Min-sup-min robust combinatorial optimization with few recourse solutions

Ayşe Nur Arslan, Michael Poss, Marco Silva

Min-sup-min robust combinatorial optimization with few recourse solutions

Ayşe Nur Arslan
IRMAR, INSA de Rennes, Rennes, France, ayse-nur.arslan@insa-rennes.fr,

Michael Poss
LIRMM, University of Montpellier, CNRS, France, michael.poss@lirmm.fr,

Marco Silva
CEGI, INESCTEC, Porto, Portugal, marco.c.silva@inesctec.pt,

In this paper, we consider a variant of adaptive robust combinatorial optimization problems where the decision maker can prepare K solutions and choose the best among them upon knowledge of the true data realizations. We suppose that the uncertainty may affect the objective and the constraints through functions that are not necessarily linear. We propose a new exact algorithm for solving these problems when the feasible set of the nominal optimization problem does not contain too many good solutions. Our algorithm enumerates these good solutions, generates dynamically a set of scenarios from the uncertainty set, and assigns the solutions to the generated scenarios using a vertex p-center formulation, solved by a binary search algorithm. Our numerical results on adaptive shortest path and knapsack with conflicts problems show that our algorithm compares favorably with the methods proposed in the literature. We additionally propose a heuristic extension of our method to handle problems where it is prohibitive to enumerate all good solutions. This heuristic is shown to provide good solutions within a reasonable solution time limit on the adaptive knapsack with conflicts problem. Finally, we illustrate how our approach handles non-linear functions on an all-or-nothing subset problem taken from the literature.

Key words: robust optimization, combinatorial optimization, vertex p-center, scenario generation
History:

1. Introduction

Robust optimization is an approach for handling uncertainty in optimization where the possible values of the parameters are described through uncertainty sets. In robust optimization, constraints are imposed for all realizations whereas the objective function is

evaluated for the worst-case realization within the uncertainty set. As such, in applications where the effects of uncertainty can be catastrophic, robust optimization presents itself as a viable modeling approach. Further, static robust optimization models with polyhedral or convex uncertainty sets lead to deterministic equivalent formulations that are often in the same complexity class as their deterministic counterparts. For these reasons, robust optimization has been enjoying a growing attention from the research community. Advances in static robust optimization are presented in Gabrel et al. (2014), Ben-Tal et al. (2009), Bertsimas et al. (2011), Buchheim and Kurtz (2018b).

The static robust optimization framework was extended to combinatorial optimization problems in Kouvelis and Yu (2013). Given the feasibility set of a combinatorial optimization problem, the framework considers that the cost vector can take any value in a given uncertainty set and seeks a solution that is optimal under the worst-case realization over this set. Assuming that the uncertainty set is a polytope described by a constant number of inequalities (such as the budgeted uncertainty set from Bertsimas and Sim (2003)), the static robust combinatorial optimization problem is not much harder than its deterministic counterpart, since solving it amounts to solve a polynomial number of problems for different cost vectors (Bertsimas and Sim (2003), Lee and Kwon (2014), Poss (2018)). However, when the uncertainty set is defined by a non-constant number of inequalities, or is simply defined by a non-constant list of possible cost vectors, the robust counterpart is typically harder than its deterministic variant (Aissi et al. (2009), Kouvelis and Yu (2013)).

Static robust combinatorial optimization problems are often considered too conservative since no recourse action can be taken to undermine the effect of uncertainty. A natural extension, coined *the min-max-min robust combinatorial optimization problem*, has therefore been proposed by Buchheim and Kurtz (2017, 2018a) allowing the decision maker to choose K solutions before the realization of uncertainty, and then, to select the best among them when the uncertain parameters materialize, thus mitigating the adverse effects. The min-max-min robust combinatorial optimization problem models the situation where the decision maker can prepare the ground for K solutions (e.g. training drivers, repairing roads, configure routing protocols) and choose the best among them upon full knowledge of the uncertain parameters. For instance, if the feasible set contains paths in a given graph, the problem seeks to prepare K different routes that can be used to evacuate citizens or transport relief supplies in case of a hazardous event (Hanasusanto et al. (2015)).

Another relevant example is related to parcel delivery companies, which would be unwilling to reschedule their delivery tours from scratch everyday. As an alternative, drivers are trained only for a small set of route plans that are to be executed in case of road closures and heavy deviations (Eufinger et al. (2019), Subramanyam et al. (2019)).

While several studies (e.g., Eufinger et al. (2019), Hanasusanto et al. (2015), Subramanyam et al. (2019)) have illustrated the practical relevance of the min-max-min robust combinatorial optimization problem, exact solution algorithms have stayed behind, and the problem has remained extremely difficult to solve, even for small instances. So far, only two algorithms are able to solve min-max-min problems with a polyhedral uncertainty set exactly: Hanasusanto et al. (2015) reformulates the problem through linear programming duality resulting in a monolithic formulation involving bilinear terms that are linearized using the McCormick linearization, and Subramanyam et al. (2019) introduces an ad-hoc branch-and-bound algorithm based on generating a relevant subset of uncertainty realizations and enumerating over their assignment to the K solutions. Unfortunately, these two approaches can hardly solve the shortest path instances proposed by Hanasusanto et al. (2015) as soon as the graph contains more than 25 nodes. A third paper has had more success with these instances, solving all of them to optimality (the largest having 50 nodes) in the special case $K \in \{2, 3\}$ (Chassein et al. (2019)). Yet this latter approach requires the uncertainty set to be the budgeted uncertainty set from Bertsimas and Sim (2003) and hardly handles any instance when K grows above 3. The theoretical complexity of the problem with budgeted uncertainty has also been investigated in Goerigk et al. (2020) (for the discrete variant) and Chassein and Goerigk (2020) (for its continuous variant). In addition, Goerigk et al. (2020) proposes heuristic algorithms with no performance guarantee. It should be noted that Hanasusanto et al. (2015) and Subramanyam et al. (2019) address, in fact, the more general K-adaptability paradigm, in which first-stage decisions can be taken in addition to the K different second-stage decisions. The paradigm of K-adaptability is strongly related to two-stage robust optimization with discrete recourse, a topic having witnessed a significant amount of work in the past few years (e.g. Bertsimas and Dunning (2016), Bertsimas and Georghiou (2018), Arslan and Detienne (2020)).

Our contribution The purpose of this work is to propose a novel algorithm for solving the generalization of the min-max-min robust combinatorial optimization problem with constraint uncertainty and non-linear dependency on the uncertain parameters. Because

of the constraint uncertainty, the adversarial problem's objective function may be discontinuous, so the maximization over the uncertainty set needs to be replaced by a supremum. We denote the resulting optimization problem as *min-sup-min*. Borrowing an idea from Chassein et al. (2019), our algorithm enumerates *good* feasible solutions. However, unlike Chassein et al. (2019) that relies on a purely enumerative scheme tailored for the budgeted uncertainty set from Bertsimas and Sim (2003), the approach presented in this paper models the problem with a generic compact uncertainty set as a variant of the p-center problem, assigning a *relevant* subset of scenarios to at most K different solutions. We solve the resulting problem by combining a row-and-column generation algorithm, binary search, and dominance rules.

We numerically compare our approach to the four algorithms available in the literature for solving *linear* min-max-min problems: the row-and-column generation algorithm from Goerigk et al. (2020), the enumeration scheme from Chassein et al. (2019), the monolithic reformulation from Hanasusanto et al. (2015) and the branch-and-bound algorithm from Subramanyam et al. (2019). We remark that the approach from Chassein et al. (2019) is restricted to problems where Ξ is the budgeted uncertainty polytope (Bertsimas and Sim (2003)), and cannot handle constraint uncertainty. Furthermore, none of these approaches is able to handle non-linear functions, in contrast with the algorithm proposed in this paper.

Our results illustrate that we are able to optimally solve many open instances from Hanasusanto et al. (2015) when K grows above 3. Our approach is not impacted by the complexity of the nominal counterpart of the problem, unlike Goerigk et al. (2020), Hanasusanto et al. (2015) and Subramanyam et al. (2019). This property is illustrated numerically on the min-max-min counterpart of the knapsack problem with conflicts, for which our algorithm compares even more favorably against the approaches from Hanasusanto et al. (2015) and Subramanyam et al. (2019), including in the case of constraint uncertainty.

We additionally propose a heuristic variant of our algorithm based on a partial enumeration of good feasible solutions that can be used when enumerating all solutions is computationally prohibitive. The latter is compared numerically to the heuristic variant of the branch-and-bound algorithm from Subramanyam et al. (2019) on the the knapsack problem with conflicts. The results indicate that our heuristic provides slightly better quality solutions for the larger values of K.

Finally, we also illustrate the numerical promise of our algorithms when the objective function is non-linear, on an all-or-nothing subset problem borrowed from Goldberg and Rudolf (2020).

The source code of the algorithms presented in this paper is available here.

Structure of the paper The rest of the paper is organized as follows. In Section 2, we formally define the min-max-min combinatorial optimization problem, and present the main components of our solution scheme, based on a dynamic scenario generation procedure combined with a p-center formulation solved through binary search. Section 3 further elaborates on the algorithm, explaining how to reduce the number of solutions and perform the binary search efficiently, as well as presenting a heuristic variant. In Section 4, we present the numerical results comparing our solution scheme to the well established methodologies from the K-adaptability and min-max-min literature on the shortest path problem and the knapsack problem with conflicts (with and without weight uncertainty), as well as illustrating it on a problem where the objective function is non-linear. We conclude the paper in Section 5.

Notations Throughout the paper, we use the short-hand notations $[n] = \{1, \dots, n\}$ for any positive integer n, and $\mathbf{x} = (x^1, \dots, x^K)$ where x^i for $i \in [K]$ is a feasible solution of the set X. We also define the cartesian product $\times_{k\in[K]} X$ as X^K.

2. Methodological development

In this section, we formally define the min-sup-min robust combinatorial optimization problem. We then present the different components of the solution scheme we propose.

2.1. Problem definition

Let $\Xi \subseteq \mathbb{R}^n$ be a compact set and consider a *nominal* combinatorial optimization problem characterized by its feasibility set $X \subseteq \{0,1\}^n$ and its objective function $f : X \to \mathbb{R} : x \mapsto f(x)$. We consider the robust counterpart of the problem, so we re-define the objective function as $f : \Xi \times X \to \mathbb{R} : (\xi, x) \mapsto f(\xi, x)$, where $\xi \in \Xi$ represents the uncertain parameters and $x \in X$ the (binary) decision variables. Similarly, we split X into components X_0 and X_1 where $X_0 \subseteq \{0,1\}^n$ contains the combinatorial structure of X, which is not affected by uncertainty, while $X_1(\xi)$ is defined by L constraints

$$g_\ell(\xi, x) \leq b_\ell, \quad \ell \in [L], \tag{1}$$

where $g_\ell : \Xi \times X \to \mathbb{R} : (\xi, x) \mapsto g_l(\xi, x)$ is the function characterizing the ℓ-th constraint. We assume throughout that f and g_ℓ are continuous functions. In this setting, the classical robust counterpart of the nominal combinatorial optimization problem seeks a solution $x \in X_0$ that satisfies constraints (1) for each $\xi \in \Xi$ and is optimal under the worst-case realization over Ξ, thus solving

$$\min_{\substack{x \in X_0 \\ x \in X_1(\xi), \forall \xi \in \Xi}} \max_{\xi \in \Xi} \; f(\xi, x). \tag{2}$$

A natural extension of (2) is to allow the decision maker to choose K solutions $x^1, \ldots, x^K$ in X_0. Then, when the uncertain parameters ξ materialize, the decision maker selects the solution $x^k \in X_1(\xi)$ having the smallest objective value $f(\xi, x^k)$. Historically, this variant was introduced without uncertainty in the constraints (Buchheim and Kurtz 2017, 2018a), resulting in the following min-max-min robust combinatorial optimization problem

$$\min_{\mathbf{x} \in X_0^K} \max_{\xi \in \Xi} \min_{k \in [K]} \; f(\xi, x^k). \tag{3}$$

In this paper, we consider a generalization of problem (3) by allowing also constraint uncertainty, leading to the following min-sup-min robust combinatorial optimization problem

$$v(X_0, \Xi) := \min_{\mathbf{x} \in X_0^K} \sup_{\xi \in \Xi} \min_{k \in [K] : x^k \in X_1(\xi)} \; f(\xi, x^k), \tag{4}$$

where we adopt the convention $v(X_0, \Xi) = +\infty$ in the case that the problem is infeasible. We illustrate problem (4) with an example on the shortest path problem.

Example 1. Consider the shortest path instance presented in Figure 1, where Ξ contains two scenarios (indicated as a tuple on each arc in Figure 1) and M is a large value representing a hazardous event that would cause road closures, $L = 0$ and f is the linear function $\xi^\top x$. Solving the classical robust problem leads to a solution of cost $M + 1$. In contrast, setting $K = 2$ we obtain $x^1 = \{(1,2),(2,4)\}$, $x^2 = \{(1,3),(3,4)\}$, so under any scenario realization, there remains an alternative route of cost 2.

