline{h}_{ij} y_j + (\bar{h}_{ij} - \underline{h}_{ij}) z_{ij} \leq f_i - \sum_{j=1}^{n_X} t_{ij} \hat{x}_j \qquad i = 1, ..., m_Y, \tag{47}$$

$$z_{ij} - y_j \leq 0 \qquad i = 1, ..., m_Y, j = 1, ..., n_Y, \tag{48}$$

$$z_{ij} \leq \hat{\xi}_{ij} \qquad i = 1, ..., m_Y, j = 1, ..., n_Y, \tag{49}$$

$$y_j - z_{ij} \leq (1 - \hat{\xi}_{ij}) \qquad i = 1, ..., m_Y, j = 1, ..., n_Y, \tag{50}$$

$$z_{ij} \geq 0 \qquad i = 1, ..., m_Y, j = 1, ..., n_Y. \tag{51}$$

A direct application of the reformulation introduced in Subramanyam (2022) leads to the introduction of variables v_{ij} replacing each occurrence of $\hat{\xi}_{ij}$. Constraints $\xi_{ij} = v_{ij}$ are then summarized as $\|v-\xi\|_2^2 = 0$, and this requirement is then relaxed in a Lagrangian way with penalty coefficient $\lambda \leq 0$. The resulting formulation, where the uncertain parameters $\hat{\xi}_{ij}$ only appear in the objective function, reads as follows

$$\min \sum_{j=1}^{n_Y} d_j y_j + \lambda \sum_{i=1}^{m_Y} \sum_{j=1}^{n_Y} (2 v_{ij} \hat{\xi}_{ij} - v_{ij} - \hat{\xi}_{ij}) \tag{52}$$

$$\text{s.t. } (47), (48), (51), \tag{53}$$

$$\boldsymbol{y} \in Y, \tag{54}$$

$$z_{ij} \leq v_{ij} \qquad i = 1, ..., m_Y, j = 1, ..., n_Y, \tag{55}$$

$$y_j - z_{ij} \leq (1 - v_{ij}) \qquad i = 1, ..., m_Y, j = 1, ..., n_Y, \tag{56}$$

$$0 \leq v_{ij} \leq 1 \qquad i = 1, ..., m_Y, j = 1, ..., n_Y. \tag{57}$$

This scheme can be viewed as a surrogate approach of the augmented Lagrangian method (see, Feizollahi et al. (2016)), as the augmented penalization $\|\boldsymbol{v} - \hat{\boldsymbol{\xi}}\|^2$ is always smaller than $\boldsymbol{e}^\top \boldsymbol{v} + \boldsymbol{e}^\top \hat{\boldsymbol{\xi}} - 2\boldsymbol{v}^\top \hat{\boldsymbol{\xi}}$. Conversely, our approach relies on a convex reformulation of the wait-and-see problem by polyhedral arguments and standard LP duality. Our approach introduces instead one multiplier λ_{ij} for each constraint $y_j - z_{ij} \leq (1 - \hat{\xi}_{ij})$. By Corollary 1, our method can easily be reduced to a single-parameter method by considering a unique multiplier $\underline{\lambda} := \min_{i,j} \underline{\lambda}_{ij}$, provided that $\underline{\lambda}_{ij}$ are valid.

We now turn our attention to comparing the Lagrangian penalization in the objective function of the two reformulations. For simplicity, consider the case that $\boldsymbol{d} \leq \boldsymbol{0}$. By Theorem 5 (point 2) in Subramanyam (2022) a multiplier $\lambda = \sum_{j=1}^{n_Y} d_j$ is sufficient. In our approach, one can take $\underline{\lambda}_{ij} = d_j$ (see Lemma 1). To simplify the comparison, we will use a single multiplier $\underline{\lambda} := \min_{i,j} \underline{\lambda}_{ij}$. Clearly, we have $\lambda \leq \underline{\lambda}$, i.e., we are making our penalization term intentionally larger, still preserving its validity. Now, for each i and j we have that

- if $\hat{\xi}_{ij} = 0$, then $\lambda(2v_{ij}\hat{\xi}_{ij} - v_{ij} - \hat{\xi}_{ij}) = -\lambda v_{ij} \geq 0 = \underline{\lambda}_{ij}\hat{\xi}_{ij}(z_{ij} - y_j)$;
- if $\hat{\xi}_{ij} = 1$, then $\lambda(2v_{ij}\hat{\xi}_{ij} - v_{ij} - \hat{\xi}_{ij}) = \lambda(v_{ij} - 1) \geq 0 \geq \underline{\lambda}(z_{ij} - y_j)$, where the last two inequalities derive from $v_{ij} \leq 1$ and from equation (50), respectively.

Thus, the penalization term used in Subramanyam (2022) is always larger than the one employed in our approach, showing that the two reformulations are different and they can only be directly compared from a computational viewpoint. In Section 5.1 we perform such a comparison on a class of problems that can be naturally cast in the setting considered in Subramanyam (2022).

Finally, we observe that the way feasibility issues are handled in Subramanyam (2022) is substantially different from what we describe in Section 3.4. Indeed, in model (45)–(46) we optimize over the set of feasible first- and second-stage decisions, while only second-stage decisions are considered in the aforementioned work. Second, our approach is aimed at proving infeasibility of the whole original problem, possibly halting the algorithm, while the one in Subramanyam (2022) is used to prove infeasibility of a single first-stage decision. If such a case is detected, a so-called *feasibility cut* is added to the master problem and the algorithm continues.

4. A Branch-and-bound algorithm

In this section, we generalize the branch-and-bound scheme introduced in Kämmerling and Kurtz (2020) to the case in which the here-and-now decisions are mixed-integer and the

uncertainty set is binary. This requires additional effort for dealing with the discrete nature of the uncertainty set. In addition, we study the convergence of the resulting algorithm, a task which requires non-trivial arguments due to the presence of continuous variables. For the sake of simplicity, we present the algorithm for the case in which integer here-and-now variables are all binary, the extension to the general case being straightforward.

