Mathematical Programming Computation (2020) 12:193–224
https://doi.org/10.1007/s12532-019-00174-2

FULL LENGTH PAPER

K-adaptability in two-stage mixed-integer robust optimization

Anirudh Subramanyam[1] · **Chrysanthos E. Gounaris**[1] · **Wolfram Wiesemann**[2]

Received: 28 July 2018 / Published online: 6 November 2019

Abstract
We study two-stage robust optimization problems with mixed discrete-continuous decisions in both stages. Despite their broad range of applications, these problems pose two fundamental challenges: (i) they constitute infinite-dimensional problems that require a finite-dimensional approximation, and (ii) the presence of discrete recourse decisions typically prohibits duality-based solution schemes. We address the first challenge by studying a K-adaptability formulation that selects K candidate recourse policies *before* observing the realization of the uncertain parameters and that implements the best of these policies *after* the realization is known. We address the second challenge through a branch-and-bound scheme that enjoys asymptotic convergence in general and finite convergence under specific conditions. We illustrate the performance of our algorithm in numerical experiments involving benchmark data from several application domains.

Keywords Robust optimization · Two-stage problems · K-adaptability · Branch-and-bound

Mathematics Subject Classification 90C11 · 90C15 · 90C34 · 90C47

This work was supported by the National Science Foundation, Grant CMMI-1434682, and the Engineering and Physical Sciences Research Council, Grants EP/M028240/1, EP/M027856/1 and EP/N020030/1.

✉ Wolfram Wiesemann
ww@imperial.ac.uk

Anirudh Subramanyam
asubram2@alumni.cmu.edu

Chrysanthos E. Gounaris
gounaris@cmu.edu

[1] Carnegie Mellon University, Pittsburgh, PA, USA

[2] Imperial College London, London, UK

1 Introduction

Dynamic decision-making under uncertainty, where actions need to be taken both in anticipation of and in response to the realization of a priori uncertain problem parameters, arguably forms one of the most challenging domains of operations research and optimization theory. Despite intensive research efforts over the past six decades, many uncertainty-affected optimization problems resist solution, and even our understanding of the complexity of these problems remains incomplete.

In the last two decades, robust optimization has emerged as a promising methodology to counter some of the intricacies associated with decision-making under uncertainty. The rich theory on static robust optimization problems, in which all decisions have to be taken before the uncertainty is resolved, is summarized in [2,4,16]. However, dynamic robust optimization problems, in which some of the decisions can adapt to the observed uncertainties, are still poorly understood.

This paper is concerned with *two-stage robust optimization problems* of the form

$$\inf_{\boldsymbol{x}\in\mathcal{X}} \sup_{\boldsymbol{\xi}\in\Xi} \inf_{\boldsymbol{y}\in\mathcal{Y}} \left\{ \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y} \;:\; \boldsymbol{T}(\boldsymbol{\xi})\boldsymbol{x} + \boldsymbol{W}(\boldsymbol{\xi})\boldsymbol{y} \leq \boldsymbol{h}(\boldsymbol{\xi}) \right\}, \tag{1}$$

where $\mathcal{X} \subseteq \mathbb{R}^{N_1}$, $\mathcal{Y} \subseteq \mathbb{R}^{N_2}$ and $\Xi \subseteq \mathbb{R}^{N_p}$ constitute nonempty and bounded mixed-integer linear programming (MILP) representable sets, $\boldsymbol{c} \in \mathbb{R}^{N_1}$, and the functions $\boldsymbol{d} : \Xi \mapsto \mathbb{R}^{N_2}$, $\boldsymbol{T} : \Xi \mapsto \mathbb{R}^{L\times N_1}$, $\boldsymbol{W} : \Xi \mapsto \mathbb{R}^{L\times N_2}$ and $\boldsymbol{h} : \Xi \mapsto \mathbb{R}^{L}$ are affine. In problem (1), the vector $\boldsymbol{x}$ represents the first-stage (or 'here-and-now') decisions which are taken before the value of the uncertain parameter vector $\boldsymbol{\xi}$ from within the uncertainty set Ξ is observed. The vector $\boldsymbol{y}$, on the other hand, denotes the second-stage ('wait-and-see' or 'adjustable') decisions that can adapt to the realized value of $\boldsymbol{\xi}$. Due to the presence of the adjustable decisions $\boldsymbol{y}$, two-stage robust optimization problems are also referred to as *adjustable robust optimization problems* [3,38]. We emphasize that problem (1) can have a random recourse, i.e., the recourse matrix $\boldsymbol{W}$ may depend on the uncertain parameters $\boldsymbol{\xi}$. Moreover, we do not assume a relatively complete recourse; that is, for some first-stage decisions $\boldsymbol{x} \in \mathcal{X}$, there can be parameter realizations $\boldsymbol{\xi} \in \Xi$ such that there is no feasible second-stage decision $\boldsymbol{y}$. Also, we do not assume that the sets $\mathcal{X}$, $\mathcal{Y}$ or Ξ are convex.

Remark 1 (*Uncertain First-Stage Objective Coefficients*) The assumption that $\boldsymbol{c}$ is deterministic does not restrict generality. Indeed, problem (1) accounts for uncertain first-stage objective coefficients $\boldsymbol{c}' : \Xi \mapsto \mathbb{R}^{N_1}$ if we augment the second-stage decisions $\boldsymbol{y}$ to $(\boldsymbol{y}, \boldsymbol{y}')$, replace the second-stage objective coefficients $\boldsymbol{d}$ with $(\boldsymbol{d}, \boldsymbol{c}')$ and impose the constraint that $\boldsymbol{y}' = \boldsymbol{x}$.

Even in the special case where $\mathcal{X}$, $\mathcal{Y}$ and Ξ are linear programming (LP) representable, problem (1) has been shown to be NP-hard [23]. Nevertheless, problem (1) simplifies considerably if the sets $\mathcal{Y}$ and Ξ are LP representable. For this setting, several approximate solution schemes have been proposed that replace the second-stage decisions with *decision rules*, i.e., parametric classes of linear or nonlinear functions of $\boldsymbol{\xi}$ [3,15,17,18,28]. If we further assume that $\boldsymbol{d}$, $\boldsymbol{T}$ and $\boldsymbol{W}$ are deterministic and Ξ

is of simple form (e.g., a budget uncertainty set), a number of exact solution schemes based on Benders' decomposition [10,26,34,40] and semi-infinite programming [1,39] have been developed.

Problem (1) becomes significantly more challenging if the set $\mathcal{Y}$ is not LP representable. For this setting, conservative MILP approximations have been developed in [20,36] by partitioning the uncertainty set Ξ into hyperrectangles and restricting the continuous and integer recourse decisions to affine and constant functions of $\boldsymbol{\xi}$ over each hyperrectangle, respectively. These a priori partitioning schemes have been extended to iterative partitioning approaches in [6,31]. Iterative solution approaches based on decision rules have been proposed in [7,8]. However, to the best of our knowledge, none of these approaches has been shown to converge to an optimal solution of problem (1). For the special case where problem (1) has a relatively complete recourse, $\boldsymbol{d}$, $\boldsymbol{T}$ and $\boldsymbol{W}$ are deterministic and the optimal value of the second-stage problem is quasi-convex over Ξ, the solution scheme of [39] has been extended in [41] to a nested semi-infinite approach that can solve instances of problem (1) with MILP representable sets $\mathcal{Y}$ and Ξ to optimality in finite time.

Instead of solving problem (1) directly, we study its *K-adaptability problem*

$$\inf_{\substack{\boldsymbol{x}\in\mathcal{X},\\ \boldsymbol{y}\in\mathcal{Y}^K}} \sup_{\boldsymbol{\xi}\in\Xi} \inf_{k\in\mathcal{K}} \left\{ \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y}_k \;:\; \boldsymbol{T}(\boldsymbol{\xi})\boldsymbol{x} + \boldsymbol{W}(\boldsymbol{\xi})\boldsymbol{y}_k \leq \boldsymbol{h}(\boldsymbol{\xi}) \right\}, \tag{2}$$

where $\mathcal{Y}^K = \times_{k=1}^K \mathcal{Y}$ and $\mathcal{K} = \{1, \ldots, K\}$. Problem (2) determines K non-adjustable second-stage policies $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_K$ here-and-now and subsequently selects the best of these policies in response to the observed value of $\boldsymbol{\xi}$. If all policies are infeasible for some realization $\boldsymbol{\xi} \in \Xi$, then the solution $(\boldsymbol{x}, \boldsymbol{y})$ attains the objective value $+\infty$. By construction, the K-adaptability problem (2) bounds the two-stage robust optimization problem (1) from above.

Our interest in problem (2) is motivated by two observations. Firstly, problem (2) has been shown to be a remarkably good approximation of problem (1), both in theory and in numerical experiments [5,24]. Secondly, and perhaps more importantly, the K-adaptability problem conforms well with human decision-making, which tends to address uncertainty by developing a small number of contingency plans, rather than devising the optimal response for every possible future state of the world. For instance, practitioners may prefer a limited number of contingency plans to full flexibility in the second stage for operational (e.g., in production planning or logistics) or organizational (e.g., in emergency response planning) reasons.

The K-adaptability problem was first studied in [5], where the authors reformulate the 2-adaptability problem as a finite-dimensional bilinear program and solve it heuristically. The authors also show that the 2-adaptability problem is NP-hard even if $\boldsymbol{d}$, $\boldsymbol{T}$ and $\boldsymbol{W}$ are deterministic, and they develop necessary conditions for the K-adaptability problem (2) to outperform the static robust problem (where all decisions are taken here-and-now). The relationship between the K-adaptability problem (2) and static robust optimization is further explored in [9] for the special case where $\boldsymbol{T}$ and $\boldsymbol{W}$ are deterministic. The authors show that the gaps between both problems and the two-stage robust optimization problem (1) are intimately related to geomet-

ric properties of the uncertainty set Ξ. Finite-dimensional MILP reformulations for problem (2) are developed in [24] under the additional assumption that both the here-and-now decisions $\boldsymbol{x}$ and the wait-and-see decisions $\boldsymbol{y}$ are binary. The authors show that both the size of the reformulations as well as their gaps to the two-stage robust optimization problem (1) depend on whether the uncertainty only affects the objective coefficients $\boldsymbol{d}$, or whether the constraint coefficients $\boldsymbol{T}$, $\boldsymbol{W}$ and $\boldsymbol{h}$ are uncertain as well. Finally, it is shown in [12,13] that for polynomial time solvable deterministic combinatorial optimization problems, the associated instances of problem (2) without first-stage decisions $\boldsymbol{x}$ can also be solved in polynomial time if all of the following conditions hold: (i) Ξ is convex, (ii) only the objective coefficients $\boldsymbol{d}$ are uncertain, and (iii) $K > N_2$ policies are sought. This result has been extended to discrete uncertainty sets in [14], in which case pseudo-polynomial solution algorithms can be developed.

In this paper, we expand the literature on the K-adaptability problem in two ways. From an *analytical viewpoint*, we compare the two-stage robust optimization problem (1) with the K-adaptability problem (2) in terms of their continuity, convexity and tractability. We also investigate when the approximation offered by the K-adaptability problem is tight, and under which conditions the two-stage robust optimization and K-adaptability problems reduce to single-stage problems. From an *algorithmic viewpoint*, we develop a branch-and-bound scheme for the K-adaptability problem that combines ideas from semi-infinite and disjunctive programming. We establish conditions for its asymptotic and finite time convergence; we show how it can be refined and integrated into state-of-the-art MILP solvers; and, we present a heuristic variant that can address large-scale instances. In contrast to existing approaches, our algorithm can handle mixed continuous and discrete decisions in both stages as well as discrete uncertainty, and allows for modeling continuous second-stage decisions via a novel class of highly flexible piecewise affine decision rules. Extensive numerical experiments on benchmark data from various application domains indicate that our algorithm is highly competitive with state-of-the-art solution schemes for problems (1) and (2).

We highlight that our C++ code is freely available for download on GitHub [33].

The remainder of this paper develops as follows. Section 2 analyzes the geometry and tractability of problems (1) and (2), and it proposes a novel class of piecewise affine decision rules that can be modeled as an instance of problem (2) with continuous second-stage decisions. Section 3 develops a branch-and-bound algorithm for the K-adaptability problem and analyzes its convergence. We present numerical results in Sect. 4, and we offer concluding remarks in Sect. 5.

Notation Vectors and matrices are printed in bold lowercase and uppercase letters, respectively, while scalars are printed in regular font. We use $\mathbf{e}_k$ to denote the kth unit basis vector and $\mathbf{e}$ to denote the vector whose components are all ones, respectively; their dimensions will be clear from the context. The ith row vector of a matrix $\boldsymbol{A}$ is denoted by $\boldsymbol{a}_i^\top$. For a logical expression $\mathcal{E}$, we define $\mathbb{I}[\mathcal{E}]$ as the indicator function which takes a value of 1 is $\mathcal{E}$ is true and 0 otherwise.

Table 1 Summary of theoretical results. Here "OBJ" refers to instances where only the objective coefficients d are uncertain, while T, W and h are constant. Similarly, "CON" refers to instances where only the constraint coefficients T, W and right-hand sides h are uncertain, while d is constant

		Continuity	Convexity	Tractability
First-stage problem	problem (1)	if feasible	if $\mathcal{X}$, $\mathcal{Y}$ convex	if $\mathcal{X}$, $\mathcal{Y}$, Ξ convex and OBJ
	problem (2)	if feasible	typically not	if $\mathcal{X}$, $\mathcal{Y}$, Ξ convex and OBJ
Evaluation problem	problem (1)	if OBJ	if Ξ convex and OBJ	if Ξ, $\mathcal{Y}$ convex and OBJ
	problem (2)	if OBJ or CON	if Ξ convex and OBJ	if Ξ convex and OBJ
Second-stage problem	problem (1)	if feasible	if $\mathcal{Y}$ convex	if $\mathcal{Y}$ convex
	problem (2)	if feasible	always	always

	Reduction to static problem
problem (1)	if Ξ convex and OBJ
problem (2)	if Ξ convex, OBJ and $K > \min\{N_2, N_p\}$

Optimality of problem (2)
if $\mathcal{Y}$, Ξ convex and OBJ
if Ξ convex, OBJ and $K > \min\{N_2, N_p\}$
if $\lvert\mathcal{Y}\rvert$ finite and $K \geq \lvert\mathcal{Y}\rvert$

2 Problem analysis

In this section, we analyze the geometry and tractability of the two-stage robust optimization problem (1) and its associated K-adaptability problem (2). Specifically, we characterize the continuity, convexity and tractability of both problems, as well as their relationship to the static robust optimization problem where all decisions are taken here-and-now. We also show how the K-adaptability problem with continuous second-stage decisions enables us to approximate the two-stage robust optimization problem (1) through highly flexible piecewise affine decision rules.

Table 1 summarizes our theoretical results. For the sake of brevity, we relegate the formal statements of these results as well as their proofs to the electronic version of the paper at arxiv.org/abs/1706.07097. In the table, the *first-stage problem* refers to the overall problems (1) and (2), the *evaluation problem* refers to the maximization over $\boldsymbol{\xi} \in \Xi$ for a fixed first-stage decision, and the *second-stage problem* refers to the inner minimization over $\boldsymbol{y} \in \mathcal{Y}$ or $k \in \mathcal{K}$ for a fixed first-stage decision and a fixed realization of the uncertain problem parameters. The table reveals that despite significant differences in their formulations, the problems (1) and (2) behave very similarly. The most significant difference is caused by the replacement of the optimization over the second-stage decisions $\boldsymbol{y} \in \mathcal{Y}$ in problem (1) with the selection of a candidate policy $k \in \mathcal{K}$ in problem (2). This ensures that the second-stage problem in (2) is always continuous, convex and tractable, whereas the first-stage problem in (2) fails to be convex even if $\mathcal{X}$ and $\mathcal{Y}$ are convex. Moreover, in contrast to problem (1), the evaluation problem in (2) remains continuous as long as either the objective function or the constraints are unaffected by uncertainty. For general problem instances, however, neither of the two evaluation problems is continuous. As we will see in Sect. 3.2, this directly impacts the convergence of our branch-and-bound algorithm, which only takes place asymptotically in general. Note that the convexity of the problems (1) and (2) does not depend on the shape of the uncertainty set Ξ.

