A Double-oracle, Logic-based Benders decomposition approach to solve the K-adaptability problem

Alireza Ghahtarani*[1], Ahmed Saif[1], Alireza Ghasemi[1], and Erick Delage †[2]

[1]Department of Industrial Engineering, Dalhousie University
[2]Department of Decision Sciences, HEC Montréal

Abstract

We propose a novel approach to solve K-adaptability problems with convex objective and constraints and integer first-stage decisions. A logic-based Benders decomposition is applied to handle the first-stage decisions in a master problem, thus the sub-problem becomes a min-max-min robust combinatorial optimization problem that is solved via a double-oracle algorithm that iteratively generates adverse scenarios and recourse decisions and assigns scenarios to K-subsets of the decisions by solving p-center problems. Extensions of the proposed approach to handle parameter uncertainty in both the first-stage objective and the second-stage constraints are also provided. We show that the proposed algorithm converges to an optimal solution and terminates in finite number of iterations. Numerical results obtained from experiments on benchmark instances of the adaptive shortest path problem, the regular knapsack problem, and a generic K-adaptability problem demonstrate the performance advantage of the proposed approach when compared to state-of-the-art methods in the literature.

Key Words: K-adaptability problem, Min-max-min robust combinatorial optimization, Adaptive robust optimization, Discrete recourse, Logic-based Benders decomposition

1 Introduction

Robust Optimization (RO) has become a classical framework for dealing with parameter uncertainty in optimization problems (Bertsimas et al., 2011). In RO, parameter uncertainty is captured through an *uncertainty set* of proper structure and size, and the optimization is conducted with respect to the *worst-case* realization in it. An important class of RO problems that has gained considerable attention recently is adaptive/adjustable robust optimization (ARO), in which some decisions are assumed to be delayable until the realized values of uncertain parameters become partially or fully known. Whereas ARO formulations often lead to better (less

*Dalhousie University, Halifax, Nova Scotia, B3J 1B6, Canada (alireza.ghahtarani@dal.ca).
†Department of Decision Sciences, HEC Montréal, Montréal, Québec, H3T 2A7, Canada.

pessimistic) solutions than their corresponding static RO models, they are computationally intractable (Ben-Tal et al., 2004). However, several exact and approximate algorithms have been proposed to solve important ARO classes such as linear two-stage RO problems (Thiele et al., 2009; Chen & Zhang, 2009; Kuhn et al., 2011; Zhao & Zeng, 2012; Bertsimas et al., 2012; Jiang et al., 2012; Iancu et al., 2013). With a few exceptions, these algorithms use duality to handle the second-stage (recourse) problem. Hence, they can be only used with continuous recourse decisions. Although some attempts have been made to develop efficient solution methods for ARO problems with discrete recourse (Dhamdhere et al., 2005; Georghiou et al., 2015; Bertsimas & Georghiou, 2015, 2018), the literature for this class of problems is still sparse.

Recently, an alternative modeling approach, referred to as *K-adaptability*, has been proposed as a conservative approximation of ARO problems with discrete recourse (Hanasusanto et al., 2015; Subramanyam et al., 2020). Rather than allowing any feasible integer recourse to be selected, the decision-maker *prepares* K solutions in advance (under uncertainty). Then, upon full knowledge of the realized value of the uncertain parameters, the best among these K solutions is selected. Apart from being better in general compared to the solutions of static RO, Buchheim & Kurtz (2017) argued that K-adaptability solutions are more easily accepted by a human user as they do not change each time but are taken from a relatively small set of candidate solutions.

Similar to Hanasusanto et al. (2015) and Subramanyam et al. (2020), we initially focus on a linear version of the problem, in which both the objective function and constraints in the first- and second-stages are affine functions of the decision variables, the uncertain parameters effect the second-stage objective function, and the uncertainty set is polyhedral. However, we later show how the algorithm that we propose can be adapted to solve other variants of the problem. Formally, the linear K-adaptability problem under consideration is formulated as follows:

$$\min_{\mathrm{x}\in\mathcal{X},\{\mathrm{y}^k\}_{k\in[K]}} \ \mathrm{c}'\mathrm{x} + \max_{\xi\in\Xi} \min_{k\in[K]} \left\{\xi'\mathrm{Q}\mathrm{y}^k : \ \mathrm{T}\mathrm{x} + \mathrm{W}\mathrm{y}^k \leq \mathrm{b}\right\} \tag{1a}$$

$$\text{s.t.} \quad \mathrm{y}^k \in \mathcal{Y}, \ k \in [K], \tag{1b}$$

where $\mathcal{X} \subset \{0,1\}^n$ and $\mathcal{Y} \subset \{0,1\}^m$ where their continuous relaxations are polyhedral, $\Xi \subset \mathbb{R}^q$ is a polyhedral set, $\mathrm{c} \in \mathbb{R}^n$, $\mathrm{Q} \in \mathbb{R}^{q\times m}$, $\mathrm{T} \in \mathbb{R}^{s\times n}$, $\mathrm{W} \in \mathbb{R}^{s\times m}$, $\mathrm{b} \in \mathbb{R}^s$, and $[K] = \{1,\ldots,K\}$. In this formulation, elements of the vector x denote the here-and-now (first-stage) decision variables, $\xi \in \Xi$ are the uncertain parameters, and y are the wait-and-see (second-stage) decision variables.

A special case of the K-adaptability problem, known in the literature as the *min-max-min robust combinatorial optimization* (MMMRCO) problem arises when the only decision that is made under uncertainty is the pre-selection of the K recourse action, *i.e.,* the problem does not have the actual first-stage decision. The MMMRCO problem has many practical applications, such as parcel delivery and finding routes in hazardous situations, that are discussed in Arslan et al. (2022). The basic MMMRCO problem (without constraint uncertainty) is formulated as follows:

$$\min_{\mathrm{y}^k\in\mathcal{Y}} \max_{\xi\in\Xi} \min_{k\in[K]} \{\xi'\mathrm{Q}\mathrm{y}^k : \ \mathrm{W}\mathrm{y}^k \leq \mathrm{b}\}. \tag{2a}$$

So far, only two solution methods have been developed for the K-adaptability problem. Hanasusanto et al. (2015) proposed a mixed-integer linear programming approximation, de-

rived using linear programming duality, which leads to a monolithic formulation involving bilinear terms. A McCormick envelope is used to linearize the bilinear terms, which requires a large number of new variables to be introduced. Moreover, the number of binary variables increases as K is increased. Consequently, this approach hardly solves the shortest path instances with more than 25 nodes. In another attempt to solve the K-adaptability problem, Subramanyam et al. (2020) proposed a branch-and-bound algorithm that enjoys asymptotic convergence in general, but has finite convergence under specific conditions. This algorithm works by generating a relevant subset of uncertainty realizations and enumerating over their assignment to the K solutions. Nevertheless, it is also inefficient for solving shortest path instances with more than 25 nodes.

Besides the aforementioned methods that can handle the general case (*i.e.,* with first-stage decisions), a few approaches to solve variants of (2) have been proposed. Chassein et al. (2019) developed a branch-and-bound algorithm that can solve large instances of the MMMRCO problem, yet only with budget uncertainty sets. They also proposed a heuristic solution algorithm based on the formulation of Hanasusanto et al. (2015). However, instead of using the McCormick linearization approach to handle the bilinear terms, they used the block-coordinate descent algorithm, which has no optimality guarantee. Moreover, their algorithms hardly solve any instance of the shortest path problem when K grows above 3. Goerigk et al. (2020) developed a heuristic algorithm based on an integer programming formulation and a row-and-column generation algorithm to solve it. Yet, this algorithm has no performance guarantee, which means this algorithm does not guarantee convergence in a finite number of iterations. Recently, Arslan et al. (2022) proposed a solution approach that iteratively generates scenarios of the uncertain parameters and assigns them to solutions by solving a p-center problem. However, this algorithm works well only if there is an effective way to restrict and enumerate the search space.

In this paper, we present a new approach to solve the K-adaptability problem with binary or integer first-stage decisions. The scenario generation step in the proposed approach enjoys finite convergence when the uncertainty set is polyhedral, but the approach can be used with any convex uncertainty set. Although we focus initially on problems with affine functions, the proposed algorithm can be extended to nonlinear objective function and constraints with respect to the decision variables for second-stage problem.

The proposed approach uses a *logic-based* (also referred to as *combinatorial*) Benders algorithm to handle the first-stage decisions such that the remaining subproblem is a MMMRCO that is solved iteratively to generate optimality cuts. To solve this subproblem, we propose a double-oracle algorithm that iterates between solving an adversary problem to iteratively generate worst-case scenarios for a K-subset of feasible solutions and determine the scenario-solution assignment, and solving a decision-maker's problem to find the optimal K-subset of solutions for all the scenarios generated so far. Although the way scenarios are generated and assigned to solutions is similar to that proposed by Arslan et al. (2022), our approach uses a more efficient way (*i.e.,* by solving an optimization problem) to identify the optimal K-subset of recourse solutions in every iteration.

We note that the K-adaptability problem formulation provided in (1) is based on that introduced by Bertsimas & Caramanis (2010), *i.e.,* with a first-stage problem that is not subject to uncertainty, whereas Hanasusanto et al. (2015) addressed an extended version in which both stages are affected by the same uncertain parameters. Hence, we show how the pro-

posed algorithm can be extended to handle K-adaptability problems with uncertainty in both stages (whether the two stages depend on the same or different uncertain parameters). Finally, extensive numerical experiments are conducted on benchmark instances of several classical optimization problems, and the computational superiority of the proposed approach *vis-á-vis* state-of-the-art solution methods is demonstrated.

The remainder of this paper is organized as follows. Section 2 presents the approach proposed to solve problem (1) (*i.e.,* with linear objective function and constraints and with uncertainty affecting only the recourse objective function). Section 3 studies the convergence properties of the proposed algorithm and proves its finite convergence. In section 4, we show how the proposed algorithm can be modified to solve different variants of the K-adaptability problem, namely, problems with integer first-stage decision variables, problems with nonlinear functions, and problems affected by uncertainty in the first and second-stages. The numerical experiments conducted to test the proposed algorithm on benchmark problems, and a detailed discussion of their results, are presented in section 5. Finally, section 6 provides some conclusions and suggests future research directions.

Notation. We use $\mathrm{upright}$ lower and upper case letters, respectively, for vectors and matrices. Individual elements of these vectors and matrices are denoted using $italic$ versions of the same letters. For example, elements of the J-dimension vector x are denoted as x_j. Depending on the context, upper case letters might be used also to denote scalars (*e.g., J*), whereas lower case letters might denote also functions (*e.g., $g(\cdot)$*). $[J]$ is used as a shorthand for the set of integers $\{1, 2, \ldots, J\}$ and a partial set of a given set $[J]$ is denoted as $[J']$. We use the symbol e to denote an all-ones vector of appropriate size. The calligraphic font is used for sets (*e.g., $\mathcal{X}$*).

2 The Proposed Solution Approach

In this section, we present the proposed approach to solve the K-adaptability problem (1) with binary first-stage decision variables and recourse objective uncertainty. First, we show how a logic-based Benders decomposition is applied to deal with the discrete first-stage variables. Then, we describe a double oracles algorithm to solve the subproblem.