We remark that (4) involves a supremum over Ξ as the adversarial objective function $F_{\mathbf{x}} : \Xi \to \mathbb{R} \cup \{+\infty\}$, defined by

$$F_{\mathbf{x}}(\xi) := \begin{cases} \min\limits_{k \in [K] : x^k \in X_1(\xi)} \; f(\xi, x^k) & \text{if } \exists k \in [K] \text{ such that } x^k \in X_1(\xi) \\ +\infty & \text{otherwise,} \end{cases} \tag{5}$$

is not continuous in $\xi \in \Xi$ in general. The following example, proposed by Subramanyam et al. (2019), illustrates this point in the case where $F_{\mathbf{x}}(\xi)$ is finite.

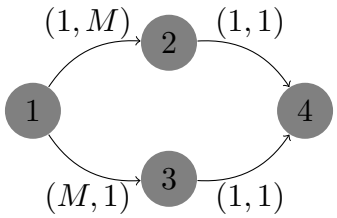

Figure 1 **Shortest path example with $X = \{\{(1,2),(2,4)\},\{(1,3),(3,4)\}\}$. Ξ contains two scenarios that are indicated as a tuple on each arc.**

EXAMPLE 2. Consider the following min-sup-min problem:

$$\min_{\mathbf{x}\in\{0,1\}^2} \sup_{\xi\in[0,1]} \min_{k\in\{1,2\}} \{(\xi-1)(1-2x^k) \mid x^k \geq \xi\}.$$

For $\mathbf{x} = (1,0)$, if $\xi = 0$ then the optimal solution of the inner minimization is obtained for $k = 2$ resulting in a value of -1, if $\xi > 0$ then the optimal solution of the inner minimization problem is $k = 1$ resulting in a value of $1-\xi$. The function $F_{\mathbf{x}}(\xi)$ is discontinuous at $\xi = 0$. Further, because X_0 and Ξ are bounded, we assume throughout without loss of generality that

$$v(X_0, \Xi) > 0, \tag{6}$$

which can always be enforced by adding, in all solutions, an artificial variable equal to 1 and having cost large enough so that $f(\xi, x) > 0$ for any $x \in X_0$ and $\xi \in \Xi$.

Let us now discuss the type of functions f, g_ℓ, $\ell \in [L]$, and set Ξ that can be handled by the algorithms proposed in what follows. On the one hand, the dependency of the functions on variables x is not a critical point of our approach, as long as the following holds.

ASSUMPTION 1. *We can efficiently compute a value UB^* such that: either problem* (4) *is infeasible, or $UB^* \geq v(X_0, \Xi)$.*

For instance, if f is bi-linear in x and ξ and $L = 0$, UB^* can be the optimal value of the static problem $\min_{x\in X} \max_{\xi\in\Xi} \xi^\top x$, or the cost provided by the more involved heuristic described in Appendix B. Whenever complex functions g_ℓ are involved, one can always consider a trivial relaxation of $\max_{x\in X_0, \xi\in\Xi} f(\xi, x)$, the optimal value of which yields the required value UB^*. On the other hand, the dependency on ξ is more critical as our algorithms will involve frequent calls to optimization problems along variables $\xi \in \Xi$ and involving the hypographs of functions f and g_ℓ, $\ell \in [L]$. Resulting optimization problems include (i) additional real variables, and/or (ii) logical constraints and binary variables.

As we discuss later in the paper, the case (ii) happens to be strongly $\mathcal{NP}$-hard even when Ξ is a polytope and f and g_ℓ are affine in ξ. In contrast, case (i), which corresponds to $L = 0$, leads to a convex problem whenever f is concave and Ξ is convex. Overall, we would like these problems to be solvable rather efficiently, typically assuming that Ξ is defined by linear, or SOCP constraints, while f and g_ℓ are affine or concave quadratic in ξ. Furthermore, we consider throughout that Assumption 1 holds.

2.2. Scenario generation

The algorithm developed in this paper lies in generating a subset of scenarios dynamically, following recent line of research addressing difficult robust optimization problems, e.g. Fischetti and Monaci (2012), Pessoa and Poss (2015), Agra et al. (2016), Ayoub and Poss (2016), Bertsimas et al. (2016), Subramanyam et al. (2019), Zeng and Zhao (2013) among others. Let $\tilde{\Xi} \subseteq \Xi$ be a subset of scenarios of finite cardinality. We consider a relaxation of (4) where Ξ is replaced by the subset $\tilde{\Xi} \subseteq \Xi$, namely

$$v(X_0, \tilde{\Xi}) := \min_{\mathbf{x} \in X_0^K} \max_{\xi \in \tilde{\Xi}} \min_{k \in [K]: x^k \in X_1(\xi)} f(\xi, x^k), \tag{7}$$

where the supremum over Ξ has been replaced by a maximization because $\tilde{\Xi}$ is finite. If problem (7) is infeasible, so is (4), and $\tilde{\Xi}$ needs not be further extended. Otherwise, let $\tilde{\mathbf{x}}$ be an optimal solution to (7), which may be infeasible or suboptimal for (4). The feasibility and optimality of $\tilde{\mathbf{x}}$ could, in theory, be verified by solving the following problem, proven to be $\mathcal{NP}$-hard (Hanasusanto et al. 2015, Theorem 3)

$$v(\tilde{\mathbf{x}}, \Xi) := \sup_{\xi \in \Xi} F_{\tilde{\mathbf{x}}}(\xi) = \sup_{\xi \in \Xi} \min_{k \in [K]: \tilde{x}^k \in X_1(\xi)} f(\xi, \tilde{x}^k). \tag{8}$$

As $v(X_0, \tilde{\Xi})$ and $v(\tilde{\mathbf{x}}, \Xi)$ respectively provide lower and upper bounds on $v(X_0, \Xi)$, testing whether $v(X_0, \tilde{\Xi}) = v(\tilde{\mathbf{x}}, \Xi)$ would prove the optimality of $\tilde{\mathbf{x}}$. Unfortunately, problem (8) involves a supremum which is hardly tractable from the numerical perspective. Therefore, we consider an alternative optimality criterion inspired by Subramanyam et al. (2019) that does not involve computing $v(\tilde{\mathbf{x}}, \Xi)$. Introducing the shorter notation $\tilde{v} = v(X_0, \tilde{\Xi})$, we define the amount of violation of the current solution, under the uncertainty realization $\xi \in \Xi$, as

$$S(\xi, \tilde{v}, \tilde{\mathbf{x}}) = \min_{k \in [K]} \max \left\{ f(\xi, \tilde{x}^k) - \tilde{v}, \max_{\ell \in [L]} \left\{ g_\ell(\xi, \tilde{x}^k) - b_\ell \right\} \right\}. \tag{9}$$

For each $k \in K$, the argument of the minimization takes the maximum among $f(\xi, \tilde{x}^k) - \tilde{v}$, which is the cost excess of solution $\tilde{x}^k$ for scenario ξ, and $g_\ell(\xi, \tilde{x}^k) - b_\ell$, which is the amount of violation of constraint $\ell \in [L]$ by solution $\tilde{x}^k$ for scenario ξ. Therefore, a non-positive value of $S(\xi, \tilde{v}, \tilde{\mathbf{x}})$ indicates that there exists $k \in [K]$ such that $\tilde{x}^k$ satisfies all constraints and does not exceed the value $\tilde{v}$. On the contrary, a positive value for $S(\xi, \tilde{v}, \tilde{\mathbf{x}})$ indicates that for each $\tilde{x}^k$, ξ leads to an objective value greater than $\tilde{v}$ or to a violated constraint. Since function $S(\xi, \tilde{v}, \tilde{\mathbf{x}})$ is continuous in ξ, we can define the separation problem as

$$S(\tilde{v}, \tilde{\mathbf{x}}) = \max_{\xi \in \Xi} S(\xi, \tilde{v}, \tilde{\mathbf{x}}). \tag{10}$$

PROPOSITION 1. *For any $\tilde{\mathbf{x}} \in X_0^K$, $v(\tilde{\mathbf{x}}, \Xi) = \tilde{v}$ if and only if $S(\tilde{v}, \tilde{\mathbf{x}}) \leq 0$.*

Proof. We first remark that by definitions of $v(\tilde{\mathbf{x}}, \Xi)$ and $\tilde{v}$, we have that $v(\tilde{\mathbf{x}}, \Xi) \geq \tilde{v}$. Further, by definition of $S(\xi, \tilde{v}, \tilde{\mathbf{x}})$, it is immediate that $S(\tilde{v}, \tilde{\mathbf{x}}) > 0$ implies that there exists $\tilde{\xi} \in \Xi$ such that for each $k \in [K]$, either $f(\tilde{\xi}, \tilde{x}^k) > \tilde{v}$ or $\tilde{x}^k \notin X_1(\tilde{\xi})$. As a result, $v(\tilde{\mathbf{x}}, \Xi) \geq F_{\tilde{\mathbf{x}}}(\tilde{\xi}) = \min_{k \in [K] | \tilde{x}^k \in X_1(\tilde{\xi})} f(\tilde{\xi}, \tilde{x}^k) > \tilde{v}$, recalling that $F_{\tilde{\mathbf{x}}}(\tilde{\xi}) = +\infty$ in the case $\{k \in [K] \mid \tilde{x}^k \in X_1(\tilde{\xi})\} = \emptyset$.

To prove the reverse implication, suppose $v(\tilde{\mathbf{x}}, \Xi) > \tilde{v}$ so there exists $\tilde{\xi} \in \Xi$ such that for each $k \in [K]$ either $f(\tilde{\xi}, \tilde{x}^k) > \tilde{v}$ or $g_\ell(\tilde{\xi}, \tilde{x}^k) > b_\ell$ for some $\ell \in [L]$. Let us denote by K_0 and K_ℓ, $\ell \in [L]$ the corresponding partition of $[K]$, breaking ties arbitrarily. We define

$$\delta = \min \left\{ \min_{k \in K_0} f(\tilde{\xi}, \tilde{x}^k) - \tilde{v}, \min_{\ell \in [L]} \min_{k \in K_\ell} g_\ell(\xi, \tilde{x}^k) - b_\ell \right\}.$$

From the definition of the partition, it is clear that δ is positive. Furthermore,

$$\begin{aligned} S(\tilde{\xi}, \tilde{v}, \tilde{\mathbf{x}}) = \min \Bigg\{ & \min_{k \in K_0} \max \left\{ f(\xi, \tilde{x}^k) - \tilde{v}, \max_{\ell \in [L]} \left\{ g_\ell(\xi, \tilde{x}^k) - b_\ell \right\} \right\}, \\ & \min_{\ell \in [L]} \min_{k \in K_\ell} \max \left\{ f(\xi, \tilde{x}^k) - \tilde{v}, \max_{\ell \in [L]} \left\{ g_\ell(\xi, \tilde{x}^k) - b_\ell \right\} \right\} \Bigg\} \\ \geq \min & \left\{ \min_{k \in K_0} f(\xi, \tilde{x}^k) - \tilde{v}, \min_{\ell \in [L]} \min_{k \in K_\ell} g_\ell(\xi, \tilde{x}^k) - b_\ell \right\} = \delta > 0, \end{aligned}$$

proving the result. □

Computing $S(\tilde{v}, \tilde{\mathbf{x}})$ can be done through an MILP formulation introducing indicator variables and logical implication constraints, see Appendix A for details. The overall scenario generation scheme is provided in Algorithm 1, wherein ξ^0 is any scenario from Ξ (typically

Algorithm 1: Scenario generation.

Input: Enumerated set of solutions X_0;
Initialization: $\tilde{\Xi} = \{\xi^0\}$, $\mathbf{x}^* = null$;
repeat
  Compute $\tilde{v} = v(X_0, \tilde{\Xi})$ and $\tilde{\mathbf{x}}$ by solving (7), set $\tilde{v} = +\infty$ and $\tilde{\mathbf{x}} = null$ if the problem is infeasible ;
  if $\tilde{\mathbf{x}} \neq null$ **then**
    Compute $S(\tilde{v}, \tilde{\mathbf{x}})$ and $\tilde{\xi} \in \arg\max_{\xi \in \Xi} S(\xi, \tilde{v}, \tilde{x})$ by solving (10);
    if $S(\tilde{v}, \tilde{\mathbf{x}}) > 0$ **then**
      $\tilde{\Xi} \leftarrow \tilde{\Xi} \cup \{\tilde{\xi}\}$;
      scenario_added = ***true***;
    else
      scenario_added = ***false***;
      $\mathbf{x}^* = \tilde{\mathbf{x}}$;
    end
  end
  else break ;
until *scenario_added* = ***false***;
Return: $\mathbf{x}^*$

chosen to be the nominal scenario if such a scenario exists). We notice that Algorithm 1 terminates either because problem (4) is infeasible or because $S(\tilde{v}, \tilde{\mathbf{x}}) \leq 0$, in which case Proposition 1 proves the optimality of the solution. We next investigate the convergence of Algorithm 1, starting with the cases in which it converges finitely.

