KATHOLIEKE UNIVERSITEIT
LEUVEN

Faculty of Business and Economics

Complexity results and exact algorithms for robust knapsack problems

Fabrice Talla Nobibon and Roel Leus

DEPARTMENT OF DECISION SCIENCES AND INFORMATION MANAGEMENT (KBI)

KBI 1118

Complexity results and exact algorithms for robust knapsack problems

Fabrice Talla Nobibon*[‡], Roel Leus[†]

Abstract. This paper studies the robust knapsack problem, for which solutions are, up to a certain point, immune to data uncertainty. We complement the works found in the literature where uncertainty affects only the profits or only the weights of the items by studying the complexity and approximation of the general setting with uncertainty regarding both the profits and the weights, for three different objective functions. Furthermore, we develop a scenario-relaxation algorithm for solving the general problem and present computational results.

Keywords: knapsack problem; robustness; scenario-relaxation algorithm; NP-hard; approximation.

1. Introduction

Many real-life problems can be modeled either as a knapsack problem or as one of its variants, we refer to [1–3] for more details. The outputs of these deterministic models, however, suffer from imprecisions that make their practical implementation almost impossible in some cases [4, 5]. The imprecisions of the deterministic models usually stem from the lack of full information about the parameters of the problem and/or the dependence of these parameters on some uncontrolled events [4, 6]. Recently, a number of models have been built to capture such uncertainty, either by including random variables and solving a stochastic model [7, 8], or by considering all possible scenarios affecting the parameters of the problem [4, 9, 10]. The former approach requires additional study for the determination of the appropriate probability distributions. The latter, on the other hand, can be chosen even when complete information on the probability of occurrence of the individual scenarios is not available, which is also the setting in which we work in this article.

In this paper we examine *robust* solution procedures for knapsack problems, meaning that the produced solutions are, up to a certain point, immune to data uncertainty [11]. For short, we will speak of *robust knapsack problems*. We consider the case where uncertainty can affect both the profits and the weights of the items, and thus complement the works found in the literature where uncertainty affects only the profits of the items [4, 6, 12, 13] or only the weights of the items [14]. We investigate both *discrete scenarios* as well as *interval scenarios* – in the former case, the possible values for the profits and the weights are in a discrete set [4], whereas the latter case assumes the values to be in a given interval [15–17]. For evaluation of the quality of a solution, three different criteria are considered: the *absolute robustness criterion*, the *min-max regret criterion* and the *min-max relative regret criterion*. For more details about these criteria and their practical interpretation, we refer to [4, 16]. In the next section, we will provide formal definitions for these three types of objective functions.

*QuantOM, HEC-Management School, University of Liège, Rue Louvrex 14, Building N1, B-4000 Liège, Belgium. E-mail: Fabrice.TallaNobibon@ulg.ac.be

[‡]KULeuven, research group ORSTAT, Naamsestraat 69, B-3000 Leuven, Belgium.

[†]KULeuven, research group ORSTAT, Naamsestraat 69, B-3000 Leuven, Belgium. E-mail: Roel.Leus@econ.kuleuven.be

Over the last decade, a number of robust knapsack problems have been studied, mainly with uncertainty affecting only the profits of the items. The results include the complexity and the approximation of the absolute robustness and the min-max regret criterion [4,12,13, 16,18]. More details will be provided in the section devoted to the review of the literature. The aim of this paper is to complement these references by studying the complexity and approximation of the remaining cases. The results are extended to the general setting with uncertainty affecting both the profits and the weights. We also develop a scenario-relaxation algorithm and a heuristic for solving the general problem and present computational results.

The remainder of this article is structured as follows. First, we provide a formal description of the problems to be studied in Section 2. In Section 3, we survey the existing literature. Section 4 looks into the knapsack problem with absolute robustness criterion, Section 5 is devoted to the study of the min-max regret criterion and in Section 6 we deal with min-max relative regret, each time for discrete scenarios. Section 7 reviews the interval-scenario case. We comment the results of our computational experiments in Section 8 and conclude in Section 9.