2.1 Incorporating decision rules in the *K*-adaptability problem

Although the K-adaptability problem (2) selects the best candidate policy $\boldsymbol{y}_k$ in response to the observed parameter realization $\boldsymbol{\xi} \in \Xi$, the policies $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_K$—

once selected in the first stage—no longer depend on $\boldsymbol{\xi}$. This lack of dependence on the uncertain problem parameters can lead to overly conservative approximations of the two-stage robust optimization problem (1) when the second-stage decisions are continuous. In this section, we show how the K-adaptability problem (2) can be used to generalize affine decision rules, which are commonly used to approximate continuous instances of the two-stage robust optimization problem (1). We note that existing schemes, such as [13,24], cannot be used for this purpose as they require the wait-and-see decisions $\boldsymbol{y}$ to be binary.

Throughout this section, we assume that problem (1) has purely continuous second-stage decisions (that is, $\mathcal{Y}$ is LP representable), a deterministic objective function (that is, $\boldsymbol{d}(\boldsymbol{\xi}) = \boldsymbol{d}$ for all $\boldsymbol{\xi} \in \Xi$) and fixed recourse (that is, $\boldsymbol{W}(\boldsymbol{\xi}) = \boldsymbol{W}$ for all $\boldsymbol{\xi} \in \Xi$). The assumption of continuous second-stage decisions allows us to assume, without loss of generality, that $\mathcal{Y} = \mathbb{R}^{N_2}$ as any potential restrictions can be absorbed in the second-stage constraints.

The affine decision rule approximation to the two-stage robust optimization problem (1) is

$$\inf_{\substack{\boldsymbol{x}\in\mathcal{X},\\ \boldsymbol{y}:\Xi\overset{1}{\mapsto}\mathbb{R}^{N_2}}} \left\{ \sup_{\boldsymbol{\xi}\in\Xi} \left\{ \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}^\top \boldsymbol{y}(\boldsymbol{\xi}) \right\} \,:\, \boldsymbol{T}(\boldsymbol{\xi})\boldsymbol{x} + \boldsymbol{W}\boldsymbol{y}(\boldsymbol{\xi}) \leq \boldsymbol{h}(\boldsymbol{\xi}) \;\; \forall \boldsymbol{\xi} \in \Xi \right\},$$

where $\boldsymbol{y} : \Xi \overset{1}{\mapsto} \mathbb{R}^{N_2}$ indicates that $\boldsymbol{y}(\boldsymbol{\xi}) = \boldsymbol{y}^0 + \boldsymbol{Y}\boldsymbol{\xi}$ for some $\boldsymbol{y}^0 \in \mathbb{R}^{N_2}$ and $\boldsymbol{Y} \in \mathbb{R}^{N_2 \times N_p}$, see [3]. This problem provides a conservative approximation to the two-stage robust optimization problem (1) since we replace the space of all (possibly non-convex and discontinuous) second-stage policies $\boldsymbol{y} : \Xi \mapsto \mathbb{R}^{N_2}$ with the subspace of all affine second-stage policies $\boldsymbol{y} : \Xi \overset{1}{\mapsto} \mathbb{R}^{N_2}$. In a similar spirit, we define the subspace of all piecewise affine decision rules $\boldsymbol{y} : \Xi \overset{K}{\mapsto} \mathbb{R}^{N_2}$ with K pieces as

$$\boldsymbol{y} : \Xi \overset{K}{\mapsto} \mathbb{R}^{N_2} \iff \left[\begin{array}{l} \exists (\boldsymbol{y}_k^0, \boldsymbol{Y}_k) \in \mathbb{R}^{N_2} \times \mathbb{R}^{N_2 \times N_p},\ k = 1, \ldots, K, \text{ such that} \\ \forall \boldsymbol{\xi} \in \Xi,\ \exists k \in \{1, \ldots, K\} \,:\, \boldsymbol{y}(\boldsymbol{\xi}) = \boldsymbol{y}_k^0 + \boldsymbol{Y}_k \boldsymbol{\xi} \end{array} \right].$$

Note that our earlier definition of $\boldsymbol{y} : \Xi \overset{1}{\mapsto} \mathbb{R}^{N_2}$ is identical to our definition of $\boldsymbol{y} : \Xi \overset{K}{\mapsto} \mathbb{R}^{N_2}$ if $K = 1$. For $K > 1$, the decision rules $\boldsymbol{y} : \Xi \overset{K}{\mapsto} \mathbb{R}^{N_2}$ may be non-convex and discontinuous, and the regions where $\boldsymbol{y}$ is affine may be non-closed and non-convex. We highlight that the points of nonlinearity are determined by the optimization problem. This is in contrast to many existing solution schemes for piecewise affine decision rules, such as [8,15,17,18], where these points are specified ad hoc by the decision-maker.

Observation 1 *The piecewise affine decision rule problem with fixed recourse*

$$\inf_{\substack{\boldsymbol{x}\in\mathcal{X},\\ \boldsymbol{y}:\Xi\overset{K}{\mapsto}\mathbb{R}^{N_2}}} \left\{ \sup_{\boldsymbol{\xi}\in\Xi} \left\{ \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}^\top \boldsymbol{y}(\boldsymbol{\xi}) \right\} \,:\, \boldsymbol{T}(\boldsymbol{\xi})\boldsymbol{x} + \boldsymbol{W}\boldsymbol{y}(\boldsymbol{\xi}) \leq \boldsymbol{h}(\boldsymbol{\xi}) \quad \forall \boldsymbol{\xi} \in \Xi \right\} \tag{3}$$

is equivalent to the K-adaptability problem with random recourse

$$\inf_{\substack{\boldsymbol{x}\in\mathcal{X},\\ (\boldsymbol{y}^0,\boldsymbol{Y},z)\in\hat{\mathcal{Y}}^K}} \sup_{\boldsymbol{\xi}\in\Xi} \inf_{k\in\mathcal{K}} \left\{ \boldsymbol{c}^\top\boldsymbol{x} + \boldsymbol{d}^\top\boldsymbol{y}_k^0 + \hat{\boldsymbol{d}}(\boldsymbol{\xi})^\top z_{k1} \,:\, \boldsymbol{T}(\boldsymbol{\xi})\boldsymbol{x} + \boldsymbol{W}\boldsymbol{y}_k^0 + \hat{\boldsymbol{W}}(\boldsymbol{\xi})z_{k2} \leq \boldsymbol{h}(\boldsymbol{\xi}) \right\}, \quad (4)$$

where $\hat{\mathcal{Y}} = \{(\boldsymbol{y}^0, \boldsymbol{Y}, \boldsymbol{z}) \in \mathbb{R}^{N_2} \times \mathbb{R}^{N_2\times N_p} \times (\mathbb{R}^{N_p} \times \mathbb{R}^{N_p L}) : \boldsymbol{z} = (\boldsymbol{z}_1, \boldsymbol{z}_2)$ *with* $\boldsymbol{z}_1 = \boldsymbol{Y}^\top\boldsymbol{d}$ *and* $\boldsymbol{z}_2 = [\boldsymbol{w}_1^\top\boldsymbol{Y} \ \ldots \ \boldsymbol{w}_L^\top\boldsymbol{Y}]^\top\}$, $\hat{\boldsymbol{d}}(\boldsymbol{\xi}) = \boldsymbol{\xi}$ *and* $\hat{\boldsymbol{W}}(\boldsymbol{\xi}) = \operatorname{diag}(\boldsymbol{\xi}^\top, \ldots, \boldsymbol{\xi}^\top) \in \mathbb{R}^{L\times N_p L}$.

Proof Problem (4) is infeasible if and only if for every $\boldsymbol{x} \in \mathcal{X}$ and $(\boldsymbol{y}^0, \boldsymbol{Y}, \boldsymbol{z}) \in \hat{\mathcal{Y}}^K$ there is a $\boldsymbol{\xi} \in \Xi$ such that $\boldsymbol{T}(\boldsymbol{\xi})\boldsymbol{x} + \boldsymbol{W}\boldsymbol{y}_k^0 + \hat{\boldsymbol{W}}(\boldsymbol{\xi})z_{k2} \not\leq \boldsymbol{h}(\boldsymbol{\xi})$ for all $k = 1, \ldots, K$, which in turn is the case if and only if for every $\boldsymbol{x} \in \mathcal{X}$ and $(\boldsymbol{y}_k^0, \boldsymbol{Y}_k) \in \mathbb{R}^{N_2} \times \mathbb{R}^{N_2\times N_p}$, $k = 1, \ldots, K$ there is a $\boldsymbol{\xi} \in \Xi$ such that $\boldsymbol{T}(\boldsymbol{\xi})\boldsymbol{x} + \boldsymbol{W}\boldsymbol{y}_k^0 + \boldsymbol{W}\boldsymbol{Y}_k\boldsymbol{\xi} \not\leq \boldsymbol{h}(\boldsymbol{\xi})$ for all $k = 1, \ldots, K$; that is, if and only if problem (3) is infeasible. We thus assume that both (3) and (4) are feasible. In this case, we verify that every feasible solution $(\boldsymbol{x}, \boldsymbol{y}^0, \boldsymbol{Y}, \boldsymbol{z})$ to problem (4) gives rise to a feasible solution $(\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{y}(\boldsymbol{\xi}) = \boldsymbol{y}_{k(\boldsymbol{\xi})}^0 + \boldsymbol{Y}_{k(\boldsymbol{\xi})}\boldsymbol{\xi}$ and $k(\boldsymbol{\xi})$ is any element of $\arg\min_{k\in\mathcal{K}}\{\boldsymbol{c}^\top\boldsymbol{x} + \boldsymbol{d}^\top\boldsymbol{y}_k^0 + \hat{\boldsymbol{d}}(\boldsymbol{\xi})^\top z_{k1} : \boldsymbol{T}(\boldsymbol{\xi})\boldsymbol{x} + \boldsymbol{W}\boldsymbol{y}_k^0 + \hat{\boldsymbol{W}}(\boldsymbol{\xi})z_{k2} \leq \boldsymbol{h}(\boldsymbol{\xi})\}$, in problem (3) that attains the same worst-case objective value. Similarly, every *optimal solution* $(\boldsymbol{x}, \boldsymbol{y})$ to problem (3) gives rise to an optimal solution $(\boldsymbol{x}, \boldsymbol{y}^0, \boldsymbol{Y}, \boldsymbol{z})$, where $\boldsymbol{z}_k = (z_{k1}, z_{k2})$ with $z_{k1} = \boldsymbol{Y}_k^\top\boldsymbol{d}$ and $z_{k2} = [\boldsymbol{w}_1^\top\boldsymbol{Y}_k \ \ldots \ \boldsymbol{w}_L^\top\boldsymbol{Y}_k]^\top$, $k = 1, \ldots, K$, in problem (4). Hence, (3) and (4) share the same optimal value and the same sets of optimal solutions. □

We close with an example that illustrates the benefits of piecewise affine decision rules.

Example 1 Consider the following instance of the two-stage robust optimization problem (1), which has been proposed in [19, Sect. 5.2]:

$$\sup_{\boldsymbol{\xi}\in[-1,1]^2} \inf_{\boldsymbol{y}\in\mathbb{R}^4_+} \left\{ \boldsymbol{e}^\top\boldsymbol{y} : y_1 \geq \xi_1+\xi_2,\ y_2 \geq \xi_1-\xi_2,\ y_3 \geq -\xi_1+\xi_2,\ y_4 \geq -\xi_1-\xi_2 \right\}.$$

The *optimal second-stage policy* is $\boldsymbol{y}^\star(\boldsymbol{\xi}) = ([\xi_1 + \xi_2]_+, [\xi_1 - \xi_2]_+, [-\xi_1 + \xi_2]_+, [-\xi_1 - \xi_2]_+)$, where $[\cdot]_+ = \max\{\cdot, 0\}$, and it results in the optimal second-stage value function $Q^\star(\boldsymbol{\xi}) = [\xi_1+\xi_2]_+ + [\xi_1-\xi_2]_+ + [-\xi_1+\xi_2]_+ + [-\xi_1-\xi_2]_+$ with a worst-case objective value of 2, see Fig. 1. The *best affine decision rule* $\boldsymbol{y}^1 : \Xi \overset{1}{\mapsto} \mathbb{R}^4_+$ is $\boldsymbol{y}^1(\boldsymbol{\xi}) = (1 + \xi_2,\ 1 + \xi_1,\ 1 - \xi_1,\ 1 - \xi_2)$, and it results in the constant second-stage value function $Q^1(\boldsymbol{\xi}) = 4$. The *best* 2*-adaptable affine decision rule* $\boldsymbol{y}^2 : \Xi \overset{2}{\mapsto} \mathbb{R}^4_+$, on the other hand, is given by

$$\boldsymbol{y}^2(\boldsymbol{\xi}) = \begin{cases} (0,\ 1 + \xi_1,\ 1 + \xi_2,\ -\xi_1 - \xi_2) & \text{if } \xi_1 + \xi_2 \leq 0 \\ (\xi_1 + \xi_2,\ 1 - \xi_2,\ 1 - \xi_1,\ 0) & \text{otherwise,} \end{cases}$$

and it results in the constant second-stage value function $Q^2(\boldsymbol{\xi}) = 2$. Thus, 2-adaptable affine decision rules are optimal in this example. Figure 2 illustrates the optimal value,

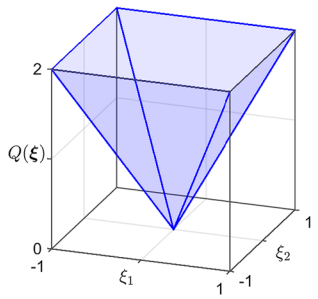

Fig. 1 The optimal second-stage value function in Example 1 is given by the first-order cone $Q^\star(\boldsymbol{\xi}) = [\xi_1 + \xi_2]_+ + [\xi_1 - \xi_2]_+ + [-\xi_1 + \xi_2]_+ + [-\xi_1 - \xi_2]_+$

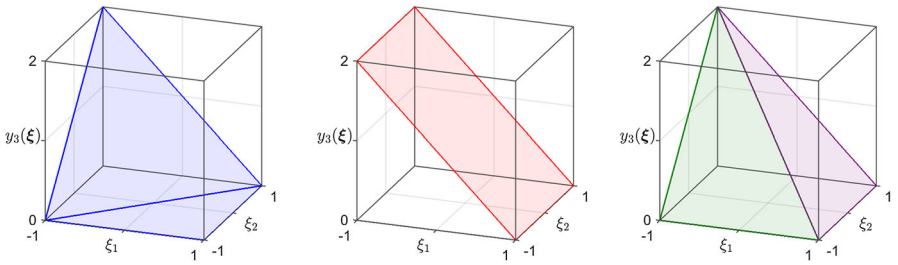

Fig. 2 Plot of the optimal second-stage policy $y_3(\boldsymbol{\xi})$ in the two-stage robust optimization problem (left), the affine decision rule problem (middle) and the 2-adaptable affine decision rule problem (right)

the affine approximation and the 2-adaptable affine approximation of the decision variable y_3.