4.1. Statement of the procedure

For a given node q with local bounds noted $\boldsymbol{l}^q$ and $\boldsymbol{u}^q$, let $W^q := \{(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) : \boldsymbol{x} \in X \cap [\boldsymbol{l}^q,\boldsymbol{u}^q], (\boldsymbol{y},\boldsymbol{z}) \in Z'(\boldsymbol{x})\}$ and let v_R^q denote the optimal objective value of (33)-(35) where W has been replaced by W^q. Additionally, we let H^q denote the set of active constraints of type (34) for the resulting problem and $\boldsymbol{x}^{[h]}$ denote the here-and-now decision which was used to generate the h-th cut of type (34), for $h \in H^q$. Finally, let us introduce $\bar{\boldsymbol{x}}^q$ defined as follows:

$$\bar{\boldsymbol{x}}^q = \frac{1}{|H^q|} \sum_{h \in H^q} \boldsymbol{x}^{[h]}. \tag{58}$$

The algorithm initially solves relaxation (33)–(35) and computes $\bar{\boldsymbol{x}}^0$. If $\bar{\boldsymbol{x}}^0 \in \{\boldsymbol{l}^0,\boldsymbol{u}^0\}$, the problem has been solved to optimality and the algorithm stops. Otherwise, a branching is executed (see below) and two descendant nodes are generated. For each descendant node q, we solve the associated relaxation and compute $\bar{\boldsymbol{x}}^q$. If $\bar{\boldsymbol{x}}^q \in \{\boldsymbol{l}^q,\boldsymbol{u}^q\}$, the node is solved, and possibly the incumbent solution is updated. Otherwise, problem q is added to a list L storing nodes associated with active problems. At each iteration, the algorithm removes from L all nodes whose associated lower bound is larger or equal to the incumbent solution value, and selects from L the node, say q, with the smallest value of lower bound. The branching variable is selected according to solution $\bar{\boldsymbol{x}}^q$, giving priority to binary variables: when a binary variable j is selected, its bounds are fixed to 0 and 1. In case a continuous variable is chosen, we update its upper bound to $\bar{x}_j^q$ in the left descendant node and its lower bound to $\bar{x}_j^q$ in the right descendant node. The algorithm stops when L is empty.

In order to compute feasible solutions during the execution of the branch-and-bound algorithm, we can exploit the following result.

PROPOSITION 1. *Let q be a given node and let us assume that $\bar{x}_j^q \in \{0,1\}$ for all j for which x_j is required to be binary. Then, a feasible solution for* (8) *can be computed by solving formulation* (33)-(35) *where W has been replaced by* $\text{vertex}(\text{conv}(\{\bar{\boldsymbol{x}}^q\} \times Z'(\bar{\boldsymbol{x}}^q)))$.

Proof: The proof is similar to that of Theorem 2.

For every node q, when appropriate, we will denote by v_U^q the objective value obtained applying Proposition 1. Notice that, even when some binary here-and-now variables are fractional, one could still try to round fractional values to the closest integer and check if the resulting solution is feasible, in which case an upper bound for problem (8) is obtained.

4.2. Identifying active cuts

At each node of the branch-and-bound tree, a (possibly infeasible) here-and-now decision is reconstructed from the solved relaxation by means of formula (58). This formula computes the "average" here-and-now decision among all active cuts, gathered in H^q.

The branch-and-bound algorithm is based on computing equation (58) at each branching node. When the uncertainty set Ξ is a convex set, identifying the set of active constraints can be easily done by checking the slack variables of each cut. We now discuss a possible way of performing this task when the uncertainty set is discrete. Though one could replace H^q in (58) by the set of all *generated* constraints, this would lead to poor branching decisions and low-quality heuristic solutions, resulting in a large branch-and-bound tree. For this reason, we instead identify set H^q by searching for an *Irreducible Infeasible Subsystem* (IIS) of the following infeasible model.

$$\max 0 \tag{59}$$

$$\text{s. t. } \theta \leq \sum_{j=1}^{n_X} c_j \hat{x}_j + \sum_{j=1}^{n_Y} \left(d_j \hat{y}_j + \sum_{i=1}^{m_Y} \lambda_{ij} \xi_{ij} (\hat{z}_{ij} - \hat{y}_j) \right) \qquad \forall (\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}) \in \hat{H}^q \tag{60}$$

$$\theta \geq v_R^q + \varepsilon \tag{61}$$

$$\boldsymbol{\xi} \in \Xi, \tag{62}$$

where $\varepsilon > 0$ is a positive input tolerance and v_R^q denotes the optimal value of the relaxation at the node. Depending on the selected tolerance ε, one can identify a subset H^q of constraints preventing θ from being strictly larger than v_R^q, i.e., being active by definition.

4.3. Convergence result

The enumerative scheme presented in the previous section solves problem (8) by iteratively partitioning the feasible set, possibly performing branching on a continuous variable. Clearly, the algorithm introduced in Kämmerling and Kurtz (2020) finitely converges as $X \subseteq \{0,1\}^{n_X}$. We now show that, in the mixed-integer case, the algorithm either finitely

converges or converges towards an optimal solution of (8) although, potentially, an infinite number of steps is required.

THEOREM 3. *Let v^* denote the optimal solution value of problem* (8)*. Then, the branch-and-bound algorithm either finitely terminates with an incumbent having value v^* or enters an infinite sequence of nodes, say $p \in P$, for which $\{\bar{\boldsymbol{x}}^p\}_{p\in P} \to \boldsymbol{x}^*$, where $\boldsymbol{x}^*$ is an optimal solution of the problem.*

Proof: We focus on the case in which the algorithm enters an infinite sequence of nodes. Since our branching strategy prioritizes binary variables over continuous ones, we may assume that all variables of the former type are fixed in the sequence. Moreover, an infinite sequence of nodes implies the existence of an infinite subsequence of nodes belonging to the same branch; we denote this subsequence by P and its generic node by p. We will denote by $[\boldsymbol{l}^p, \boldsymbol{u}^p]$ the local bounds of node p, by $(\theta^{p*}, \boldsymbol{\xi}^{p*})$ the optimal solution of problem (33)-(35), and by $\bar{\boldsymbol{x}}^{p*}$ the here-and-now decision computed according to (58).
Since branching always reduces the domain of the here-and-now variables, we have $[\boldsymbol{l}^{p+1}, \boldsymbol{u}^{p+1}] \subseteq [\boldsymbol{l}^p, \boldsymbol{u}^p]$. Thus, there exists a subsequence of P for which $[\boldsymbol{l}^p, \boldsymbol{u}^p]$ converges to an interval $[\boldsymbol{l}^*, \boldsymbol{u}^*]$. In addition, there exists a subsequence for which $\boldsymbol{\xi}^{p*}$ converges to a point $\boldsymbol{\xi}^*$, as $\boldsymbol{\xi}$ variables are binary and the number of their feasbile combinations is finite.