2. Problem statement

Given is a set $N = \{1, \dots, n\}$ of n items, a set S of scenarios affecting the items and a capacity b of the knapsack. We assume that $S \neq \emptyset$. Each scenario $s \in S$ is a $2n$-vector (V^s, A^s), where $V^s = (v_1^s, \dots, v_n^s)$ is the vector of profits and $A^s = (a_1^s, \dots, a_n^s)$ the vector of weights. The quantity v_i^s (respectively a_i^s) is the profit (respectively the weight) of item i under scenario s. We assume that for every scenario $s \in S$, $0 \le a_i^s \le b$ for $i = 1, \dots, n$ and there exists at least one $s \in S$ with $\sum_{i=1}^n a_i^s > b$. With each scenario $s \in S$ corresponds a (classic) knapsack problem defined by:

$$\begin{aligned} (\mathrm{KP}_s) \qquad \max_X \qquad & F_s(X) = \sum_{i=1}^n v_i^s x_i \\ s.t. \qquad & \sum_{i=1}^n a_i^s x_i \le b, \\ & x_i \in \{0,1\} \quad i = 1, \dots, n. \end{aligned}$$

Let F_s^* be the optimal objective value of KP_s. For a given solution $X = (x_1, \dots, x_n) \in \{0,1\}^n$ satisfying $\sum_{i=1}^n a_i^s x_i \le b$, the regret of X under the scenario s is the value $F_s^* - F_s(X)$. For a specific scenario $s \in S$ and solution $X \in \{0,1\}^n$, if $\sum_{i=1}^n a_i^s x_i > b$ we adopt the convention $F_s(X) = -\infty$. For a given $X \in \{0,1\}^n$, we define the *maximum regret* $Z(X) = \max_{s \in S} \{F_s^* - F_s(X)\}$. Note that if there exists $s \in S$ with $\sum_{i=1}^n a_i^s x_i > b$ then $Z(X) = +\infty$.

Let K be the set of feasible solutions for all scenarios, i.e., $K = \{X \in \{0,1\}^n : \sum_{i=1}^n a_i^s x_i \le b, \ \forall s \in S\}$. In this paper, we examine robust knapsack problems with the following three objective functions:

(1) maximization of the *absolute robustness*:

$$\max_{X \in K} \min_{s \in S} F_s(X);$$

(2) minimization of the *maximum regret* (or the *worst-case regret*), which corresponds with

$$\min_{X\in K} Z(X) = \min_{X\in K} \max_{s\in S} \left\{F_s^* - F_s(X)\right\};$$

and

(3) minimization of the *maximum relative regret* (or *min-max relative regret*):

$$\min_{X\in K} \max_{s\in S} \left\{\frac{F_s^* - F_s(X)}{F_s^*}\right\}.$$

Our objectives are to study for each of the above criteria the complexity and the approximability of the problem and to present an algorithm to solve the general setting. We consider separately the case with discrete scenarios and the case with interval scenarios.

3. Literature review

The deterministic knapsack problem is a well studied problem, we refer to [1, 2] for an overview. This review will therefore only discuss the most closely related articles on the subject of robust knapsack problems.

There are a handful of existing articles that explicitly deal with robust knapsack problems. Yu [19] and Kouvelis and Yu [4] consider a special case of the absolute robustness criterion where uncertainty affects only the profits of items in a discrete manner. They prove that the problem is strongly NP-hard when the number of scenarios is unbounded and devise a pseudo-polynomial-time algorithm for solving the problem when the number of scenarios is bounded by a constant. Lida [6] describes exact algorithms able to solve instances with up to 60 items. Taniguchi et al. [13] study the same problem and present a fast heuristic and an exact branch-and-bound algorithm. Recently, Sbihi [12] has presented a local-search algorithm able to provide solutions for instances with up to 10 000 items and 100 scenarios. Aissi et al. [18] study the approximability of the problem and find that there exists a fully-polynomial-time approximation algorithm (FPTAS) for this problem when S is bounded. They also prove that when S is unbounded there is no FPTAS unless $P = NP$, but there still exists a polynomial-time approximation algorithm (PTAS).

For the min-max regret criterion, Kouvelis and Yu [4] provide a pseudo-polynomial algorithm for solving the problem when S is bounded and Aissi et al. [18] show that there is no approximation scheme unless $P = NP$. When S is unbounded, the problem is strongly NP-hard and there is no approximation scheme [16, 18]. Kress et al. [20] consider a robust multi-dimensional knapsack problem where the objective is to minimize the maximum regret. They show that the problem is NP-hard and develop a practically efficient algorithm for solving it.