2.1 A Logic-based Benders Decomposition

We apply Benders decomposition by projecting the model onto the subspace defined by the first-stage variables x to get the master problem (MP):

$$\min_{\mathrm{x} \in \mathcal{X} \cap \mathcal{V}} \mathrm{c}'\mathrm{x} + \nu(\mathrm{x}), \tag{3}$$

where $\mathcal{V} := \left\{\mathrm{x} : \mathrm{W}\mathrm{y}^k \leq \mathrm{b} - \mathrm{T}\mathrm{x} \text{ for some } \mathrm{y}^k \in \mathcal{Y}, k \in [K]\right\}$. For a given $\bar{\mathrm{x}} \in \mathcal{X} \cap \mathcal{V}$, $\nu(\bar{\mathrm{x}})$ is the optimal value of the sub-problem (SP)

$$\min_{\{\mathrm{y}^k\}_{k\in[K]}} \max_{\xi \in \Xi} \min_{k \in [K]} \quad \xi' \mathrm{Q}\mathrm{y}^k \tag{4a}$$

$$\text{s.t.} \quad \mathrm{y}^k \in \mathcal{Y},\ \mathrm{W}\mathrm{y}^k \leq \mathrm{b} - \mathrm{T}\bar{\mathrm{x}} \qquad \forall k \in [K]. \tag{4b}$$

We note that since $\mathrm{x} \in \mathcal{X} \cap \mathcal{V}$, MP enjoys *relatively complete recourse*, *i.e.,* it has feasible solutions for all $\bar{\mathrm{x}} \in \mathcal{X} \cap \mathcal{V}$, and $\xi \in \Xi$. Without this property, the SP might be infeasible for some $\bar{x}$, thus requiring feasibility cuts to be generated.

The basic idea of the classical Benders algorithm is to approximate the function $\nu(\mathrm{x})$ using hyper-planes (referred to as *optimality cuts*) generated by solving the dual SP for fixed values of x. However, since SP has binary decision variables, it is not possible to use the duality theory to generate cuts. Assuming that we have an oracle to solve SP, in any iteration r, the r-th feasible solution x^r is used to define the sets $\mathcal{S}_r := \{i \in [n] : x_i^r = 1\}$ and to evaluate its corresponding worst-case second-stage objective function value θ_r. We use this solution and value to generate the valid combinatorial cut, first proposed by Laporte & Louveaux (1993),

$$\theta \geq (\theta_r - L_r)(\sum_{i \in \mathcal{S}_r} x_i - \sum_{\notin \mathcal{S}_r} x_i) - (\theta_r - L_r)(|\mathcal{S}_r| - 1) + L_r,$$

where $|\mathcal{S}_r|$ is the cardinality of $\mathcal{S}_r$, L is a lower bound on the optimal value of the SP and θ is a MP decision variable that defines the epigraph of $\nu(\mathrm{x})$. Hence, MP can be written as follows:

$$\min_{\mathrm{x} \in \mathcal{X}, \theta} \ \mathrm{c}'\mathrm{x} + \theta \tag{5a}$$

$$\text{s.t.} \quad \theta \geq (\theta_r - L_r)(\sum_{i \in \mathcal{S}_r} x_i - \sum_{\notin \mathcal{S}_r} x_i) - (\theta_r - L_r)(|\mathcal{S}_r| - 1) + L_r \quad \forall r, \tag{5b}$$

The Logic-based Benders decomposition algorithm is summarized as follows:

Algorithm 1: Logic-based Benders decomposition algorithm for solving K-adaptability problem

Initiate with an arbitrary feasible $\bar{\mathrm{x}}$, set $UB = \infty$, $LB = -\infty$, $r = 1$
while $UB - LB \geq \epsilon$ **do**
  Solve SP (4) with $\bar{\mathrm{x}}$ and find θ_r
  Find a lower bound L_r for SP (4) as will be explained later
  Generate the optimality cut: $\theta \geq (\theta_r - L_r)(\sum_{i \in \mathcal{S}_r} x_i - \sum_{i \notin \mathcal{S}_r} x_i) - (\theta_r - L_r)(|\mathcal{S}_r| - 1|) + L_r$
  Update the upper bound: $UB = \min(UB, \mathrm{c}'\bar{\mathrm{x}} + \theta_r)$
  Solve MP (5) with the new optimality cut added
  Set LB equal to the optimal value of MP (5)
  Extract the optimal partial solution x^* and use it as $\bar{\mathrm{x}}$ in the next iteration and set $r = r + 1$
end
Return: Declare the pair $\left(\mathrm{x}^*, \mathrm{y}^{k^*}\right)$ as the optimal solution

We note that SP (4) is a MMMCRO problem. In the next section, we propose a novel approach to solve it and also to obtain the lower bound L_r.

2.2 Double-oracle for Solving SP

We define the partial sets $\mathcal{Y}' \subseteq \mathcal{Y}$ with $|\mathcal{Y}| \geq K$ and $\Xi' \subset \Xi$ and use $j \in [J]$ ($[J']$) and $h \in [H]$ ($[H']$) to index the elements of $\mathcal{Y}$ ($\mathcal{Y}'$) and the vertices of Ξ (Ξ'), also referred to as "scenarios", respectively. SP (4) can be reformulated over these partial sets as the p-center

problem $\mathtt{P}(\mathcal{Y}', \Xi')$ (Arslan et al., 2022):

$$\min_{\{z_j\}_{j\in[J']},\{v_{jh}\}_{j\in[J'],h\in[H']},w} \quad w \tag{6a}$$

$$\text{s.t.} \quad w \geq \sum_{j\in[J']} \xi_h' \mathrm{Q} \mathrm{y}_j v_{jh} \quad \forall h \in [H'] \tag{6b}$$

$$\sum_{j\in[J']} v_{jh} = 1 \quad \forall h \in [H'] \tag{6c}$$

$$\sum_{j\in[J']} z_j = K \tag{6d}$$

$$v_{jh} \leq z_j \quad \forall j \in [J'], \forall h \in [H'] \tag{6e}$$

$$v_{jh}, z_j \in \{0,1\} \quad \forall j \in [J'], \forall h \in [H']. \tag{6f}$$

The binary variable z_j takes value 1 if the feasible solution y_j is selected to be among the K "prepared" solutions, and v_{jh} takes value 1 if scenario ξ_h is assigned to solution y_j, and 0 otherwise. Constraint (6b) finds the scenario-solution pair with the worst cost among all assignments. Constraint (6c) ensures that each scenario is assigned to exactly one solution, whereas (6d) and (6e), respectively, stipulate that K solutions are selected and that scenarios can be assigned to selected solution only.

To solve (6), the following algorithm is proposed:

1. Solve the problem $\mathtt{P}(\mathcal{Y}', \Xi)$, *i.e.,* the problem with the subset $\mathcal{Y}'$ of all recourse solutions generated so far (carried forward from Step 2 in the previous iteration) and all scenarios in Ξ to obtain an upper bound UB. To solve this problem, we begin with a subset Ξ' of scenarios and perform the following steps.
   (a) Solve the problem $\mathtt{P}(\mathcal{Y}', \Xi')$ (*i.e.,* Problem (6)) to find z^*, v^* and w^*. Identify the optimal K-subset of recourses as $\left\{\mathrm{y}^k \in \mathcal{Y}' : z_k^* = 1\right\}$
   (b) Given the current optimal K-subset $\left\{\mathrm{y}^k\right\}_{k\in[K]}$ of recourses, try to find a scenario $\xi_{|H'|+1} \in \Xi$ that violates (6b) by solving the problem

$$\max_{\xi\in\Xi,\eta} \quad \eta \tag{7a}$$

$$\text{s.t.} \quad \eta \leq \xi' \mathrm{Q} \mathrm{y}^k \quad k \in [K] \tag{7b}$$

   (c) If $\eta^* > w^*$, add the new scenario to Ξ' and repeat steps (a) and (b). Otherwise, stop and move to Step 2.
2. In this step, we find the optimal K-subset $\left\{\mathrm{y}^{k*}\right\}_{k\in[K]}$ of recourses that minimizes the

worst-case loss for the discrete scenario set Ξ' by solving the problem $\mathtt{P}(\mathcal{Y}, \Xi')$:

$$\min_{\{\mathrm{y}^k\}_{k\in[K]},\gamma,\{u_{kh}\}_{k\in[K],h\in[H']}} \quad \gamma \tag{8a}$$

$$\text{s.t.} \quad \xi_h' \mathrm{Q}\mathrm{y}^k \le \gamma + M(1-u_{kh}) \quad \forall k \in [K],\, \forall h \in [H'] \tag{8b}$$

$$\sum_{k\in[K]} u_{kh} = 1 \quad \forall h \in [H'] \tag{8c}$$

$$\mathrm{y}^k \in \mathcal{Y} \quad \forall k \in [K] \tag{8d}$$

$$\mathrm{W}\mathrm{y}^k \le \mathrm{b} - \mathrm{T}\bar{\mathrm{x}} \quad \forall k \in [K] \tag{8e}$$

$$u_{kh} \in \{0,1\} \quad \forall k \in [K],\, \forall h \in [H']. \tag{8f}$$

In problem (8), u_{kh} is a binary assignment variable which takes value 1 if scenario h is assigned to recourse k. We update the solution pool as $\mathcal{Y}' \leftarrow \mathcal{Y}' \cup \left\{\mathrm{y}^{k^*}\right\}_{k\in[K]}$, where $\left\{\mathrm{y}^{k^*}\right\}_{k\in[K]}$ is the partial optimal solution of problem (8). Moreover we set $LB = \gamma$.

3. Iterate between steps (1) and (2) until $UB - LB < \varepsilon$. Declare the incumbent $\left\{\mathrm{y}^{k^*}\right\}_{k\in[K]}$ as the optimal solution.