Since our node selection strategy always picks the node with lowest bound, we have that θ^{p*} is a lower bound on v^*. In addition, branching operation implies that $\theta^{(p+1)*} \geq \theta^{p*}$, and thus θ^{p*} converges to a value θ^*. Finally, $\bar{\boldsymbol{x}}^{p*} \in [\boldsymbol{l}^p, \boldsymbol{u}^p] \subseteq [\boldsymbol{l}^0, \boldsymbol{u}^0]$, hence there exists a subsequence for which $\bar{\boldsymbol{x}}^{p*}$ converges to a solution $\bar{\boldsymbol{x}}^*$ that belongs to $[\boldsymbol{l}^*, \boldsymbol{u}^*]$; otherwise there would exist a node p for which $\bar{\boldsymbol{x}}^{p*} \notin [\boldsymbol{l}^p, \boldsymbol{u}^p]$. In addition, $\bar{\boldsymbol{x}}^* \in X$ as X is a compact set.

We now show that $\bar{\boldsymbol{x}}^* \in \text{vert}([\boldsymbol{l}^*, \boldsymbol{u}^*])$. Indeed, let $j \in \{1, ..., n_X\}$ be a here-and-now variable index which is infinitely branched on, and consider the subsequence $P' \subseteq P$ such that, for all $p \in P'$, $u_j^{p+1} = x_j^{p*}$. If P' is infinite, then $\{u_j^{p*}\}_{p\in P'} \to x_j^*$ since $\{x_j^{p*}\}_{p\in P'} \to x_j^*$ as $P' \subseteq P$. If instead P' is finite, then the subsequence $P'' \subseteq P$ such that, for all $p \in P''$, $l_j^{p+1} = x_j^{p*}$ is infinite and, for the same reason, we have $\{l_j^{p*}\}_{p\in P''} \to x_j^*$.

Thus, we have that $\bar{\boldsymbol{x}}^* \in \text{vert}([\boldsymbol{l}, \boldsymbol{u}])$ and, by Theorem 2, its cost is θ^*. In addition, $\bar{\boldsymbol{x}}^*$ is feasible for (8) and thus $\theta^* \geq v^*$. Since $\theta^{p*} \leq v^*$ for all $p \in P$, this is true for θ^* as well, and hence θ^* coincides with the optimal solution value v^*.

5. Applications and Computational experiments

In this section, we report extensive computational results performed on two different applications. [1] Our implementation was done in C++17 using GuRoBi version 10 to solve every sub-problem. All the experiments were run on an AMD 3960 running at 3.8 GHz. with a time limit equal to 3,600 CPU seconds per run.

5.1. Facility Location Problem

In our first application, we consider a Facility Location Problem (FLP) with set of facilities V_1 and set of customers V_2. For each facility $u \in V_1$, we denote by f_u its opening cost and by q_u its capacity. For each customer $v \in V_2$, we let d_v be her demand and p_v be the profit earned if one decides to serve the customer. In this case, all the associated demand must be provided by a unique facility without splitting across multiple facilities. For each connection $(u, v) \in V_1 \times V_2$, we denote by t_{uv} the transportation cost for delivering d_v units of goods from u to v. The objective is to maximize the overall profit, deducted of opening and transportation costs.

In an uncertain context, we consider an adjustable robust decision process in which here-and-now decisions concern the facilities to be opened. Uncertainty materializes into random disruptions of the opened facilities, possibly resulting in an actual capacity $q_u \leq \bar{q}_u$, where $\bar{q}_u$ denotes the nominal capacity value. More precisely, let $R \in \mathbb{N}\backslash\{0\}$ denote the number of possible realizations of the capacity of each facility u that differ from the nominal value. In our model, for a given R we let $q_u \in \{\bar{q}_u(1 - \frac{r}{R}), r = 0, \ldots, R\}$. In other words, when $R = 1$ we consider the case of full disruptions only, whereas $R \geq 2$ allows for partial disruptions.

We introduce a binary variable ξ_{ur} taking value 1 whenever facility u has actual capacity $\bar{q}_u(1 - \frac{r}{R})$. By following the traditional Γ-uncertainty approach, we control the level of uncertainty by means of a parameter $\Gamma \in \mathbb{N}$ and accordingly define the uncertainty set as

$$\Xi = \left\{ \boldsymbol{\xi} \in \{0,1\}^{|V_1| \times (R+1)} : \sum_{u \in V_1} \sum_{r=0}^{R} r\, \xi_{ur} \leq \Gamma,\ \sum_{r=0}^{R} \xi_{ur} = 1 \,\forall u \in V_1 \right\} \tag{63}$$

For a given scenario $\boldsymbol{\xi} \in \Xi$ the actual capacity of each facility $u \in V_1$ is thus defined as $q_u(\boldsymbol{\xi}) = \sum_{r=0}^{R} \bar{q}_u(1 - \frac{r}{R})\xi_{ur}$.

[1] All instances considered in this paper are publicly available at `https://github.com/hlefebvr/AC_AdjustableRobustOptimizationWithDiscreteUncertainty`.

Our adjustable robust model is given as

$$\min_{\boldsymbol{x}\in\{0,1\}^{|V_1|}} \left\{ \sum_{u\in V_1} f_u x_u + \max_{\boldsymbol{\xi}\in\Xi} \min_{\boldsymbol{y}\in Y(\boldsymbol{x},\boldsymbol{\xi})} \sum_{u\in V_1}\sum_{v\in V_2} (t_{uv}-p_v) y_{uv} \right\} \tag{64}$$

where, given $\boldsymbol{x}\in\{0,1\}^{|V_1|}$ indicating the facilities to be opened and $\boldsymbol{\xi}\in\Xi$, the wait-and-see variables $y\in\{0,1\}^{|V_1|\times|V_2|}$ assign facilities to customers and have feasible space $Y(\boldsymbol{x},\boldsymbol{\xi})$ defined as

$$Y(\boldsymbol{x},\boldsymbol{\xi}) = \left\{ \boldsymbol{y}\in\{0,1\}^{|V_1|\times|V_2|} : \begin{array}{ll} \displaystyle\sum_{u\in V_1} y_{uv} \le 1 & \forall v\in V_2 \\ \displaystyle\sum_{v\in V_2} d_v y_{uv} \le q_u(\boldsymbol{\xi}) x_u & \forall u\in V_1 \end{array} \right\}. \tag{65}$$

Here, the first set of constraints imposes that each customer is served at most once, while the second states that the amount of goods leaving a facility does not exceed its actual capacity.