PROPOSITION 2. *If the function $F_{\mathbf{x}}$ is continuous for $\mathbf{x} \in X_0^K$, Algorithm 1 converges in at most $|X_0|^K$ iterations.*

Proof. Let us denote by $\tilde{\mathbf{x}}^i$ the i-th optimal solution to (7) computed along the algorithm, corresponding to the set $\tilde{\Xi}^i$, and yielding the upper bound $v(\tilde{\mathbf{x}}^i, \Xi) = \max_{\xi \in \Xi} \min_{k \in [K]: \tilde{x}^{k,i} \in X_1(\xi)} f(\xi, \tilde{x}^{k,i})$ obtained by the uncertain vector $\tilde{\xi}^i \in \Xi$, where the sup operator is replaced by the max operator as a result of the continuity of $F_{\mathbf{x}}$. Consider $j > i$

such that $\tilde{\mathbf{x}}^j = \tilde{\mathbf{x}}^i$ and recall that by definition of $\tilde{\xi}^i$, $v(\tilde{\mathbf{x}}^i, \Xi) = \min_{k\in[K]:\tilde{x}^{k,i}\in X_1(\tilde{\xi}^i)} f(\tilde{\xi}^i, \tilde{x}^{k,i})$. Thus, $\tilde{\xi}^i \in \tilde{\Xi}^j$, where $\tilde{\Xi}^j$ is the uncertainty set at iteration j, implies that

$$\begin{aligned} v(\tilde{\mathbf{x}}^j, \tilde{\Xi}^j) = \max_{\xi\in\tilde{\Xi}^j} \min_{k\in[K]:\tilde{x}^{k,j}\in X_1(\xi)} f(\xi, \tilde{x}^{k,j}) = \max_{\xi\in\tilde{\Xi}^j} \min_{k\in[K]:\tilde{x}^{k,i}\in X_1(\xi)} f(\xi, \tilde{x}^{k,i}) \geq \min_{k\in[K]:\tilde{x}^{k,i}\in X_1(\tilde{\xi}^i)} f(\tilde{\xi}^i, \tilde{x}^{k,i}) \\ = v(\tilde{\mathbf{x}}^i, \Xi) = v(\tilde{\mathbf{x}}^j, \Xi), \end{aligned}$$

where the last equality holds by continuity of $F_{\mathbf{x}}$, proving the optimality of solution $\tilde{\mathbf{x}}^j = \tilde{\mathbf{x}}^i$. The algorithm therefore visits every feasible solution $\mathbf{x} \in X_0^K$ in the worst case, before proving the optimality of one of these solutions. The result follows by finiteness of X_0. □

COROLLARY 1. *If only the objective or the constraints are uncertain, Algorithm 1 converges in at most $|X_0|^K$ iterations.*

Proof. If only the objective or the constraints are uncertain then $F_{\mathbf{x}}$ is continuous in $\xi \in \Xi$ for $\mathbf{x} \in X_0^K$ (Subramanyam et al. 2019, Proposition B.1). □

In general however, the algorithm may not converge in finite time. Fortunately in that case, it will converge asymptotically to an optimal solution of problem (4).

PROPOSITION 3. *If problem (4) is feasible, every accumulation point $(\hat{\mathbf{x}}, \hat{v})$ of the solutions $(\tilde{\mathbf{x}}^i, \tilde{v}^i)$ generated along an infinite repeat-loop is composed of an optimal solution $\hat{\mathbf{x}}$ of problem (4) together with its objective value $\hat{v}$.*

Proof. Let us denote by $\tilde{\mathbf{x}}^i$ the i-th optimal solution to (7) computed along the algorithm, corresponding to the set $\tilde{\Xi}^i$, and yielding the value $\tilde{v}^i$, and by ξ^i to the i-th optimal solution of (10). If $(\tilde{\mathbf{x}}^i, \tilde{v}^i)$ forms an infinite sequence, then by compactness of X_0 and Ξ, there exists an accumulation point of $(\tilde{\mathbf{x}}^i, \tilde{v}^i)$ which we denote by $(\hat{\mathbf{x}}, \hat{v})$, as well as of ξ^i which we denote by $\hat{\xi}$. Let us assume that $(\tilde{\mathbf{x}}^i, \tilde{v}^i)$ converges to $(\hat{\mathbf{x}}, \hat{v})$ and ξ^i converges to $\hat{\xi}$, possibly considering subsequences.

We first show that solution $\hat{\mathbf{x}}$ is feasible for problem (4) and that $\hat{v}$ is its objective value. Suppose for a contradiction that $(\hat{\mathbf{x}}, \hat{v})$ is not feasible. Then there exists $\xi^* \in \Xi$ such that $S(\xi^*, \hat{v}, \hat{\mathbf{x}}) = \delta > 0$. This implies that for i sufficiently large we have that $S(\xi^i, \tilde{v}^i, \tilde{\mathbf{x}}^i) = \max_{\xi\in\Xi} S(\xi, \tilde{v}^i, \tilde{\mathbf{x}}^i) \geq S(\xi^*, \tilde{v}^i, \tilde{\mathbf{x}}^i) \geq \delta/2 > 0$. As S is continuous, by taking limits we obtain that $S(\hat{\xi}, \hat{v}, \hat{\mathbf{x}}) \geq S(\xi^*, \hat{v}, \hat{\mathbf{x}}) \geq \delta/2 > 0$. On the other hand, as $\xi^i \in \Xi^{i+1}$, we must have that $S(\xi^i, \tilde{v}^{i+1}, \tilde{\mathbf{x}}^{i+1}) \leq 0$. Once again taking limits, we obtain $S(\hat{\xi}, \hat{v}, \hat{\mathbf{x}}) \leq 0$, which is a contradiction.

Now suppose $(\hat{\mathbf{x}}, \hat{v})$ is not optimal, so there exists a feasible solution $(\mathbf{x}', v')$ such that $v' < \hat{v}$. Thus, for i large enough, $v' < \tilde{v}^i$. This is in contradiction with $v' \geq v(X_0, \Xi) \geq v(X_0, \tilde{\Xi}^i) = \tilde{v}^i$. Therefore, $(\hat{\mathbf{x}}, \hat{v})$ must be optimal for problem (4). □

We remark that Proposition 3 implies that if problem (4) is infeasible then the algorithm converges in a finite number of iterations, detecting infeasibility. Further, practical implementations of Algorithm 1 replace condition $S(\tilde{v}, \tilde{\mathbf{x}}) > 0$ with $S(\tilde{v}, \tilde{\mathbf{x}}) > \epsilon$ for some $\epsilon > 0$. Therefore, Proposition (3) guarantees that any such implementation will converge after a finite number of iterations, returning a solution that is feasible (if one exists) and optimal up to error ϵ.

2.3. Vertex p-center formulation

One of the numerical challenges of Algorithm 1 is the efficient solution of problem (7) to optimality. In this section, we detail how problem (7) can be cast as a vertex $p-$center problem and be solved using a binary search algorithm as a result. To this end, let X_0 be the enumerated set $(x_1, \ldots, x_r)$, meaning that the underlying nominal problem has r different feasible solutions. We remark that r is typically very large, we will thus provide techniques to restrict the enumeration to a relevant subset of these feasible solutions (see Section 2.4). We also let $\tilde{\Xi}$ be the enumerated set $(\xi_1, \ldots, \xi_m)$ where m denotes the number of scenarios generated so far.

Having the aforementioned two enumerated sets in mind, problem (7) can be reformulated as follows: choose at most K feasible solutions such that each scenario is assigned to one of these K solutions and such that the worst-case assignment cost is minimized. Let the binary decision variable z_s, for $s \in [r]$, equal to 1 if and only if feasible solution x_s is selected. Furthermore, for $s \in [r]$ and $j \in [m]$, let the binary decision variable y_{sj} equal to 1 if the feasible solution x_s is assigned to the scenario ξ_j, 0 otherwise. The problem then formalizes as:

$$\min \quad \omega \tag{11}$$

$$\text{s.t.} \quad \omega \geq \sum_{s \in [r]} f(\xi^j, x_s) y_{sj}, \qquad \forall j \in [m] \tag{12}$$

$$\sum_{s \in [r]} y_{sj} = 1, \qquad \forall j \in [m] \tag{13}$$

$$\sum_{s \in [r]} z_s \leq K, \tag{14}$$

$$y_{sj} \leq z_s, \qquad \forall\ s \in [r], j \in [m] \tag{15}$$

$$y_{sj} = 0, \qquad \forall\ s \in [r], j \in [m] : \exists \ell \in L \text{ s.t. } g(\xi^j, x_s) > b_\ell \tag{16}$$

$$z \in \{0,1\}^r, y \in \{0,1\}^{r \times m}. \tag{17}$$

We remark that the above formulation handles constraint uncertainty by setting some of the components of y to 0, as expressed by constraints (16). Based on this formulation, problem (7) is a vertex p-center problem (introduced in Daskin (2011)) where X_0 contains the possible centers, $\tilde{\Xi}$ contains the clients that need to be served, and $p = K$.

We next consider the decision version of this problem and look for a solution of cost not greater than a given threshold $\rho > 0$ (recall assumption (6)). Since we are looking for a solution with a cost not greater than ρ, we can disregard assignments with costs greater than ρ, as well as those that lead to constraint violations. We model this with the help of the 0/1 coverage matrix $cov(\rho)$ defined as follows: given a threshold $\rho > 0$, $s \in [r]$ and $j \in [m]$, $cov_{sj}(\rho)$ equals 1 if and only if $f(\xi^j, x_s) \leq \rho$ and $g(\xi^j, x_s) \leq b_\ell$ for all $\ell \in L$. With the above definitions, the optimal value of problem (7) is not greater than ρ if and only if there exists a feasible solution to the following system (see Minieka (1970)):

$$\sum_{s \in [r]} cov_{sj}(\rho)\, y_{sj} = 1, \qquad \forall j \in [m] \tag{18}$$

$$\sum_{s \in [r]} z_s \leq K, \tag{19}$$

$$y_{sj} \leq z_s, \qquad \forall s \in [r],\ j \in [m] \tag{20}$$

$$y \in \{0,1\}^{r \times m}, z \in \{0,1\}^r. \tag{21}$$

We remark that checking the feasibility of the above system is $\mathcal{NP}$-hard, and, theoretically, the best one can do is to enumerate all possible assignments (Chalermsook et al. (2017)). Herein, we solve the problem by a mixed-integer linear program (MILP), replacing constraints (18) with

$$\sum_{s \in [r]} cov_{sj}(\rho)\, y_{sj} \geq 1 - \alpha_j, \qquad \forall j \in [m], \tag{22}$$

where $\alpha \geq 0$, and minimizing the objective function

$$\sum_{j \in [m]} \alpha_j. \tag{23}$$

If the optimal value of the resulting MILP is zero, then the optimal value of (7) is at most ρ, so that we can reduce the threshold ρ. Otherwise, there is no feasible solution with cost at most ρ, so we must increase the threshold to recover feasibility. This results in a binary search on the optimal value of ρ, described in Algorithm 2 allowing to solve problem (7) to optimality. Similar algorithms have been proposed for solving the vertex p-center problem, e.g. Chen and Chen (2009), Contardo et al. (2019).

The above scheme can be improved by removing dominated solutions and scenarios, where a solution s is said to be *dominated* if there exists $s' \in [r]$ such that $cov_{sj}(\rho) \leq cov_{s'j}(\rho)$ for each $j \in [m]$, *i.e.*, s' covers not less scenarios than s, while a scenario j is said to be *dominated* if there exists $j' \in [m]$ such that $cov_{sj}(\rho) \geq cov_{sj'}(\rho)$ for each $s \in [r]$, *i.e.*, j' is covered by not more solutions than j. Removing dominated elements can significantly reduce the number of variables and constraints of the above MILP. We remark that if two solutions dominate each other, this means $cov_{sj}(\rho) = cov_{s'j}(\rho)$ for each $j \in [m]$ and the two solutions are indistinguishable in the covering constraints (22); we thus keep only one of them, breaking ties arbitrarily. The same holds for any pair of scenarios that dominate each other.

Algorithm 2: Solving (7) through binary search.

Input: Enumerated set of solutions X_0, set of scenarios $\tilde{\Xi}$, an upper bound UB and a lower bound LB on $v(X_0, \tilde{\Xi})$;
$LB_{\text{bs}} = LB$, $UB_{\text{bs}} = UB$;
repeat
  $\rho = \frac{LB_{\text{bs}} + UB_{\text{bs}}}{2}$;
  Test feasibility of ρ by calling Algorithm 3;
  if $\omega = 0$ **then** $LB_{\text{bs}} \leftarrow \rho$;
  else $UB_{\text{bs}} \leftarrow \rho$;
until $\frac{UB_{\text{bs}} - LB_{\text{bs}}}{UB_{\text{bs}}} \leq \epsilon$;
Return: ρ

An optimal solution $\mathbf{x}$ to problem (7) is recovered by calling Algorithm 3 with the value of ρ returned by Algorithm 2. The solution vector z^* has K positive components each corresponding to a feasible solution $x_s \in X_0$, and forming together the solution $\mathbf{x}$.

Algorithm 3: Test feasibility of ρ.

Input: Enumerated set of solutions X_0, set of scenarios $\tilde{\Xi}$, objective target ρ ;

Compute $cov(\rho)$;

Remove dominated solutions;

Remove dominated scenarios;

Compute $\omega := \{\min(23) : (19) - (21), (22), \alpha \geq 0\}$;

Return: ω, z^*

An upper bound and a lower bound on the optimal value of problem (7), $v(X_0, \tilde{\Xi})$, are used as input in Algorithm 2. To initialize the algorithm, we can use a lower bound of 0 (thanks to (6)) and the upper bound UB^* mentioned in Assumption 1. We remark, that in the case that the problem (4) is infeasible, the solution of $\{\min(23) : (19) - (21), (22), \alpha \geq 0\}$ after updating $cov(\rho)$ with the value of ρ returned by Algorithm 2 prove infeasible, indicating that problem (7) is infeasible. In this case, the algorithm terminates without an incumbent solution, thereby concluding infeasibility.