The piecewise affine decision rules presented here can be readily combined with discrete second-stage decisions. For the sake of brevity, we omit the details of this straightforward extension.

3 Solution scheme

Our solution scheme for the K-adaptability problem (2) is based on a reformulation as a semi-infinite disjunctive program which we present next.

Observation 2 *The K-adaptability problem (2) is equivalent to*

$$\begin{array}{l} \textit{minimize} \;\; \theta \\ \textit{subject to} \; \theta \in \mathbb{R}, \;\; \boldsymbol{x} \in \mathcal{X}, \;\; \boldsymbol{y} \in \mathcal{Y}^K \\ \qquad \bigvee\limits_{k \in \mathcal{K}} \begin{bmatrix} \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y}_k \leq \theta \\ \boldsymbol{T}(\boldsymbol{\xi})\boldsymbol{x} + \boldsymbol{W}(\boldsymbol{\xi})\boldsymbol{y}_k \leq \boldsymbol{h}(\boldsymbol{\xi}) \end{bmatrix} \quad \forall \boldsymbol{\xi} \in \Xi. \end{array} \tag{5}$$

Moreover, if some of the constraints in problem (5) are deterministic, i.e., they do not depend on $\boldsymbol{\xi}$, then they can be moved outside the disjunction and instead be enforced for all $k \in \mathcal{K}$.

In the following, we stipulate that the optimal value of (5) is $+\infty$ whenever it is infeasible.

Proof of Observation 2 Problem (2) is infeasible if and only if (iff) for every $\boldsymbol{x} \in \mathcal{X}$ and $\boldsymbol{y} \in \mathcal{Y}^K$ there is a $\boldsymbol{\xi} \in \Xi$ such that $\boldsymbol{T}(\boldsymbol{\xi})\boldsymbol{x} + \boldsymbol{W}(\boldsymbol{\xi})\boldsymbol{y}_k \not\leq \boldsymbol{h}(\boldsymbol{\xi})$ for all $k \in \mathcal{K}$, which in turn is the case iff for every $\boldsymbol{x} \in \mathcal{X}$ and $\boldsymbol{y} \in \mathcal{Y}^K$, the disjunction in (5) is violated for at least one $\boldsymbol{\xi} \in \Xi$; that is, iff problem (5) is infeasible. We thus assume that both (2) and (5) are feasible. In this case, one readily verifies that every feasible solution $(\boldsymbol{x}, \boldsymbol{y})$ to problem (2) gives rise to a feasible solution $(\theta, \boldsymbol{x}, \boldsymbol{y})$, where $\theta = \sup_{\boldsymbol{\xi}\in\Xi} \inf_{k\in\mathcal{K}} \{\boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y}_k : \boldsymbol{T}(\boldsymbol{\xi})\boldsymbol{x} + \boldsymbol{W}(\boldsymbol{\xi})\boldsymbol{y}_k \leq \boldsymbol{h}(\boldsymbol{\xi})\}$, in problem (5) with the same objective value. Likewise, any *optimal* solution $(\theta, \boldsymbol{x}, \boldsymbol{y})$ to problem (5) corresponds to an optimal solution $(\boldsymbol{x}, \boldsymbol{y})$ in problem (2). Hence, (2) and (5) share the same optimal value and the same sets of optimal solutions.

We now claim that if $\boldsymbol{t}_l(\boldsymbol{\xi})^\top = \boldsymbol{t}_l^\top$, $\boldsymbol{w}_l(\boldsymbol{\xi})^\top = \boldsymbol{w}_l^\top$ and $h_l(\boldsymbol{\xi}) = h_l$ for all $l \in \{1, \ldots, L\}\backslash\mathcal{L}$, where $\mathcal{L} \subseteq \{1, \ldots, L\}$, then problem (5) is equivalent to

$$
\begin{array}{l}
\text{minimize } \theta \\
\text{subject to } \theta \in \mathbb{R}, \quad \boldsymbol{x} \in \mathcal{X}, \quad \boldsymbol{y} \in \mathcal{Y}^K \\
\qquad \boldsymbol{t}_l^\top \boldsymbol{x} + \boldsymbol{w}_l^\top \boldsymbol{y}_k \leq h_l, \quad \forall l \in \{1, \ldots, L\}\backslash\mathcal{L}, \ \forall k \in \mathcal{K} \\
\qquad \bigvee_{k\in\mathcal{K}} \left[\begin{array}{l} \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y}_k \leq \theta \\ \boldsymbol{t}_l(\boldsymbol{\xi})^\top \boldsymbol{x} + \boldsymbol{w}_l(\boldsymbol{\xi})^\top \boldsymbol{y}_k \leq h_l(\boldsymbol{\xi}) \quad \forall l \in \mathcal{L} \end{array} \right] \quad \forall \boldsymbol{\xi} \in \Xi.
\end{array} \tag{5'}
$$

By construction, every feasible solution $(\theta, \boldsymbol{x}, \boldsymbol{y})$ to problem (5′) is feasible in problem (5). Conversely, fix any feasible solution $(\theta, \boldsymbol{x}, \boldsymbol{y})$ to problem (5) and assume that $\boldsymbol{t}_l^\top \boldsymbol{x} + \boldsymbol{w}_l^\top \boldsymbol{y}_k > h_l$ for some $k \in \mathcal{K}$ and $l \in \{1, \ldots, L\}\backslash\mathcal{L}$. In that case, the kth disjunct in (5) is violated for *every* realization $\boldsymbol{\xi} \in \Xi$. We can therefore replace $\boldsymbol{y}_k$ with a different candidate policy $\boldsymbol{y}_{k'}$ that satisfies $\boldsymbol{t}_l^\top \boldsymbol{x} + \boldsymbol{w}_l^\top \boldsymbol{y}_{k'} \leq h_l$ for all $l \in \{1, \ldots, L\}\backslash\mathcal{L}$ without sacrificing feasibility. [Note that such a candidate policy $\boldsymbol{y}_{k'}$ exists since $(\theta, \boldsymbol{x}, \boldsymbol{y})$ is assumed to be feasible in (5)]. Replacing any infeasible policy in this way results in a solution that is feasible in problem (5′). □

Problem (5) cannot be solved directly as it contains infinitely many disjunctive constraints. Instead, our solution scheme iteratively solves a sequence of (increasingly tighter) relaxations of this problem that are obtained by enforcing the disjunctive constraints over finite subsets of Ξ. Whenever the solution of such a relaxation violates the disjunction for some realization $\boldsymbol{\xi} \in \Xi$, we create K subproblems that enforce the disjunction associated with $\boldsymbol{\xi}$ to be satisfied by the kth disjunct, $k = 1, \ldots, K$. Our solution scheme is reminiscent of discretization methods employed in *semi-infinite programming*, which iteratively replace an infinite set of constraints with finite subsets and solve the resulting discretized problems. Indeed, our scheme can be interpreted as a generalization of Kelley's cutting-plane method [11,27] applied to semi-infinite disjunctive programs. In the special case where $K = 1$, our method reduces to the cutting-plane method for (static) robust optimization problems proposed in [30].

In the remainder of this section, we describe our basic branch-and-bound scheme (Sect. 3.1), we study its convergence (Sect. 3.2), we discuss algorithmic variants to the basic scheme that can enhance its numerical performance (Sect. 3.3), and we present a heuristic variant that can address problems of larger scale (Sect. 3.4).

3.1 Branch-and-bound algorithm

Our solution scheme iteratively solves a sequence of scenario-based K-adaptability problems and separation problems. We define both problems first, and then we describe the overall algorithm.

The Scenario-Based K-Adaptability Problem For a collection $\Xi_1, \ldots, \Xi_K$ of finite subsets of the uncertainty set Ξ, we define the *scenario-based K-adaptability problem* as

$$\mathcal{M}(\Xi_1, \ldots, \Xi_K) \;=\; \begin{array}{ll} \text{minimize} & \theta \\ \text{subject to} & \theta \in \mathbb{R},\;\; \boldsymbol{x} \in \mathcal{X},\;\; \boldsymbol{y} \in \mathcal{Y}^K \\ & \left.\begin{array}{l} \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y}_k \le \theta \\ \boldsymbol{T}(\boldsymbol{\xi})\boldsymbol{x} + \boldsymbol{W}(\boldsymbol{\xi})\boldsymbol{y}_k \le \boldsymbol{h}(\boldsymbol{\xi}) \end{array}\right\} \;\; \forall \boldsymbol{\xi} \in \Xi_k,\; \forall k \in \mathcal{K}. \end{array} \tag{6}$$

If $\mathcal{X}$ and $\mathcal{Y}$ are convex, problem (6) is an LP; otherwise, it is an MILP. The problem is closely related to a relaxation of the semi-infinite disjunctive program (5) that enforces the disjunction only over the realizations $\boldsymbol{\xi} \in \bigcup_{k\in\mathcal{K}} \Xi_k$. More precisely, problem (6) can be interpreted as a restriction of that relaxation which requires the kth candidate policy $\boldsymbol{y}_k$ to be worst-case optimal for all realizations $\boldsymbol{\xi} \in \Xi_k$, $k \in \mathcal{K}$. We obtain an optimal solution $(\theta, \boldsymbol{x}, \boldsymbol{y}_k)$ to the relaxed semi-infinite disjunctive program by solving $\mathcal{M}(\Xi_1, \ldots, \Xi_K)$ for all partitions $(\Xi_1, \ldots, \Xi_K)$ of $\bigcup_{k\in\mathcal{K}} \Xi_k$ and reporting the optimal solution $(\theta, \boldsymbol{x}, \boldsymbol{y}_k)$ of the problem $\mathcal{M}(\Xi_1, \ldots, \Xi_K)$ with the smallest objective value.

If $\Xi_k = \emptyset$ for all $k \in \mathcal{K}$, then problem (6) is unbounded, and we stipulate that its optimal value is $-\infty$ and that its optimal value is attained by any solution $(-\infty, \boldsymbol{x}, \boldsymbol{y})$ with $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{X} \times \mathcal{Y}^K$. Otherwise, if problem (6) is infeasible for $\Xi_1, \ldots, \Xi_K$, then we define its optimal value to be $+\infty$. In all other cases, the optimal value of problem (6) is finite and it is attained by an optimal solution $(\theta, \boldsymbol{x}, \boldsymbol{y})$ since $\mathcal{X}$ and $\mathcal{Y}$ are compact.

Remark 2 (Decomposability) For K-adaptability problems without first-stage decisions $\boldsymbol{x}$, problem (6) decomposes into K scenario-based static robust optimization problems that are only coupled through the constraints referencing the epigraph variable θ. In this case, we can recover an optimal solution to problem (6) by solving each of the K static problems individually and identifying the optimal θ as the maximum of their optimal values.

The Separation Problem. For a feasible solution $(\theta, \boldsymbol{x}, \boldsymbol{y})$ to the scenario-based K-adaptability problem (6), we define the *separation problem* as

$$\begin{aligned} \mathcal{S}(\theta, \boldsymbol{x}, \boldsymbol{y}) &= \max_{\boldsymbol{\xi}\in\Xi} \; S(\theta, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi}), \;\; \text{where} \\ S(\theta, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi}) &= \min_{k\in\mathcal{K}} \; \max \\ &\qquad \left\{ \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y}_k - \theta,\; \max_{l\in\{1,\ldots,L\}} \left\{ \boldsymbol{t}_l(\boldsymbol{\xi})^\top \boldsymbol{x} + \boldsymbol{w}_l(\boldsymbol{\xi})^\top \boldsymbol{y}_k - h_l(\boldsymbol{\xi}) \right\} \right\}, \end{aligned} \tag{7}$$

for $\mathcal{S} : \mathbb{R} \cup \{-\infty\} \times \mathcal{X} \times \mathcal{Y}^K \mapsto \mathbb{R} \cup \{+\infty\}$ and $S : \mathbb{R} \cup \{-\infty\} \times \mathcal{X} \times \mathcal{Y}^K \times \Xi \mapsto \mathbb{R} \cup \{+\infty\}$. Whenever it is positive, the innermost maximum in the definition of $S(\theta, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi})$ records the maximum constraint violation of the candidate policy $\boldsymbol{y}_k$ under the parameter realization $\boldsymbol{\xi} \in \Xi$. Likewise, the quantity $\boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y}_k - \theta$ denotes the excess of the objective value of $\boldsymbol{y}_k$ under the realization $\boldsymbol{\xi}$ over the current candidate value of the worst-case objective, θ. Thus, $S(\theta, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi})$ is strictly positive if and only if every candidate policy $\boldsymbol{y}_k$ either is infeasible or results in an objective value greater than θ under the realization $\boldsymbol{\xi} \in \Xi$. Whenever θ is finite, the separation problem is feasible and bounded, and it has an optimal solution since Ξ is nonempty and compact. Otherwise, we have $\mathcal{S}(\theta, \boldsymbol{x}, \boldsymbol{y}) = +\infty$, and the optimal value is attained by any $\boldsymbol{\xi} \in \Xi$.

Observation 3 *The separation problem (7) is equivalent to the MILP*

$$\begin{array}{ll} \textit{maximize} & \zeta \\ \textit{subject to} & \zeta \in \mathbb{R}, \quad \boldsymbol{\xi} \in \Xi, \quad z_{kl} \in \{0,1\}, \ (k,l) \in \mathcal{K} \times \{0,1,\ldots,L\} \\ & \left.\begin{array}{l} \displaystyle\sum_{l=0}^{L} z_{kl} = 1 \\ z_{k0} = 1 \quad \Rightarrow \quad \zeta \le \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y}_k - \theta \\ z_{kl} = 1 \quad \Rightarrow \quad \zeta \le \boldsymbol{t}_l(\boldsymbol{\xi})^\top \boldsymbol{x} + \boldsymbol{w}_l(\boldsymbol{\xi})^\top \boldsymbol{y}_k - h_l(\boldsymbol{\xi}) \quad \forall l \in \{1,\ldots,L\} \end{array}\right\} \ \forall k \in \mathcal{K}. \end{array} \tag{8}$$

This problem can be solved in polynomial time if Ξ is convex and the number of policies K is fixed.

Proof Fix any feasible solution $(\theta, \boldsymbol{x}, \boldsymbol{y})$ to the scenario-based K-adaptability problem (6). For every $\boldsymbol{\xi} \in \Xi$, we can construct a feasible solution $(\zeta, \boldsymbol{\xi}, \boldsymbol{z})$ to problem (8) with $\zeta = S(\theta, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi})$ by setting $z_{k0} = 1$ if $\boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y}_k - \theta \ge \boldsymbol{t}_l(\boldsymbol{\xi})^\top \boldsymbol{x} + \boldsymbol{w}_l(\boldsymbol{\xi})^\top \boldsymbol{y}_k - h_l(\boldsymbol{\xi})$ for all $l = 1, \ldots, L$ and $z_{kl} = 1$ for $l \in \arg\max_{l \in \{1,\ldots,L\}} \left\{\boldsymbol{t}_l(\boldsymbol{\xi})^\top \boldsymbol{x} + \boldsymbol{w}_l(\boldsymbol{\xi})^\top \boldsymbol{y}_k - h_l(\boldsymbol{\xi})\right\}$ otherwise (where ties can be broken arbitrarily). We thus conclude that $\mathcal{S}(\theta, \boldsymbol{x}, \boldsymbol{y})$ is less than or equal to the optimal value of problem (8). Likewise, every feasible solution $(\zeta, \boldsymbol{\xi}, \boldsymbol{z})$ to problem (8) satisfies $\zeta \le \max\left\{\boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y}_k - \theta, \ \max_{l \in \{1,\ldots,L\}} \{\boldsymbol{t}_l(\boldsymbol{\xi})^\top \boldsymbol{x} + \boldsymbol{w}_l(\boldsymbol{\xi})^\top \boldsymbol{y}_k - h_l(\boldsymbol{\xi})\}\right\}$ for all $k \in \mathcal{K}$; that is, $\zeta \le S(\theta, \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\xi})$. Thus, the optimal value of problem (8) is less than or equal to $\mathcal{S}(\theta, \boldsymbol{x}, \boldsymbol{y})$ as well.