Algorithm 2: The Double-Oracle algorithm for solving SP (4)

```
initialization: y', Ξ', LB = −∞, UB = +∞
while UB − LB ≥ ε do
    while Scenario-added=true do
        Compute w*, z*, v*, and y^{k*} by solving (6)
        Compute ξ_{|H|+1} ∈ Ξ, and η* by solving (7)
        if η* > w* then
            Ξ' ← Ξ' ∪ {ξ_{|H|+1}}_{k∈[K]}
            Scenario-added=true
        else
            Ξ' ← Ξ'
            Scenario-added=false
        end
    end
    Return: Ξ', UB = w*
    Compute {y^{k*}}_{k∈[K]}, γ* by solving (8)
    Y' ← Y' ∪ {y^{k*}}_{k∈[K]}
    LB = γ*
end
Return: {y^{k*}}_{k∈[K]}
```

To generate an optimality cut, the logic-based Benders decomposition algorithm explained in the previous section requires a valid lower bound (L) on the optimal value of the second-stage problem. A valid lower bound should be $L \le \min_{\mathrm{x}}\{\nu(\mathrm{x}) |\ \mathrm{x} \in \mathcal{X}\}$. In every iteration

of the proposed algorithm, we calculate a lower bound on the optimal value of SP for a fixed first-stage decision $\bar{\mathrm{x}}$ by solving (8). However, a lower bound on the optimal value of SP for all $\mathrm{x} \in \mathcal{X}$ is required, which can be obtained by solving the following problem:

$$\min_{\{\mathrm{y}^k\}_{k\in[K]},\gamma,\{u_{kh}\}_{k\in[K],h\in[H']},\mathrm{x}} \quad \gamma \tag{9a}$$

$$\text{s.t.} \quad \xi'_h \mathrm{Q}\mathrm{y}^k \leq \gamma + M(1-u_{kh}) \quad \forall k \in [K],\, \forall h \in [H'] \tag{9b}$$

$$\sum_{k\in[K]} u_{kh} = 1 \quad \forall h \in [H'] \tag{9c}$$

$$\mathrm{y}^k \in \mathcal{Y} \quad \forall k \in [K] \tag{9d}$$

$$\mathrm{W}\mathrm{y}^k \leq \mathrm{b} - \mathrm{T}\mathrm{x} \quad \forall k \in [K] \tag{9e}$$

$$\mathrm{x} \in \mathcal{X} \tag{9f}$$

$$u_{kh} \in \{0,1\} \quad \forall k \in [K],\, \forall h \in [H']. \tag{9g}$$

Note that the lower bond (L_r) changes at each Benders iteration since we solve problem (9) in each iteration by using an updated subset of scenarios Ξ'. We also suggest "warm-starting" the SP in every Benders iteration by re-using some of the y variables generated in previous iterations. Given a subset $\{\mathrm{y}_j\}_{j\in[J']}$ of recourse solutions, one can "filter" them using constraint (8e) and reuse the ones that satisfy this constraint for the new $\bar{\mathrm{x}}$ in problem (6) right away. Likewise, the scenarios (vertices of Ξ) generated in an iteration can be re-used in subsequent iterations of the Benders algorithm vertices since they do not depend on $\bar{\mathrm{x}}$. Such warm-staring techniques can substantially improve the performance of the proposed algorithm, even though we have not use them in our numerical tests. Figure 1 illustrates the proposed algorithm:

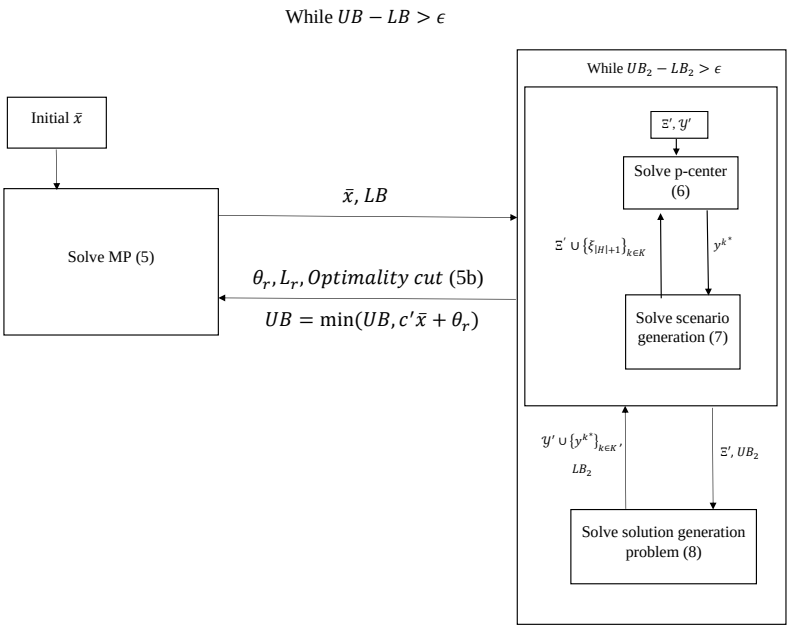

Figure 1: Double-oracle, Logic-based Benders decomposition algorithm

3 Finite Convergence

The proposed algorithm consists of three loops. The outer loop handles the first-stage decision variables by using logic-based Benders decomposition. The other two loops are used for solving SP, which is a MMMRCO problem. The inner loop has two step: scenario generation and solution generation. We show that the outer and the scenario generation loops terminate in a finite number of iterations, and the solution generation loop leads to convergence of LB and UB. These lemmas do not exploit the linearity of the cost and recourse constraint functions in problem (1).

Lemma 1. *Given that $\mathcal{X}$ is bounded, the number of optimality cuts* (5b) *generated and, thus, the number of iterations of the outer loop are finite.*

Proof. Since $\mathcal{X}$ is bounded and x are binary, there is a finite number of feasible solutions $\bar{\mathrm{x}}$ for the first-stage problem (5). Each first-stage solution corresponds to a single optimality cut (5b), generated by solving the the SP to obtain x^r, θ_r and L. Hence, the number of optimality cuts is finite, and so is the number of outer loop iterations. □

Lemma 2. *The maximum number of scenarios $\{\xi_h\}_{h\in H}$ that can be generated through the double oracle algorithm is finite, and equal to $\frac{|J|!}{K!(|J|-K)!}$.*

Proof. Given that $\mathcal{Y}$ is a bounded discrete set, its elements (feasible solutions) are finite and thus can be enumerated. The p-center problem (6) selects K solutions and assigns those selected solutions to scenarios. Moreover, those K selected solutions are used to generate worst-case scenarios through problem (7); and if the generated scenario violates constraint (6b), it is added to the set of scenarios. However, if all possible worst-case scenarios of all feasible $(\bar{\mathrm{x}}, \mathrm{y}^k)$ assignment in our subset of scenarios are available, then no generated scenario from problem (7) can violate (6b) and the iteration between (6) and (7) will terminate. If all combination of K out of $|\mathcal{Y}|$ are selected and used in problem (7), all possible worst-case scenarios for all feasible $(\bar{\mathrm{x}}, \mathrm{y}^k)$ are generated. Since $\mathcal{Y}$ is a bounded discrete set, there are exactly $\frac{|J|!}{k!(|J|-k)!}$ possible ways to select K elements from the set $\mathcal{Y}$. Consequently, at most there are $\binom{|J|}{k} = \frac{|J|!}{k!(|J|-k)!}$ possible solutions for (6). Hence, in the worst-case situation, with the finite number of iterations between the p-center problem, (6), and the scenario generation problem, (7), all possible worst-case scenarios are generated. therefore, no new scenario can violate (6b) and then, this step terminates. □

Lemma 3. *The upper and lowed bounds, obtained respectively by solving problems $P(\mathcal{Y}', \Xi)$ and $P(\mathcal{Y}, \Xi')$, will converge.*

Proof. The upper bound is achieved by solving $\mathtt{P}(\mathcal{Y}', \Xi)$. In order to get the optimal assignment, problem (6) with $\mathcal{Y}'$ must be solved. However, the optimal value of its objective function, which is the upper bound, can be achieved by solving $UB = \max_{h\in[H]} \min_{j\in[J']} \xi'_h \mathrm{Q}\mathrm{y}_j$, the proof of the reformulation is provided in the appendix A. On the other hand, By using the fixed set of scenarios, a set of K solutions are generated by solving (8), which can be reformulated as $LB = \max_{h\in[H']} \min_{\mathrm{y}^k\in\mathcal{Y}} \xi'_h \mathrm{Q}\mathrm{y}^k$, the detail of this reformulation is provided in the appendix A. There are two cases.

First: The optimal solutions y^{k^*} already exist in the subset of solutions $\mathcal{Y}'$, in this case, the optimal pair of solutions and scenarios will be found by solving (6) and its objective value is equivalent to $UB = \max_{h\in[H]} \min_{j\in[J']} \xi'_h \mathrm{Q}\mathrm{y}_j$. On the other hand, the optimization problem related to lower bound, (8), will generate y^{k^*} as optimal solution since the scenarios are fixed for both the lower and upper bound problems. In this case $LB = UB$ and the algorithm will terminate.

Second: The optimal solutions y^{k^*} are not in the subset of solutions $\mathcal{Y}'$. In this case, by using fixed scenarios and solving (8) a set of K solutions are generated that includes y^{k^*}. This set of solutions are added to the subset of solutions in the p-center problem (6). Consequently, the optimal pair of solutions and scenarios will be in the $\mathcal{Y}'$, and Ξ', respectively. Hence, by solving the p-center problem (6), the optimal pair of solutions and scenarios will be selected. Consequently, $LB = UB$ and the algorithm will terminate. □

Based on above mentioned lemmas, we can conclude that all three loops in proposed algorithm will terminate in finite number of iterations. Consequently, the proposed algorithm will converge in finite number of iterations.

4 Extensions

So far, we focused on the K-adaptability problem with linear objectives and constraints, binary decision variables in the first- and second-stage that can be extend to integer second-stage decision variables, and with objective uncertainty only in the second-stage. In this section, we show how the proposed algorithm can be extended to more general cases beyond the basic setting outlined earlier.

4.1 Second-stage Constraint Uncertainty

Similar to the algorithm proposed by Arslan et al. (2022) to solve the MMMRCO problem, our approach can be extended to the case when uncertainty affects both the objective function and constraints of the recourse problem. The extended problem can be formulated as follows:

$$\min_{\mathrm{x}\in\mathcal{X},\{\mathrm{y}^k\}_{k\in[K]}} \quad \mathrm{c}'\mathrm{x} + \sup_{\xi\in\Xi} \min_{k\in[K]} \{\xi'\mathrm{Q}\mathrm{y}^k : \ \mathrm{T}\mathrm{x} + \mathrm{W}(\xi)\mathrm{y}^k \leq \mathrm{b}\} \tag{10a}$$

$$\text{s.t.} \quad \mathrm{y}^k \in \mathcal{Y}, \ k \in [K], \tag{10b}$$

where $\mathrm{W}(\xi)$ is an affine mapping of uncertain parameters. Note the dependency of the SP constraints on ξ. To solve this problem, we use the same iterative algorithm explained earlier

but with a modified p-center problem $\mathtt{P}'(\mathcal{Y}', \Xi')$ by adding (11f), as follows:

$$\min_{\{z_j\}_{j\in[J']},\{v_{jh}\}_{j\in[J'],h\in[H']},w} \quad w \tag{11a}$$

$$\text{s.t.} \quad w \geq \sum_{j\in[J']} \xi_h' \mathrm{Q} \mathrm{y}_j v_{jh} \quad \forall h \in [H'] \tag{11b}$$

$$\sum_{j\in[J']} v_{jh} = 1 \quad \forall h \in [H'] \tag{11c}$$

$$\sum_{j\in[J']} z_j = K \tag{11d}$$

$$v_{jh} \leq z_j \quad \forall j \in [J'], \forall h \in [H'] \tag{11e}$$

$$v_{jh} = 0 \quad \forall j \in [J'],\ h \in [H'] : \exists s \in [S] \text{ s.t. } \mathrm{e}'_s(\mathrm{W}(\xi_h)\mathrm{y}_j - \mathrm{b} + \mathrm{T}\bar{\mathrm{x}}) > 0 \tag{11f}$$

$$v_{jh}, z_j \in \{0,1\} \quad \forall j \in [J'], \forall h \in [H']. \tag{11g}$$