2.4. Putting things together

The overall solution scheme, presented in Algorithm 4, is comprised of three steps. First, we rely on Assumption 1 to compute the bound UB^*. This upper bound is used for generating only a subset of the set X_0, referred to as $X_0(UB^*)$ in the following, in the second step. The subset $X_0(UB^*) \subseteq X_0$, represents the set of solutions whose worst case value is better than the upper bound, and that can therefore potentially improve the current best solution. The validity of restricting the enumeration to $X_0(UB^*)$ as well as the details of how this enumeration is done are given in Section 3.1. Finally, in the third step, the scenario generation algorithm is invoked. This algorithm calls in turn Algorithms 2 and 3 with a potentially updated lower and upper bounds each time problem (7) needs to be solved with an updated set of scenarios $\tilde{\Xi}$.

We remark that the lower bound is updated with the ρ value returned by Algorithm 2 at each iteration provided that a feasible solution to problem (7) is obtained. One can additionally update the upper bound, in the case that the function $F_{\mathbf{x}}(\xi)$ is continuous in $\xi \in \Xi$, by solving directly problem (8) by replacing the sup operator by max, instead of problem (10). In this case, the convergence criteria can also be changed to a relative gap between the upper and lower bound.

Algorithm 4: Complete solution procedure.

Heuristic: Obtain an upper bound UB^* that is valid for $v(X_0, \Xi)$ if problem (4) is feasible, see Assumption 1;
Generate solutions: Enumerate the set $X_0(UB^*)$;
Initialization: $\tilde{\Xi} = \{\xi^0\}$, $\mathbf{x}^* = null$, $LB = 0$, $UB = UB^*$;
repeat
  Compute $\tilde{v} = v(X_0, \tilde{\Xi}) = \rho$ by calling Algorithm 2 with UB and LB ;
  Compute $\tilde{\mathbf{x}}$ by calling Algorithm 3 , set $\tilde{v} = +\infty$ and $\tilde{\mathbf{x}} = null$ if $\omega > 0$;
  if $\tilde{\mathbf{x}} \neq null$ **then**
    $LB = \rho$;
    Steps 6–13 of Algorithm 1;
  end
  else break ;
until *scenario_added* = ***false***;
Return: $\mathbf{x}^*$

3. Numerical improvements

The scenario generation algorithm described above reposes on an increasingly accurate relaxation that is strengthened at each iteration through the solution of a separation problem. This idea has been repeatedly used in mathematical programming within the framework of cutting plane algorithms, Benders' decomposition and robust optimization. However, as it is all too common in practice, a naive implementation of this idea is often not numerically efficient. In this section, we outline some of the algorithmic details that improve significantly the numerical efficiency of Algorithm 1. The interested reader is deferred to our open-source code for the full details of the implementation.

3.1. Restricting the set of enumerated solutions

The second step of our solution scheme enumerates the set of solutions $X_0(UB^*)$ for a given upper bound UB^*. This step is critical as enumerating the set X_0 without this restriction can be time consuming itself while the cardinality of the set of enumerated solutions affects the complexity of the rest of the algorithm. In the following, we first prove that restricting X_0 to $X_0(UB^*)$ in the scenario generation framework does not change the correctness of

the algorithm. We then formalize the definition of the set $X_0(UB^*)$, and detail how this set can be efficiently generated by combining partial enumeration with bound-based filtering.

PROPOSITION 4. *Assume that $|X_0| > 1$, let $x' \in X_0$ be such that at least one of the following holds:*

1. $\min_{\xi\in\Xi} f(\xi, x') \geq UB^*$
2. $\min_{\xi\in\Xi} g_\ell(\xi, x') > b_\ell$ *for some* $\ell \in [L]$.

Defining $X_0' = X_0 \setminus \{x'\}$, we have that either $v(X_0', \Xi) = v(X_0, \Xi)$ or $v(X_0, \Xi) \geq UB^*$.

Proof. The case $v(X_0, \Xi) = +\infty$ is trivial so we assume next that $v(X_0, \Xi) < +\infty$. We first remark that $v(X_0', \Xi) \geq v(X_0, \Xi)$ since removing solutions from X_0 creates a restriction of problem (4). Let $\tilde{\mathbf{x}} = (\tilde{x}^1, \ldots, \tilde{x}^K)$ be an optimal solution to (4) and suppose that $v(X_0', \Xi) > v(X_0, \Xi)$. Then, x' must be one of the K solutions used in $\tilde{\mathbf{x}}$, since otherwise we would have $v(X_0', \Xi) = v(X_0, \Xi)$. Suppose, without loss of generality, that $\tilde{x}^1 = x'$. We define $\hat{\mathbf{x}} = (\tilde{x}^2, \tilde{x}^2, \tilde{x}^3, \ldots, \tilde{x}^K)$. If $1 \notin \operatorname{arg\,min}_{k\in[K]:x^k\in X_1(\xi)} f(\xi, \tilde{x}^k)$ for all $\xi \in \Xi$, then $\sup_{\xi\in\Xi} \min_{k\in[K]:\hat{x}^k\in X_1(\xi)} f(\xi, \hat{x}^k) = \sup_{\xi\in\Xi} \min_{k\in[K]:\tilde{x}^k\in X_1(\xi)} f(\xi, \tilde{x}^k)$, proving $v(X_0', \Xi) = v(X_0, \Xi)$. Otherwise, $1 \in \operatorname{arg\,min}_{k\in[K]:x^k\in X_1(\xi)} f(\xi, \tilde{x}^k)$ for some $\xi \in \Xi$ (implying point 2. does not hold and therefore point 1. must hold) so that $v(X_0, \Xi) \geq v(x', \Xi) = \max_{\xi\in\Xi} f(\xi, x') \geq \min_{\xi\in\Xi} f(\xi, x') = UB^*$. This last inequality implies that, $v(X_0, \Xi) \geq UB^*$, and therefore solving (4) will provide no better solution than the heuristic solution already available. □

As a direct consequence of Proposition 4, we may define the set of non-dominated solutions as

$$X_0(UB^*) = \left\{ x \in X_0 : \min_{\xi\in\Xi} f(\xi, x) < UB, \min_{\xi\in\Xi} g_\ell(\xi, x) \leq b_\ell, \forall \ell \in [L] \right\}.$$

The calculation of $X_0(UB^*)$ is one of the steps of the algorithm where the structures of X_0, X_1, and f play a critical role. While computing $X_0(UB^*)$ can in general be done using a constraint-programming or a branch-and-bound algorithm, see Chassein et al. (2019), these solutions can typically be enumerated much more quickly by leveraging the specific structure of the problem at hand. For instance, the non-dominated solutions of the problems presented in our numerical experiments can be efficiently computed using dynamic programming algorithms. Each of the above algorithms (branch-and-bound, constraint programming, dynamic programming) relies on the computation of partial solutions, which can be defined as

$$X^{part}(UB^*) = \Big\{ I \subseteq [n] : \exists x \in X_0 \text{ s.t. } x_i = 1 \, \forall i \in I, \min_{\xi\in\Xi} f(\xi, x) < UB^*,$$

$$\min_{\xi\in\Xi} g_\ell(\xi,x) \le b_\ell, \forall \ell \in [L] \Big\}.$$

The constraints of $X^{part}(UB^*)$ involve a complete solution $x \in X_0$ that is typically unknown when partial solution I is constructed. It is therefore necessary to replace $f(\xi,x)$ and $g_\ell(\xi,x)$ by a possibly smaller expression depending only on I in order to eliminate non-relevant partial solutions as early as possible. With this in mind, we define $\underline{f}(I)$ as a lower bound on the cost required to complement I to a feasible solution in X_0, formally,

$$\underline{f}(I) \le \min_{\substack{\xi\in\Xi, x\in X_0 \\ x_i=1\ \forall i\in I}} f(\xi,x) - f(\xi,x_I), \tag{24}$$

where x_I is the binary vector having I as a support. We define similarly $\underline{g}_\ell(I)$ for each $\ell \in L$ such that

$$\underline{g}_\ell(I) \le \min_{\substack{\xi\in\Xi, x\in X_0 \\ x_i=1\ \forall i\in I}} g_\ell(\xi,x) - d_\ell(\xi,x_I). \tag{25}$$

Let us illustrate (24) further in the case where f is the bilinear function $\xi^\top x$ (handling a bilinear function g_ℓ lead to similar derivations). In this setting, the right-hand-side of (24) becomes

$$\min_{\substack{\xi\in\Xi, x\in X_0 \\ x_i=1\ \forall i\in I}} \sum_{i\in[n]\setminus I} \xi_i x_i,$$

which is a bilinear mixed-integer optimization problem whose exact solution can be very difficult. However, we do not need to solve this problem exactly but instead only to ensure that $\underline{f}(I)$ is not greater than its optimal value. In general, the computation of $\underline{f}(I)$ depends on both the structure of Ξ and X_0, and is therefore problem dependent. In Section 4, we provide examples of how this bound can be calculated in practice. Once $\underline{f}(I)$ is calculated, the condition $f(\xi,x) < UB^*$ is then replaced by the weaker condition $\min_{\xi\in\Xi} f(\xi,x_I) + \underline{f}(I) < UB$, where $\underline{f}(I)$ can be computed off-line in a pre-processing step. In our numerical experiments, we leverage these observations in order to reduce the number of solutions enumerated and alleviate the numerical burden of the enumeration step of our algorithm.

3.2. Eliminating dominated solutions

After $X_0(UB^*)$ has been generated, we can further reduce its cardinality by running a pairwise comparison among its elements, in the line of the dominance checks described in Section 2.3. Specifically, given $x \in X_0(UB^*)$ and $\xi \in \Xi$, we denote by $L(\xi,x) \subseteq [L]$ the indexes of the constraints of $X_1(\xi)$ satisfied by x for ξ. We say that x is *dominated* by x' if

$f(\xi, x) \geq f(\xi, x')$ and $L(\xi, x) \subseteq L(\xi, x')$ for all $\xi \in \Xi$. Compared to the dominance between solutions discussed in Section 2.3, here, we consider Ξ instead of $\tilde{\Xi}$, and we do not reduce the objective value to threshold ρ. Unfortunately, verifying dominance for a single pair of solutions x, x' is already intractable, as formalized in the proposition below, the proof of which is provided in Appendix C.

PROPOSITION 5. *Verifying whether $x \in X_0$ is dominated by $x' \in X_0$ is $\mathcal{NP}$-hard even when $L = 0$ and f is concave quadratic in ξ for each $x \in X_0$.*

Despite the discouraging result of Proposition 5, there are some cases where this dominance check can be performed efficiently. For instance, if $L = 0$ and f is linear in ξ, then the dominance check amounts to optimizing a linear function over Ξ, specifically

$$\min_{\xi \in \Xi} f(\xi, x) - f(\xi, x'). \tag{26}$$

As many pairs of x and x' must be considered, it may also be useful to consider a superset of Ξ in (26), such as its bounding box or its smallest enclosing ellipsoid. While the resulting condition is only sufficient for dominance, it may be much faster to check leading to a better trade-off from the numerical perspective. In the case $L > 0$ with f and g_ℓ linear in ξ for each ℓ, we may consider the sufficient condition $f(\xi, x) \geq f(\xi, x')$ and $g_\ell(\xi, x) \geq g_\ell(\xi, x')$ for all $\xi \in \Xi$.

3.3. Special case of objective uncertainty

We next turn our attention to the case where $L = 0$, that is only uncertainty involved in the problem is in the objective function. As mentioned previously, in this case, the function $F_{\mathbf{x}}(\xi)$ is continuous in $\xi \in \Xi$, and the separation problem can be stated as:

$$v(\tilde{\mathbf{x}}, \Xi) := \max_{\xi \in \Xi} F_{\tilde{\mathbf{x}}}(\xi) = \max_{\xi \in \Xi} \min_{k \in [K]: \tilde{x}^k \in X_1(\xi)} f(\xi, \tilde{x}^k). \tag{27}$$

Problem (27) may be further rewritten as

$$\begin{aligned} v(\tilde{\mathbf{x}}, \Xi) = \max \quad & \eta \\ \text{s.t.} \quad & \xi \in \Xi \\ & \eta \leq f(\xi, \tilde{x}^k) \quad k \in [K], \end{aligned}$$

which is a convex optimization problem if f is concave in $\xi \in \Xi$, and Ξ is convex. As $v(\tilde{\mathbf{x}}, \Xi)$ is an upper bound on the optimal value of (4), each time the separation problem is solved,

the upper bound can be updated if the value $v(\tilde{\mathbf{x}}, \Xi)$ improves upon the current upper bound. We remark that updating the lower and upper bound on the optimal value of (4) is beneficial as this reduces the search space for the binary search algorithm.

The solution time of the binary search algorithm can further be reduced by realizing that a solution can be returned even before convergence between the binary search upper and lower bound is achieved. Indeed, at any iteration of the algorithm a feasible solution to (7) with value UB_{bs} is available, and can be returned as a heuristic solution to (7). Further, at earlier iterations of the scenario generation algorithm it is less crucial to solve problem (7) to exact optimality as the relaxation probably misses many important scenarios. The tolerance of the binary search algorithm can therefore be adapted according to this observation, a higher tolerance can be chosen in earlier iterations and decreased along with the global optimality gap throughout the algorithm.