If the number of policies K is fixed and the uncertainty set Ξ is convex, then problem (8) can be solved by enumerating all $(L+1)^K$ possible choices for $\boldsymbol{z}$, solving the resulting linear programs in ζ and $\boldsymbol{\xi}$ and reporting the solution with the maximum value of ζ. □

The Algorithm Our solution scheme solves a sequence of scenario-based K-adaptability problems (6) over monotonically increasing scenario sets Ξ_k, $k \in \mathcal{K}$. At each iteration, the separation problem (8) identifies a new scenario $\boldsymbol{\xi} \in \Xi$ to be added to these sets.

1. *Initialize* Set $\mathcal{N} \leftarrow \{\tau^0\}$ (node set), where $\tau^0 = (\Xi_1^0, \ldots, \Xi_K^0)$ with $\Xi_k^0 = \emptyset$ for all $k \in \mathcal{K}$ (root node). Set $(\theta^{\mathrm{i}}, \boldsymbol{x}^{\mathrm{i}}, \boldsymbol{y}^{\mathrm{i}}) \leftarrow (+\infty, \emptyset, \emptyset)$ (incumbent solution).
2. *Check convergence* If $\mathcal{N} = \emptyset$, then stop and declare infeasibility (if $\theta^{\mathrm{i}} = +\infty$) or report $(\boldsymbol{x}^{\mathrm{i}}, \boldsymbol{y}^{\mathrm{i}})$ as an optimal solution to problem (2).

3. *Select node* Select a node $\tau = (\Xi_1, \ldots, \Xi_K)$ from $\mathcal{N}$. Set $\mathcal{N} \leftarrow \mathcal{N} \backslash \{\tau\}$.
4. *Process node* Let $(\theta, \boldsymbol{x}, \boldsymbol{y})$ be an optimal solution to the scenario-based K-adaptability problem (6) If $\theta \geq \theta^{\mathrm{i}}$, then go to Step 2..
5. *Check feasibility* Let $(\zeta, \boldsymbol{\xi}, \boldsymbol{z})$ be an optimal solution to the separation problem (8). If $\zeta \leq 0$, then set $(\theta^{\mathrm{i}}, \boldsymbol{x}^{\mathrm{i}}, \boldsymbol{y}^{\mathrm{i}}) \leftarrow (\theta, \boldsymbol{x}, \boldsymbol{y})$ and go to Step 2.; otherwise, if $\zeta > 0$, go to Step 6..
6. *Branch* Instantiate K new nodes $\tau_1, \ldots, \tau_K$ as follows: $\tau_k = (\Xi_1, \ldots, \Xi_k \cup \{\boldsymbol{\xi}\}, \ldots, \Xi_K)$ for each $k \in \mathcal{K}$. Set $\mathcal{N} \leftarrow \mathcal{N} \cup \{\tau_1, \ldots, \tau_K\}$ and go to Step 3..

Our branch-and-bound algorithm can be interpreted as an uncertainty set partitioning scheme. For a solution $(\theta, \boldsymbol{x}, \boldsymbol{y})$ in Step 4., the sets

$$\Xi(\theta, \boldsymbol{x}, \boldsymbol{y}_k) = \left\{ \boldsymbol{\xi} \in \Xi : \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y}_k \leq \theta,\ \boldsymbol{T}(\boldsymbol{\xi})\boldsymbol{x} + \boldsymbol{W}(\boldsymbol{\xi})\boldsymbol{y}_k \leq \boldsymbol{h}(\boldsymbol{\xi}) \right\}, \quad k \in \mathcal{K},$$

describe the regions of the uncertainty set Ξ for which at least one of the candidate policies is feasible and results in an objective value smaller than or equal to θ. Step 5. of the algorithm attempts to identify a realization $\boldsymbol{\xi} \in \Xi \setminus \bigcup_{k \in \mathcal{K}} \Xi(\theta, \boldsymbol{x}, \boldsymbol{y}_k)$ for which every candidate policy either is infeasible or results in an objective value that exceeds θ. If there is no such realization, then the solution $(\boldsymbol{x}, \boldsymbol{y})$ is feasible in the K-adaptability problem (2). Otherwise, Step 6. assigns the realization $\boldsymbol{\xi}$ to each scenario subset Ξ_k, $k \in \mathcal{K}$, in turn. Figure 3 illustrates our solution scheme.

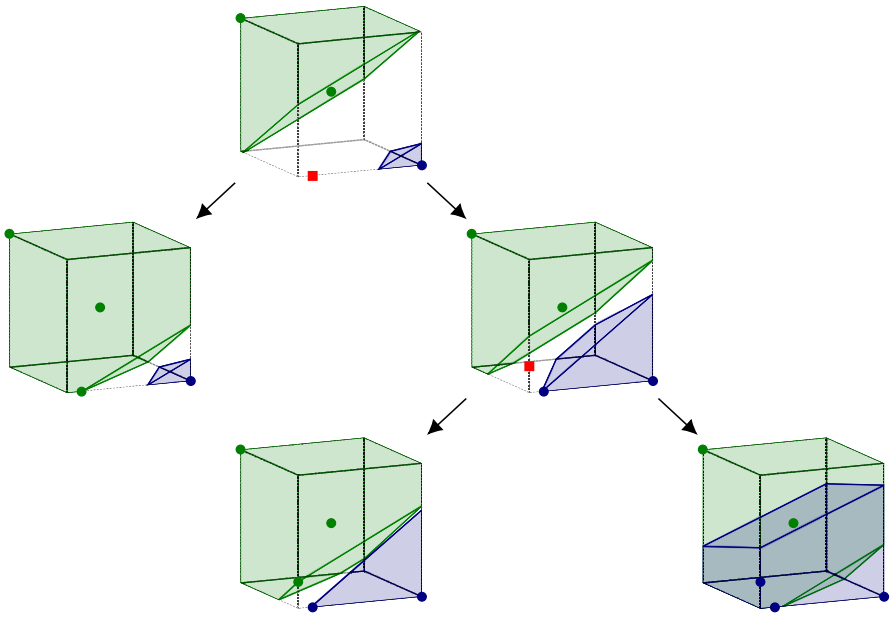

Fig. 3 An illustrative example with $K = 2$ policies. Each cube represents the uncertainty set Ξ while the shaded regions represent $\Xi(\theta, \boldsymbol{x}, \boldsymbol{y}_1)$ and $\Xi(\theta, \boldsymbol{x}, \boldsymbol{y}_2)$. The green and blue dots represent elements of the sets Ξ_1 and Ξ_2, respectively, while the red squares represent the candidate realizations $\boldsymbol{\xi}$ identified in Step 5. of the algorithm (color figure online)

3.2 Convergence analysis

We now establish the correctness of our branch-and-bound scheme, as well as conditions for its asymptotic and finite convergence.

Theorem 1 (Correctness) *If the branch-and-bound scheme terminates, then it either returns an optimal solution to problem* (2) *or correctly identifies the latter as infeasible.*

Proof We first show that if the branch-and-bound scheme terminates on an infeasible problem instance, then the final incumbent solution is $(\theta^{\mathrm{i}}, \boldsymbol{x}^{\mathrm{i}}, \boldsymbol{y}^{\mathrm{i}}) = (+\infty, \emptyset, \emptyset)$. Indeed, the algorithm can only update the incumbent in Step 5. if the objective value of the separation problem is non-positive. By construction, this is only possible if the algorithm has determined a feasible solution.

We now show that if the branch-and-bound scheme terminates on a feasible problem instance, then the algorithm returns an optimal solution $(\boldsymbol{x}^{\mathrm{i}}, \boldsymbol{y}^{\mathrm{i}})$ of problem (2). To this end, assume that $(\boldsymbol{x}^\star, \boldsymbol{y}^\star)$ is an optimal solution of problem (2) with objective value $\theta^\star$. Let $\mathcal{T}$ be the set of all nodes of the branch-and-bound tree for which $(\theta^\star, \boldsymbol{x}^\star, \boldsymbol{y}^\star)$ is feasible in the corresponding scenario-based K-adaptability problem (6). Note that $\mathcal{T} \neq \emptyset$ since $(\theta^\star, \boldsymbol{x}^\star, \boldsymbol{y}^\star)$ is feasible in the root node. Let $\mathcal{T}' \subseteq \mathcal{T}$ be the set of those nodes which have children in $\mathcal{T}$, that is, nodes that have been branched in Step 6 of our algorithm. Consider the set $\mathcal{T}'' = \mathcal{T} \backslash \mathcal{T}'$. Since the branch-and-bound scheme has terminated, we must have $\mathcal{T}'' \neq \emptyset$. Consider an arbitrary node $\tau \in \mathcal{T}''$. Since τ has been selected in Step 3. but not been branched in Step 6., either (i) τ has been fathomed in Step 4. or if (ii) τ has been fathomed in Step 5.. In the former case, the solution $(\theta^\star, \boldsymbol{x}^\star, \boldsymbol{y}^\star)$ must have been weakly dominated by the incumbent solution $(\theta^{\mathrm{i}}, \boldsymbol{x}^{\mathrm{i}}, \boldsymbol{y}^{\mathrm{i}})$, which therefore must be optimal as well. In the latter case, the incumbent solution must have been updated to $(\theta^\star, \boldsymbol{x}^\star, \boldsymbol{y}^\star)$. □

We now show that our branch-and-bound scheme converges asymptotically to an optimal solution of the K-adaptability problem (2). Our result has two implications: (i) for infeasible problem instances, the algorithm always terminates after finitely many iterations, i.e., infeasibility is detected in finite time; (ii) for feasible problem instances, the algorithm eventually only inspects solutions in the neighborhood of optimal solutions.

Theorem 2 (Asymptotic Convergence) *Every accumulation point* $(\hat{\theta}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ *of the solutions to the scenario-based* K*-adaptability problem* (6) *in an infinite branch of the branch-and-bound tree gives rise to an optimal solution* $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ *of the* K*-adaptability problem* (2) *with objective value* $\hat{\theta}$.

Proof We denote by $(\theta^\ell, \boldsymbol{x}^\ell, \boldsymbol{y}^\ell)$ and $(\zeta^\ell, \boldsymbol{\xi}^\ell, \boldsymbol{z}^\ell)$ the sequences of optimal solutions to the scenario-based K-adaptability problem in Step 4. and the separation problem in Step 5. of the algorithm, respectively, that correspond to the node sequence τ^ℓ, $\ell = 0, 1, \ldots$, of some infinite branch of the branch-and-bound tree. Since $\mathcal{X}$, $\mathcal{Y}$ and Ξ are compact, the Bolzano–Weierstrass theorem implies that $(\theta^\ell, \boldsymbol{x}^\ell, \boldsymbol{y}^\ell)$ and $(\zeta^\ell, \boldsymbol{\xi}^\ell, \boldsymbol{z}^\ell)$ each have at least one accumulation point.

We first show that every accumulation point $(\hat{\theta}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ of the sequence $(\theta^\ell, \boldsymbol{x}^\ell, \boldsymbol{y}^\ell)$ corresponds to a *feasible* solution $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ of the K-adaptability problem (2) with objective value $\hat{\theta}$. By possibly going over to subsequences, we can without loss of generality

assume that the two sequences $(\theta^\ell, \boldsymbol{x}^\ell, \boldsymbol{y}^\ell)$ and $(\zeta^\ell, \boldsymbol{\xi}^\ell, z^\ell)$ converge themselves to $(\hat{\theta}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ and $(\hat{\zeta}, \hat{\boldsymbol{\xi}}, \hat{z})$, respectively. Assume now that $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ does *not* correspond to a feasible solution of the K-adaptability problem (2) with objective value $\hat{\theta}$. Then there is $\boldsymbol{\xi}^\star \in \Xi$ such that

$S(\hat{\theta}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \boldsymbol{\xi}^\star) \geq \delta$ for some $\delta > 0$. By construction of the separation problem (8), this implies that

$$S(\theta^\ell, \boldsymbol{x}^\ell, \boldsymbol{y}^\ell, \boldsymbol{\xi}^\ell) = \max_{\boldsymbol{\xi} \in \Xi} S(\theta^\ell, \boldsymbol{x}^\ell, \boldsymbol{y}^\ell, \boldsymbol{\xi}) \geq S(\theta^\ell, \boldsymbol{x}^\ell, \boldsymbol{y}^\ell, \boldsymbol{\xi}^\star) \geq \delta/2$$

for all ℓ sufficiently large. By taking limits and exploiting the continuity of S, we conclude that

$$S(\hat{\theta}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{\xi}}) \geq S(\hat{\theta}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \boldsymbol{\xi}^\star) \geq \delta/2.$$

Note, however, that $S(\theta^{\ell+1}, \boldsymbol{x}^{\ell+1}, \boldsymbol{y}^{\ell+1}, \boldsymbol{\xi}^\ell) \leq 0$ since $\boldsymbol{\xi}^\ell \in \Xi_k^{\ell+1}$ for some $k \in \mathcal{K}$. Since the sequence $(\theta^{\ell+1}, \boldsymbol{x}^{\ell+1}, \boldsymbol{y}^{\ell+1})$ also converges to $(\hat{\theta}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ and $\boldsymbol{\xi}^\ell$ converges to $\hat{\boldsymbol{\xi}}$, we thus conclude that $S(\hat{\theta}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{\xi}}) \leq 0$, which yields the desired contradiction.

We now show that every accumulation point $(\hat{\theta}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ of the sequence $(\theta^\ell, \boldsymbol{x}^\ell, \boldsymbol{y}^\ell)$ corresponds to an *optimal* solution $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ of the K-adaptability problem (2) with objective value $\hat{\theta}$. Assume to the contrary that $(\hat{\theta}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ is feasible but suboptimal, that is, there is a feasible solution $(\boldsymbol{x}', \boldsymbol{y}')$ to the K-adaptability problem (2) with objective value $\theta' < \hat{\theta}$. Let $\mathcal{T}$ be the set of all nodes of the branch-and-bound tree for which $(\theta', \boldsymbol{x}', \boldsymbol{y}')$ is feasible in the corresponding scenario-based K-adaptability problem (6). Note that $\mathcal{T} \neq \emptyset$ since $(\theta', \boldsymbol{x}', \boldsymbol{y}')$ is feasible in the root node. In the following, we distinguish the two cases (i) $|\mathcal{T}| < \infty$ and (ii) $|\mathcal{T}| = \infty$.

If $|\mathcal{T}| < \infty$, then $\mathcal{T}$ must contain nodes that were fathomed in Steps 4 or 5 of the algorithm. In that case, however, the incumbent solution must have been updated to $(\tilde{\theta}, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}})$ with $\tilde{\theta} \leq \theta'$ after finitely many iterations. Since the objective values θ^ℓ of the scenario-based K-adaptability problems will be arbitrarily close to $\hat{\theta}$ for ℓ sufficiently large, the corresponding nodes τ^ℓ will also be fathomed in Step 4.. This contradicts the assumption that the node sequence τ^ℓ corresponds to an infinite branch of the branch-and-bound tree.