In this formulation, there are s constraints with uncertain parameters, *i.e.,* $\mathrm{b} \in \mathbb{R}^s$, and e_s denotes the s-th column of the identity matrix I_s. Constraint (11f) prevents infeasible assignment of solution-scenario pairs. Moreover, problem (7) is modified by adding (12c) as follows:

$$\max_{\xi\in\Xi,\eta,\lambda\in\{0,1\}^K} \quad \eta \tag{12a}$$

$$\text{s.t.} \quad \eta \leq \xi' \mathrm{Q} \mathrm{y}^k + M\lambda^k \quad k \in [K] \tag{12b}$$

$$\mathrm{W}(\xi)\mathrm{y}^k \geq \mathrm{b} - \mathrm{T}\bar{\mathrm{x}} - M(1-\lambda^k) + \epsilon \quad k \in [K], \tag{12c}$$

where $\lambda^k = 1$ if the scenario ξ is such that y^k is infeasible for any of the s uncertain constraints, thus enforces that we do not consider $\xi' \mathrm{Q} \mathrm{y}^k$ for calculating the upper bound on η. In the case where $\mathrm{W}(\xi)\mathrm{y}^k = \mathrm{b} - \mathrm{T}\bar{\mathrm{x}}$, we would want $\lambda^k = 0$. Consequently, the small ϵ on the RHS of (12c) prevents $\lambda^k = 1$ for equality case. Moreover, the solution generation problem (8) is modified by considering constraint (13e) with uncertain parameters.

$$\min_{\{\mathrm{y}^k\}_{k\in[K]},\gamma,\{u_{kh}\}_{k\in[K],h\in[H']}} \quad \gamma \tag{13a}$$

$$\text{s.t.} \quad \xi_h' \mathrm{Q} \mathrm{y}^k \leq \gamma + M(1-u_{kh}) \quad \forall k \in [K],\ \forall h \in [H'] \tag{13b}$$

$$\sum_{k\in[K]} u_{kh} = 1 \quad \forall h \in [H'] \tag{13c}$$

$$\mathrm{y}^k \in \mathcal{Y} \quad \forall k \in [K] \tag{13d}$$

$$\mathrm{W}(\xi_h')\mathrm{y}^k \leq \mathrm{b} - \mathrm{T}\bar{\mathrm{x}} + M(1-u_{kh}) \quad \forall k \in [K], \forall h \in [H'] \tag{13e}$$

$$u_{kh} \in \{0,1\} \quad \forall k \in [K],\ \forall h \in [H']. \tag{13f}$$

One should note that, even though $\mathcal{V}$ can be modified to ensure that MP satisfies relatively complete recourse, it might happen that for some $\bar{\mathrm{x}} \in \mathcal{X} \cap \mathcal{V}$, there exist no set of K solutions

$\{\mathrm{y}^k\}_{k=1}^K$ that ensure that some y^k is always feasible under all $\xi \in \Xi$. For this reason, at the r-th iteration of the the logic-based Benders decomposition algorithm, SP might become infeasible for x_i^r, which is identified when problem (13) becomes infeasible for some Ξ'. At this point, Algorithm 1 should be modified to returns to the MP a feasibility cut of the form:

$$\sum_{i \in \mathcal{S}_r} \mathrm{x}_i - \sum_{\notin \mathcal{S}_r} \mathrm{x}_i \leq |\mathcal{S}_r| - 1,$$

in order to discard x_i^r from the set of feasible candidates, instead of producing an optimality cut of the form (5b).

4.2 First-stage Integer Decision Variables

The outer loop in the proposed algorithm depends on the first-stage variables being binary (*i.e.,* $\mathrm{x} \in \{0,1\}^n$) to generate logic-based Benders cuts of the type (5b). However, if the first-stage integer variables are not binary, but rather general integer, *i.e.,* $\mathcal{X} \in \mathbb{Z}_+^n$, one can simply apply the transformation $x_i = \sum_{p_i=0}^{P_i} 2^p u_{ip_i},\ i = 1, \ldots, n$, where $u_{ip_i} \in \{0,1\}$, and P_i depends on upper bound of x_i. Clearly, this generalization comes at the expense of increasing the number of variables in the first-stage problem, thus it might be efficient only for small values of P. It should be noted that the basic algorithm described in Section 2 can handle general integer recourse decisions since none of the algorithm steps depends on y^k being binary.

4.3 First-stage Objective Uncertainty

Even though Bertsimas & Caramanis (2010) defined K-adaptability such that the first-stage objective is deterministic, we extend our algorithm to the case when there is first-stage objective uncertainty, similar to Hanasusanto et al. (2015). A practical example of the K-adaptability problem with uncertainty in the first-stage is the multi-period portfolio selection problem where decisions about the allocation of budget among assets have to be made at the beginning of the investment horizon, thus are first-stage decisions. Indeed, asset returns are uncertain even at the outset, and the initial capital allocation decision cannot be postpone until this uncertainty is revealed. Hence, the here-and-now decisions are directly affected by the uncertain parameters. In this section, we differentiate between two cases of first-stage objective uncertainty: when it is independent from the second-stage uncertainty, and when some (or all) uncertain parameters affect both stages, *i.e.,* dependent uncertainty. We show how the proposed approach is tailored for each case.

4.3.1 Independent Uncertainty

Let us consider the following K-adaptability problem:

$$\min_{\mathrm{x} \in \mathcal{X}} \max_{\xi \in \Xi, \omega \in \Omega} \min_{k \in K} \{\omega' \mathrm{C}\mathrm{x} + \xi' \mathrm{Q}\mathrm{y}^k :\ \mathrm{T}\mathrm{x} + \mathrm{W}\mathrm{y}^k \leq \mathrm{b}\} \tag{14a}$$

$$\text{s.t.}\quad \mathrm{y}^k \in \mathcal{Y},\ k \in [K], \tag{14b}$$

where $\omega \in \mathbb{R}^c$, $C \in \mathbb{R}^{c \times n}$, and Ω is a compact and convex uncertainty set. Other variables and parameters are the same as those used in formulation (1). We assume that ω and ξ are disjoint sets of uncertain parameters. In this case, MP (5) is modified as follows:

$$\min_{\mathrm{x}\in\mathcal{X},\,\theta}\ \max_{\omega\in\Omega}\ \omega'\mathrm{C}\mathrm{x}+\theta \tag{15a}$$

$$\text{s.t. } \theta \geq (\theta_r - L)(\sum_{i\in\mathcal{S}_r} x_i - \sum_{\notin\mathcal{S}_r} x_i) - (\theta_r - L)(|\mathcal{S}_r| - 1) + L, \tag{15b}$$

which is a static RO problem that can be tractably reformulated by applying convex duality on the inner maximization. The SP does not change, and we can apply the double-oracle algorithm described in Section 2.2 to solve it.

4.3.2 Dependent Uncertainty

Next, we consider the K-adaptability problem variant addressed by (Hanasusanto et al., 2015), where the same uncertain parameters affect both stages, formulated as follows:

$$\min_{\mathrm{x}\in\mathcal{X},\{y^k\}_{k\in[K]}}\ \max_{\xi\in\Xi}\ \min_{k\in K}\ \xi'\mathrm{C}\mathrm{x}+\xi'\mathrm{Q}\mathrm{y}^k \tag{16a}$$

$$\text{s.t.}\quad \mathrm{y}^k\in\mathcal{Y},\ \mathrm{T}x+\mathrm{W}\mathrm{y}^k\leq \mathrm{b},\ k\in[K], \tag{16b}$$

where $\mathrm{C}\in\mathbb{R}^{q\times n}$ and the rest of parameters and variables are the same as in (1). This case can be reformulate as second stage uncertainty.

Let $\bar{\mathrm{Q}}=[\mathrm{C},\mathrm{Q}]\in\mathbb{R}^{q\times(n+m)}$, $\bar{\mathrm{T}}=\begin{bmatrix}\mathrm{I}\\ \mathrm{I}\\ \mathrm{T}\end{bmatrix}$, $\bar{\mathrm{W}}=\begin{bmatrix}-\mathrm{I} & 0\\ \mathrm{I} & 0\\ 0 & \mathrm{W}\end{bmatrix}$, and $\bar{b}=\begin{bmatrix}0\\ 0\\ \mathrm{b}\end{bmatrix}$. We have that problem (16) can be equivalently formulated as:

$$\min_{\mathrm{x}\in\mathcal{X},\{\bar{\mathrm{y}}^k\}_{k\in[K]}}\ \max_{\xi\in\Xi}\ \min_{k\in K}\ \xi'\bar{\mathrm{Q}}\bar{\mathrm{y}}^k \tag{17a}$$

$$\text{s.t.}\quad \bar{\mathrm{y}}\in\{0,1\}^n\times\mathcal{Y},\ \bar{\mathrm{T}}x+\bar{\mathrm{W}}\bar{\mathrm{y}}^k\leq\bar{\mathrm{b}},\ k\in[K]. \tag{17b}$$

4.4 Nonlinear Objective and Constraint Functions

Our algorithm can handle nonlinear K-adaptability problems of the form:

$$\min_{\mathrm{x}\in\mathcal{X}} f(\mathrm{x})+\max_{\xi\in\Xi}\ \min_{k\in K}\{g(\xi,\mathrm{y}^k):\ h(\xi,\mathrm{x},\mathrm{y}^k)\leq\mathrm{b}\} \tag{18a}$$

$$\text{s.t.}\quad \mathrm{y}^k\in\mathcal{Y},\ k\in[K], \tag{18b}$$

where $f:\mathcal{X}\mapsto\mathbb{R}$ is convex in x, $g:\Xi\times\mathcal{Y}\mapsto\mathbb{R}$ is affine in ξ and convex in y, and $h:\Xi\times\mathcal{X}\times\mathcal{Y}\mapsto\mathbb{R}$ is affine in ξ and jointly convex in x and y. In this case, `MP` (5) is modified by using $f(\mathrm{x})$ instead of $\mathrm{c}'\mathrm{x}$. Moreover, the p-center problem (11) is modified by replacing $\xi_h'\mathrm{Q}y_jv_{jh}$ and $\xi'\mathrm{W}y+\mathrm{T}\bar{\mathrm{x}}$ with $g(\xi_h,y_j)$ and $h(\xi_h,y_j,\bar{\mathrm{x}})$, respectively. In the p-center problem, v_{jh} and z_j are decision variables while y_j, ξ_h, and $\bar{\mathrm{x}}$ are constant. Consequently, regardless of the type of functions $g(\cdot,\cdot)$ and $h(\cdot,\cdot,\cdot)$, the p-center problem finds the optimal K solutions and assigns them to scenarios. Moreover, the scenario generation problem (7) is rewritten as follows:

$$\max_{\xi\in\Xi,\eta,\lambda\in\{0,1\}^K}\quad \eta \tag{19a}$$

$$\text{s.t.}\quad \eta\leq g(\xi,\mathrm{y}^k)+M\lambda^k \qquad k\in[K] \tag{19b}$$

$$h(\xi,\mathrm{y}^k,\bar{\mathrm{x}})\geq\mathrm{b}-M(1-\lambda^k)+\epsilon \quad k\in[K], \tag{19c}$$

which again takes the form of a mixed integer LP given our assumption that $g(\cdot,\cdot)$ and $h(\cdot,\cdot,\cdot)$ be affine in ξ. Finally, the solution generation problem (8) is changed by replacing $\xi_h' \mathrm{Q} \mathrm{y}^k$ and $\mathrm{W}\mathrm{y}^k + \mathrm{T}\bar{\mathrm{x}}$ with $g(\xi_h, \mathrm{y}^k)$ and $h(\xi_h, \mathrm{y}^k, \bar{\mathrm{x}})$, respectively.