5.1.1. Instance generation We generated facility location instances according to Cornuéjols et al. (1991), and considered the following sizes $(|V_1|,|V_2|)$: (10, 20), (10, 30), (10, 40), (10, 50), (15, 20), (15, 30), (15, 40) and (15, 50). For each facility $u\in V_1$, the nominal capacity $\bar{q}_u$ was uniformly generated between 10 and 160, while the opening cost was computed as $f_u = \alpha_u + \beta_u\sqrt{\bar{q}_u}$ where α_u and β_u were generated between 0 and 90 and 100 and 110, respectively. Demands were randomly generated in $[0,1]$ and scaled so that $\sum_{u\in V_1}\bar{q}_u/\sum_{v\in V_2} d_v = \mu$ where μ is a parameter taking value 2 or 3. The candidate positions for opening facilities and the location of customers were randomly generated in the unitary square. Then, for each pair $(u,v)\in V_1\times V_2$, the transportation cost t_{uv} was defined as the associated Euclidean distance multiplied by $10\times d_v$. Finally, the profit of each customer was set equal to $p_v = 4\times \text{median}\{t_{uv} : u\in V_1\}$. For each combination of $(|V_1|,|V_2|)$ and μ, we generated 10 instances.

5.1.2. Full disruption In this section, we consider the case where unpredictable events may fully disrupt the facilities. In other words, we consider that $R=1$, i.e., the actual capacity for each facility u is either $\bar{q}_u$ (for $\xi_{u0}=1$) or 0 (for $\xi_{u1}=1$). In this setting, the wait-and-see feasible set $Y(\boldsymbol{x},\boldsymbol{\xi})$ includes those $\boldsymbol{y}\in\{0,1\}^{|V_1|\times|V_2|}$ fulfilling conditions

$$\sum_{u\in V_1} y_{uv} \le 1 \qquad \forall v\in V_2 \tag{66}$$

$$\sum_{v\in V_2} d_v y_{uv} \le \bar{q}_u x_u \qquad \forall u\in V_1 \tag{67}$$

$$y_{uv} \le \xi_{u0} \qquad \forall (u,v)\in V_1\times V_2. \tag{68}$$

Given in this form, our model directly fulfills the assumptions made in Subramanyam (2022), i.e., right-hand side uncertainty. This allowed us to directly run the publicly available code of the column-and-constraint algorithm proposed in Subramanyam (2022) with no need of introducing further reformulations, which would make the comparison unclear. We denote this algorithm as CCG.

Table 1 gives the results of our Branch-and-Cut (B&C) algorithm for $\Gamma = 2, 3$ and 4. The leftmost part of the table refers to instances with $\mu = 2$, whereas the rightmost columns refer to $\mu = 3$. We report the number of instances solved to proven optimality (out of 10), the average computing time, the average number of nodes, and the average percentage optimality gap at the end of the computation. All averages are computed with respect to instances solved to optimality only, but the optimality gap which is computed over all instances. In addition, we report the number of solved instances, the average computing time and average percentage optimality gap for algorithm CCG (we report *na* when the optimality gap is not available for some instance). In rows where an algorithm can solve a larger number of instances, we report this figure in boldface.

Algorithm B&C can solve 80% of the instances with $\mu = 2$ and almost 95% of those with $\mu = 3$. Optimality gaps are typically small but for few cases, these correspond to runs in which the algorithm fails in finding a high quality feasible solution. The number of branch-and-bound nodes is always reasonable, meaning that a manageable enumeration allows to converge to a proven optimal solution. Not surprisingly, larger instances require a larger computational effort; we also observe that the value of Γ has an impact on the effort required by our algorithm.

The comparison with algorithm CCG is also depicted in Figure 1, which shows, for $\mu = 2$ and $\mu = 3$, separately, the performance profiles of the two algorithms in terms of fraction of instances solved within a certain ratio of computing times with respect to the winning algorithm. While for $\mu = 2$ algorithm B&C is slower but solves a slightly larger number of instances when more computing time is allowed, when $\mu = 3$, the better performance of algorithm B&C clearly emerges: indeed, the number instances solved to optimality is around 20% larger, and very soon the curve associated with B&C dominates the CCG one.