Kalai and Vanderpooten [21] introduce a new notion of absolute robustness for the knapsack problem, called the lexicographic α-robustness criterion. Here, a profit vector containing the lowest profits is associated with every feasible solution, and two profit vectors are compared by considering the first distinct coordinates of these vectors. They show that the complexity of the lexicographic α-robust knapsack problem does not increase compared to the absolute robustness version and present a pseudo-polynomial algorithm in the case of a bounded number of scenarios.

Bertsimas and Sim [11] propose a new approach to robust optimization: they look into the trade-off between the probability of violation of constraints and the effect on the objective function of the deterministic problem, which is what they call the price of robustness. This approach is applied to the knapsack problem for the case where uncertainty affects only the weights. Klopfenstein and Nace [14] define a robust chance-constrained knapsack problem following the approach of Bertsimas and Sim and propose an approximation algorithm.

Finally, we mention that Deineko and Woeginger [22] prove that the robust knapsack problem with interval scenarios is complete for the second level of the polynomial hierarchy. We refer to [23] for more details about the polynomial hierarchy approach in computational complexity.

4. Absolute robustness

In this section, we study the robust knapsack problem with the absolute robustness criterion when the scenario set is discrete. Explicitly, the problem is given by:

$$\begin{array}{llll} (\text{AbKP}) & \max\limits_{X} \min\limits_{s\in S} & \sum\limits_{i=1}^{n} v_i^s x_i & \\ & s.t. & \sum\limits_{i=1}^{n} a_i^s x_i \le b & \forall s \in S, \\ & & x_i \in \{0,1\} & i = 1,\dots,n, \end{array}$$

with S a discrete set of distinct scenarios. The results below reduce the set S to be considered.

Definition 4.1. *Given two scenarios $s, u \in S$, we say that scenario s dominates scenario u if both of the following sets of inequalities are satisfied:*

$$(i) \quad a_i^s \ge a_i^u, \quad i = 1,\dots,n; \qquad (ii) \quad v_i^s \le v_i^u, \quad i = 1,\dots,n.$$

A scenario u is non-dominated if there exists no $s \in S$ such that (i) and (ii) are true. A non-dominated scenario is called a *maximal scenario.* Let $\bar{S} \subseteq S$ be the set of maximal scenarios. The next result states that it suffices to consider $\bar{S}$ instead of S in the definition of AbKP.

Proposition 4.2. *An optimal solution to AbKP can be obtained by solving a reduced problem in which the set of scenarios S is replaced by $\bar{S}$.*

Proof: Let Z and $\bar{Z}$ be defined by:

$$\begin{array}{lll} Z = \max\limits_{X} \left\{ \min\limits_{s\in S} \sum\limits_{i=1}^{n} v_i^s x_i \right\} & \qquad & \bar{Z} = \max\limits_{X} \left\{ \min\limits_{s\in \bar{S}} \sum\limits_{i=1}^{n} v_i^s x_i \right\} \\ s.t. \quad \sum\limits_{i=1}^{n} a_i^s x_i \le b \quad \forall s \in S, & & s.t. \quad \sum\limits_{i=1}^{n} a_i^s x_i \le b \quad \forall s \in \bar{S}, \\ \qquad x_i \in \{0,1\} \quad i = 1,\dots,n. & & \qquad x_i \in \{0,1\} \quad i = 1,\dots,n. \end{array}$$

We want to show that $Z = \bar{Z}$. If $S = \bar{S}$, the equality is true. Suppose now that $S \neq \bar{S}$. The inclusion $\bar{S} \subseteq S$ implies that $Z \leq \bar{Z}$.

On the other hand, let $X^0 \in \{0,1\}^n$ be a feasible solution for the reduced problem and $s^0 \in \bar{S}$ such that $\bar{Z} = \sum_{i=1}^n v_i^{s^0} x_i^0$. For each scenario $s \in S \setminus \bar{S}$ there exists a scenario $u \in \bar{S}$ such that s is dominated by u. We have:

$$u \in \bar{S} \text{ implies that } \sum_{i=1}^n a_i^u x_i^0 \leq b, \qquad (\star)$$

$$u \text{ dominates } s \text{ implies that } \sum_{i=1}^n a_i^s x_i^0 \leq \sum_{i=1}^n a_i^u x_i^0. \qquad (\star\star)$$

$(\star)$ and $(\star\star)$ imply that X^0 is feasible for any scenario $s \in S \setminus \bar{S}$. Consequently, X^0 is feasible for the unreduced problem and we have $\bar{Z} = \sum_{i=1}^n v_i^{s^0} x_i^0 \leq Z$. Because $Z \leq \bar{Z}$ and $\bar{Z} \leq Z$, we conclude that $Z = \bar{Z}$. □

In the remainder of this section, we present some special cases of AbKP before looking at the general setting.