5 Numerical Results

This section presents and discusses the numerical results obtained by implementing the proposed algorithm to solve three problems: the shortest path problem, the knapsack problem, and a generic K-adaptability problem. These results are compared to those obtained from state-of-the-art algorithms proposed in (MILP reformulation of Hanasusanto et al. (2015), the iterative algorithm (IA) of Chassein et al. (2019), the row-and-column generation (RCG) algorithm proposed by Goerigk et al. (2020), the scenario generation (SG) approach developed by Arslan et al. (2022), and Branch-and-Bound (BB) method proposed by Subramanyam et al. (2020)). The proposed algorithm, Double-Oracle (DO), MILP, IA, and RCG were coded on Python 3.10.4 using Jupyter, SG is available here, and BB is also available here). The subproblems were solved using CPLEX called through CPLEX-CMD on a Linux laptop with an 8th generation Intel Core i7 7700 processor and 16 GB RAM. The time limit and the relative optimality gap were set, respectively, to two hours (7200 seconds) and 5%.

5.1 Shortest Path Problem

The first problem used to evaluate the proposed algorithm is the adaptive shortest path problem, previously studied in Hanasusanto et al. (2015), Chassein et al. (2019). This problem aims to select a subset of network arcs with the least total cost to form a path from a source s to a destination t when arc costs are uncertain. In the K-adaptability variant of the problem, K paths are pre-formed and the shortest (least costly) among them is selected once the actual costs ate realized. We used test instances from Arslan et al. (2022), available here.

Formally, the problem can be described as follows: A network $(\mathcal{V}, \mathcal{A})$ has the cost of each arc $(i,j) \in \mathcal{A}$ characterized as $c_{ij} = \bar{c}_{ij} + \xi_{ij}\hat{c}_{ij}$, where $\bar{c}_{ij}$ is the nominal cost and $\hat{c}_{ij}$ is the maximal deviation. The primary uncertain parameter ξ belongs to the budgeted uncertainty set $\Xi = \{\xi \in [0,1] | \sum_{(i,j)\in A} \xi_{i,j} \leq \Gamma\}$, where Γ is an "uncertainty budget" that controls the size of uncertainty set. With that, the problem is formulated as follows:

$$\min_{x_{ij}\in[0,1]^n} \sum_{(i,j)\in\mathcal{A}} c_{ij}x_{ij} \tag{20a}$$

$$\text{s.t.} \sum_{(i,j)\in\delta^+(i)} x_{ij} - \sum_{(i,j)\in\delta^-(i)} x_{ij} = b_i,\ \forall i \in \mathcal{V}, \tag{20b}$$

where $b_s = -1$, $b_t = 1$, and $b_i = 0$ for $i \in \mathcal{V}/\ \{s,t\}$ and the sets δ_i^+ and δ_i^-represent the forward and backward starts of node $i \in \mathcal{V}$, respectively.

We solve the problem in different sizes of $|\mathcal{V}| \in \{20, 25, 40, 50\}$. For each problem size, we considered $k \in \{2,3,4,5,6\}$. Finally, each instance is solved based on different $\Gamma \in \{3,6\}$. Ten randomly-generated instances were solved for each combination of ν, k, and Γ. We compare the results of our algorithm with those based on the MILP reformulation

of Hanasusanto et al. (2015), the row-and-column generation (RCG) algorithm proposed by Goerigk et al. (2020), the iterative algorithm (IA) of Chassein et al. (2019) and the scenario generation (SG) approach developed by Arslan et al. (2022).

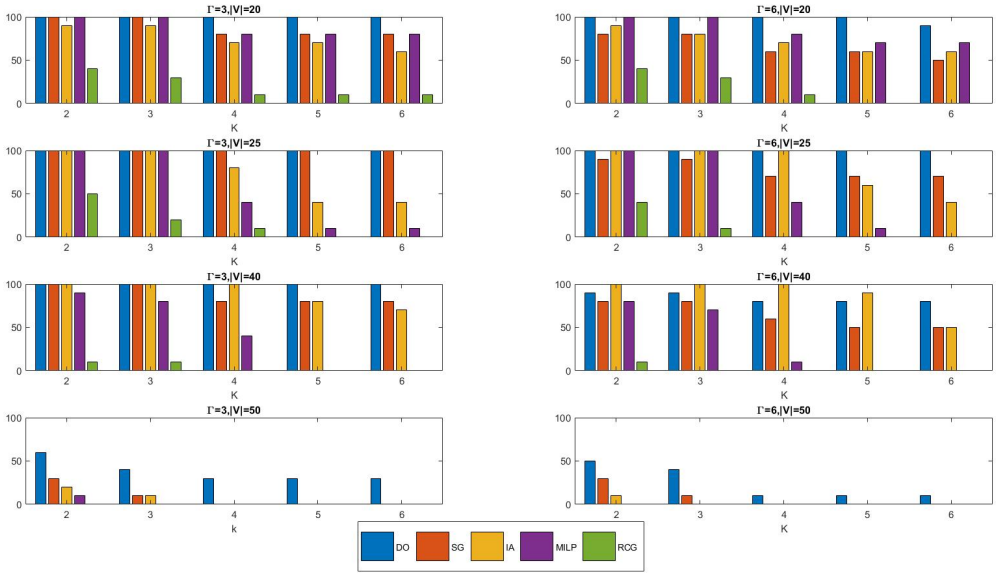

Figure 2: Percentage of solved instances of the shortest path problem

Figure 2 shows the percentage of instances solved by each algorithm within the cut-off time of two hours. Intuitively, as the problem size increases (in terms of both $|\mathcal{V}|$ and K), fewer instances are solved to optimality by all algorithms. Nevertheless, our proposed algorithm, labeled "Double-oracle", shows better performance than all other algorithms. For example, our proposed algorithm solved all instances with $\Gamma = 3$ and $k \in \{2, 3, 4, 5\}$, while none of the other algorithms could solve all of these instances within the cut-off time. Moreover, the proposed algorithm solved $30 - 60\%$ of the instances with $|\mathcal{V}| = 50$, $\Gamma = 3$, and $k \in \{2, 3, 4, 5, 6\}$, while the next best algorithm is the scenario generation method proposed by Arslan et al. (2022) that could not solve any instances with size 50 and $K > 3$. Details of the results can be found in Tables 1 and 2 in Appendix B.

The comparison results for $\Gamma = 6$, shown in Figure 2, exhibit the same pattern. Our algorithm has a significant performance over the benchmark algorithms, except the iterative approach proposed by Chassein et al. (2019), when the uncertainty budget Γ is doubled. The iterative approach solved 100% of instances with $|\mathcal{V}| = 40$, $k \in \{2, 3, 4\}$, and $\Gamma = 6$, while our algorithm solved 80% to 90% of instances with the same size. However, for $|\mathcal{V}| = 50$, performance of the iterative algorithm deteriorates, as it could only solve $0-10\%$ of the instances, whereas the double-oracle algorithm solved $10 - 50\%$ of these instances. These results clearly show the performance advantage of the proposed algorithm over other algorithms proposed in the literature, especially for large-size problems. It is worthy to note that the iterative approach of Chassein et al. (2019), unlike ours, cannot guarantee global optimality since it uses a fixing heuristic to handle a bilinear term in each iteration.

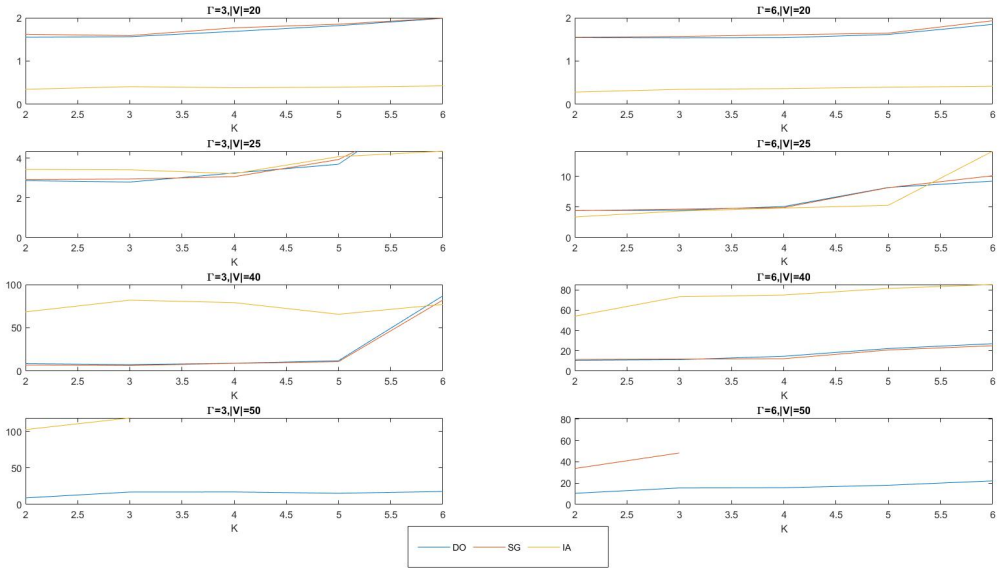

Figure 3: Average CPU times for the solved instances of the adaptive shortest path problem

Another important performance measure for comparing algorithms is the processing (CPU) time. We compare the CPU time of our algorithm to those of the iterative algorithm by Chassein et al. (2019) and the scenario generation approach by Arslan et al. (2022). Based on preliminary results, the other algorithms were too slow in comparison to the ones selected, thus they were excluded. Figure 3 illustrate the average CPU times for different algorithms of the instanced solved within the cut-off time for different problem sizes and with $\Gamma = 3$ and $\Gamma = 6$. It can be seen that the iterative algorithm had better performance than our algorithm in small-size instances (with an average difference of about 2 seconds). However, for the largest problem size ($\mathcal{V} = 50$), the iterative algorithm could not solve any instance with $K > 3$ within the cut-off time, whereas our approach solved instances of the same size with $K = 6$ in less than 20 seconds. We observe that the double-oracle algorithm has much smaller CPU times for large instances in comparison to the iterative algorithm, which also does not guarantee optimality. The average CPU times for the double-oracle algorithm and the scenario generation algorithm of Arslan et al. (2022) are almost identical. For example, the average CPU times of our algorithm for instances with $\mathcal{V} \in \{20, 25, 40\}$, $\Gamma = 3$, and $K = \{2, 3, 4, 5, 6\}$ were between 1.55 and 86.95 seconds, while those for the scenario generation algorithm were $1.62 - 82.07$ seconds. However, the scenario generation algorithm could not solve any instances with $\mathcal{V} = 50$, and $K > 3$, while our algorithm solved some of these instances within the cut-off time.