If $|\mathcal{T}| = \infty$, on the other hand, the compactness of $\mathcal{X}$, $\mathcal{Y}$ and Ξ implies that $(\theta', \boldsymbol{x}', \boldsymbol{y}')$ constitutes the accumulation point of another infinite sequence $(\theta'^{,\ell}, \boldsymbol{x}'^{,\ell}, \boldsymbol{y}'^{,\ell})$. In that case, too, the objective values θ^ℓ and $\theta'^{,\ell}$ of the scenario-based K-adaptability problems will be arbitrarily close to $\hat{\theta}$ and θ', respectively, for ℓ sufficiently large. Since $\theta' < \hat{\theta}$, the algorithm will fathom the tree nodes corresponding to the sequence $(\theta^\ell, \boldsymbol{x}^\ell, \boldsymbol{y}^\ell)$ in Step 4.. This again contradicts the assumption that the node sequence τ^ℓ corresponds to an infinite branch of the branch-and-bound tree.

Since both cases $|\mathcal{T}| < \infty$ and $|\mathcal{T}| = \infty$ lead to contradictions, we conclude that the assumed suboptimality of $(\hat{\theta}, \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})$ is wrong. The statement of the theorem thus follows. □

Theorem 2 guarantees that after sufficiently many iterations of the algorithm, our scheme generates feasible solutions that are close to an optimal solution of the K-adaptability problem (2). In general, our algorithm may not converge after finitely

many iterations. In the following, we discuss a class of problem instances for which finite convergence is guaranteed.

Theorem 3 (Finite Convergence) *The branch-and-bound scheme terminates after finitely many iterations, if $\mathcal{Y}$ has finite cardinality and only the objective function in problem (2) is uncertain.*

Proof If only the objective function in the K-adaptability problem (2) is uncertain, then the corresponding semi-infinite disjunctive program (5) can be written as

$$\begin{array}{lll} \text{minimize} & \theta & \\ \text{subject to} & \theta \in \mathbb{R}, \quad \boldsymbol{x} \in \mathcal{X}, \quad \boldsymbol{y} \in \mathcal{Y}^K & \\ & \boldsymbol{T}\boldsymbol{x} + \boldsymbol{W}\boldsymbol{y}_k \leq \boldsymbol{h} & \forall k \in \mathcal{K} \\ & \bigvee\limits_{k \in \mathcal{K}} \left[\boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y}_k \leq \theta \right] & \forall \boldsymbol{\xi} \in \Xi, \end{array}$$

see Observation 2. Thus, the scenario-based K-adaptability problem (6) becomes

$$\begin{array}{rlll} \mathcal{M}(\Xi_1, \ldots, \Xi_K) = & \text{minimize} & \theta & \\ & \text{subject to} & \theta \in \mathbb{R}, \quad \boldsymbol{x} \in \mathcal{X}, \quad \boldsymbol{y} \in \mathcal{Y}^K & \\ & & \boldsymbol{T}\boldsymbol{x} + \boldsymbol{W}\boldsymbol{y}_k \leq \boldsymbol{h} & \forall k \in \mathcal{K} \\ & & \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y}_k \leq \theta & \forall \boldsymbol{\xi} \in \Xi_k, \ \forall k \in \mathcal{K}, \end{array}$$

and the separation problem (7) can be written as

$$\begin{aligned} \mathcal{S}(\theta, \boldsymbol{x}, \boldsymbol{y}) &= \max_{\boldsymbol{\xi} \in \Xi} \min_{k \in \mathcal{K}} \left\{ \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y}_k - \theta \right\} \\ &= \boldsymbol{c}^\top \boldsymbol{x} - \theta + \max_{\boldsymbol{\xi} \in \Xi} \min_{k \in \mathcal{K}} \left\{ \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y}_k \right\}. \end{aligned}$$

We now show that if $\mathcal{Y}$ has finite cardinality, then our branch-and-bound algorithm terminates after finitely many iterations. To this end, assume that this is not the case, and let τ^ℓ, $\ell = 0, 1, \ldots$ be some rooted branch of the tree with infinite length. We denote by $(\theta^\ell, \boldsymbol{x}^\ell, \boldsymbol{y}^\ell)$ and $(\zeta^\ell, \boldsymbol{\xi}^\ell, \boldsymbol{z}^\ell)$ the corresponding sequences of optimal solutions to the master and the separation problem, respectively. Since $\mathcal{Y}$ has finite cardinality, we must have $\boldsymbol{y}^{\ell_1} = \boldsymbol{y}^{\ell_2}$ for some $\ell_1 < \ell_2$.

The solution $(\theta^{\ell_2}, \boldsymbol{x}^{\ell_2}, \boldsymbol{y}^{\ell_2})$ satisfies $\mathcal{S}(\theta^{\ell_2}, \boldsymbol{x}^{\ell_2}, \boldsymbol{y}^{\ell_2}) > 0$ since τ^ℓ, $\ell = 0, 1, \ldots$, is a branch of infinite length. Since $\boldsymbol{y}^{\ell_2} = \boldsymbol{y}^{\ell_1}$, we thus conclude that

$$\boldsymbol{c}^\top \boldsymbol{x}^{\ell_2} - \theta^{\ell_2} + \max_{\boldsymbol{\xi} \in \Xi} \min_{k \in \mathcal{K}} \left\{ \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y}_k^{\ell_1} \right\} > 0.$$

Since $\boldsymbol{\xi}^{\ell_1}$ is optimal in $\mathcal{S}(\theta^{\ell_1}, \boldsymbol{x}^{\ell_1}, \boldsymbol{y}^{\ell_1})$ and $S(\theta^{\ell_1}, \boldsymbol{x}^{\ell_1}, \boldsymbol{y}^{\ell_1}, \boldsymbol{\xi}^{\ell_1}) > 0$, we have

$$\boldsymbol{c}^\top \boldsymbol{x}^{\ell_2} - \theta^{\ell_2} + \min_{k \in \mathcal{K}} \left\{ \boldsymbol{d}(\boldsymbol{\xi}^{\ell_1})^\top \boldsymbol{y}_k^{\ell_1} \right\} = \boldsymbol{c}^\top \boldsymbol{x}^{\ell_2} - \theta^{\ell_2} + \max_{\boldsymbol{\xi} \in \Xi} \min_{k \in \mathcal{K}} \left\{ \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y}_k^{\ell_1} \right\} > 0.$$

However, since the node $\tau^{\ell_2} = (\Xi_1^{\ell_2}, \ldots, \Xi_K^{\ell_2})$ is a descendant of the node $\tau^{\ell_1} = (\Xi_1^{\ell_1}, \ldots, \Xi_K^{\ell_1})$, we must have $\boldsymbol{\xi}^{\ell_1} \in \Xi_k^{\ell_2}$ for some $k \in \mathcal{K}$. This, along with the fact that $(\theta^{\ell_2}, \boldsymbol{x}^{\ell_2}, \boldsymbol{y}^{\ell_2})$ is a feasible solution to the master problem $\mathcal{M}(\Xi_1^{\ell_2}, \ldots, \Xi_K^{\ell_2})$ and that $\boldsymbol{y}^{\ell_2} = \boldsymbol{y}^{\ell_1}$, implies that

$$\boldsymbol{c}^\top \boldsymbol{x}^{\ell_2} - \theta^{\ell_2} + \min_{k \in \mathcal{K}} \left\{ \boldsymbol{d}(\boldsymbol{\xi}^{\ell_1})^\top \boldsymbol{y}_k^{\ell_1} \right\} \leq 0.$$

This yields the desired contradiction and proves the theorem. □

We note that the assumption of deterministic constraints is critical in the previous statement.

Example 2 Consider the following instance of the K-adaptability problem (2):

$$\inf_{y_1, y_2 \in \{0,1\}} \sup_{\xi \in [0,1]} \inf_{k \in \{1,2\}} \{(\xi - 1)(1 - 2y_k) \,:\, y_k \geq \xi\}$$

On this instance, our branch-and-bound algorithm generates a tree in which all branches have finite length, except (up to permutations) the sequence of nodes $\tau^\ell = (\Xi_1^\ell, \Xi_2^\ell)$, where $(\Xi_1^0, \Xi_2^0) = (\emptyset, \emptyset)$ and $(\Xi_1^\ell, \Xi_2^\ell) = \left(\left\{\xi^0 2^{-i} : i = 0, 1, \ldots, \ell - 1\right\}, \{0\}\right)$, $\ell > 0$, for some $\xi^0 \in (0, 1]$. For the node τ^ℓ, $\ell > 1$, the optimal solution of the scenario-based K-adaptability problem (6) is $(\theta^\ell, y_1^\ell, y_2^\ell) = (1 - \xi^0 2^{-\ell+1}, 1, 0)$, while the optimal solution of the separation problem is $(\zeta^\ell, \xi^\ell) = (\xi^0 2^{-\ell}, \xi^0 2^{-\ell})$. Thus, our branch-and-bound algorithm does not terminate after finitely many iterations.

We note that every practical implementation of our branch-and-bound scheme will fathom nodes in Step 5. whenever the objective value of the separation problem (7) is sufficiently close to zero (within some ϵ-tolerance). This ensures that the algorithm terminates in finite time in practice. Indeed, in Example 2 the objective value of the separation problem is less than ϵ for all nodes τ^ℓ with $\ell \geq \log_2(\xi^0 \epsilon^{-1})$, and our branch-and-bound algorithm will fathom the corresponding path of the tree after $\mathcal{O}(\log \epsilon^{-1})$ iterations if we seek ϵ-precision solutions.

3.3 Improvements to the basic algorithm

The algorithm of Sect. 3.1 serves as a blueprint that can be extended in multiple ways. In the following, we discuss three enhancements that improve the numerical performance of our algorithm.

Breaking symmetry For any feasible solution $(\boldsymbol{x}, \boldsymbol{y})$ of the K-adaptability problem (2), every solution $(\boldsymbol{x}, \boldsymbol{y}')$, where $\boldsymbol{y}'$ is one of the $K!$ permutations of the second-stage policies $(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_K)$, is also feasible in (2) and attains the same objective value. This implies that our branch-and-bound tree is highly isomorphic since the scenario-based problems (6) and (8) are identical (up to a permutation of the policies) across many nodes. We can reduce this undesirable symmetry by modifying Step 6. of our branch-and-bound scheme as follows:

6.′. *Branch* Let $K' = 1$ if $\Xi_1 = \ldots = \Xi_K = \emptyset$ and let $K' = \min\left\{K,\ 1+\max_{k\in\mathcal{K}}\{k : \Xi_k \neq \emptyset\}\right\}$ otherwise. Instantiate K' new nodes $\tau_k = (\Xi_1, \ldots, \Xi_k \cup \{\boldsymbol{\xi}\}, \ldots, \Xi_K)$, $k = 1, \ldots, K'$. Set $\mathcal{N} \leftarrow \mathcal{N} \cup \{\tau_1, \ldots, \tau_{K'}\}$ and go to Step 3..

The modification above is effective only when there are empty scenario sets, *i.e.*, $\cup_{k\in\mathcal{K}'} \Xi_k = \emptyset$ for some $\mathcal{K}' \subseteq \mathcal{K}$. Nevertheless, it can be shown that empty scenario sets can arise at any tree depth, and the modification causes a significant reduction in the number of generated nodes. Despite generating only a subset of the nodes that our original algorithm constructs, however, the modification above always maintains at least one of the $K!$ solutions symmetric to every feasible solution.

Integration into MILP Solvers. Step 4. of our algorithm solves the scenario-based problem (6) from scratch in every node, despite the fact that two successive problems along any branch of the branch-and-bound tree differ only by the addition of a few constraints. We can leverage this commonality if we integrate our branch-and-bound algorithm into the solution scheme of the MILP solver used for problem (6). In doing so, we can also exploit the advanced facilities commonly present in the state-of-the-art solvers such as warm-starts and cutting planes, among others.

In order to integrate our branch-and-bound algorithm into the solution scheme of the MILP solver, we initialize the solver with the scenario-based problem (6) corresponding to the root node τ^0 of our algorithm, see Step 1.. The solver then proceeds to solve this problem using its own branch-and-bound procedure. Whenever the solver encounters an *integral solution* $(\theta, \boldsymbol{x}, \boldsymbol{y}) \in \mathbb{R} \times \mathcal{X} \times \mathcal{Y}^K$, we solve the associated separation problem (8). If $\mathcal{S}(\theta, \boldsymbol{x}, \boldsymbol{y}) > 0$, then we execute Step 6. of our algorithm through a *branch callback*: we report the K new branches to the solver, which will discard the current solution. Otherwise, if $\mathcal{S}(\theta, \boldsymbol{x}, \boldsymbol{y}) \leq 0$, then we do not create any new branches, and the solver will accept $(\theta, \boldsymbol{x}, \boldsymbol{y})$ as the new incumbent solution. This ensures that only those solutions which are feasible in problem (5) are accepted as incumbent solutions, and furthermore, that all nodes of our branch-and-bound scheme (not just the root node) are fully explored within the same search tree generated by the MILP solver for solving problem (6) corresponding to the root node τ^0 of our algorithm.

Whenever the solver encounters a *fractional solution*, it will by default branch on an integer variable that is fractional in the current solution. However, if the fractional solution also happens to satisfy $\mathcal{S}(\theta, \boldsymbol{x}, \boldsymbol{y}) > 0$, then it is possible to override this default strategy and instead execute Step 6. of our algorithm. In such cases, a heuristic rule can be used to decide whether to branch on integer variables or to branch as in Step 6.. In our computational experience, a simple rule that alternates between the default branching rule of the solver and the one defined by Step 6. appears to perform well in practice.

3.4 Modification as a heuristic algorithm

Whenever the number of policies K is large, the solution of the scenario-based K-adaptability problem (6) can be time consuming. In such cases, only a limited number of nodes will be explored by the algorithm in a given amount of computation time, and

the quality of the final incumbent solution may be poor. As a remedy, we can reduce the size and complexity of the scenario-based K-adaptability problem (6) by fixing some of its second-stage policies and limiting the total computation time. In doing so, we obtain a heuristic variant of our algorithm that can scale to large values of K.

In our computational experience, a simple heuristic that sequentially solves the 1-, 2-, …, K-adaptability problems by fixing in each K-adaptability problem all but one of the second-stage policies, $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{K-1}$, to their corresponding values in the $(K-1)$-adaptability problem, performs well in practice. This heuristic is motivated by two observations. First, the resulting scenario-based K-adaptability problems (6) have the same size and complexity as the corresponding scenario-based 1-adaptability problems. Second, in our experiments on instances with uncertain objective coefficients $\boldsymbol{d}$, we often found that some optimal second-stage policies of the $(K-1)$-adaptability problem also appear in the optimal solution of the K-adaptability problem. In fact, it can be shown that this heuristic can obtain K-adaptable solutions that improve upon 1-adaptable solutions only if the objective coefficients $\boldsymbol{d}$ are affected by uncertainty.

4 Numerical results

We now analyze the computational performance of our branch-and-bound scheme in a variety of problem instances from the literature. We consider a shortest path problem with uncertain arc weights (Sect. 4.1), a capital budgeting problem with uncertain cash flows (Sect. 4.2), a variant of the capital budgeting problem with the additional option to take loans (Sect. 4.3), a project management problem with uncertain task durations (Sect. 4.4), and a vehicle routing problem with uncertain travel times (Sect. 4.5). Of these, the first two problems involve only binary decisions, and they can therefore also be solved with the approach described in [24]. In these cases, we show that our solution scheme is highly competitive, and it frequently outperforms the approach of [24]. In contrast, the third and fourth problems also involve continuous decisions, and there is no existing solution approach for their associated K-adaptability problems. However, the project management problem from Sect. 4.4 involves *only* continuous second-stage decisions, and therefore the corresponding two-stage robust optimization problem (1) can also be approximated using affine decision rules [3], which represent the most popular approach for such problems. In this case, we elucidate the benefits of K-adaptable constant and affine decisions over standard affine decision rules. Finally, the first and last problems involve only binary second-stage decisions and deterministic constraints, and they can therefore also be addressed with the heuristic approach described in [13]. In these cases, we show that the heuristic variant of our algorithm often outperforms the latter approach in terms of solution quality.