			$\mu=2$							$\mu=3$						
			B&C				CCG			B&C				CCG		
Γ	$\|V1\|$	$\|V2\|$	opt	time	%gap	nodes	opt	time	%gap	opt	time	%gap	nodes	opt	time	%gap
2	10	20	10	13.61	0.00	19	10	4.98	0.00	10	13.42	0.00	47	10	12.83	0.00
		30	10	64.50	0.00	36	10	106.81	0.00	**10**	18.50	0.00	32	9	251.17	1.98
		40	**10**	254.90	0.00	27	8	40.25	7.95	10	15.92	0.00	30	10	70.54	0.00
		50	10	1058.59	0.00	30	10	106.21	0.00	**10**	158.93	0.00	47	7	94.25	19.77
	15	20	**10**	122.09	0.00	63	9	134.87	12.46	**10**	69.06	0.00	72	9	110.83	2.42
		30	**10**	742.82	0.00	60	6	553.81	5.19	**10**	102.49	0.00	33	7	406.04	9.19
		40	6	1509.53	1.93	82	**7**	92.54	4.96	**9**	324.90	0.07	87	5	142.36	13.47
		50	5	1882.22	2.70	50	**6**	560.80	5.93	**10**	250.11	0.00	45	9	445.83	4.72
3	10	20	10	22.74	0.00	31	10	8.46	0.00	**10**	29.79	0.00	99	9	37.91	*na*
		30	10	75.81	0.00	38	10	37.17	0.00	10	104.54	0.00	80	10	49.70	0.00
		40	10	213.18	0.00	36	10	95.80	0.00	10	45.13	0.00	65	10	36.49	0.00
		50	8	658.77	27.12	34	8	451.47	2.04	**10**	230.14	0.00	66	7	254.23	62.61
	15	20	**10**	218.62	0.00	88	7	69.42	32.40	**10**	126.99	0.00	89	8	109.51	5.53
		30	**10**	1320.35	0.00	55	4	177.90	19.18	**10**	359.67	0.00	60	6	264.51	16.85
		40	4	1353.41	5.55	58	**7**	468.41	2.57	**9**	1093.62	0.85	100	2	190.89	23.41
		50	1	3572.38	8.89	51	**3**	856.32	39.34	**10**	908.34	0.00	65	6	768.34	13.62
4	10	20	10	34.25	0.00	84	10	3.38	0.00	10	41.74	0.00	227	10	7.63	0.00
		30	10	121.93	0.00	162	10	37.88	0.00	10	185.98	0.00	380	10	31.53	0.00
		40	10	360.44	0.00	165	10	198.41	0.00	10	56.94	0.00	110	10	18.36	0.00
		50	8	1000.25	7.72	107	**9**	306.56	4.66	**10**	607.93	0.00	117	8	610.69	2.16
	15	20	**10**	452.02	0.00	496	6	246.01	*na*	**10**	376.20	0.00	675	9	576.29	2.47
		30	**7**	806.49	13.49	72	3	147.91	*na*	**10**	1195.95	0.00	207	5	182.24	53.27
		40	3	1210.74	7.00	78	**4**	1280.90	13.95	**5**	1875.86	11.97	113	2	562.71	30.50
		50	0	-	39.03	-	**2**	1241.74	108.12	**4**	1190.49	21.23	77	2	704.55	41.09

Table 1 Computational results on FLP instances with full disruption.

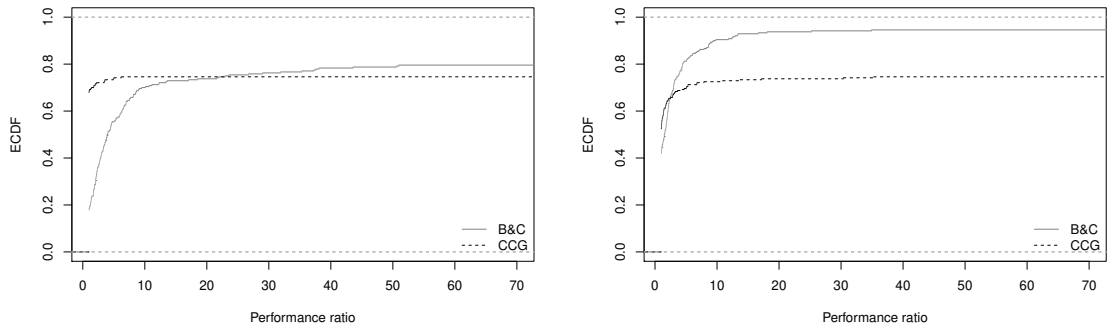

Figure 1 Performance profile for algorithms B&C and CCG. Parameter $\mu=2$ (left) and $\mu=3$ (right).

5.1.3. Partial disruption In this section, we consider the case where facilities may be partially disrupted, which happens for $R > 1$. To evaluate the effect of an increased number of outcomes for each uncertain coefficient, we tested algorithm B&C on instances with $|V_1| = 10$, $\mu = 3$ and $\Gamma = 2$ by considering $R = 1, 2, 3$ and 4. For each group of instances and each considered value of R, we computed the median solution time for instances with $|V_2| = 20, 30, 40$ and 50, separately, and report it as a line in Figure 2. The figure shows that instances with $|V_2| \leq 40$ are still effectively solved when R increases, as the median computing time does not exceed 500 seconds. For $|V_2| = 50$ the median computing time grows significantly, though the algorithm is still capable of solving 33 out of 40 instances.

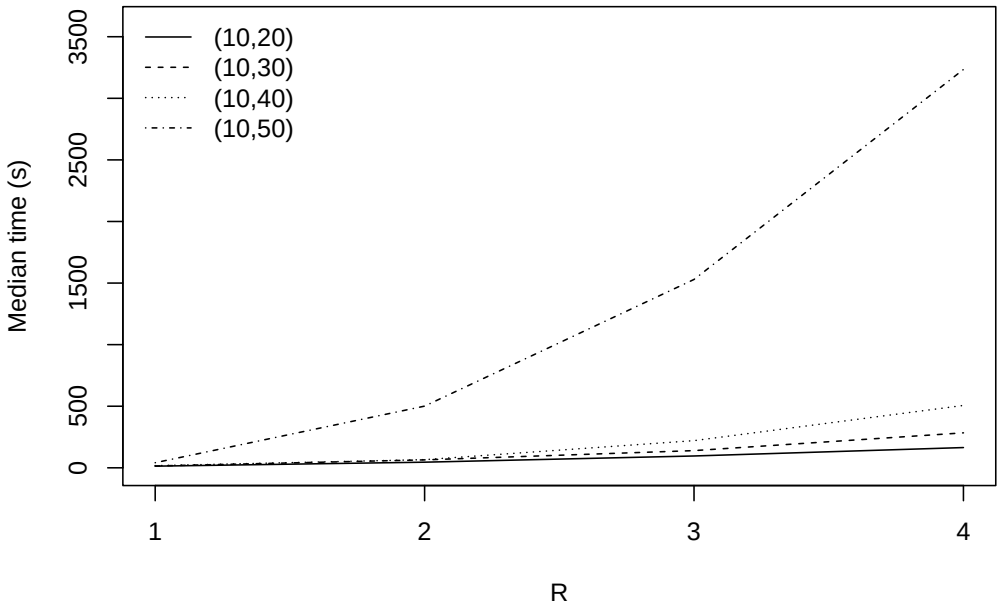

Figure 2 **Median time of algorithm B&C for instances with $|V_1| = 10$ and $\mu = 3$ for $\Gamma = 2$ and increasing values of R.**

5.2. Multiple Knapsack Problem

In the second set of experiments we consider a problem in which uncertainty directly affects the coefficients of the constraint matrix, namely, a variant of the Multiple Knapsack Problem (MKP). In the deterministic MKP, given a collection of n items with associated weights w_j and profits p_j, one should decide a maximum-profit subset of items to be packed into a set of K identical knapsacks of fixed capacity W. We consider here an adjustable robust counterpart of the MKP where here-and-now decisions dispatch each item to one knapsack. At a second stage, the exact weight of each item is revealed and, for each knapsack, a subset of items having maximum profit is determined while respecting the capacity constraint. More in details, the actual weight of each item j is either its nominal value $\underline{w}_j$ or an increased value $\underline{w}_j + \tilde{w}_j$, with $\tilde{w}_j > 0$. Although the here-and-now decisions do not have explicit costs in the objective function, not all here-and-now decisions are equivalent for the second stage. For example, while assigning all items to one knapsack is a feasible here-and-now decision, this policy may be sub-optimal with respect to the wait-and-see problem which, therefore, would optimize over a single knapsack only.