5.2 Adaptive Knapsack Problem

We then tested on the adaptive knapsack problem, for which the objective is to prepare K different combinations of items that respect the capacity constraint without exact knowledge of

their profit. Let $n \in \{1, ..., n\}$ be the set of items, w_i and p_i, respectively, be the weight and profit of item $i \in n$, and b is the available budget. The feasible set is defined as $\mathcal{X} = \{\mathrm{x} \in \{0,1\}^n | \sum_{i \in N} w_i x_i \leq b\}$. The goal is to find the best combination of items that maximizes the profit $\mathrm{p}'\mathrm{x}$. The uncertain parameter p_i is assumed to follow $p_i = (1 + \sum_{j \in m} \frac{\Phi_{ij}\xi_j}{2})\bar{p}_i$, where $\bar{p}_i$ is the nominal profit, $|m|$ is the number of uncertain factors and $\Phi \in \mathbb{R}^{|n| \times |m|}$ is the factor loading matrix. The i-th row of Φ is characterized by the set $\{\Phi_i \in [-1,1]^{|m|} \mid \sum_{j \in m} |\Phi_{ij}| = 1\}$. As a result, the realized profit of each object $i \in n$ remains within the interval $[\bar{p}_i - \frac{\bar{p}_i}{2}, \bar{p}_i + \frac{\bar{p}_i}{2}]$. We solve the problem in different sizes $n \in \{100, 150, 200, 300\}$ and different values of $K \in \{2, 3, 4, 5, 6\}$. Ten instances of each combination of n, and K are solved, and the results obtained from our algorithm are compared to those of the the IA of Chassein et al. (2019), the MILP reformulation of Hanasusanto et al. (2015), and the SG method of Arslan et al. (2022). These results are summarized in Tables 3 and 4.

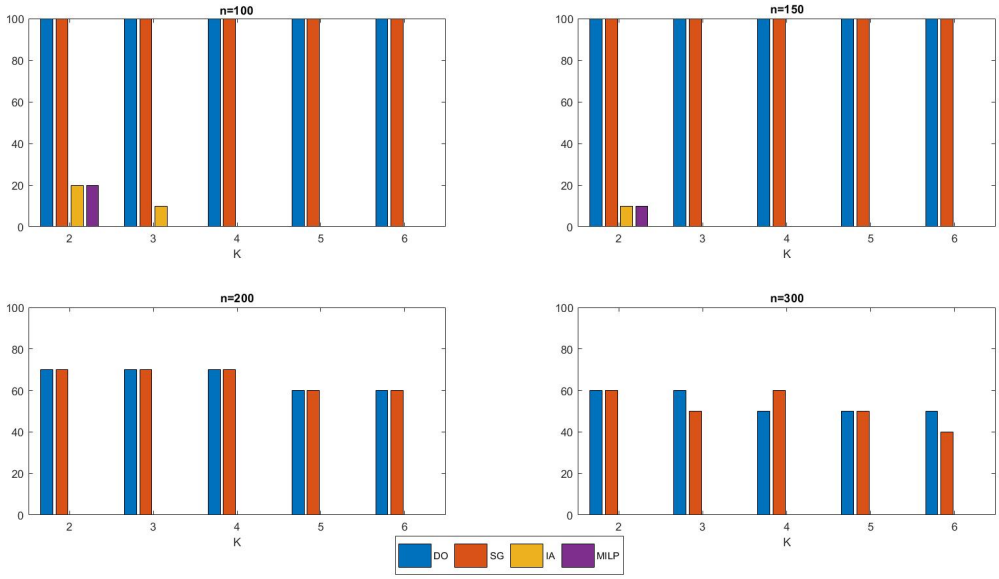

Figure 4: Percentage of solved instances of the adaptive knapsack problem

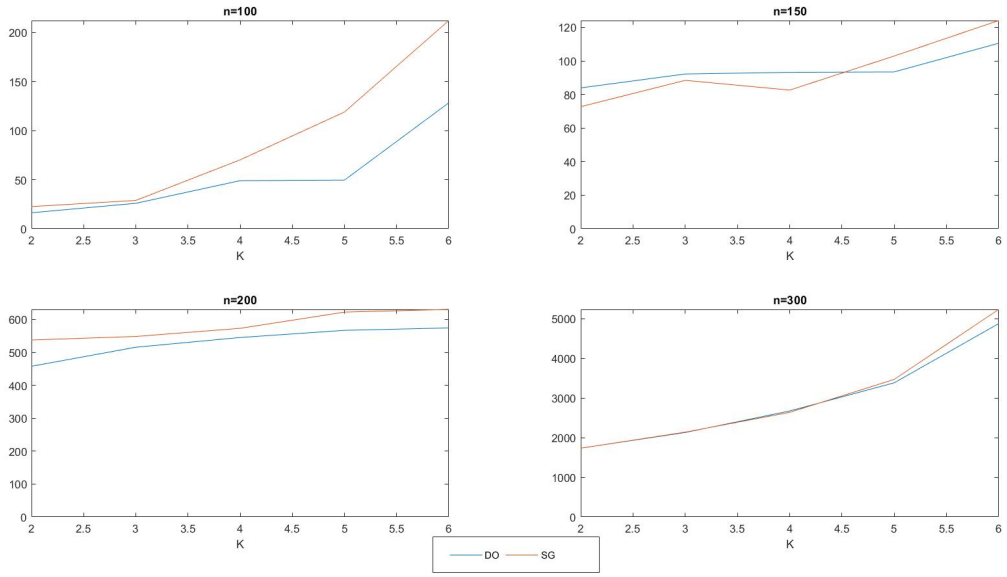

Figure 5: Average CPU times for the solved instances of the adaptive knapsack problem

Figure 4 shows the percentage of problems solved to proven optimality by each algorithm. It can be seen that for different problem sizes, our algorithm and the scenario generation method were able to solve almost the same percentage of test instances. However, MILP and IA algorithms could not solve any instances with $n = 100$ and $K > 2$. By increasing n and K, the percentage of solved instances by SG and DO dropped to $50 - 60\%$, showing the significant impact of n and K on their performance. Figure 5 shows the average CPU times for the algorithms with different problem sizes. Since the MILP reformulation and IA could not solve large-size instances, we only compared against the SG algorithm in this round of experiments. It can be seen that the average CPU times of our algorithm and SG approach are very close, with no clear advantage for either algorithm. However, our algorithm is faster than IA and MILP reformulation.

5.3 Problems with Actual First-Stage Decisions

Finally, we tested on binary K-adaptability problems with actual first-stage decisions of the form:

$$\min_{\mathrm{x},\{\mathrm{y}^k\}_{k\in[K]}} \max_{\xi\in\Xi} \min_{k\in[K]} \sum_i a_i x_i + \sum_j c_j y_j^k \tag{21a}$$

$$\text{s.t.} \sum_i x_i = b, \tag{21b}$$

$$\sum_i d_i x_i + \sum_j f_j y_j \geq l, \tag{21c}$$

where $\mathrm{x} \in \{0,1\}^n, \mathrm{y} \in \{0,1\}^m$ are the first- and second-stage decision variables, respectively. Moreover, we set $c_j = \bar{c}_j - \xi_j \hat{c}_j$, where $\bar{c}_j$ is the nominal value that is drawn at random from the uniform distribution $U(8,12)$ and $\hat{c}_j$ is the maximal deviation, set equal to 25% of the nominal value. Moreover, a_i, d_i f_j are generated randomly based on the uniform distributions $U(8,12)$, $U(50,100)$ and $U(80,90)$, respectively. Finally, we set $b = 10$, and $l = 0$. The uncertain parameter ξ_j follows $\Xi = \{\xi \in [0,1] | \sum_j \xi_j \leq \Gamma\}$. 10 random instances of each size and uncertainty budget combination were solved. We compared the results of our algorithm to those of the MILP reformulation of Hanasusanto et al. (2015) and Branch-and-Bound (BB) approach of Subramanyam et al. (2020), which is adapted to each application using the authors' implementation available here, for different instance sizes $n \in \{20,30,40,50\}$ and $m \in \{20,30,40,50\}$. Complete results are presented in Tables 5 and 6 in Appendix B.