For each problem category, we investigate the tradeoffs between computational effort and improvement in objective value of the K-adaptability problem for increasing values of K. We demonstrate that (i) the K-adaptability problem can provide significant improvements over static robust optimization (which corresponds to the case $K = 1$), and that (ii) our solution scheme can quickly determine feasible solutions of high quality.

We implemented our branch-and-bound algorithm in C++ using the C callable library of CPLEX 12.6 [25]. We note that CPLEX does not directly support the creation of more than two child nodes during branching. However, as the branching Step 6′ may create up to K child nodes, we use a binary representation of the underlying K-ary tree (see, e.g., [32]), and implement the branching step using a combination of *incumbent* and *branch* callbacks. By using callback functions in this manner, we can integrate our algorithm within CPLEX in a modular fashion while still leveraging advanced MILP technology such as cutting planes, backtracking rules and other heuristics. We caution, however, that the creation of up to K child nodes can result in very large search trees, and potentially cause memory issues. Therefore, to limit the size of our search trees, we use a CPLEX-imposed memory limit of 16GB RAM. We used a constraint feasibility tolerance of $\epsilon = 10^{-4}$ to accept any incumbent solutions, whereas all other solver options were kept at their default values. Unless mentioned otherwise, we employ a time limit of 7200 s for our branch-and-bound scheme and a time limit of 60 s for the heuristic variant of our algorithm. All experiments were conducted on a single core of an Intel Xeon 2.8 GHz computer.

4.1 Shortest paths

We consider the shortest path problem from [24]. Let $G = (V, A)$ be a directed graph with nodes $V = \{1, \ldots, N\}$, arcs $A \subseteq V \times V$ and arc weights $d_{ij}(\boldsymbol{\xi}) = (1+\xi_{ij}/2)d^0_{ij}$, $(i, j) \in A$. Here, $d^0_{ij} \in \mathbb{R}_+$ represents the nominal weight of the arc $(i, j) \in A$ and ξ_{ij} denotes the uncertain deviation from the nominal weight. The realizations of the uncertain vector $\boldsymbol{\xi}$ are known to belong to the set

$$\Xi = \left\{ \boldsymbol{\xi} \in [0, 1]^{|A|} : \sum_{(i,j)\in A} \xi_{ij} \leq \Gamma \right\},$$

which stipulates that at most Γ arc weights may maximally deviate from their nominal values.

Let $s \in V$ and $t \in V, s \neq t$, denote the source and terminal nodes of G, respectively. The decision-maker aims to choose K paths from s to t here-and-now, i.e., before observing the actual arc weights, such that the worst-case weight of the shortest among the chosen paths is minimized. This problem can be formulated as an instance of the K-adaptability problem (2):

$$\inf_{\boldsymbol{y}\in\mathcal{Y}^K} \sup_{\boldsymbol{\xi}\in\Xi} \inf_{k\in\mathcal{K}} \boldsymbol{d}(\boldsymbol{\xi})^\top \boldsymbol{y}_k$$

Here, $\mathcal{Y}$ denotes the set of all $s - t$ paths in G; that is,

$$\mathcal{Y} = \left\{ \boldsymbol{y} \in \{0, 1\}^{|A|} : \sum_{(j,l)\in A} y_{jl} - \sum_{(i,j)\in A} y_{ij} \geq \mathbb{I}[j = s] - \mathbb{I}[j = t] \;\; \forall\, j \in V \right\}.$$

Table 2 Results for the shortest path problem

N	$K=2$			$K=3$			$K=4$		
	Opt (#)	Time (s)	Gap (%)	Opt (#)	Time (s)	Gap (%)	Opt (#)	Time (s)	Gap (%)
20	99	6	1.23	97	408	2.51	70	539	1.74
25	91	222	4.14	64	847	2.91	33	885	2.89
30	64	744	4.40	31	1237	4.10	16	827	4.27
35	37	1083	5.36	14	1020	5.01	10	896	5.23
40	10	808	6.28	6	1670	6.43	2	39	6.10
45	9	1152	7.70	1	16	7.06	1	15	6.61
50	2	3307	8.55	1	2308	7.90	0	–	7.10

For each value of K, the "Opt (#)" column reports the number of instances (out of 100) which were solved to optimality, while the "Time (s)" column reports the average time to solve these instances to optimality. For those instances which could not be solved to optimality within the time limit of 7200 s, the average gap $|(\theta^{\mathrm{i}} - \mathrm{lb})/\theta^{\mathrm{i}}| \times 100\%$ between the global lower bound (lb) and incumbent θ^{i} of the branch-and-bound tree is reported in the "Gap (%)" column

Note that this problem only contains second-stage decisions and as such, the corresponding two-stage robust optimization problem (1) may be of limited interest in practice. Nevertheless, the K-adaptability problem (2) has important applications in logistics and disaster relief [24].

For each graph size $N \in \{20, 25, \ldots, 50\}$, we randomly generate 100 problem instances as follows. We assign the coordinates $(u_i, v_i) \in \mathbb{R}^2$ to each node $i \in V$ uniformly at random from the square $[0, 10]^2$. The nominal weight of the arc $(i, j) \in A$ is defined to be the Euclidean distance between the nodes i and j; that is, $d^0_{ij} = \sqrt{(u_i - u_j)^2 + (v_i - v_j)^2}$. The source node s and the terminal node t are defined to be the nodes with the maximum Euclidean distance between them. The arc set A is obtained by removing from the set of all pairwise links the $\lfloor 0.7(N^2 - N) \rfloor$ connections with the largest nominal weights. We set the uncertainty budget to $\Gamma = 3$. Further details on the parameter settings can be found in [24].

Table 2 summarizes the numerical performance of our branch-and-bound scheme for $K \in \{2, 3, 4\}$. Table 2 indicates that our scheme is able to reliably compute optimal solutions for small values of N and K, while the average optimality gap for large values of N and K is less than 9%. The numerical performance is strongly affected by the value of K; very few of the 4-adaptable instances are solved to optimality within the time limit. This decrease in tractability is partly explained in Fig. 4, which shows the improvement in objective value of the K-adaptability problem over the static problem (where $K = 1$). Figure 4a shows that the computed 4-adaptable solutions are typically of high quality since they improve upon the static solutions by as much as 13% for large values of N. Moreover, Fig. 4b shows that these solutions are obtained within 1 minute (on average), even for the largest instances. This indicates that the gaps in Table 2 are likely to be very conservative since the majority of computation time is spent on obtaining a certificate of optimality for these solutions. Finally, although not shown in Table 2, instances with $K = 1$ are solved in less than 10 s, on average. This motivates the heuristic variant of our algorithm, which we present next.

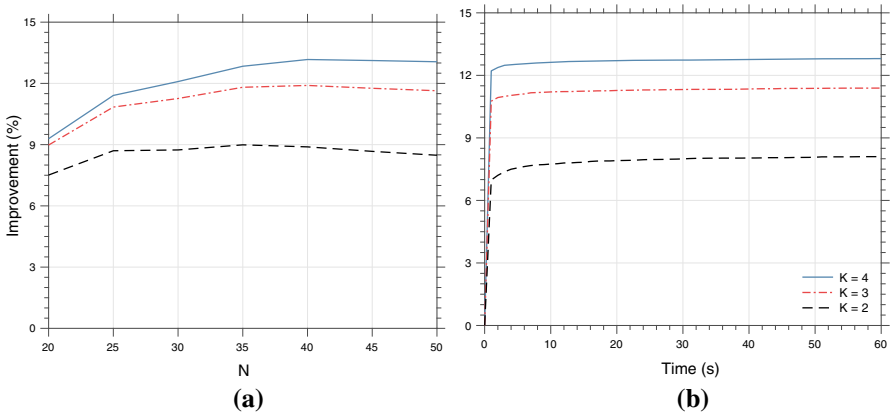

Fig. 4 Results for the shortest path problem using the exact algorithm. The graphs show the average improvement $|(\theta_1 - \theta_K)/\theta_1| \times 100\%$ of the objective value of the K-adaptability problem (θ_K) over the static problem (θ_1). The left graph shows the improvement after 2 h (for increasing N), while the right graph shows the time profile of the improvement of the incumbent solution in the first 60 s (for $N = 50$)

Figure 5 illustrates the quality of the solutions obtained using the heuristic variant of our algorithm, described in Sect. 3.3, and contrasts it with the quality of the solutions obtained using the heuristic algorithm described in [13]. We make several observations from Fig. 5. First, the 2-, 3- and 4-adaptable solutions obtained using our heuristic algorithm are about 2% better than those obtained using the heuristic algorithm described in [13], on average; the differences in the qualities of the 6-, 8- and 10-adaptable solutions are smaller. Second, when compared across instances for which the exact algorithm was able to determine an optimal solution, the corresponding 2-, 3- and 4-adaptable solutions determined using our heuristic are only 0.3% suboptimal, on average. Third, 6-adaptable heuristic solutions (obtained in less than 1 min) are already better than 4-adaptable exact solutions (obtained after 2 h), for all values of N, indicating that we can compensate for not using K optimal policies by using more than K, albeit suboptimal, policies. Fourth, the figure shows that the marginal gain in objective value decreases rapidly as we increase the number of policies K. Indeed, while the 2-adaptable solutions are about 8.3% better than the 1-adaptable (i.e., static) solutions, the 10-adaptable solutions are only about 0.1% better than the 8-adaptable solutions. This is explained by the observation that the objective values of the corresponding K-adaptable solutions are very close to the optimal value of the two-stage robust optimization problem (1), as determined using the method described in [24, Remark 1].

4.2 Capital budgeting

We consider the capital budgeting problem from [24], where a company wishes to invest in a subset of N projects. Each project i has an uncertain cost $c_i(\boldsymbol{\xi})$ and an uncertain profit $r_i(\boldsymbol{\xi})$ that are governed by factor models of the form

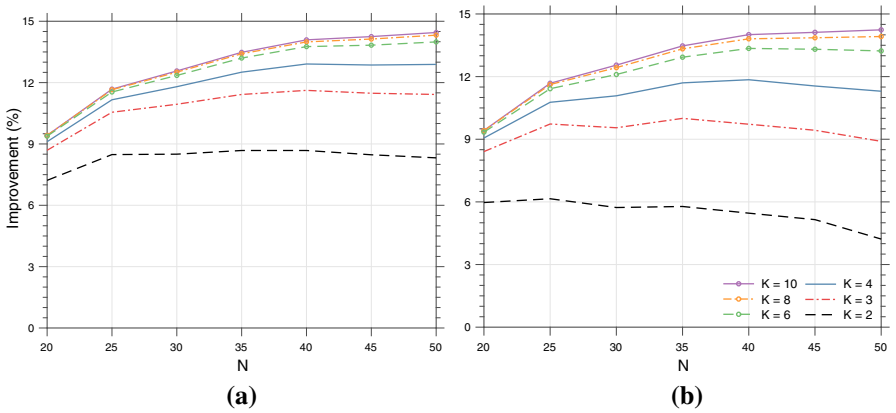

Fig. 5 Results for the shortest path problem using the heuristic algorithm. The graphs show the average improvement after 1 min obtained using the heuristic variant of our algorithm (left) and the heuristic algorithm described in [13] (right)

$$c_i(\boldsymbol{\xi}) = \left(1 + \boldsymbol{\Phi}_i^\top \boldsymbol{\xi}/2\right) c_i^0 \quad \text{and} \quad r_i(\boldsymbol{\xi}) = \left(1 + \boldsymbol{\Psi}_i^\top \boldsymbol{\xi}/2\right) r_i^0 \quad \text{for } i = 1, \ldots, N.$$

In these models, c_i^0 and r_i^0 represent the nominal cost and the nominal profit of project i, respectively, while $\boldsymbol{\Phi}_i^\top$ and $\boldsymbol{\Psi}_i^\top$ represent the ith row vectors of the factor loading matrices $\boldsymbol{\Phi}, \boldsymbol{\Psi} \in \mathbb{R}^{N \times 4}$. The realizations of the uncertain vector of risk factors $\boldsymbol{\xi}$ belong to the uncertainty set $\Xi = [-1, 1]^4$.

The company can invest in a project either before or after observing the risk factors $\boldsymbol{\xi}$. In the latter case, the company generates only a fraction κ of the profit (reflecting a penalty for postponement) but incurs the same cost as in the case of an early investment. Given an investment budget B, the capital budgeting problem can then be formulated as the following instance of the two-stage robust optimization problem (1):

$$\sup_{\boldsymbol{x} \in \mathcal{X}} \inf_{\boldsymbol{\xi} \in \Xi} \sup_{\boldsymbol{y} \in \mathcal{Y}} \left\{ \boldsymbol{r}(\boldsymbol{\xi})^\top (\boldsymbol{x} + \kappa \boldsymbol{y}) \;:\; \boldsymbol{c}(\boldsymbol{\xi})^\top (\boldsymbol{x} + \boldsymbol{y}) \leq B,\; \boldsymbol{x} + \boldsymbol{y} \leq \mathbf{e} \right\},$$

where $\mathcal{X} = \mathcal{Y} = \{0, 1\}^N$.

For our numerical experiments, we randomly generate 100 instances for each problem size $N \in \{5, 10, \ldots, 30\}$ as follows. The nominal costs $\boldsymbol{c}^0$ are chosen uniformly at random from the hyperrectangle $[0, 10]^N$. We then set $\boldsymbol{r}^0 = \boldsymbol{c}^0/5$, $B = \mathbf{e}^\top \boldsymbol{c}^0/2$ and $\kappa = 0.8$. The rows of the factor loading matrices $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ are sampled uniformly from the unit simplex in $\mathbb{R}^4$; that is, the ith row vector is sampled from $[0, 1]^4$ such that $\boldsymbol{\Phi}_i^\top \mathbf{e} = \boldsymbol{\Psi}_i^\top \mathbf{e} = 1$ is satisfied for all $i = 1, \ldots, N$.