4.1 S is a Cartesian product

For each item $i \in \{1, \ldots, n\}$, we distinguish a set S_i^v of possible values for v_i, defined as follows: $S_i^v = \{v_i^s : s \in S\}$; set S_i^a similarly contains the possibilities for a_i. Notice that $|S_i^a|$ and $|S_i^v|$ can be significantly less than $|S|$, where $|\cdot|$ denotes the cardinality. We define the Cartesian product $\Pi = (\prod_{i=1}^n S_i^v) \times (\prod_{i=1}^n S_i^a)$; each element of Π is also called a scenario. Remark that $S \subseteq \Pi$. Consider the scenario $\bar{s} \in \Pi$ defined as follows: for each item $i \in \{1, \ldots, n\}$, the profit and the weight of item i under scenario $\bar{s}$ are given by $v_i^{\bar{s}} = \min S_i^v$ and $a_i^{\bar{s}} = \max S_i^a$. It is easy to see that $\bar{s}$ is the only maximal scenario of Π.

Lemma 4.3. *If $\bar{s} \in S$ then it is the only maximal scenario of S.*

Proof: This follows from the fact that $S \subseteq \Pi$ and $\bar{s}$ is the only maximal scenario of Π. □

Note that $\bar{s} \in S$ if $S = \Pi$.

Lemma 4.4. *If $\bar{s} \in S$ then the problem AbKP can be solved in pseudo-polynomial time.*

Proof: As a direct consequence of Proposition 4.2 and Lemma 4.3, the problem is then equivalent to a deterministic knapsack problem. □

4.2 Uncertainty affects only the profits of items

This is a special case where $A^s = A$ for each scenario $s \in S$. This special case has been studied by Yu [19] and Taniguchi et al. [13]. Yu proves that if S is unbounded then the problem is strongly NP-hard. He also provides a pseudo-polynomial-time algorithm based on dynamic programming (DP) by weight for solving the problem when $|S|$ is bounded by a constant. Recently, Taniguchi et al. [13] have presented a heuristic and an exact algorithm for solving the robust knapsack problem when uncertainty affects only the profits of items and Sbihi [12] has described an efficient local-search algorithm for solving the same problem. Aissi et al. [18] prove that there exists a FPTAS for solving the problem when S is bounded. When S is unbounded, however, they show that there is no approximation scheme.

4.3 Uncertainty affects only the weights of items

In this case, $V^s = V$ for each scenario $s \in S$. The problem AbKP reduces to:

$$\begin{array}{lll} (\text{AbKP\_W}) \quad \max & \displaystyle\sum_{i=1}^{n} v_i x_i & \\ \qquad\qquad\quad s.t. & \displaystyle\sum_{i=1}^{n} a_i^s x_i \leq b & \forall s \in S, \\ & x_i \in \{0,1\} & i = 1,\ldots,n. \end{array}$$

If the size of the scenario set S is bounded by a constant, the problem AbKP_W is a special case of the multi-dimensional knapsack problem [1] because the right-hand sides of the constraints are identical; the latter is solved in pseudo-polynomial time. We obtain the following approximation result.

Proposition 4.5. *When the set S of scenarios is bounded, the problem AbKP_W has a PTAS but does not have a FPTAS unless $P = NP$.*

Proof: The fact that AbKP_W has a PTAS follows from Theorem 9.4.3 in [1] while the proof of the non-existence of a FPTAS is obtained via a slight modification of Theorem 9.4.1 in [1]. □

When S is unbounded, on the other hand, the next theorem shows that the problem AbKP_W is strongly NP-hard. The proof uses a reduction from the 3-Dimensional Matching (3DM) problem defined as follows:

Instance: Set $M \subseteq W \times X \times Y$, where W, X and Y are disjoint sets having the same number q of elements.
Question: Does M contain a matching, i.e., a subset $M' \subseteq M$ such that $|M'| = q$ and no two elements of M' agree in any coordinate?