3.4. A heuristic scenario generation algorithm

The exactness of Algorithm 1 is crucially dependent on the enumeration of all *relevant* solutions, that is the set of solutions $X_0(UB^*)$ needs to contain *all* feasible solutions whose worst-case value is less than or equal to the value of the best solution known, UB^*. For combinatorial optimization problems that are loosely constrained, the cardinality of the set $X_0(UB^*)$ may therefore grow relatively fast, and the enumeration of this set may take a considerable amount of computational time (even taking the improvements described previously into account). In this case, it is desirable to devise an algorithm that can provide good feasible solutions to problem (4) without enumerating the entire set $X_0(UB^*)$. Such an algorithm may be used on its own or in combination with our exact algorithm presented in Algorithm 1 so as to reduce the cardinality of the set $X_0(UB^*)$ (the lower the UB^* the smaller the cardinality of $X_0(UB^*)$. We next describe how our algorithm can be altered in order to work with a partially enumerated set of feasible solutions.

The main difference between this heuristic version and Algorithm 1 is the way $\tilde{v}$ and $\tilde{\mathbf{x}}$ are obtained. While the exact algorithm enumerates set $X_0(UB^*)$, here we assume that only a subset of solutions $\tilde{X}_0$ with $|\tilde{X}_0| \leq X_0(UB^*)$ is available. We therefore solve

$$v(\tilde{X}_0, \tilde{\Xi}) := \min_{\mathbf{x} \in \tilde{X}_0^K} \max_{\xi \in \tilde{\Xi}} \min_{k \in [K]: x^k \in X_1(\xi)} f(\xi, x^k). \tag{28}$$

As a result of the solution of (28) with a partially enumerated set of solutions and scenarios, one obtains a solution $\tilde{\mathbf{x}}$ as in Algorithm 1. This solution is sent to the separation problem to

Algorithm 5: Heuristic scenario generation.

Initialization: $\tilde{\Xi} = \{\xi'\}$, $\tilde{X}_0 = \{\mathbf{x}'\}$, $\mathbf{x}^* = null$;
repeat
  Compute $\tilde{v} = v(\tilde{X}_0, \tilde{\Xi})$ and $\tilde{\mathbf{x}}$ by solving (28);
  Compute $S(\tilde{v}, \tilde{\mathbf{x}})$ and $\tilde{\xi}$ by solving (10);
  if $S(\tilde{v}, \tilde{\mathbf{x}}) > 0$ **then**
    $\tilde{\Xi} \leftarrow \tilde{\Xi} \cup \{\tilde{\xi}\}$;
    scenario_added = ***true***;
    Compute x' by solving (29), possibly heuristically;
    $\tilde{X}_0 \leftarrow x' \cup N(x')$;
  else
    scenario_added = ***false***;
    $\mathbf{x}^* = \tilde{\mathbf{x}}$;
  end
until *scenario_added* = ***false***;
Return: $\mathbf{x}^*$

obtain a new scenario, $\tilde{\xi}$, as well as a potentially better upper bound and a new incumbent solution (in the case of objective uncertainty). The set $\tilde{X}_0$ is then enriched with the solution,

$$\tilde{x} \in \underset{x \in X_0 \cap X_1(\tilde{\xi})}{\arg\min} f(\tilde{\xi}, x), \tag{29}$$

as well as its neighbors $N(\tilde{x})$, which is defined in a problem specific manner. The heuristic algorithm just described is formally presented in Algorithm 5.

Solving (29) can be time-consuming for difficult combinatorial optimization problems. As the overall heuristic algorithm does not need an exact solution to (29), it can be convenient to obtain instead a good feasible solution typically obtained through ad-hoc heuristic algorithms. In the numerical results presented in Section 4.2 for a linear function f, we use a relax-and-fix heuristic that first solves the linear relaxation of problem (29). The algorithm then proceeds to fix the variables that take an integer value in the relaxation solution to their current values and solve the restricted problem (29) to return an integer feasible solution. In order to improve the quality of solutions returned by this relax-and-fix heuristic

we impose that the relaxation solution should improve the best known integer solution for scenario $\tilde{\xi}$ that is already in the set $\tilde{X}_0$. To this end, we impose the constraint $f(\tilde{\xi}, x) \leq \min_{x' \in \tilde{X}_0 \cap X_1(\tilde{\xi})} f(\tilde{\xi}, x')$ in solving the linear relaxation of (29). For the all-or-nothing subset problem described in Section 4.3, which involves a non-linear objective, we first replace f by a linear approximation and then proceed with the aforementioned relax-and-fix algorithm.

4. Computational experiments

In this section, we present numerical results, using our scenario generation algorithm to solve three robust adaptive combinatorial optimization problems, namely, the shortest path problem, the knapsack problem with conflicts (with and without objective uncertainty), and the non-linear smuggler problem presented by Goldberg and Rudolf (2020). We compare the numerical performance of our algorithm (denoted SG) to the existing methods from the min-max-min and K-adaptability literature, more specifically, the direct solution of the monolithic formulation of Hanasusanto et al. (2015) by an optimization solver (denoted M), the branch-and-bound algorithm of Subramanyam et al. (2019) (denoted BB), the iterative algorithm from Chassein et al. (2019) (denoted IT), and the row-and-column generation algorithm from Goerigk et al. (2020) (denoted RCG). We remark that only the shortest path problem is solved by all approaches, as IT requires Ξ to be the budgeted uncertainty set, which is not the case for the knapsack instances. Further, RCG and M are not considered for the the knapsack problem with weight uncertainty as these methods do not handle constraint uncertainty efficiently. Finally, only SG is considered the all-or-nothing subset problem as it is the only method that can solve problems involving nonlinear functions to exact optimality.

For each algorithm tested the time limit is set to two hours (7200 seconds). All mixed integer linear programming and linear programming models are solved using IBM ILOG Cplex 12.10 solver with a single thread. Our proposed scenario generation algorithm, the monolithic formulation of Hanasusanto et al. (2015), the row-and-column generation algorithm from Goerigk et al. (2020), and the iterative algorithm of Chassein et al. (2019) are implemented with Julia 1.2.0 (Bezanson et al. (2017)) using the mathematical programming package JuMP (Dunning et al. (2017)). The source code and all data used are available here. The branch-and-bound algorithm of Subramanyam et al. (2019) is adapted to each application using the authors' implementation available here.

All our experiments are conducted using a 2 Dodeca-core Haswell Intel Xeon E5-2680 v3 2.5 GHz machine with 128Go RAM running Linux OS. The resources of this machine are strictly partitioned using the Slurm Workload Manager[1] to run several tests in parallel. The resources available for each run (algorithm-instance) are set to two threads and a 20 Go RAM limit. This virtually creates 12 independent machines, each running a single instance at a time.

4.1. Shortest path problem

Setting The first problem on which we assess our algorithm is the adaptive shortest path problem, which has been previously studied in Chassein et al. (2019), Hanasusanto et al. (2015) and Subramanyam et al. (2019). In this problem, K paths are prepared without exact knowledge of the arc costs, where the shortest among them is to be implemented once the actual arc costs are revealed. We provide numerical results on instances generated randomly according to the procedure proposed by Hanasusanto et al. (2015). As such, we consider a network (V, A) where the cost of each arc $(i, j) \in A$ is characterized as $\bar{c}_{ij} + \xi_{ij}\hat{c}_{ij}$, with $\bar{c}_{ij}$ the nominal cost, and $\hat{c}_{ij}$ the maximal deviation. The uncertain parameter ξ can take any value within the budgeted uncertainty set $\Xi^\Gamma = \{\xi \in [0,1]^{|\mathcal{A}|} | \sum_{(i,j)\in\mathcal{A}} \xi_{ij} \leq \Gamma\}$. The feasibility set X is characterized by the classical flow constraints, $X = \{x \in \{0,1\}^{|A|} \mid \sum_{(i,j)\in\delta^+(i)} x_{ij} - \sum_{(j,i)\in\delta^-(i)} x_{ij} = b_i, \forall i \in V\}$, where $b_s = -1$, $b_t = 1$, and $b_i = 0$ for $i \in V \setminus \{s, t\}$, and sets $\delta^+(i)$ and $\delta^-(i)$ represent the forward and backward stars of node $i \in V$, respectively. We use the heuristic solution of the bilinear formulation presented in Appendix B as an initial step of our algorithm. Previously published results (Chassein et al. (2019)) as well as preliminary experiments prove this heuristic to provide very good quality solutions in a short amount of computational time. Further, to calculate $\underline{f}(I)$ where I contains the set of arcs of a path from s to u, we consider the shortest path from u to t using costs $\bar{c}$. We remark that the shortest path from each vertex $i \in V$ to t can be calculated in polynomial time in a preprocessing step. The dominance check described in Section 3.2 is realized by solving (26) exactly for all pairs $x, x' \in X_0(UB^*)$.

Results In Figure 2, we present the percentage of instances solved to optimality by each solution method over hundred instances. We consider three different instance sizes $|V| \in \{20, 25, 30\}$, two different uncertainty budgets $\Gamma \in \{3, 6\}$, and five different values

[1] `https://slurm.schedmd.com/` (accessed June 2021)

of $K \in \{2,3,4,5,6\}$. From left to right, the first two graphics illustrate the evolution of the percentage of instances solved to optimality as K increases, presented as an average over the instance sizes, $|V| \in \{20,25,30\}$. The last two graphics illustrate the evolution of the same indicator as $|V|$ increases, presented as an average over the number of policies, $K \in \{2,3,4,5,6\}$.

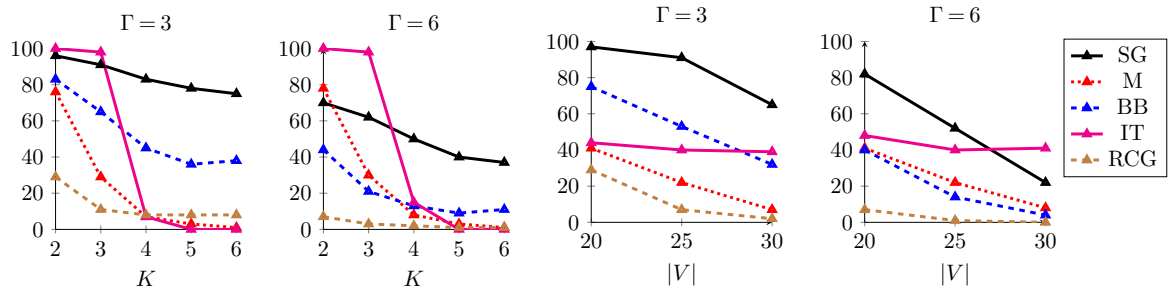

Figure 2 **Percentages of instances solved for the adaptive shortest path problem.**

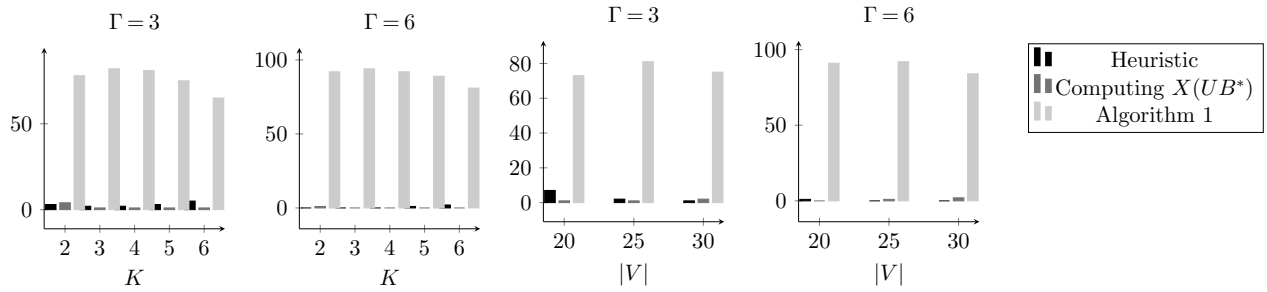

Figure 3 **Geometric averages of the percentages of the time spent in each of the three steps of the scenario generation algorithm, considering only the instances solved to optimality.**

As it is clear from these graphics, the proposed scenario generation algorithm compares favorably to the existing methods in the literature, solving on average %85 of all instances for $\Gamma = 3$, and %52 of all instances for $\Gamma = 6$. Based on these graphics, the scenario generation algorithm is superior to all other methods as K grows beyond 4. Although the iterative scheme of Chassein et al. (2019), which is tailored for min-max-min problems with the budgeted uncertainty set, manages to solve more instances for $K = 2,3$, it clearly reaches its limits as K grows. On the other hand, this method solves more instances when $n = 30$ and $\Gamma = 6$. We additionally observe that, among the general purpose ($K-$adaptability) methods, the branch-and-bound algorithm of Subramanyam et al. (2019) is the most effective, and suffers less from increasing K.

Our results in Figure 2 are complemented by the information given in Figure 3, where we represent the percentage of total solution time allocated to each step of our solution framework. Accordingly, the enumeration step can be done very efficiently for this problem, with a significant amount of time spent in scenario generation iterations.