In the first stage, each item has to be assigned to exactly one knapsack. The here-and-now feasible space X is therefore modeled by means of binary variables x_{jk}, defined for each item $j \in \{1, ..., n\}$ and knapsack $k \in \{1, ..., K\}$. Each variable takes value 1 if and only if item j is dispatched to knapsack k. More formally, X is defined as follows

$$X = \left\{ \boldsymbol{x} \in \{0,1\}^{n \times K} : \sum_{k=1}^{K} x_{jk} = 1 \, \forall j = 1, \ldots, n \right\}. \tag{69}$$

As to uncertainty, we assume that up to Γ items take their largest weight $\bar{w}_j$, i.e., the uncertainty set is

$$\Xi = \left\{ \boldsymbol{\xi} \in \{0,1\}^n : \sum_{j=1}^{n} \xi_j \leq \Gamma \right\} \tag{70}$$

Accordingly, given $\boldsymbol{\xi} \in \Xi$, the actual weight of each item j is $\underline{w}_j + \xi_j \tilde{w}_j$.

Once uncertainty reveals, the decision maker selects, for each knapsack, the subset of items to be packed. Only items dispatched to that knapsack are available for packing, and

capacity constraint has to be satisfied with respect to actual weights. This leads to the following definition of the wait-and-see feasible space

$$Y(\boldsymbol{x},\boldsymbol{\xi}) = \left\{ \boldsymbol{y} \in \{0,1\}^{n\times K} : \begin{array}{ll} y_{jk} \le x_{jk} & \forall j = 1,\dots,n, k = 1,...,K \\ \sum_{j=1}^{n} (\underline{w}_j + \xi_j \tilde{w}_j) y_{jk} \le W & \forall k = 1,\dots,K \end{array} \right\} \quad (71)$$

5.2.1. Instance generation We generated random MKP instances characterized by number of items $n \in \{10, 15\}$ and number of knapsacks $K \in \{2, 3, 4\}$. For each item j, both the profit p_j and the nominal weight $\underline{w}_j$ were generated according to a discrete uniform distribution in $[1, 1000]$. The capacity of each knapsack was defined as $W = \frac{\alpha \sum_{j=1}^{n} \underline{w}_j}{K}$, where $\alpha \in \{0.25, 0.50, 0.75\}$. The weight of each item j in the worst case was defined as $\bar{w}_j = \underline{w}_j(1 + \delta_j)$, where δ_j is randomly generated in $[0, H]$, rounding the resulting value to the closest integer, and $H \in \{0.1, 1.0\}$.

For each combination of these parameters, we generated 5 different instances, thus producing a benchmark with 180 instances. Each instance was solved with different values of $\Gamma \in \{1, 2, 3, 4\}$.

5.2.2. Results Table 2 reports the outcome of our experiments. Each entry of the table refers to the 10 instances associated with the same values of n, α and K for a specific value of Γ. Column "opt" gives the number of instances (out of 10) that are solved to proven optimality within the time limit, whereas "time" reports the average computing time, computed with respect to the instances solved to optimality only.

The results show that our algorithm is able to solve more than 90% of the instances with $n = 10$, the most challenging ones in this group being those with $\alpha = 0.50$ and $K = 4$, and about 55% of the instances with $n = 15$. More in details, when $\Gamma = 1$ almost all the instances (168 out of 180) are solved to optimality, whereas performances get worse for increasing values of Γ, the number of solved instances being 136 for $\Gamma = 2$, 117 for $\Gamma = 3$, and 103 for $\Gamma = 4$. As frequently observed in deterministic knapsack problems (Pisinger 2005), the hardest instances appear with intermediate values of the capacity, i.e., $\alpha = 0.50$ in our context.

Table 3 gives some additional statistics on the behaviour of algorithm B&C, and reports the number of generated nodes (column "nodes"), the average number of generated cuts (column "cuts"), and the average time spent for solving the relaxation and the separation

			$K=2$		$K=3$		$K=4$	
n	α	Γ	opt	time	opt	time	opt	time
10	0.25	1	10	0.45	10	4.05	10	237.85
		2	10	2.15	10	195.4	7	36.65
		3	10	5.85	10	148	8	0.95
		4	10	7.1	10	146.45	8	1.35
	0.50	1	10	2.2	10	183.65	10	0.6
		2	10	14.15	10	754.65	3	0.1
		3	10	36.75	8	1184.65	3	0.45
		4	10	62.85	10	1471.45	3	0.95
	0.75	1	10	1.95	10	0.1	10	0.2
		2	10	30	10	375.1	10	3.2
		3	10	59.85	9	687.3	10	12.7
		4	10	45.05	8	636	10	48.45
15	0.25	1	10	4.95	4	996	6	4.45
		2	10	193.05	1	3591	4	2.7
		3	10	623.2	0	–	4	40.2
		4	8	746.65	2	2070.9	1	6.5
	0.50	1	10	103.2	8	34.7	10	0.3
		2	10	880.8	0	–	6	447.2
		3	5	752.8	0	–	0	–
		4	5	1915.9	0	–	0	–
	0.75	1	10	0.1	10	0.1	10	0.1
		2	5	1446.45	10	37.35	10	6.1
		3	1	486	9	609.9	10	83.65
		4	–	–	2	817.1	6	1019.75

Table 2 Computational results on MKP instances.

problems (columns "t_{rel}" and "t_{sep}", respectively). Each line of the table refers to specific values of n, α, and Γ, i.e., the values refer to 30 instances defined by different values of K in an aggregated way. Figures are computed with respect to the instances solved to optimality only.