The 3DM problem is proved to be strongly NP-complete by Garey and Johnson [24].

Theorem 4.6. *The problem AbKP_W is strongly NP-hard for an unbounded scenario set S.*

Proof: Consider an arbitrary instance of 3DM. We describe a polynomial transformation into an instance of AbKP_W.

The set of scenarios $S = W \cup X \cup Y$, therefore $|S| = 3q$. We have $n = |M|$ items and we write $M = \{M_1, M_2, \ldots, M_n\}$. The capacity of the knapsack is $b = 1$ and for each item $i \in \{1,\ldots,n\}$, the profit $v_i = 1$. For a given scenario s and item i, the weight of item i under scenario s is defined by $a_i^s = 1$ if $s \in M_i$ and 0 otherwise. In every scenario there are three unit-weight items. Clearly, this transformation can be done in polynomial time. We will show that the instance of 3DM is a YES instance if and only if the AbKP_W instance has a solution with an objective value greater than or equal to q.

If the 3DM instance is a YES instance then there exists $M' \subseteq M$ with $|M'| = q$ and M' is a matching. We set $x_i = 1$ if $M_i \in M'$ and $x_i = 0$ otherwise. This is a feasible solution to AbKP_W and it achieves an objective value of q.

Conversely, suppose that the constructed AbKP_W instance has a feasible solution with an objective value greater than or equal to q. If there were a solution with objective value strictly greater than q then there would be M_i and M_j in M' such that $M_i \cap M_j \neq \emptyset$, and for a scenario $s \in M_i \cap M_j$ the capacity b would be exceeded. Consequently, the objective value is equal to q. Let $M' = \{M_i \in M : x_i = 1\}$. Because the objective value is exactly q, we have $|M'| = q$. The fact that $(x_1, \dots, x_n)$ is a feasible solution to AbKP_W implies that M' is a matching. □

4.4 General case

We now consider the general problem AbKP with uncertainty regarding both the weights and the profits. We assume that $S = \bar{S}$; if this is not the case, $\bar{S}$ can be identified in polynomial time via the following procedure. Given a set S of scenarios, let $M = \max_{s \in S} \max_{1 \le i \le n} v_i^s$. The quantity M is a maximum over a set of $n \times |S|$ elements. We associate a $2n$-vector T^s with each scenario $s \in S$, with $T^s = \left(t_1^s, \dots, t_n^s, t_{n+1}^s, \dots, t_{2n}^s\right)$ where $t_i^s = M - v_i^s$ if $1 \le i \le n$ and $t_i^s = a_{i-n}^s$ if $n + 1 \le i \le 2n$. Given two scenarios s and u, s dominates u if and only if $t_i^s \ge t_i^u, \;\; i = 1, \dots, 2n$. Efficient algorithms for identifying dominated scenarios with the latter input data are available in the literature, see [25], for instance.

The next result states that the problem can be solved in pseudo-polynomial time when S is bounded.

Lemma 4.7. *If $|S|$ is bounded by a constant then AbKP can be solved in pseudo-polynomial time.*

We describe a pseudo-polynomial-time algorithm based on DP. We use the following value function:

$$F_k\left(\alpha_1, \dots, \alpha_{|S|}; b_1, \dots, b_{|S|}\right) = \max_X \min_{s \in S} \left\{ \sum_{i=k}^{n} v_i^s x_i + \alpha_s \;\middle|\; \sum_{i=k}^{n} a_i^s x_i \le b_s, \;\; s \in S \text{ and } x_i \in \{0,1\}, \;\; i = k, \dots, n \right\},$$

i.e., $F_k\left(\alpha_1, \dots, \alpha_{|S|}; b_1, \dots, b_{|S|}\right)$ is the max-min value when the optimal selection is made among the items $k, k+1, \dots, n$ under the knapsack capacity b_s for scenario s, and having already collected a profit α_s for scenario $s \in S$ in the items $1, 2, \dots, k-1$. The initial condition is:

$$F_n\left(\alpha_1, \dots, \alpha_{|S|}; b_1, \dots, b_{|S|}\right) = \begin{cases} \min\left\{\alpha_1 + v_n^1, \dots, \alpha_{|S|} + v_n^{|S|}\right\} & \text{if } b_s \ge a_n^s, \;\; \forall s \in S, \\ \min\left\{\alpha_1, \dots, \alpha_{|S|}\right\} & \text{otherwise.} \end{cases}$$