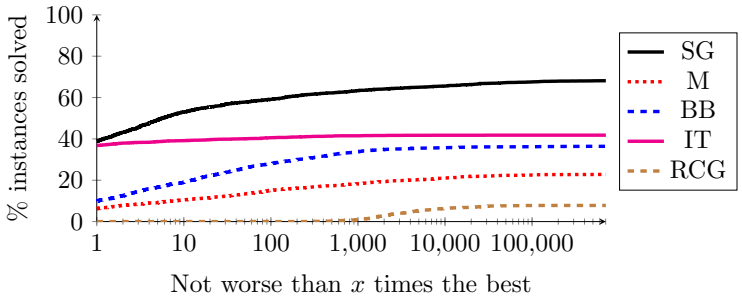

Figure 4 **Performance profile for the shortest path problem.**

In Figure 4, we additionally present a performance profile that shows the percentage of instances solved not slower than the value of the abscissa times the best method. This performance profile was obtained by aggregating all 3000 instances (100 instances for each combination of $|V|$, Γ, and K). Accordingly, the scenario generation method and IT solved a little less than 40% of all instances faster than the other algorithms, while these numbers fall to about 7% and 12% for M and BB, respectively. The nearly flat curve for IT also underlines that most instances that are not solved very quickly are not solved at all, contrasting with the increasing curve for CG.

4.2. Knapsack problem with conflicts

Setting We next study an adaptive knapsack problem with conflicts that is inspired by the capital budgeting problem and its variants studied in the $K-$adaptability literature (Hanasusanto et al. (2015),Subramanyam et al. (2019)). Here, the objective is to prepare K different combinations of items that respect the capacity and conflict constraints without exact knowledge of their profits (and weights). The most profitable among these K combinations is to be implemented upon actual realization of the uncertain profits. We consider two variants of the problem: in the first one, only the profits are considered uncertain, while the second one also involves weight uncertainty.

Let $N = \{1, \ldots, n\}$ be the set of items considered. In this version of the problem, the feasibility set X can be characterized in different ways, which affects the efficiency of the

resulting algorithms. Let w_i be the weight of an item, B be the knapsack budget and $C \subseteq N \times N$ be the set of conflicts. Herein we consider either the conflict formulation defined by the constraints $X^{\text{conf}} = \{x \in \{0,1\}^{|N|} \mid \sum_{i \in N} w_i x_i \leq B, x_i + x_j \leq 1 \quad \forall (i,j) \in C\}$, or the aggregated formulation defined by the constraints $X^{\text{agg}} = \{x \in \{0,1\}^{|N|} \mid \sum_{i \in N} w_i x_i \leq B, |E_i| x_i + \sum_{j \in E_i} x_j \leq |E_i| \quad \forall i \in N\}$ where $E_i = \{j \in N \mid (i,j) \in C \text{ or } (j,i) \in C\}$ is the set of items that are in conflict with i. The latter was proposed by Hifi and Michrafy (2007) and was used as the basis of heuristic algorithms for solving the knapsack with conflicts problem in Hifi (2014). The uncertain profits are characterized as $(1 + \sum_{j \in M} \frac{\Phi_{ij} \xi_j}{2}) \bar{p}_i$, where $\bar{p}_i$ is the nominal profit of item $i \in N$, $|M|$ 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, denoted Φ_i, is characterized by the set $\{\Phi_i \in [-1,1]^{|M|} \mid \sum_{j \in M} |\Phi_{ij}| = 1\}$, whereas the uncertainty set Ξ is characterized as $[-1,1]^{|M|}$. As a result, the realized profit of each object $i \in N$ remains within the interval $[\bar{p}_i - \frac{\bar{p}_i}{2}, \bar{p}_i + \frac{\bar{p}_i}{2}]$. If the weights are also considered uncertain, we consider a second factor loading matrix $\Psi \in \mathbb{R}^{|N| \times |M|}$ such that $\{\Psi_i \in [-1,1]^{|M|} \mid \sum_{j \in M} |\Psi_{ij}| = 1\}$, and define $w_i = (1 + \sum_{j \in M} \frac{\Psi_{ij} \xi_j}{2}) \bar{w}_i$, where $\bar{w}_i$ is the nominal profit of item $i \in N$.

The instances (available online) are generated randomly following the procedure proposed by Hanasusanto et al. (2015). Specifically, the parameter $|M|$ is set to 4 and we generate 10 instances for each instance size $|N| \in \{100, 150, 200, 250, 300, 350\}$: w_i is uniformly generated in $[0, 100]$, $B = \sum_{i \in N} w_i / 2$, $\bar{p}_i = c_i / 5$, and Φ and Ψ are uniformly generated within the sets described above. For each instance, we additionally introduce randomly generated conflicts between items. To do so, we use conflict graphs parametrized by a given density inspired by the instances used in Sadykov and Vanderbeck (2013) in the context of the bin packing problem with conflicts. In this context, the density of a graph $G = (N, E)$ is defined as the ratio between $|E|$ and the cardinality of the edge set of the complete graph having the same number of vertices. In our numerical experiments, we consider conflict graphs with density 0.5.

Regarding $\underline{f}(I)$, a strong lower bound of the bilinear mixed-integer problem that can be computed efficiently is not immediately available, because the set $[-1,1]^{|M|}$ does not have a clear dominated scenario, unlike the aforementioned budgeted uncertainty set. For this reason, we only perform the optimality check when the full solution x is known. In this case, $\underline{f}(I) = 0$ and $x \in X^{part}(UB^*)$ only if $\min_{\xi \in \Xi} f(\xi, x) \leq UB^*$. For the case without constraint uncertainty, the dominance check is realized by solving (26) exactly for all

pairs $x, x' \in X_0(UB^*)$. This is extended to the case with uncertainty in the constraints by considering the sufficient condition $f(\xi, x) \geq f(\xi, x')$ and $g_1(\xi, x) \geq g_1(\xi, x')$ for all $\xi \in \Xi$.

Results for the exact algorithms for cost uncertainty only We test all four methods for the two formulations X^{conf} and X^{agg} on the aforementioned instances. We use the heuristic version of the scenario generation algorithm presented in Section 3.4 in order to obtain UB^* (see Appendix D for details). In Figure 12, we present the number of instances solved to optimality by each solution method over ten instances. The graphic on the left illustrates the evolution of the number of instances solved to optimality as K increases, presented as an average over the instance sizes, $|N| \in \{100, 150, 200, 250, 300\}$. The graphic on the right illustrates the evolution of the same indicator as $|N|$ increases, presented as an average over the number of policies, $K \in \{2, 3, 4, 5, 6\}$.

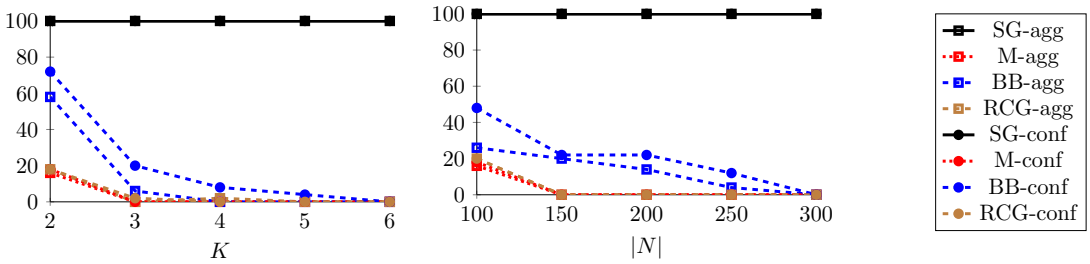

Figure 5 **Instances solved for the adaptive knapsack problem.**

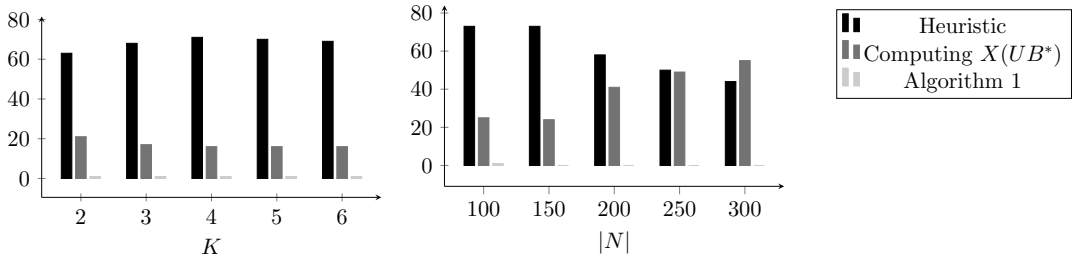

Figure 6 **Geometric averages of the percentages of the time spent in each of the three steps of the scenario generation algorithm based on the aggregated formulation, considering only the instances solved to optimality.**

As illustrated by these results, the comparative efficiency of the four algorithms is not impacted by the underlying formulations. The monolithic approach M, and the row-and-column generation approach RCG, were able to solve only a small percentage of the smallest

instances with 100 items, regardless of the underlying formulation. The branch-and-bound algorithm BB was more effective and more sensitive to the underlying formulation than M. Its implementation with X^{conf} solved %72 of all instances with $K = 2$, while it failed to scale with increasing values of K. When comparing formulations, BB was able to solve more instances in each category coupled with the conflict formulation X^{conf} than the aggregated formulation X^{agg}. The scenario generation algorithm was able to solve all instances up to $|N| = 300$ regardless of the underlying formulation.

Figure 6 presents the percentage of total solution time allocated to each step of our solution framework for this problem with the aggregated formulation (results for the conflict formulation are similar). As it is clear from this graphic, the combination of the heuristic and enumerative steps makes up almost all of the solution time. In contrast, for BB, the difference between the strengths of the two formulations has an important effect, as evident from the superior performance of this method with X^{conf}. These results highlight the particularities of branch-and-bound and scenario generation algorithms. The scenario generation algorithm is more sensitive to the time spent in the heuristic step and the quality of the upper bound obtained as this affects directly the number of solutions that should be enumerated. On the other hand, the branch-and-bound algorithm is more sensitive to the mathematical formulation, the strength of its linear relaxation and the required solution time being important factors.

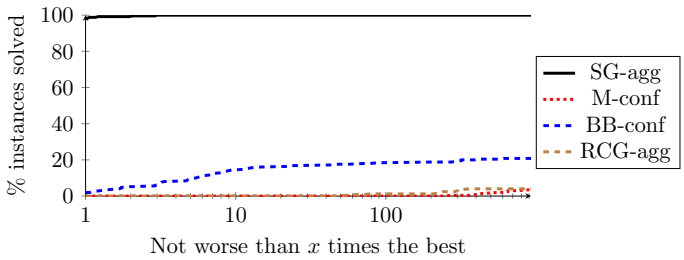

Figure 7 **Performance profile for the knapsack problem.**

In Figure 7, we additionally present a performance profile that shows the percentage of instances solved not slower than the value of the abscissa times the best method. This performance profile was obtained by aggregating all 250 instances (10 instances for each combination of N and K) for the best formulation for each algorithm, that is, SG-agg, M-conf, BB-conf, and RCG-agg.

Results for the exact algorithms for cost and weight uncertainty Figures 8 and 9 present the counterparts of Figures 12 and 7 for the variant of the problem with constraint uncertainty. The conclusions are similar to the case without weight uncertainty, SG-agg and SG-conf solving nearly all instances within the time limit, while BB-conf solves about 20% and is much slower than SG-agg.

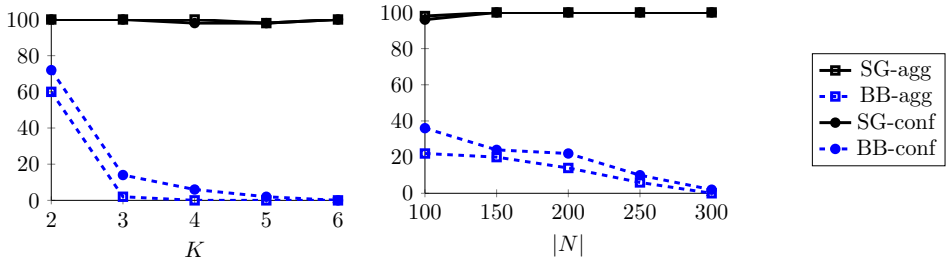

Figure 8 **Instances solved for the adaptive knapsack problem with constraint uncertainty.**

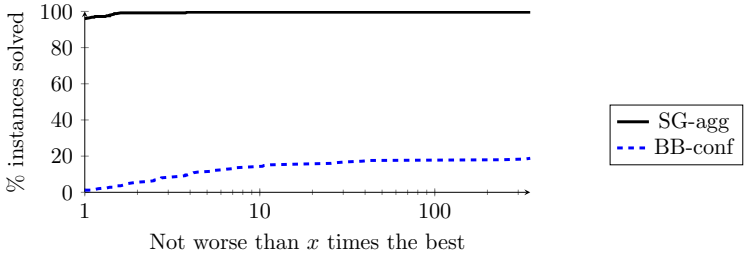

Figure 9 **Performance profile for the knapsack problem with constraint uncertainty.**

4.3. All-or-nothing subset

Setting Our last application is inspired by the utility-maximizing evader who wishes to select a subset of illicit activities that are being monitored. The evader receives a positive value only if he or she successfully completes all activities without being exposed Goldberg and Rudolf (2020). Many variants can be considered for this problem, depending on the constraints linking the feasible sets of illicit activities. In what follows, we consider that the smuggler may choose up to $B \in \mathbb{Z}$ activities which must be free of any conflicts. We remark that the objective function in this problem is non-linear, so, to the best of our knowledge, approaches from the literature cannot solve it to exact optimality. The purpose of our experiments is therefore only to illustrate the capability of our algorithm for handling non-linearities.