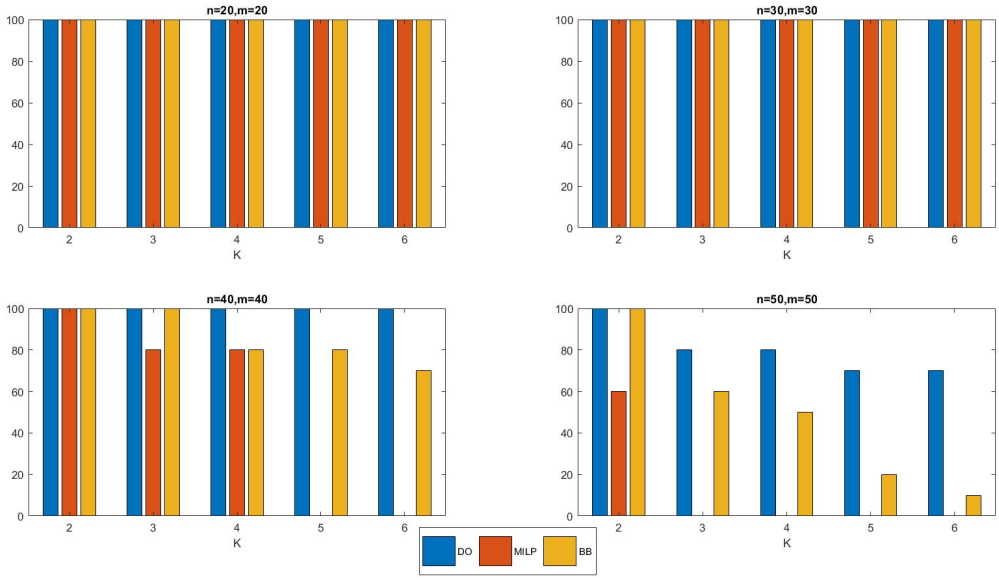

Figure 6: Percentage of solved instances of the two-stage problem

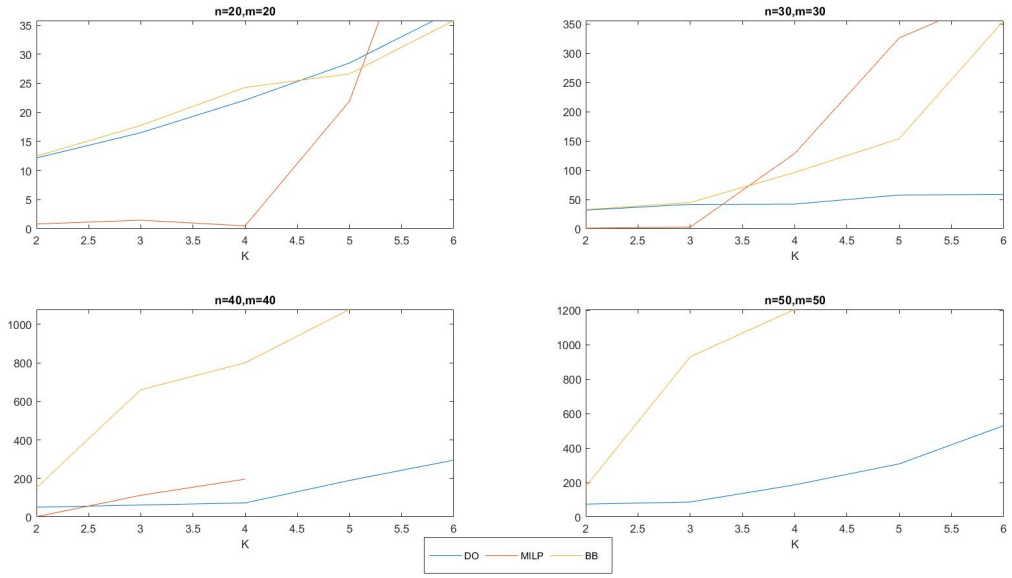

Figure 7: Average CPU times for the solved instances of the two-stage problem

Figure 6 shows the percentage of instances solved by each algorithm. For small problem sizes (*i.e.,* $n = m = 20, 30$), all algorithms were able to solve all instances. However, as the problem sizes were increased, the performance advantage of our algorithm became clear, especially for large values of K. For example, the MILP reformulation could solve 60% of instances of size $n = m = 50$ only with $K = 2$, but none when $K > 2$. In contrast, our algorithm solved the vast majority of instances of the same size to proven optimality within the cut-off time even with $K = 6$. Similar insights could be drawn from Figure 7, which shows the CPU times of all algorithms for the problems that were solved within the cutoff time. Again, it is clear that while the MILP reformulation could solve small instances with small K values efficiently, our algorithm significantly outperforms it in large instances. Furthermore, the proposed algorithm scales well in K, thus can result in high adaptability in the face of parameter uncertainty.

6 Conclusion

In this paper, we proposed a new algorithm for solving K-adaptability problems, possibly involving constraint and objective function uncertainty and non-linear functions. The algorithm combines discrete scenario generation and p-center assignment problem with Logic-based Benders decomposition, where the sub-problem is a MMMRCO. Numerical experiments on benchmark test instances demonstrated that the proposed algorithm performs well with large-size instances and for large K values. It was able to solve larger instances to optimality and produce the result faster than the state-of-the-art algorithms from the existing K-adaptability and MMMRCO literature. The proposed algorithm could solve instances of adaptive knapsack problem with up to 300 items with k up to 6, while the existing algorithms could solve only a

small percentage of the instances with up to 100 items and 250 items with k up to 4. We also showed that the proposed algorithm can solve the shortest path problem with up to 50 nodes while the existing algorithm barely can solve this problem with more than 50 nodes.

The proposed algorithm dominates the iterative scheme of Chassein et al. (2019) from different points of view. First, our algorithm is faster than the iterative algorithm in large-scale adaptive Knapsack and shortest path problems. Second, The iterative scheme of Chassein et al. (2019) reaches its limit with $K = 2$ with 50 nodes for the shortest path problem while our algorithm easily solves problems with larger K values up to 6. Our algorithm also performs better than the RCG algorithm of Goerigk et al. (2020) that is tailored solely for budgeted uncertainty sets. The RCG algorithm still cannot handle the shortest path problem with $n > 40$ and $K > 3$, whereas results show that our algorithm works efficiently for $n > 50$. Our algorithm also dominates the MILP reformulation of Hanasusanto et al. (2015) that cannot solve the shortest path problem with 40 nodes and $K > 3$. Moreover, the MILP reformulation cannot handle the knapsack problem with more than 100 items. Our algorithm also shows better performance with respect to the percentage of solved instances and processing time than the SG method of Arslan et al. (2022) for the shortest path problem. However, the proposed double-oracle algorithm and the SG method have almost the same performance on the adaptive Knapsack problem.

Finally, the proposed algorithm leverages a logic-based Benders decomposition to solve the K-adaptability faster than any algorithm in the literature. It can handle instances with up to 50 binary decision variables in each of the first- and second-stage problems. Consequently, we can conclude that not only is our algorithm more efficient with regard to solving large-scale problems in the reasonable time, but also the algorithm is generic and can be extended to tackle different variants of the problems such as nonlinear objective functions and constraints, uncertain parameters in constraints and first-stage objective function.

For future research, we plan to investigate how the proposed algorithm can be extended to general RO problems with combinatorial recourse and to K-adaptability problems with continuous first-stage variables. We also propose using efficient methods to solve the p-center problem for future research to increase the efficiency of the proposed algorithm. Another important future research can be improving the lower bound used in optimality cut, a better lower bound can lead to faster convergence to optimal solution in our proposed algorithm.

References

Arslan, A. N., Poss, M., & Silva, M. (2022). Min-max-min robust combinatorial optimization with few recourse solutions. Working paper or preprint.
URL `https://hal.archives-ouvertes.fr/hal-02939356`

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

Bertsimas, D., Brown, D. B., & Caramanis, C. (2011). Theory and applications of robust optimization. *SIAM review*, *53*(3), 464–501.

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

Bertsimas, D., & Georghiou, A. (2015). Design of near optimal decision rules in multistage adaptive mixed-integer optimization. *Operations Research*, *63*(3), 610–627.

Bertsimas, D., & Georghiou, A. (2018). Binary decision rules for multistage adaptive mixed-integer optimization. *Mathematical Programming*, *167*(2), 395–433.

Bertsimas, D., Litvinov, E., Sun, X. A., Zhao, J., & Zheng, T. (2012). Adaptive robust optimization for the security constrained unit commitment problem. *IEEE transactions on power systems*, *28*(1), 52–63.

Buchheim, C., & Kurtz, J. (2017). Min–max–min robust combinatorial optimization. *Mathematical Programming*, *163*(1-2), 1–23.

Chassein, A., Goerigk, M., Kurtz, J., & Poss, M. (2019). Faster algorithms for min-max-min robustness for combinatorial problems with budgeted uncertainty. *European Journal of Operational Research*, *279*(2), 308–319.

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

Dhamdhere, K., Goyal, V., Ravi, R., & Singh, M. (2005). How to pay, come what may: Approximation algorithms for demand-robust covering problems. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, (pp. 367–376). IEEE.

Georghiou, A., Wiesemann, W., & Kuhn, D. (2015). Generalized decision rule approximations for stochastic programming via liftings. *Mathematical Programming*, *152*(1), 301–338.

Goerigk, M., Kurtz, J., & Poss, M. (2020). Min–max–min robustness for combinatorial problems with discrete budgeted uncertainty. *Discrete Applied Mathematics*, *285*, 707–725.

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

Iancu, D. A., Sharma, M., & Sviridenko, M. (2013). Supermodularity and affine policies in dynamic robust optimization. *Operations Research*, *61*(4), 941–956.

Jiang, R., Zhang, M., Li, G., & Guan, Y. (2012). Benders' decomposition for the two-stage security constrained robust unit commitment problem. In *IIE Annual Conference. Proceedings*, (p. 1). Institute of Industrial and Systems Engineers (IISE).

Kuhn, D., Wiesemann, W., & Georghiou, A. (2011). Primal and dual linear decision rules in stochastic and robust optimization. *Mathematical Programming*, *130*(1), 177–209.

Laporte, G., & Louveaux, F. V. (1993). The integer l-shaped method for stochastic integer programs with complete recourse. *Operations research letters*, *13*(3), 133–142.

Subramanyam, A., Gounaris, C. E., & Wiesemann, W. (2020). K-adaptability in two-stage mixed-integer robust optimization. *Mathematical Programming Computation*, *12*(2), 193–224.

Thiele, A., Terry, T., & Epelman, M. (2009). Robust linear optimization with recourse. *Rapport technique*, (pp. 4–37).

Zhao, L., & Zeng, B. (2012). Robust unit commitment problem with demand response and wind energy. In *2012 IEEE power and energy society general meeting*, (pp. 1–8). IEEE.

A Proofs

Proposition 4. *The objective function value, w, of the p-center problem* (6) *can be achieved by solving* $w^* = \max_{h\in[H]} \min_{j\in[J']} \xi'_h \mathrm{Q}\mathrm{y}_j$

Proof. There are $|H|$ constraints in the form of (6b) while constraints (6c) force the problem to select one pair of scenario and solution in each constraint (6b). The objective value should be minimum of w that is greater than selected pairs of scenarios and solutions in each h constraint (6b). To find the optimal value of w, minimum of $\xi'_h \mathrm{Q}\mathrm{y}_j$ for each $h \in [H']$ is selected, then optimal w^* will be maximum of selected pairs of scenarios and solutions in each constraint (6b). If $w^* > \max_{h\in[H]} \min_{j\in[J']} \xi'_h \mathrm{Q}\mathrm{y}_j$, then it cannot be optimal because there is a feasible solution with lower objective value which is $\max_{h\in[H]} \min_{j\in[J']} \xi'_h \mathrm{Q}\mathrm{y}_j$. On the other hand, if $w^* < \max_{h\in[H]} \min_{j\in[J']} \xi'_h \mathrm{Q}\mathrm{y}_j$, then some of constraints (6b) whose pair of solutions and scenarios are greater than w will be violated. Consequently, $UB = w^* = \max_{h\in[H]} \min_{j\in[J']} \xi'_h \mathrm{Q}\mathrm{y}_j$. □

Proposition 5. *The objective function value, γ, of the problem* (8) *can be achieved by solving* $\gamma^* = \max_{h\in[H']} \min_{\mathrm{y}^k\in\mathcal{Y}} \xi'_h \mathrm{Q}\mathrm{y}^k$.