These results show that, although we considered instances of medium size, the number of nodes is typically quite large (more than 1200, on average), showing that some enumeration

n	α	Γ	opt	time	nodes	cuts	t_{rel}	t_{sep}
10	0.25	1	30	80.8	1597.0	3103.5	73.6	7.1
		2	27	84.0	1398.5	6957.5	78.3	5.7
		3	28	55.2	730.3	4960.5	52.6	2.6
		4	28	55.2	605.6	4639.6	52.3	2.9
	0.50	1	30	62.2	1292.6	5668.2	59.6	2.5
		2	23	334.3	2107.1	23262.9	314.1	20.2
		3	21	497.3	2207.8	33429.4	475.0	22.3
		4	23	667.2	2374.0	41392.0	626.4	40.7
	0.75	1	30	0.7	42.1	195.1	0.7	0.0
		2	30	136.1	635.9	10553.4	133.9	2.2
		3	29	215.2	528.6	16955.0	212.8	2.4
		4	28	174.9	264.1	12433.4	173.4	1.5
15	0.25	1	20	146.1	1426.3	6329.8	113.3	32.8
		2	15	368.8	2098.3	19669.2	293.5	75.3
		3	14	456.6	1612.9	33544.8	399.5	57.1
		4	11	973.8	2122.5	53311.2	911.0	62.8
	0.50	1	28	44.5	790.7	4955.1	42.1	2.3
		2	16	658.5	2300.9	45363.0	632.9	25.6
		3	5	1128.6	3365.8	71782.6	1120.4	8.2
		4	5	1825.1	2263.0	76708.6	1814.3	10.8
	0.75	1	30	0.1	1.3	20.3	0.1	0.0
		2	25	202.5	285.7	14262.9	200.3	2.1
		3	20	312.1	29.8	7396.3	310.9	1.2
		4	8	861.4	7.0	5828.5	858.4	3.0

Table 3 Additional statistics on MKP instances.

is needed for closing the optimality gap, although the number of nodes remains manageable. Despite the quite large number of separated cuts (more than 20,000, on average), the time for separation is very small, and accounts for only 5% of the total solution time, the remaining 95% of the time being spent for solving the relaxation. The average number of cuts per node is equal to 60, though this figure strongly depends on the size of the problem, as it is around 15 for instances with $n = 10$ and grows by one order of magnitude for $n = 15$.

6. Conclusions

In this paper, we considered adjustable robust optimization problems with mixed-integer wait-and-see decisions and discrete uncertainty set. For this class of problems, we proposed a novel reformulation in which uncertainty appears in the objective function only. This allows us to derive a new exact algorithm for this class of problems, and to perform a computational analysis on a facility location problem with full and partial facility disruptions and on a variant of the multiple knapsack problem. Our computational results show that our approach is able to solve instances of medium size in a reasonable amount of time and compares favourably with the current state-of-the-art approaches from the literature.

Acknowledgements

The authors are grateful to two anonymous reviewers for their constructive comments and remarks. This research was supported by the Air Force Office of Scientific Research under awards number FA8655-20-1-7012 and FA8655-20-1-7019. This work was carried out when the first author was a PhD student at the University of Bologna.

References

Arslan AN, Detienne B (2022) Decomposition-based approaches for a class of two-stage robust binary optimization problems. *INFORMS Journal on Computing* 34(2):857–871.

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

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

Bertsimas D, Caramanis C (2010) Finite adaptability in multistage linear optimization. *IEEE Transactions on Automatic Control* 55(12):2751–2766.

Bertsimas D, Dunning I (2016) Multistage robust mixed-integer optimization with adaptive partitions. *Operations Research* 64(4):980–998.

Bertsimas D, Georghiou A (2017) Binary decision rules for multistage adaptive mixed-integer optimization. *Mathematical Programming* 167(2):395–433.

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

Birge J, Louveaux F (2011) *Introduction to Stochastic Programming*. Springer Series in Operations Research and Financial Engineering (Springer New York).

Buchheim C, Kurtz J (2018) Robust combinatorial optimization under convex and discrete cost uncertainty. *EURO Journal on Computational Optimization* 6(3):211–238.

Cornuéjols G, Sridharan R, Thizy JM (1991) A comparison of heuristics and relaxations for the capacitated plant location problem. *European journal of operational research* 50(3):280–297.

Detienne B, Lefebvre H, Malaguti E, Monaci M (2021) Adaptive robust optimization with objective uncertainty. *Technical Report OR-21-1, DEI-University of Bologna* .

Feizollahi MJ, Ahmed S, Sun A (2016) Exact augmented lagrangian duality for mixed integer linear programming. *Mathematical Programming* 161(1-2):365–387, URL `http://dx.doi.org/10.1007/s10107-016-1012-8`.

Fischetti M, Ljubić I, Monaci M, Sinnl M (2019) Interdiction games and monotonicity, with application to knapsack problems. *INFORMS Journal on Computing* 31(2):390–410.

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

Hanasusanto GA, Kuhn D, Wiesemann W (2015) K-adaptability in two-stage robust binary programming. *Operations Research* 63(4):877–891.

Kämmerling N, Kurtz J (2020) Oracle-based algorithms for binary two-stage robust optimization. *Computational Optimization and Applications* 77(2):539–569.

Malaguti E, Monaci M, Pruente J (2022) K-adaptability in stochastic optimization. *Mathematical Programming* 196:567–595.

Pisinger D (2005) Where are the hard knapsack problems? *Computers & Operations Research* 32(9):2271–2284.

Postek K, den Hertog D (2016) Multi-stage adjustable robust mixed-integer optimization via iterative splitting of the uncertainty set. *INFORMS Journal on Computing* 28(3):553–574.

Prékopa A (2013) *Stochastic Programming*. Mathematics and Its Applications (Springer Netherlands).

Romeijnders W, Postek K (2021) Piecewise constant decision rules via branch-and-bound based scenario detection for integer adjustable robust optimization. *INFORMS Journal on Computing* 33(1):390–400.

Shapiro A, Dentcheva D, Ruszczynski A (2021) *Lectures on stochastic programming: modeling and theory*. MOS-SIAM Series on Optimization (SIAM, Philadelphia, PA).