Let $N = \{1, \ldots, n\}$ be the set of activities considered and $C \subseteq N \times N$ be the set of conflicts. We associate to each activity a reward r_i and a success probability p_i, where success probabilities are assumed independent. For any $i \in N$, binary variable x_i indicates whether activity i is carried out. Following Goldberg and Rudolf (2020), we wish to maximize the product of the overall probability with the total reward, leading to the deterministic objective function $\sum_{i \in N} r_i x_i \prod_{j \in N} p_j^{x_j}$. In the context of this paper, we suppose the rewards are uncertain and belong to the intersection of a box and an ellipsoid $\Xi = \{\xi \mid \sum_{j \in N} \left(\frac{\xi_i - \hat{r}_i}{\hat{r}_i}\right)^2 \leq \Omega, \underline{r}_i \leq \xi_i \leq \overline{r}_i \quad \forall i \in N\}$, where $\Omega > 0$ is a given radius and $\underline{r} \leq \hat{r} \leq \overline{r}$ are given vectors, so the objective function is given by

$$f(\xi, x) = \sum_{i \in N} \xi_i x_i \prod_{j \in N} p_j^{x_j}.$$

Function $f(\xi, x)$ is linear in ξ, which makes the separation problems tractable.

We consider the feasibility set $X^{\text{agg}} = \{x \in \{0,1\}^{|N|} \mid \sum_{j \in N} x_j \leq B, |E_i| x_i + \sum_{j \in E_i} x_j \leq |E_i| \quad \forall i \in N\}$ where $E_i = \{j \in N \mid (i,j) \in C \text{ or } (j,i) \in C\}$ is the set of activities that are in conflict with i.

For each $|N| \in \{20, 40, 60\}$ we generate 10 instances as follows: for each $i \in N$, $\hat{p}_i$ is uniformly generated in $[0.5, 1.0]$, $\hat{r}_i$ is uniformly generated in $[0, 1000]$, $\underline{r}_i = (1 - \rho)\hat{r}_i$, and $\overline{r}_i = (1 + \rho)\hat{r}_i$ with $\rho = 0.5$. The conflict graph is generated as in Section 4.2, using a density of 0.5. Finally, the budget B is equal to 5 and Ω to 1.

The initial heuristic solution is obtained by replacing the non linear function $f(\xi, x)$ with the linearized function $f'(\xi, x) = \sum_{i \in N} \xi_i x_i$ and solving exactly the min-max robust counterpart of the resulting optimization problem. Additionally, we replace (29) by a relax-and-fix heuristic optimizing the linear function $f'(\tilde{\xi}, x)$ within the context of our heuristic algorithm. As in the case of the knapsack problem, a strong lower bound $\underline{f}(I)$ is not immediately available and we only perform the optimality check when the full solution x is known. Finally, the dominance check described in Section 3.2 is realized by solving (26) for the enclosing ellipsoid $\Xi' = \{\xi \mid \sum_{j \in N} \left(\frac{\xi_i - \hat{r}_i}{\hat{r}_i}\right)^2 \leq \Omega\}$.

Results We report on Figure 10 the average solution times for the different values of $|N| \in \{20, 40, 60\}$ and $K \in \{2, 3, 4, 5, 6\}$. We see that the impact of $|N|$ is significantly more marked than that of K. Figure 11 complements the picture by providing the number of solutions in $X_0(UB^*)$ before and after the dominance check, as well as the percentage

of total solution time allocated to each step of our solution framework. These results illustrate the significant impact of performing dominance comparisons on the number of solutions. Moreover, they show that most of the time is spent in the enumeration step, which includes the time spent in dominance comparisons. Overall, our results underline that the non-linearity present in the objective function seems to cause no particular problem, the difficulty lying mostly in the generation of a large number of good solutions for the problem. Unreported results indicate the heuristic algorithm does not provide a sufficiently low UB^*, which is not surprising given that it ignores the non-linear part of f. The results could possibly be improved by further refining the heuristic algorithm.

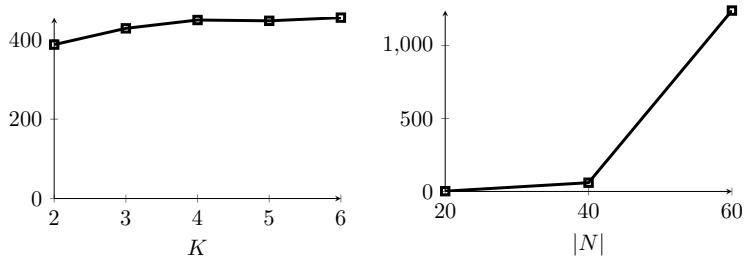

Figure 10 **Average solution times for the all-or-nothing subset problem.**

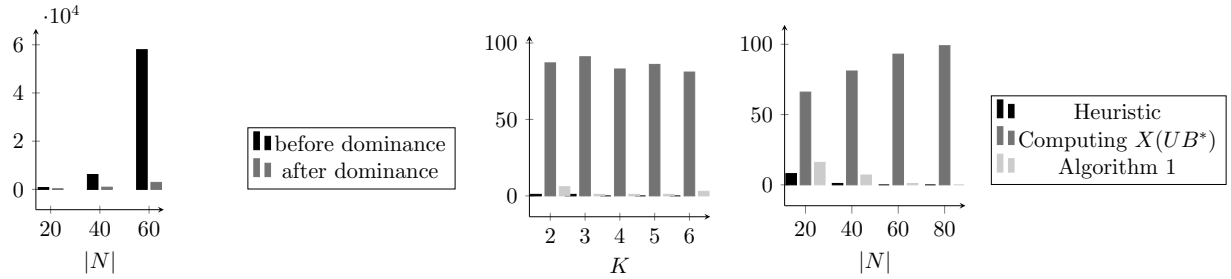

Figure 11 **Number of solutions in $X(UB^*)$ before and after dominance checks (left), and percentages of the times spent in each of the three steps of the scenario generation algorithm (center and right).**

5. Concluding remarks

In this paper, we present a new exact algorithm for solving the min-sup-min robust combinatorial optimization problem, possibly involving constraint uncertainty and non-linear functions. The algorithm combines the enumeration scheme presented in Chassein et al. (2019) with a scenario-based relaxation, reformulated as a p-center problem and solved through a binary-search algorithm. The algorithm proposed herein scales reasonably well

with K and is able to solve more instances to optimality than the state-of-the-art algorithms from the $K-$adaptability literature (Hanasusanto et al. (2015), Subramanyam et al. (2019)) and min-max-min literature (Chassein et al. (2019), Goerigk et al. (2020)), in particular for large values of K. The main advantage of our approach is that it is little impacted by the complexity of solving the underlying deterministic problem. This is illustrated for the knapsack problem with conflicts, with and without uncertainty in the constraints, for which our algorithm can solve nearly all instances with up to 300 items. In contrast, Goerigk et al. (2020), Hanasusanto et al. (2015) and Subramanyam et al. (2019) can only solve a small part of the instances with up to 100 and 250 items, respectively, when the constraint is certain, and Subramanyam et al. (2019) obtain similar results with an uncertain constraint. We additionally propose a heuristic variant of our algorithm based on a partial enumeration of good feasible solutions that can be used when enumerating all good feasible solutions is computationally prohibitive and it is hard to optimize over the set X_0. On the one hand, when optimizing over X_0 is easy, as for the shortest path problem, our new heuristic cannot compete with the one from Chassein et al. (2019). On the other hand, the new heuristic behaves well for harder problems and even compares favorably to the heuristic variant of the branch-and-bound algorithm from Subramanyam et al. (2019) on the knapsack problem with conflicts for the larger values of K (the heuristic from Chassein et al. (2019) being able to cope only with the smallest instances for that problem). Finally, our results on the all-or-nothing subset problem illustrate how our approach is able to solve a problem featuring a non-convex objective function with up to 60 binary variables.

References

Agra A, Santos MC, Nace D, Poss M (2016) A dynamic programming approach for a class of robust optimization problems. *SIAM Journal on Optimization* 26(3):1799–1823.

Aissi H, Bazgan C, Vanderpooten D (2009) Min-max and min-max regret versions of combinatorial optimization problems: A survey. *Eur. J. Oper. Res.* 197(2):427–438.

Arslan A, Detienne B (2020) Decomposition-based approaches for a class of two-stage robust binary optimization problems Available at https://hal.inria.fr/hal-02190059/.

Ayoub J, Poss M (2016) Decomposition for adjustable robust linear optimization subject to uncertainty polytope. *Comput. Manag. Science* 13(2):219–239.

Ben-Tal A, El Ghaoui L, Nemirovski A (2009) *Robust optimization*, volume 28 (Princeton University Press).

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

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

Bertsimas D, Dunning I, Lubin M (2016) Reformulation versus cutting-planes for robust optimization. *Comput. Manag. Science* 13(2):195–217.

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

Bertsimas D, Sim M (2003) Robust discrete optimization and network flows. *Math. Program.* 98(1-3):49–71.

Bezanson J, Edelman A, Karpinski S, Shah VB (2017) Julia: A fresh approach to numerical computing. *SIAM review* 59(1):65–98, URL `https://doi.org/10.1137/141000671`.

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

Buchheim C, Kurtz J (2018a) Complexity of min–max–min robustness for combinatorial optimization under discrete uncertainty. *Discrete Optimization* 28:1–15.

Buchheim C, Kurtz J (2018b) Robust combinatorial optimization under convex and discrete cost uncertainty. *EURO J. Comput. Optim.* 6(3):211–238, URL `http://dx.doi.org/10.1007/s13675-018-0103-0`.

Chalermsook P, Cygan M, Kortsarz G, Laekhanukit B, Manurangsi P, Nanongkai D, Trevisan L (2017) From gap-eth to fpt-inapproximability: Clique, dominating set, and more. *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, 743–754 (IEEE).

Chassein A, Goerigk M (2020) On the complexity of min–max–min robustness with two alternatives and budgeted uncertainty. *Discrete Applied Mathematics* .

Chassein AB, 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 D, Chen R (2009) New relaxation-based algorithms for the optimal solution of the continuous and discrete p-center problems. *Computers & Operations Research* 36(5):1646–1655.

Contardo C, Iori M, Kramer R (2019) A scalable exact algorithm for the vertex p-center problem. *Computers & OR* 103:211–220.

Daskin MS (2011) *Network and discrete location: models, algorithms, and applications* (John Wiley & Sons).

Dunning I, Huchette J, Lubin M (2017) Jump: A modeling language for mathematical optimization. *SIAM Review* 59(2):295–320, URL `http://dx.doi.org/10.1137/15M1020575`.

Eufinger L, Kurtz J, Buchheim C, Clausen U (2019) A robust approach to the capacitated vehicle routing problem with uncertain costs. *INFORMS Journal on Optimization* To appear.

Fischetti M, Monaci M (2012) Cutting plane versus compact formulations for uncertain (integer) linear programs. *Math. Program. Comput.* 4(3):239–273.

Gabrel V, Murat C, Thiele A (2014) Recent advances in robust optimization: An overview. *European Journal of Operational Research* 235(3):471–483.

Goerigk M, Kurtz J, Poss M (2020) Min-max-min robustness for combinatorial problems with discrete budgeted uncertainty. *Discrete Applied Mathematics* In press.

Goldberg N, Rudolf G (2020) On the complexity and approximation of the maximum expected value all-or-nothing subset. *Discrete Applied Mathematics* 283:1–10.

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

Hifi M (2014) An iterative rounding search-based algorithm for the disjunctively constrained knapsack problem. *Engineering Optimization* 46(8):1109–1122.

Hifi M, Michrafy M (2007) Reduction strategies and exact algorithms for the disjunctively constrained knapsack problem. *Computers & operations research* 34(9):2657–2673.

Kouvelis P, Yu G (2013) *Robust discrete optimization and its applications*, volume 14 (Springer Science & Business Media).

Lee T, Kwon C (2014) A short note on the robust combinatorial optimization problems with cardinality constrained uncertainty. *4OR* 12(4):373–378.

Minieka E (1970) The m-center problem. *Siam Review* 12(1):138–139.

Pessoa AA, Poss M (2015) Robust network design with uncertain outsourcing cost. *INFORMS J. Comput.* 27(3):507–524.

Poss M (2018) Robust combinatorial optimization with knapsack uncertainty. *Discrete Optimization* 27:88–102.

Sadykov R, Vanderbeck F (2013) Bin packing with conflicts: a generic branch-and-price algorithm. *INFORMS Journal on Computing* 25(2):244–255.

Sahni S (1974) Computationally related problems. *SIAM Journal on computing* 3(4):262–279.

Subramanyam A, Gounaris CE, Wiesemann W (2019) K-adaptability in two-stage mixed-integer robust optimization. *Mathematical Programming Computation* 1–32.

Zeng B, Zhao L (2013) Solving two-stage robust optimization problems using a column-and-constraint generation method. *Operations Research Letters* 41(5):457–461.