Proof. Let us assume that y^{k^*} is the optimal solution of problem (8). There are $|K| \times |H'|$ constraints of (8b). For each $h \in [H']$ there are $|K|$ constraints in form of (8b). Constraints (8c) force the problem to select one pair of scenarios and solutions for each $h \in [H']$. The objective function is minimization, consequently, minimum of $(\mathrm{y}^{k^*}, \xi'_h)$ for each $h \in [H']$ are selected. Since γ should be greater than $\xi'_h \mathrm{Q}\mathrm{y}^{k^*}$, then maximum of selected pairs will be optimal objective value γ^*. Consequently, the optimal value of objective function, γ^* can be achieved by solving $\gamma^* = \max_{h\in[H']} \min_{\mathrm{y}^k\in\mathcal{Y}} \xi'_h \mathrm{Q}\mathrm{y}^k$. □

B Results

Table 1: Percentage of solved shortest path instances for $\Gamma = 3$

Size	Algorithms	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 6$
20	Double-Oracle	100%	100%	100%	100%	100%
	MILP Hanasusanto et al. (2015)	100%	100%	80%	80%	80%
	IA Chassein et al. (2019)	90%	90%	70%	70%	60%
	RCG Goerigk et al. (2020)	40%	30%	10%	10%	10%
	SG Arslan et al. (2022)	100%	100%	80%	80%	80%
25	Double-Oracle	100%	100%	100%	100%	100%
	MILP Hanasusanto et al. (2015)	100%	100%	40%	10%	10%
	IA Chassein et al. (2019)	100%	100%	80%	40%	40%
	RCG Goerigk et al. (2020)	50%	20%	10%	0%	0%
	SG Arslan et al. (2022)	100%	100%	100%	100%	100%
40	Double-Oracle	100%	100%	100%	100%	100%
	MILP Hanasusanto et al. (2015)	90%	80%	40%	0%	0%
	IA Chassein et al. (2019)	100%	100%	100%	80%	70%
	RCG Goerigk et al. (2020)	10%	10%	0%	0%	0%
	SG Arslan et al. (2022)	100%	100%	80%	80%	80%
50	Double-Oracle	60%	40%	30%	30%	30%
	MILP Hanasusanto et al. (2015)	10%	0%	0%	0%	0%
	IA Chassein et al. (2019)	20%	10%	0%	0%	0%
	RCG Goerigk et al. (2020)	0%	0%	0%	0%	0%
	SG Arslan et al. (2022)	30%	10%	0%	0%	0%

Table 2: Percentage of solved shortest path instances for $\Gamma = 6$

Size	Algorithms	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 6$
20	Double-Oracle	100%	100%	100%	100%	90%
	MILP Hanasusanto et al. (2015)	100%	100%	80%	70%	70%
	IA Chassein et al. (2019)	90%	80%	70%	60%	60%
	RCG Goerigk et al. (2020)	40%	30%	10%	0%	0%
	SG Arslan et al. (2022)	80%	80%	60%	60%	50%
25	Double-Oracle	100%	100%	100%	100%	100%
	MILP Hanasusanto et al. (2015)	100%	100%	40%	10%	0%
	IA Chassein et al. (2019)	100%	100%	100%	60%	40%
	RCG Goerigk et al. (2020)	40%	10%	0%	0%	0%
	SG Arslan et al. (2022)	90%	90%	70%	70%	70%
40	Double-Oracle	90%	90%	80%	80%	80%
	MILP Hanasusanto et al. (2015)	80%	70%	10%	0%	0%
	IA Chassein et al. (2019)	100%	100%	100%	90%	50%
	RCG Goerigk et al. (2020)	10%	0%	0%	0%	0%
	SG Arslan et al. (2022)	80%	80%	60%	50%	50%
50	Double-Oracle	50%	40%	10%	10%	10%
	MILP Hanasusanto et al. (2015)	0%	0%	0%	0%	0%
	IA Chassein et al. (2019)	10%	0%	0%	0%	0%
	RCG Goerigk et al. (2020)	0%	0%	0%	0%	0%
	SG Arslan et al. (2022)	30%	10%	0%	0%	0%

Table 3: Average CPU time for solved shortest path instances for $\Gamma = 3$

Size	Algorithms	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 6$
20	Double-Oracle	1.55	1.56	1.69	1.82	1.99
	MILP Hanasusanto et al. (2015)	0.46	0.96	3.09	6.54	433.08
	IA Chassein et al. (2019)	0.35	0.41	0.38	0.40	0.43
	RCG Goerigk et al. (2020)	1749.49	2569.55	2955.92	3491.93	3564.87
	SG Arslan et al. (2022)	1.62	1.59	1.77	1.85	1.99
25	Double-Oracle	2.87	2.80	3.26	3.70	7.23
	MILP Hanasusanto et al. (2015)	12.69	53.75	524.45	2.50	16.55
	IA Chassein et al. (2019)	3.44	3.42	3.23	4.08	4.36
	RCG Goerigk et al. (2020)	4373.73	6423.87	7129.75	NA	NA
	SG Arslan et al. (2022)	2.93	2.95	3.07	3.94	6.52
40	Double-Oracle	8.63	7.50	9.04	11.97	86.95
	MILP Hanasusanto et al. (2015)	90.61	299.26	873.76	NA	NA
	IA Chassein et al. (2019)	68.45	81.99	79.13	65.68	77.08
	RCG Goerigk et al. (2020)	3498.99	5139.10	NA	NA	NA
	SG Arslan et al. (2022)	6.91	6.51	8.93	10.85	82.07
50	Double-Oracle	9.00	17.04	17.15	15.36	17.78
	MILP Hanasusanto et al. (2015)	355.99	NA	NA	NA	NA
	IA Chassein et al. (2019)	102.68	118.69	NA	NA	NA
	RCG Goerigk et al. (2020)	NA	NA	NA	NA	NA
	SG Arslan et al. (2022)	28.93	NA	NA	NA	NA

Table 4: Average CPU time (s) for solved shortest path instances for $\Gamma = 6$

Size	Algorithms	$k = 2$	$k = 3$	$k = 4$	$k = 5$	$k = 6$
20	Double-Oracle	1.54	1.54	1.54	1.61	1.85
	MILP Hanasusanto et al. (2015)	0.72	1.26	3.31	7.21	119.40
	IA Chassein et al. (2019)	0.28	0.35	0.36	0.40	0.42
	RCG Goerigk et al. (2020)	1766.99	2595.24	2985.48	NA	NA
	SG Arslan et al. (2022)	1.55	1.57	1.61	1.64	1.93
25	Double-Oracle	4.43	4.47	5.08	8.19	9.23
	MILP Hanasusanto et al. (2015)	14.31	75.40	151.24	389.98	NA
	IA Chassein et al. (2019)	3.39	4.34	4.84	5.28	14.11
	RCG Goerigk et al. (2020)	4417.47	6488.11	NA	NA	NA
	SG Arslan et al. (2022)	4.40	4.66	4.91	8.16	10.10
40	Double-Oracle	10.59	11.27	14.58	22.20	26.96
	MILP Hanasusanto et al. (2015)	70.59	664.92	1044.60	NA	NA
	IA Chassein et al. (2019)	54.03	73.24	74.91	81.40	85.20
	RCG Goerigk et al. (2020)	3533.98	NA	NA	NA	NA
	SG Arslan et al. (2022)	11.41	11.81	12.20	20.76	24.96
50	Double-Oracle	10.46	15.45	15.70	18.01	22.08
	MILP Hanasusanto et al. (2015)	NA	NA	NA	NA	NA
	IA Chassein et al. (2019)	81.04	NA	NA	NA	NA
	RCG Goerigk et al. (2020)	NA	NA	NA	NA	NA
	SG Arslan et al. (2022)	33.67	48.21	NA	NA	NA

Table 5: Percentage of solved knapsack problem instances

Size	Algorithms	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$
100	Double-Oracle	100%	100%	100%	100%	100%
	MILP Hanasusanto et al. (2015)	20%	0%	0%	0%	0%
	IA Chassein et al. (2019)	20%	10%	0%	0%	0%
	SG Arslan et al. (2022)	100%	100%	100%	100%	100%
150	Double-Oracle	100%	100%	100%	100%	100%
	MILP Hanasusanto et al. (2015)	10%	0%	0%	0%	0%
	IA Chassein et al. (2019)	10%	0%	0%	0%	0%
	SG Arslan et al. (2022)	100%	100%	100%	100%	100%
200	Double-Oracle	70%	70%	70%	60%	60%
	MILP Hanasusanto et al. (2015)	0%	0%	0%	0%	0%
	IA Chassein et al. (2019)	0%	0%	0%	0%	0%
	SG Arslan et al. (2022)	70%	70%	70%	60%	60%
300	Double-Oracle	60%	60%	50%	50%	50%
	MILP Hanasusanto et al. (2015)	0%	0%	0%	0%	0%
	IA Chassein et al. (2019)	0%	0%	0%	0%	0%
	SG Arslan et al. (2022)	60%	50%	60%	50%	40%

Table 6: Average CPU time (s) of solved knapsack problem instances

Size	Algorithms	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$
100	Double-Oracle	16.29	25.92	49.01	49.60	128.35
	MILP Hanasusanto et al. (2015)	3829.95	NA	NA	NA	NA
	IA Chassein et al. (2019)	999.12	1460.23	NA	NA	NA
	SG Arslan et al. (2022)	22.60	28.85	70.23	118.96	212.19
150	Double-Oracle	83.93	92.21	93.19	93.45	110.52
	MILP Hanasusanto et al. (2015)	4072.81	NA	NA	NA	NA
	IA Chassein et al. (2019)	70.94	NA	NA	NA	NA
	SG Arslan et al. (2022)	72.73	88.41	82.61	102.92	124.05
200	Double-Oracle	457.84	515.72	545.24	567.05	574.33
	MILP Hanasusanto et al. (2015)	NA	NA	NA	NA	NA
	IA Chassein et al. (2019)	NA	NA	NA	NA	NA
	SG Arslan et al. (2022)	537.63	548.42	573.13	622.36	630.55
300	Double-Oracle	1734.74	2129.96	2673.10	3382.99	4875.86
	MILP Hanasusanto et al. (2015)	NA	NA	NA	NA	NA
	IA Chassein et al. (2019)	NA	NA	NA	NA	NA
	SG Arslan et al. (2022)	1733.66	2138.93	2638.38	3468.43	5233.18

Table 7: Percentage of solved generic K-adaptability instances

Size	Algorithms	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$
n=20,m=20	Double-Oracle	100%	100%	100%	100%	100%
	MILP Hanasusanto et al. (2015)	100%	100%	100%	100%	100%
	BB Subramanyam et al. (2020)	100%	100%	100%	100%	100%
n=30,m=30	Double-Oracle	100%	100%	100%	100%	100%
	MILP Hanasusanto et al. (2015)	100%	100%	100%	100%	100%
	BB Subramanyam et al. (2020)	100%	100%	100%	100%	100%
n=40,m=40	Double-Oracle	100%	100%	100%	100%	100%
	MILP Hanasusanto et al. (2015)	100%	80%	80%	0%	0%
	BB Subramanyam et al. (2020)	100%	100%	80%	80%	70%
n=50,m=50	Double-Oracle	100%	80%	80%	70%	70%
	MILP Hanasusanto et al. (2015)	60%	0%	0%	0%	0%
	BB Subramanyam et al. (2020)	100%	60%	50%	20%	10%

Table 8: Average CPU time (s) of solved generic K-adaptability instances

Size	Algorithms	$k=2$	$k=3$	$k=4$	$k=5$	$k=6$
n=20,m=20	Double-Oracle	12.17	16.51	22.11	28.48	37.52
	MILP Hanasusanto et al. (2015)	0.82	1.47	0.50	21.93	70.54
	BB Subramanyam et al. (2020)	12.47	17.75	24.29	26.62	35.78
n=30,m=30	Double-Oracle	31.84	41.69	42.13	57.68	58.45
	MILP Hanasusanto et al. (2015)	1.10	2.78	128.62	325.99	402.52
	BB Subramanyam et al. (2020)	32.62	44.82	96.30	153.90	355.75
n=40,m=40	Double-Oracle	50.67	62.05	73.02	189.71	294.91
	MILP Hanasusanto et al. (2015)	1.50	112.55	196.46	NA	NA
	BB Subramanyam et al. (2020)	81.91	660.71	800.25	1077.28	NA
n=50,m=50	Double-Oracle	74.63	86.93	186.54	308.29	530.35
	MILP Hanasusanto et al. (2015)	3.87	NA	NA	NA	NA
	BB Subramanyam et al. (2020)	176.47	930.46	1205.01	NA	NA