Table 3 summarizes the numerical performance of our branch-and-bound scheme for $K \in \{2, 3, 4\}$. Table 3 demonstrates that our branch-and-bound scheme performs very well for this problem class since the majority of instances is solved to optimality for $K \in \{2, 3\}$. Moreover, the optimality gaps for the unsolved instances are less than 4% for $K \in \{2, 3\}$ and less than 9% for $K = 4$ on average. Additionally,

Table 3 Results for the capital budgeting problem

N	$K = 2$			$K = 3$			$K = 4$		
	Opt (#)	Time (s)	Gap (%)	Opt (#)	Time (s)	Gap (%)	Opt (#)	Time (s)	Gap (%)
5	100	1	–	100	1	–	100	3	–
10	100	1	–	100	16	–	100	149	–
15	100	10	–	99	566	0.33	69	2245	1.42
20	100	419	–	34	2787	1.65	5	3710	4.02
25	29	2238	1.12	4	2281	2.63	0	–	6.22
30	1	188	3.01	1	6687	3.35	0	–	8.27

The columns have the same interpretation as in Table 2

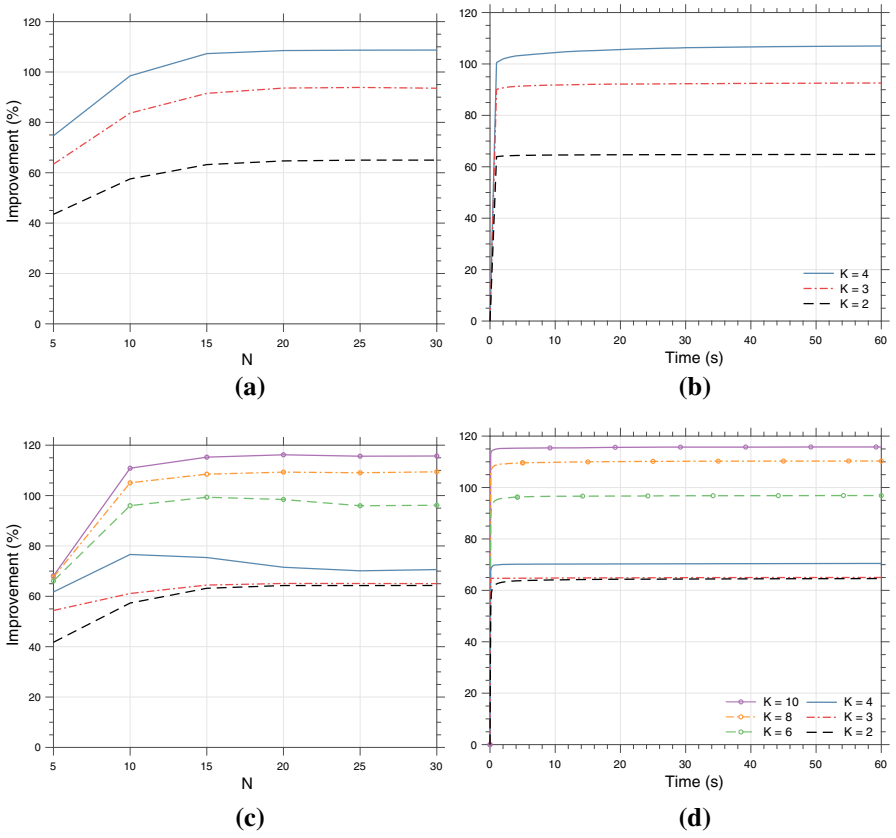

Fig. 6 Results for the capital budgeting problem. The top (bottom) graphs are for the exact (heuristic) algorithm under a time limit of 2 h (1 min). The left graphs show the average improvement at the time limit (for increasing N), while the right graphs show the time profile of the improvement of the incumbent solution in the first 60 s (for $N = 30$)

Fig. 6 shows that even for the largest instances, high-quality incumbent solutions which significantly improve ($\approx 100\%$) upon the static robust solutions are obtained within 1 minute of computation time. In obtaining the results of Fig. 6c, d using

our heuristic, we observed that limiting the computation time to 1 minute is often insufficient for the successful termination of the algorithm. Therefore, the kth second-stage policy determined by the algorithm is not always deterministic, and it can affect the solution of the $(k+1)$-adaptability problem as well as the resulting objective value improvements. To eliminate this random effect, we ran our heuristic 10 times for each instance, using different random seeds for the underlying MILP solver, and report the averaged results in Fig. 6c, d. Our results compare favorably with those of [24] as well as those of the partition-and-bound approach for the corresponding two-stage robust optimization problem presented in [6].

4.3 Capital budgeting with loans

We consider a generalization of the capital budgeting problem from Sect. 4.2 where the company can increase its investment budget by purchasing a loan from the bank at a unit cost of $\lambda > 0$ before the risk factors $\boldsymbol{\xi}$ are observed as well as purchasing a loan at a unit cost of $\mu\lambda$ (with $\mu > 1$) after the observation occurs. If the company does not purchase any loan, then the problem reduces to the one described in Sect. 4.2. Therefore, we expect the worst-case profits to be at least as large as in that setting. The generalized capital budgeting problem can be formulated as the following instance of problem (1):

$$\sup_{(x_0,\boldsymbol{x})\in\mathcal{X}} \inf_{\boldsymbol{\xi}\in\Xi} \sup_{(y_0,\boldsymbol{y})\in\mathcal{Y}} \left\{ \boldsymbol{r}(\boldsymbol{\xi})^\top(\boldsymbol{x}+\kappa\boldsymbol{y}) - \lambda(x_0+\mu y_0) \;:\; \begin{bmatrix} \boldsymbol{x}+\boldsymbol{y} \le \mathbf{e} \\ \boldsymbol{c}(\boldsymbol{\xi})^\top\boldsymbol{x} \le B + x_0 \\ \boldsymbol{c}(\boldsymbol{\xi})^\top(\boldsymbol{x}+\boldsymbol{y}) \le B + x_0 + y_0 \end{bmatrix} \right\}$$

Here, $\mathcal{X} = \mathcal{Y} = \mathbb{R}_+ \times \{0,1\}^N$. The constraint $\boldsymbol{c}(\boldsymbol{\xi})^\top\boldsymbol{x} \le B + x_0$ ensures that the first-stage expenditures $\boldsymbol{c}(\boldsymbol{\xi})^\top\boldsymbol{x}$ are fully covered by the budget B as well as the loan x_0 taken here-and-now.

We consider problems with $N \in \{5, 10, \ldots, 30\}$ projects. For each value of N, we solve the same 100 instances from Sect. 4.2 with $\lambda = 0.12$ and $\mu = 1.2$. Table 4 shows the computational performance of our branch-and-bound scheme for $K \in \{2, 3, 4\}$. As in the case of the problems discussed so far, the numerical tractability of our algorithm decreases as the value of K increases. However, a comparison of Tables 3 and 4 suggests that the numerical tractability is not significantly affected by the presence of the additional continuous variables x_0 and y_0. Indeed, the majority of instances for $K = 2$ are solved to optimality and the average gap across all unsolved instances is less than 5% for $K = 3$ and less than 9% for $K = 4$. Figure 7 shows that the 4-adaptable solutions improve upon the static solutions by as much as 115% in the largest instances. Although not shown in the figure, a comparison of the objective values of the final incumbent solutions with those of the capital budgeting problem without loans (Sect. 4.2) reveals that for $N \ge 15$, the option to purchase loans has no effect on the worst-case profit of the static solution and results in less than 1% improvement in the worst-case profit of the 2-adaptable solution. Indeed, the option

Table 4 Results for the capital budgeting problem with loans

N	$K = 2$			$K = 3$			$K = 4$		
	Opt (#)	Time (s)	Gap (%)	Opt (#)	Time (s)	Gap (%)	Opt (#)	Time (s)	Gap (%)
5	100	1	–	100	9	–	98	80	3.14
10	100	3	–	100	78	–	98	938	1.92
15	100	62	–	96	1265	0.91	23	3989	2.23
20	85	1680	0.80	20	3941	1.71	0	–	4.94
25	12	3363	2.29	1	2693	3.34	0	–	6.88
30	1	424	3.78	0	–	4.73	0	–	8.17

The columns have the same interpretation as in Table 2

to purchase loans results in significantly better worst-case profits only if $K \geq 3$. The average relative gain in objective value is 4.3% for $K = 3$ and 5.9% for $K = 4$.

4.4 Project management

We define a project as a directed acyclic graph $G = (V, A)$ whose nodes $V = \{1, \ldots, N\}$ represent the tasks (e.g., 'build foundations' or 'develop prototype') and whose arcs $A \subseteq V \times V$ denote the temporal precedences, i.e., $(i, j) \in A$ implies that task j cannot be started before task i has been completed. We assume that each task $i \in V$ has an uncertain duration $d_i(\boldsymbol{\xi})$ that depends on the realization of an uncertain parameter vector $\boldsymbol{\xi} \in \Xi$. Without loss of generality, we stipulate that the project graph G has the unique sink $N \in V$, and that the last task N has a duration of zero. This can always be achieved by introducing dummy nodes and/or arcs.

In the following, we want to calculate the worst-case makespan of the project, i.e, the smallest amount of time that is required to complete the project under the worst realization of the parameter vector $\boldsymbol{\xi} \in \Xi$. This problem can be cast as the following instance of problem (1):

$$\sup_{\boldsymbol{\xi} \in \Xi} \inf_{\boldsymbol{y} \in \mathcal{Y}} \left\{ y_N \,:\, y_j - y_i \geq d_i(\boldsymbol{\xi}) \;\; \forall (i, j) \in A \right\}$$

Here $\mathcal{Y} = \mathbb{R}^N_+$, and y_i denotes the start time of task i, $i = 1, \ldots, N$. This problem is known to be NP-hard [37, Theorem 2.1], and we will employ affine decision rules as well as K-adaptable constant and affine decisions to approximate the optimal value of this problem. Note that the problem does not contain any first-stage decisions, but such decisions could be readily included, for example, to allow for resource allocations that affect the task durations.

For our numerical experiments, we consider the instance class presented in [37, Example 2.2]. To this end, we set $N = 3m + 1$ and $A = \{(3l + 1, 3l + p), (3l + p, 3l + 4) \,:\, l = 0, \ldots, m \text{ and } p = 2, 3\}$, $d_{3l+2}(\boldsymbol{\xi}) = \xi_{l+1}$ and $d_{3l+3}(\boldsymbol{\xi}) = 1 - \xi_{l+1}$, $l = 0, \ldots, m - 1$, as well as $d_{3l+1}(\boldsymbol{\xi}) = 0$, $l = 0, \ldots, m$. Figure 8 illustrates the

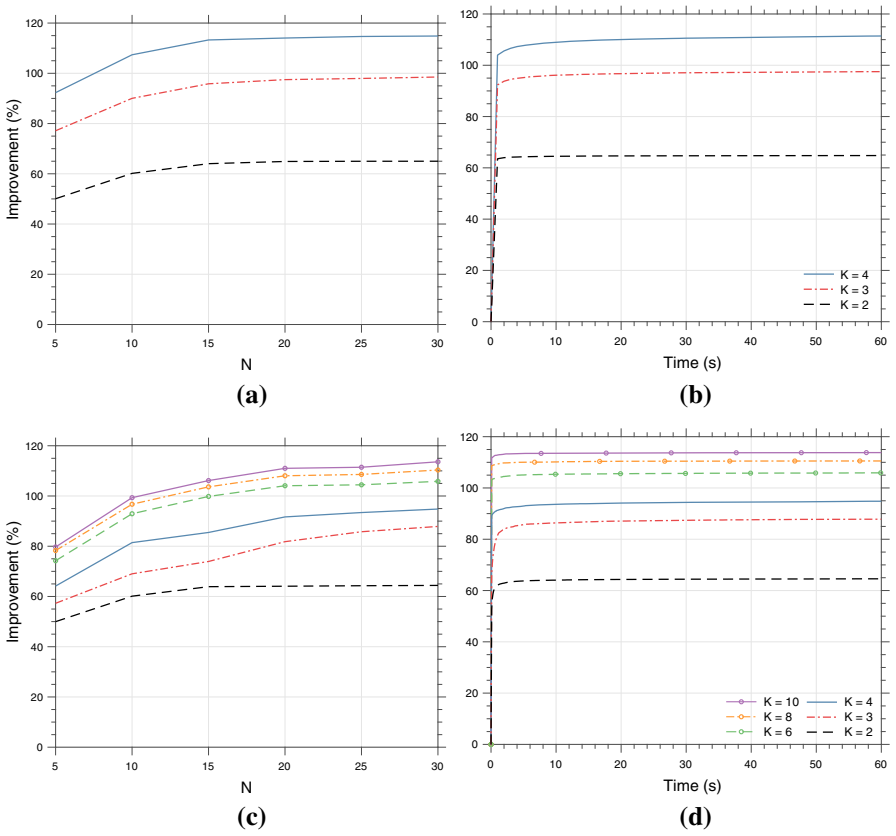

Fig. 7 Results for the capital budgeting problem with loans. The graphs have the same interpretation as in Fig. 6

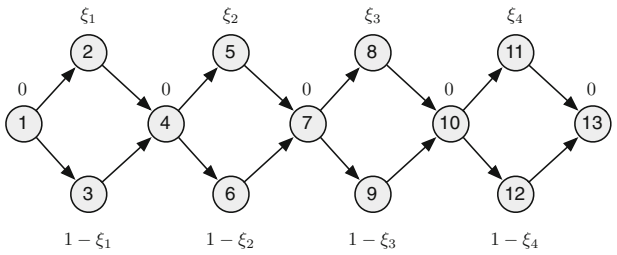

Fig. 8 Project network with $N = 3m + 1$ nodes for $m = 4$

project network corresponding to $m = 4$. Similar to [37], we consider the uncertainty set $\mathcal{Z} = \{\boldsymbol{\xi} \in \mathbb{R}^m_+ : \|\boldsymbol{\xi} - \mathbf{e}/2\|_1 \leq 1/2\}$.

We consider project networks of size $N \in \{3m + 1 : m = 3, 4, \ldots, 8\}$. One can show that for each network size N, the optimal value of the corresponding

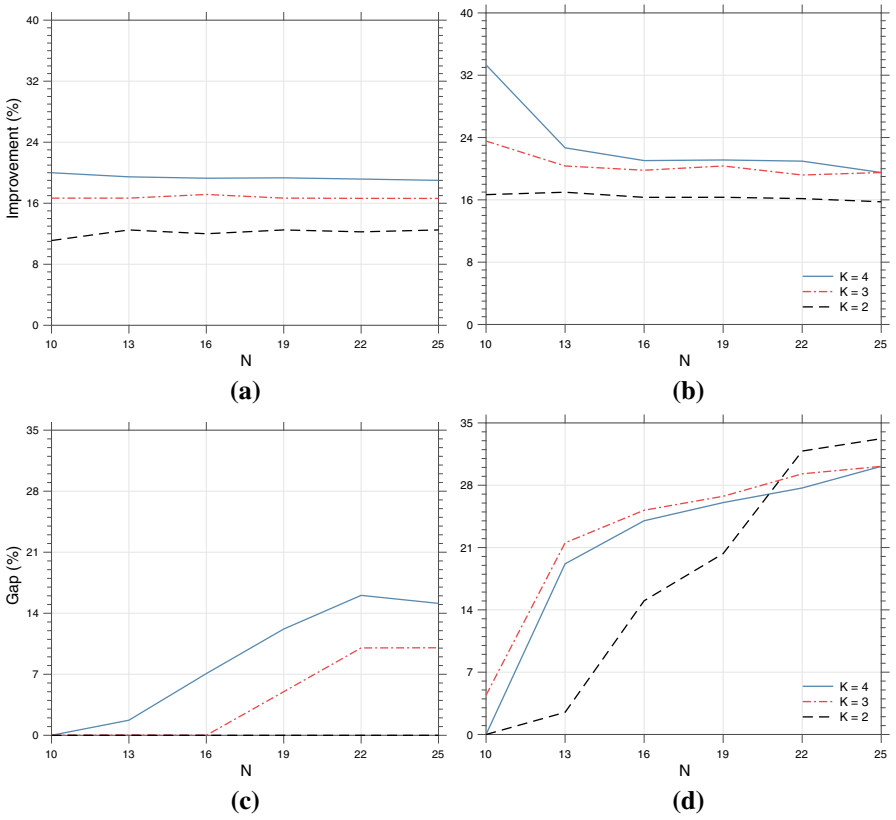

Fig. 9 Results for the project management problem. The left (right) graphs show results for the piecewise constant (affine) K-adaptability problems for increasing values of N. Graphs **a** and **b** depict improvements in objective value, while graphs **c** and **d** show optimality gaps after 2 h. The y-axes in graphs **a** and **b** have the same interpretation as those in Fig. 4 while the y-axes in graphs **c** and **d** have the same interpretation as the column "Gap (%)" in Table 2

static robust optimization problem as well as the affine decision rule problem is m, while the optimal value of the unapproximated two-stage robust optimization problem is $(m + 1)/2$, see [37, Example 2.2]. For $K \in \{2, 3, 4\}$, Fig. 9 summarizes the computational performance of the branch-and-bound scheme and the improvement in objective value of the resulting piecewise constant and piecewise affine decision rules with K pieces over the corresponding 1-adaptable solutions. Figure 9a, b show that using only two pieces, piecewise constant decision rules can improve upon the affine approximation by more than 12%, while a piecewise affine decision rule can improve by more than 15%. Figure 9c, d show that piecewise constant decision rules require smaller computation times than piecewise affine decision rules. This is not surprising since piecewise constant decision rules are parameterized by $\mathcal{O}(KN)$ variables, whereas piecewise affine decision rules are parameterized by $\mathcal{O}(KN^2)$ variables.