Appendix A: Separation problem when $L > 0$

Let $z_{k\ell}$ be a binary variable for each $k \in K$ ad $\ell \in [L] \cup \{0\}$. Following Subramanyam et al. (2019), the value of $S(\tilde{v}, \tilde{\mathbf{x}})$ can be computed by solving

$$\max \quad \eta$$

$$
\begin{aligned}
\text{s.t.} \quad & \sum_{\ell=0}^{L} z_{k\ell} = 1, && \forall k \in K \\
& z_{k0} = 1 \implies \eta \leq f(\xi, \tilde{x}^k) - \tilde{v}, && \forall k \in K \\
& z_{k\ell} = 1 \implies \eta \leq g_\ell(\xi, \tilde{x}^k) - b_\ell, && \forall k \in K, \ell \in L \\
& \xi \in \Xi \\
& z \in \{0,1\}^{K \times (L+1)}.
\end{aligned}
$$

Observation 3 from Subramanyam et al. (2019) proves the validity of the above reformulation in the case f and g_ℓ are linear functions. One readily extends their reasoning to the more general context considered herein.

Appendix B: Dualization

We present next the dualized reformulation proposed by Hanasusanto et al. (2015) as well as its heuristic variant first suggested by Chassein et al. (2019). Let us define Ξ as the polytope $\Xi := \{A\xi \leq b, \xi \geq 0\}$. Then, applying an epigraph reformulation to problem (4), we have

$$
\min_{\mathbf{x} \in X^K} \left\{ \max \eta : \xi \in \Xi,\, \eta \leq f(\xi, x^k),\, k \in [K] \right\}. \tag{30}
$$

Introducing the vectors of dual variables α and β for the constraints $A\xi \leq b$ and $\eta \leq f(\xi, x^k)$, $k \in [K]$, we can dualize the inner linear program of (30) to obtain

$$
\begin{aligned}
\min \quad & b^\top \alpha \\
\text{s.t.} \quad & \alpha^\top A - \sum_{k \in [K]} \beta_k x^k \geq 0 \\
& \sum_{k \in [K]} \beta_k = 1 \\
& x^1, \ldots, x^K \in X \\
& \alpha, \beta \geq 0.
\end{aligned}
$$

The above non-linear mixed-integer linear program is addressed heuristically in Chassein et al. (2019), by iteratively optimizing over $(\alpha, \mathbf{x})$ and (α, β) until no further improvement is obtained. Alternatively, Hanasusanto et al. (2015) apply the McCormick linearization to each product $\beta_k x^k$ to obtain an exact MILP reformulation.

Appendix C: Proof of Proposition 5

We adapt the reduction from the subset sum problem proposed by Sahni (1974). Specifically, given the set of integers $\{s_i, i = 1, \ldots, n\}$, and the integer M, the subset sum problem asks whether there exists a subset of integers that sums up to M. The reduction considers

$$
f(\xi, x) = -M^* \delta(x, x') - \delta(x, \bar{x}) \left(\sum_{i \in [n]} \xi_i (\xi_i - 1) + \sum_{i \in [n]} s_i \xi_i \right),
$$

where function $\delta(x, y) = 1$ if $x = y$ and 0 otherwise, $M^* = M - \min_{i \in [n]} \frac{s_i - 1}{s_i^2}$, and $\Xi = \left\{ \xi \in [0,1]^n : \sum_{i \in [n]} s_i \xi_i = M \right\}$. We observe that $f(\xi, x') = -M^*$ and $f(\xi, \bar{x}) = -\sum_{i \in [n]} \xi_i (\xi_i - 1) - \sum_{i \in [n]} s_i \xi_i$, the latter verifying the property below.

LEMMA 1. *Let z be the optimal objective value of*

$$\max\left\{\sum_{i\in[n]}\xi_i(\xi_i-1)+\sum_{i\in[n]}s_i\xi_i : \sum_{i\in[n]}s_i\xi_i = M, 0\le\xi\le 1\right\},$$

where all s_i are integers. Either $z = M$ or $z \le M - \min_{i\in[n]}\frac{s_i-1}{s_i^2}$.

Proof. The objective function is strictly convex so the maximum is reached on one of the extreme points. Furthermore, any extreme point has at most one fractional component. Thus, if there exists one extreme point ξ^* with no fractional component, $\sum_{i\in[n]}\xi_i^*(\xi_i^*-1)=0$ and the corresponding objective value is M. Otherwise, let i be the fractional component. As all s_i are integer, $s_i\xi_i^*$ must also be integer to satisfy the constraint $\sum_{i\in[n]}s_i\xi_i = M$, so that $\xi_i^*\in\{1/s_i,\dots,(s_i-1)/s_i\}$. Thus, the objective value of that solution is at most $M-\frac{s_i-1}{s_i^2}$, proving the result. □

Now, $\bar{x}$ is dominated by x' if and only if $f(\xi,x')\le f(\xi,\bar{x})$ for all $\xi\in\Xi$. As $-f(\xi,x')=M^*$, the previous equality is equivalent to

$$M^*\ge\max\left\{\sum_{i\in[n]}\xi_i(\xi_i-1)+\sum_{i\in[n]}s_i\xi_i : \sum_{i\in[n]}s_i\xi_i = M, 0\le\xi\le 1\right\}.$$

To conclude the reduction, we notice that the answer to the subset sum instance is *yes* if and only if $\max_{\xi\in\Xi}f(\xi,\bar{x})=M>M^*$, which happens if and only if $\bar{x}$ is not dominated. Furthermore, thanks to the above lemma, the answer to the subset sum instance is *no* if and only if $\max_{\xi\in\Xi}f(\xi,\bar{x})\le M^*$ meaning that $\bar{x}$ is dominated.

Appendix D: Heuristic results

We present in this section an evaluation of different heuristic approches that can be used as the first step of our algorithm. Our preliminary results showed that the heuristic resolution of the bilinear formulation presented in Appendix B was too slow for this problem, taking nearly two hours of computational time for instances with 100 items only. This behavior is due to the optimization over variables $\mathbf{x}$, which takes too much time as N grows. We therefore focus on the two other approaches: the heuristic version of our algorithm presented in Section 3.4, and the heuristic version of the branch-and-bound algorithm of Subramanyam et al. (2019) where $k-$adaptability problems are solved in sequence for $k=1,\dots,K$, and the optimal solution of the $(k-1)-$adaptable problem is fixed in the $k-$adaptable problem.

For this comparison, we consider the scenario generation algorithm SG and the branch-and-bound algorithm BB, with two formulations X^{agg} and X^{conf}. For each of the four methods, we compare the relative distance of the upper bound provided by the algorithm to the true optimal value of each instance obtained through our exact algorithm SG. In Figure 12, we represent a scatter plot with this relative gap along with the solution time of each algorithm for each value of $K\in\{2,3,4,5,6\}$ and $|N|=300$. The detailed results for instances of size $|N|\in\{100,150,200,250,300\}$ are given in Tables 1 and 2. Comparing the results of the two formulations for BB displayed on Figure 12, we realize that while the solution times may differ based on the formulation, the solution obtained does not change. Similar results were obtained for smaller instances and the details of hBB are provided with X^{conf}, which is faster than X^{agg}, on Tables 1 and 2. This behavior is

due to the fact that hBB solves an MILP to exact optimality at every iteration, so in the absence of multiple optima the same optimal solutions are returned irrespective of the formulations used. On the other hand, the heuristic variant of the scenario generation algorithm solves (29) heuristically and therefore the solution quality depends on the linear relaxation of the respective formulation.

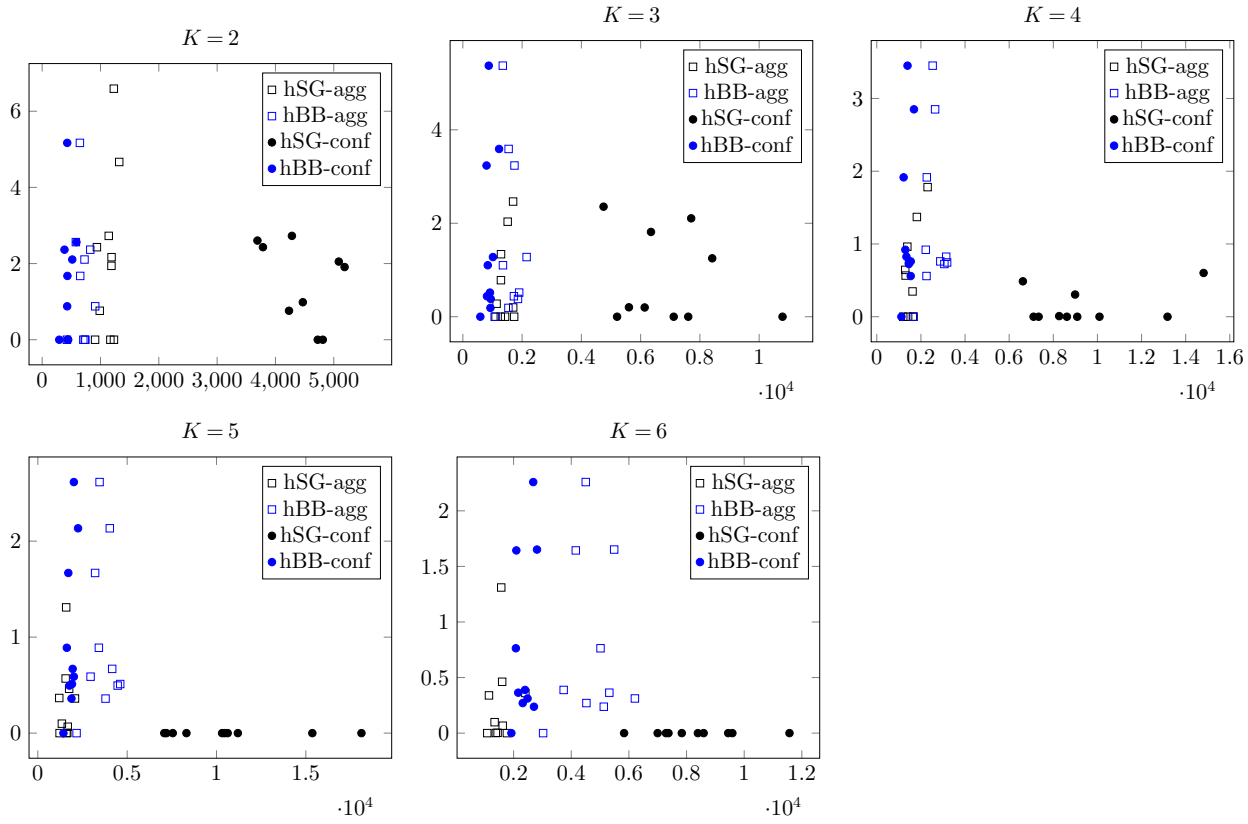

Figure 12 **Solution times (horizontal axis) and ratio to the best solution (vertical axis) for the instances with 300 items.**

The results presented in Figure 12 lead to the conclusion that the heuristic scenario generation algorithm with the formulation X^{conf}, hSG-conf, leads to better quality solutions, especially as K increases. However, this algorithm is costly in terms of computation time which prohibits its utilisation for larger instances. Further, among the remaining three methods, hBB-conf and hSG-agg, are compatible in terms of the solution time required, while the solution time for hBB-agg increases significantly with K. Therefore, hBB-conf and hSG-agg, emerge as computationally viable heuristic methods as N and K increase. Among these two methods hSG-agg provides consistently good quality solutions.

	100			150			200			250			300		
K	hSG-agg	hSG-conf	hBB-conf	hSG-agg	hSG-conf	hBB-conf	hSG-agg	hSG-conf	hBB-conf	hSG-agg	hSG-conf	hBB-conf	hSG-agg	hSG-conf	hBB-conf
2	9	20	2	52	94	8	149	311	32	474	1649	120	1130	4574	439
3	10	24	3	73	132	17	182	402	66	539	2450	258	1416	6968	894
4	10	26	5	96	181	25	207	505	101	685	3236	399	1564	9410	1403
5	10	25	7	89	170	35	206	565	139	638	3687	577	1565	10621	1855
6	10	25	8	78	172	44	198	529	187	625	3196	730	1537	8292	2365

Table 1 Average solution time for each heuristic algorithm in seconds.

	100			150			200			250			300		
K	hSG-agg	hSG-conf	hBB-conf	hSG-agg	hSG-conf	hBB-conf	hSG-agg	hSG-conf	hBB-conf	hSG-agg	hSG-conf	hBB-conf	hSG-agg	hSG-conf	hBB-conf
2	1.0	0.7	1.8	2.5	2.2	1.9	1.2	1.2	1.4	1.3	2.3	1.1	2.1	1.8	1.5
3	0.5	0.3	1.3	1.4	1.0	1.9	0.2	0.4	1.0	1.0	1.0	1.3	0.7	0.8	1.6
4	0.5	0.0	0.5	0.4	0.3	1.4	0.3	0.2	0.5	0.6	0.3	0.8	0.6	0.1	1.3
5	0.5	0.0	0.1	0.2	0.0	0.7	0.2	0.0	0.3	0.3	0.0	0.4	0.3	0.0	1.0
6	0.5	0.0	0.0	0.4	0.0	0.2	0.2	0.0	0.1	0.3	0.0	0.2	0.3	0.0	0.8

Table 2 Average relative optimality gap obtained with each heuristic algorithm.