Soyster AL (1973) Convex programming with set-inclusive constraints and applications to inexact linear programming. *Operations Research* 21(5):1154–1157.

Subramanyam A (2022) A Lagrangian dual method for two-stage robust optimization with binary uncertainties. *Optimization and Engineering* 23(4):1831–1871.

Subramanyam A, Gounaris CE, Wiesemann W (2019) K-adaptability in two-stage mixed-integer robust optimization. *Mathematical Programming Computation* 12(2):193–224.

Zeng B, Zhao L (2012) Solving two-stage robust optimization problems using a column-andconstraint generation method. *Operations Research Letters* 41(5):457–461.

Zhao L, Zeng B (2013) An exact algorithm for two-stage robust optimization with mixed integer recourse problems. *Technical report* .

Appendix

Appendix A: Convex-hull splitting property

The following theorem is derived, and extended, from Arslan and Detienne (2022).

THEOREM 4. *Let* $Y \subseteq \Pi_{j=1}^{n}[l_j, u_j]$ *and let* $L(\boldsymbol{x})$ *be defined, for* $\boldsymbol{x} \in \{0,1\}^n$ *as follows,*

$$L(\boldsymbol{x}) = \left\{ \boldsymbol{y} \in \mathbb{R}_+^n : \forall j \in \{1,...,n\}, \begin{cases} x_j = 1 \Rightarrow y_j \in [\alpha_j^1, \beta_j^1] \\ x_j = 0 \Rightarrow y_j \in [\alpha_j^0, \beta_j^0] \end{cases} \right\} \tag{72}$$

with $\alpha_j^0, \alpha_j^1, \beta_j^0, \beta_j^1 \in \{l_j, u_j\}$. *Then, the following equality holds,*

$$\forall \boldsymbol{x} \in \{0,1\}^n, \quad \text{conv}(Y \cap L(\boldsymbol{x})) = \text{conv}(Y) \cap L(\boldsymbol{x}) \tag{73}$$

Proof: First, it is clear that $\text{conv}(Y \cap L(\boldsymbol{x})) \subseteq \text{conv}(Y) \cap L(\boldsymbol{x})$ for any $\boldsymbol{x} \in \{0,1\}^n$. Thus, assume that there exists $\alpha_1, ..., \alpha_{n+1} \geq 0$ and $\bar{\mathbf{y}}^1, ..., \bar{\mathbf{y}}^{n+1} \in Y$ such that $\sum_{k=1}^{n+1} \alpha_k = 1$ and $\boldsymbol{y} = \sum_{k=1}^{n+1} \alpha_k \bar{\mathbf{y}}^k \in \text{conv}(Y) \cap L(\boldsymbol{x})$ while $\boldsymbol{y} \notin \text{conv}(Y \cap L(\boldsymbol{x}))$. Thus, there exists some $\bar{k}$ and $\bar{j}$ such that $\alpha_{\bar{k}} > 0$, $\bar{\mathbf{y}}^{\bar{k}} \in Y$ and $((x_{\bar{j}} = 1) \wedge (\bar{\mathbf{y}}_{\bar{j}}^{\bar{k}} \notin [\alpha_{\bar{j}}^1, \beta_{\bar{j}}^1]) \vee ((x_{\bar{j}} = 0) \wedge (\bar{\mathbf{y}}_{\bar{j}}^{\bar{k}} \notin [\alpha_{\bar{j}}^0, \beta_{\bar{j}}^0]))$.

We only treat the case where $x_{\bar{j}} = 1$, since the case $x_{\bar{j}} = 0$ can be treated similarly. We then have four possibilities:

- $\alpha_{\bar{j}}^1 = l_{\bar{j}}$, $\beta_{\bar{j}}^1 = l_{\bar{j}}$: Then, $\bar{\mathrm{y}}_{\bar{j}}^{\bar{k}} \notin [\alpha_{\bar{j}}^1, \beta_{\bar{j}}^1]$ implies that $\bar{\mathrm{y}}_{\bar{j}}^{\bar{k}} > l_{\bar{j}}$ and since $\boldsymbol{y} \in L(\boldsymbol{x})$ we must have $y_{\bar{j}} = l_{\bar{j}}$. Then, we can write the following,

$$l_{\bar{j}} = y_{\bar{j}} = \sum_{k=1}^{n+1} \alpha_k \bar{\mathrm{y}}_{\bar{j}}^k > \sum_{k : \bar{\mathrm{y}}_{\bar{j}}^k \in [\alpha_{\bar{j}}^1, \beta_{\bar{j}}^1]} \alpha_k l_{\bar{j}} + \sum_{k : \bar{\mathrm{y}}_{\bar{j}}^k \notin [\alpha_{\bar{j}}^1, \beta_{\bar{j}}^1]} \alpha_k l_{\bar{j}} = l_{\bar{j}} \tag{74}$$

which is absurd.

- $\alpha_{\bar{j}}^1 = u_{\bar{j}}$, $\beta_{\bar{j}}^1 = l_{\bar{j}}$: Impossible, unless $u_{\bar{j}} = l_{\bar{j}}$
- $\alpha_{\bar{j}}^1 = l_{\bar{j}}$, $\beta_{\bar{j}}^1 = u_{\bar{j}}$: Then, $\bar{y}_{\bar{j}}^{\bar{k}} \notin [\alpha_{\bar{j}}^1, \beta_{\bar{j}}^1]$ implies $\bar{\mathrm{y}}^{\bar{k}} \notin Y$, which violates the assumption.
- $\alpha_{\bar{j}}^1 = u_{\bar{j}}$, $\beta_{\bar{j}}^1 = u_{\bar{j}}$: this case yields a contradiction with the same argument as in the first case.

REMARK 4. Special cases of Theorem 4 are $L(\boldsymbol{x}) = \{\boldsymbol{y} : \boldsymbol{y} \leq \boldsymbol{x}\}$, $L(\boldsymbol{x}) = \{\boldsymbol{y} : \boldsymbol{y} \geq \boldsymbol{x}\}$, $L(\boldsymbol{x}) = \{\boldsymbol{y} : \boldsymbol{y} = \boldsymbol{x}\}$ and $L(\boldsymbol{x}) = \{\boldsymbol{y} : y_j \geq (1 - x_j)u_j\}$ for binary $\boldsymbol{x}$.