4.5 Vehicle routing

We consider the classical capacitated vehicle routing problem [21,22,35] defined on a complete, undirected graph $G = (V, E)$ with nodes $V = \{0, 1, \ldots, N\}$ and edges $E = \{(i, j) \in V \times V : i < j\}$. Node 0 represents the unique depot, while each node $i \in V_C = \{1, \ldots, N\}$ corresponds to a customer with demand $d_i \in \mathbb{R}_+$. The depot is equipped with M homogeneous vehicles; each vehicle has capacity C and it incurs an uncertain travel time $t_{ij}(\boldsymbol{\xi}) = (1 + \xi_{ij}/2)t_{ij}^0$ when it traverses the edge $(i, j) \in E$. Here, $t_{ij}^0 \in \mathbb{R}_+$ represents the nominal travel time along the edge $(i, j) \in E$, while ξ_{ij} denotes the uncertain deviation from the nominal value. Similar to the shortest path problem from Sect. 4.1, the realizations of the uncertain vector $\boldsymbol{\xi}$ are known to belong to the set

$$\Xi = \left\{ \boldsymbol{\xi} \in [0, 1]^{|E|} : \sum_{(i,j)\in E} \xi_{ij} \leq \Gamma \right\},$$

which stipulates that at most Γ travel times may maximally deviate from their nominal values.

A *route plan* $(R_1, \ldots, R_M)$ corresponds to a partition of the customer set V_C into M vehicle routes, $R_m = (R_{m,1}, \ldots, R_{m,N_m})$, where $R_{m,l}$ represents the lth customer and N_m the number of customers served by the mth vehicle. This route plan is feasible if the total demand served on each route is less than the vehicle capacity; that is, if $\sum_{l=1}^{N_m} d_{R_{m,l}} \leq C$ is satisfied for all $m \in \{1, \ldots, M\}$. The total travel time of a feasible route plan under the uncertainty realization $\boldsymbol{\xi}$ is given by $\sum_{m=1}^{M} \sum_{l=0}^{N_m} t_{R_{m,l} R_{m,l+1}}(\boldsymbol{\xi})$, where we define $R_{m,0} = R_{m,N_m+1} = 0$; that is, each vehicle starts and ends at the depot. The decision-maker aims to choose K route plans here-and-now, i.e., before observing the actual travel times, such that the worst-case total travel time of the shortest among the chosen route plans is minimized. This problem can be formulated as an instance of the K-adaptability problem (2):

$$\inf_{\boldsymbol{y}\in\mathcal{Y}^K} \sup_{\boldsymbol{\xi}\in\Xi} \inf_{k\in\mathcal{K}} \boldsymbol{t}(\boldsymbol{\xi})^\top \boldsymbol{y}_k$$

Here, $\mathcal{Y}$ denotes the set of all feasible route plans in G; that is,

$$\mathcal{Y} = \left\{ \boldsymbol{y} \in \mathbb{Z}_+^{|E|} : \begin{array}{l} 0 \leq y_{ij} \leq 1 \quad \forall (i, j) \in E : i, j \in V_C, \\ \sum_{j\in V_C} y_{0j} = 2M, \\ \sum_{j\in V:(i,j)\in E} y_{ij} = 2 \quad \forall i \in V_C, \\ \sum_{(i,j)\in E: i,j\in U} y_{ij} \leq |U| - \left\lceil \frac{1}{D} \sum_{i\in U} d_i \right\rceil \quad \forall U \subseteq V_C \end{array} \right\}.$$

Similar to the shortest path problem, the K-adaptability formulation of the vehicle routing problem only contains second-stage decisions, and as such, the corresponding two-stage robust optimization problem (1) is of limited interest in practice. However, the K-adaptability problem (2) has important applications in logistics enterprises, where the time available between observing the travel times in a road network and determining the route plan is limited, or because the drivers must be trained to a small set of route plans that are to be executed daily over the course of a year.

We note that the set $\mathcal{Y}$ represents the so-called *two-index vehicle flow* formulation of the vehicle routing problem, in which the first equation ensures that M vehicles are used; the second set of equations ensures that each customer is visited by exactly one vehicle; while the third set of inequalities ensure that there are no *subtours* disconnected from the depot and that all vehicle capacities are respected. This formulation is known to be extremely challenging to solve because it consists of an exponential number of inequalities. For $K > 1$, the corresponding K-adaptability problem is naturally even more challenging and it is practically intractable to solve it using the approach described in [24]. In contrast, the heuristic variant of our algorithm described in Sect. 3.3 as well as the heuristic approach of [13] only require the solution of vehicle routing subproblems that are of similar complexity as the associated 1-adaptability problems. Therefore, in the following, we only present results using these algorithms. In both cases, we solved all vehicle routing subproblems using the branch-and-cut algorithm described in [29].

For our numerical experiments, we consider all 49 instances from [29] with $N \leq 50$, which are commonly used to benchmark vehicle routing algorithms. We set an overall time limit of 2 h. For the heuristic variant of our algorithm, we further set a time limit of 10 min per vehicle routing subproblem. We note that the heuristic of [13] requires the successful termination of an expensive preprocessing step to determine good K-adaptable solutions. Therefore, to prevent bias in favor of our algorithm, Fig. 10 compares the two algorithms only across the 39 instances for which this step terminated successfully. The figure shows that when the number of policies is small,

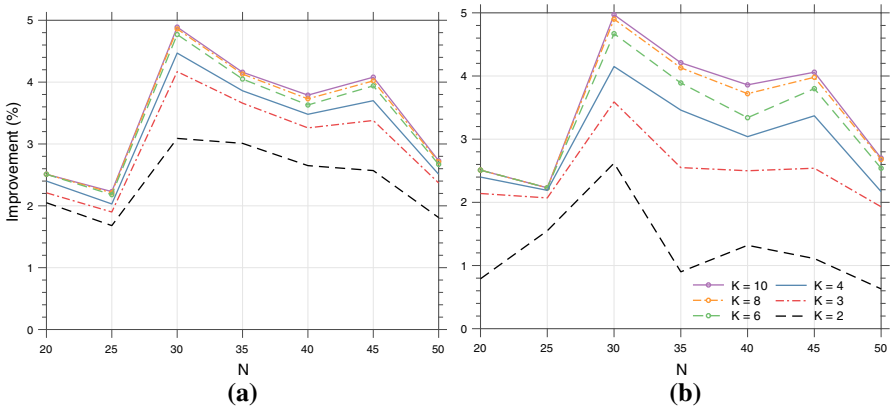

Fig. 10 Results for the vehicle routing problem. The graphs show the average improvement after 2 h obtained using the heuristic variant of our algorithm (left) and the heuristic algorithm in [13] (right)

the K-adaptable solutions obtained using our algorithm are about 1% better than those obtained using the heuristic algorithm of [13]. Moreover, the differences in their objective values are relatively higher for larger instances.

5 Conclusions

In contrast to single-stage robust optimization problems, which are typically solved via monolithic reformulations, there is growing evidence that two-stage and multi-stage robust optimization problems are best solved algorithmically [6–8,31,39]. Our findings in this paper appear to confirm this observation, as our proposed branch-and-bound algorithm compares favorably with the reformulations proposed in [24]. In terms of modeling flexibility, our algorithm can accommodate mixed continuous and discrete decisions in both stages, can incorporate discrete uncertainty, and allows us to model flexible piecewise affine decision rules. At the same time, our numerical results indicate that the algorithmic approach is highly competitive in terms of computational performance as well. From a practical viewpoint, a notable feature of our algorithm is that it admits a lightweight implementation by integrating it into the branch-and-bound schemes of commercial solvers via branch callbacks, while allowing easy modification as a heuristic for large-scale instances.

Our results open up multiple fruitful avenues for future research. The scope of the presented algorithm can be broadened further by generalizing it to two-stage *distributionally* robust optimization problems, where the uncertain problem parameters are modeled as random variables following a probability distribution that is only partially known. More ambitiously, it would be instructive to explore how the concept of K-adaptability, as well as our proposed branch-and-bound algorithm, extend to dynamic robust optimization problems with more than two stages.

Acknowledgements Anirudh Subramanyam gratefully acknowledges support from the John and Claire Bertucci Graduate Fellowship program at Carnegie Mellon University.

References

1. Ayoub, J., Poss, M.: Decomposition for adjustable robust linear optimization subject to uncertainty polytope. Comput. Manag. Sci. **13**(2), 219–239 (2016)
2. Ben-Tal, A., Ghaoui, L.E., Nemirovski, A.: Robust Optimization. Princeton University Press, Princeton (2009)
3. Ben-Tal, A., Goryashko, A., Guslitzer, E., Nemirovski, A.: Adjustable robust solutions of uncertain linear programs. Math. Program. **99**(2), 351–376 (2004)
4. Bertsimas, D., Brown, D.B., Caramanis, C.: Theory and applications of robust optimization. SIAM Rev. **53**(3), 464–501 (2011)
5. Bertsimas, D., Caramanis, C.: Finite adaptibility in multistage linear optimization. IEEE Trans. Autom. Control **55**(12), 2751–2766 (2010)

6. Bertsimas, D., Dunning, I.: Multistage robust mixed integer optimization with adaptive partitions. Oper. Res. **64**(4), 980–998 (2016)
7. Bertsimas, D., Georghiou, A.: Design of near optimal decision rules in multistage adaptive mixed-integer optimization. Oper. Res. **63**(3), 610–627 (2015)
8. Bertsimas, D., Georghiou, A.: Binary decision rules for multistage adaptive mixed-integer optimization. Math. Program. **167**(2), 395–433 (2018)
9. Bertsimas, D., Goyal, V., Sun, X.A.: A geometric characterization of the power of finite adaptability in multistage stochastic and adaptive optimization. Math. Oper. Res. **36**(1), 24–54 (2011)
10. Bertsimas, D., Litvinov, E., Sun, X.A., Zhao, J., Zheng, T.: Adaptive robust optimization for the security constrained unit commitment problem. IEEE Trans. Power Syst. **28**(1), 52–63 (2013)
11. Blankenship, J.W., Falk, J.E.: Infinitely constrained optimization problems. J. Optim. Theory Appl. **19**(2), 261–281 (1976)
12. Buchheim, C., Kurtz, J.: Min-max-min robustness: a new approach to combinatorial optimization under uncertainty based on multiple solutions. Electron Notes Discrete Math. **52**, 45–52 (2016)
13. Buchheim, C., Kurtz, J.: Min–max–min robust combinatorial optimization. Math. Program. **163**(1), 1–23 (2017)
14. Buchheim, C., Kurtz, J.: Complexity of min-max-min robustness for combinatorial optimization under discrete uncertainty. Discrete Optim. **28**(1), 1–15 (2018)
15. Chen, X., Zhang, Y.: Uncertain linear programs: extended affinely adjustable robust counterparts. Oper. Res. **57**(6), 1469–1482 (2009)
16. Gabrel, V., Murat, C., Thiele, A.: Recent advances in robust optimization: an overview. Eur. J. Oper. Res. **235**(3), 471–483 (2014)
17. Georghiou, A., Wiesemann, W., Kuhn, D.: Generalized decision rule approximations for stochastic programming via liftings. Math. Program. **152**(1), 301–338 (2015)
18. Goh, J., Sim, M.: Distributionally robust optimization and its tractable approximations. Oper. Res. **58**(4), 902–917 (2010)
19. Gorissen, B.L., den Hertog, D.: Robust counterparts of inequalities containing sums of maxima of linear functions. Eur. J. Oper. Res. **227**(1), 30–43 (2013)
20. Gorissen, B.L., Yanıkoğlu, I., den Hertog, D.: A practical guide to robust optimization. Omega **53**, 124–137 (2015)
21. Gounaris, C., Repoussis, P., Tarantilis, C., Wiesemann, W., Floudas, C.: An adaptive memory programming framework for the robust capacitated vehicle routing problem. Transp. Sci. **50**(4), 1239–1260 (2016)
22. Gounaris, C., Wiesemann, W., Floudas, C.: The robust capacitated vehicle routing problem under demand uncertainty. Oper. Res. **61**(3), 677–693 (2013)
23. Guslitser, E.: Uncertainty-immunized solutions in linear programming. Master's Thesis, Technion, Israeli Institute of Technology (2002)
24. Hanasusanto, G.A., Kuhn, D., Wiesemann, W.: *K*-adaptability in two-stage robust binary programming. Oper. Res. **63**(4), 877–891 (2015)
25. IBM: ILOG CPLEX Optimizer (2018). https://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/. Accessed 27 July 2018
26. Jiang, R., Zhang, M., Li, G., Guan, Y.: Two-stage robust power grid optimization problem. Available on Optimization Online (2010)
27. Kelley, J.E.: The cutting-plane method for solving convex programs. J. Soc. Ind. Appl. Math. **8**(4), 703–712 (1960)
28. Kuhn, D., Wiesemann, W., Georghiou, A.: Primal and dual linear decision rules in stochastic and robust optimization. Math. Program. **130**(1), 177–209 (2011)
29. Lysgaard, J., Letchford, A., Eglese, R.: A new branch-and-cut algorithm for the capacitated vehicle routing problem. Math. Program. **100**(2), 423–445 (2004)
30. Mutapcic, A., Boyd, S.: Cutting-set methods for robust convex optimization with pessimizing oracles. Optim. Methods Softw. **24**(3), 381–406 (2009)
31. Postek, K., den Hertog, D.: Multistage adjustable robust mixed-integer optimization via iterative splitting of the uncertainty set. INFORMS J. Comput. **28**(3), 553–574 (2016)
32. Rubin, P.: When the OctoMom Solves MILPs (2010). https://orinanobworld.blogspot.com/2010/06/when-octomom-solves-milps.html. Accessed 29 Jan 2019
33. Subramanyam, A., Gounaris, C., Wiesemann, W.: KAdaptabilitySolver (GitHub repository) (2019). https://doi.org/10.5281/zenodo.3490004

34. Thiele, A., Terry, T., Epelman, M.: Robust linear optimization with recourse. Technical Report, Lehigh University and University of Michigan, (2010)
35. Toth, P., Vigo, D.: Vehicle routing: problems, methods, and applications. Society for Industrial and Applied Mathematics, 2nd edn. (2014)
36. Vayanos, P., Kuhn, D., Rustem, B.: Decision rules for information discovery in multi-stage stochastic programming. In: Proceedings of the 50th IEEE Conference on Decision and Control and European Control Conference, pp. 7368–7373 (2011)
37. Wiesemann, W., Kuhn, D., Rustem, B.: Robust resource allocations in temporal networks. Math. Program. **135**(1), 437–471 (2012)
38. Yanıkoğlu, I., Gorissen, B.L., den Hertog, D.: A survey of adjustable robust optimization. Eur. J. Oper. Res. **277**(3), 799–813 (2019)
39. Zeng, B., Zhao, L.: Solving two-stage robust optimization problems using a column-and-constraint generation method. Oper. Res. Lett. **41**(5), 457–561 (2013)
40. Zhao, C., Wang, J., Watson, J.P., Guan, Y.: Multi-stage robust unit commitment considering wind and demand response uncertainties. IEEE Trans. Power Syst. **28**(3), 2708–2717 (2013)
41. Zhao, L., Zeng, B.: An exact algorithm for two-stage robust optimization with mixed integer recourse problems. Technical Report, University of South Florida, (2012)