We have the recursive relation (for $k = n-1, n-2, \ldots, 2$):

$$F_{k-1}\left(\alpha_1, \ldots, \alpha_{|S|}; b_1, \ldots, b_{|S|}\right) = \begin{cases} F_k\left(\alpha_1, \ldots, \alpha_{|S|}; b_1, \ldots, b_{|S|}\right) & \text{if } \exists s \in S: \; b_s < a_{k-1}^s, \\ \max\Big\{ F_k\left(\alpha_1, \ldots, \alpha_{|S|}; b_1, \ldots, b_{|S|}\right), \\ \quad F_k\left(\alpha_1 + v_k^1, \ldots, \alpha_{|S|} + v_k^{|S|}; b_1 - a_k^1, \ldots, b_{|S|} - a_k^{|S|}\right) \Big\} & \text{otherwise.} \end{cases}$$

The optimal objective value is $F_1(0, \ldots, 0; b, \ldots, b)$. The time complexity of the DP algorithm is $O\left(nb^{|S|}L^{|S|}\right)$, with $L = \max_{s \in S} \sum_{i=1}^n v_i^s$. Thus, if $|S|$ is bounded by a constant, this algorithm runs in pseudo-polynomial time. Example 1 illustrates the application of the above DP algorithm.

Example 1. *Consider the following instance of AbKP with $|S| = 2$ scenarios, a knapsack capacity $b = 8$ and $n = 4$ items.*

$$\begin{aligned} \max_X \quad & \min\{3x_1 + 7x_2 + 4x_3 + 5x_4, \;\; 5x_1 + 4x_2 + 2x_3 + 3x_4\} \\ s.t. \quad & 3x_1 + 6x_2 + 2x_3 + 4x_4 \le 8 \\ & 2x_1 + 5x_2 + 4x_3 + 3x_4 \le 8 \\ & x_i \in \{0, 1\}, \quad i = 1, \ldots, 4. \end{aligned}$$

We find that $F_1(0,0;8,8) = F_2(3,5;5,6) = F_3(3,5;5,6) = F_4(3,5;5,6) = 8$ and an optimal solution is $x_1 = x_4 = 1$, $x_2 = x_3 = 0$.

Observe that the number of states in the above DP is strongly affected by the order of magnitude of the profit and the weight of the items, the number of scenarios and the value of the budget. More concretely, the DP is expected to be efficient if the profit and the weight of the items are small, and there are few scenarios. Because our experiments in Section 8 focus on instances with many scenarios (up to 10 000) and items having profit and weight between 1 and 190, running the above DP does not seem practical and hence the algorithm has not been implemented. A different algorithm will be proposed further in this section.

The next proposition contains the approximation results for the general setting.

Proposition 4.8. *If the set S is bounded, the problem AbKP has a PTAS but not a FPTAS.*

Proof: The existence of a PTAS for AbKP follows from the application of Theorem 1 in [18] since the multi-objective version of the multi-dimensional knapsack problem has a PTAS [26]. On the other hand, there is no FPTAS for AbKP because AbKP_W does not have a FPTAS. □

We will next study the case where $|S|$ is unbounded. In this case, the problem is strongly NP-hard since it contains the special cases studied in the previous subsections. The negative results obtained for the above cases are also valid for this general setting. As a result, we infer that there is no approximation scheme when $|S|$ is unbounded.

A linear formulation for AbKP is:

$$
\begin{aligned}
(\mathrm{M}_1) \quad \max \quad & y \\
s.t. \quad & y \leq \sum_{i=1}^{n} v_i^s x_i \quad \forall s \in S, \\
& \sum_{i=1}^{n} a_i^s x_i \leq b \quad \forall s \in S, \\
& x_i \in \{0,1\} \qquad i = 1, \ldots, n, \\
& y \geq 0.
\end{aligned}
$$

As mentioned above, we may assume that $S = \bar{S}$, where $\bar{S}$ is the set of maximal scenarios. Note that this assumption is not necessary for the application of the algorithm described below.

The cardinality of S guides the choice of the algorithm for solving AbKP. On the one hand, if S contains only few scenarios, then it is practical to directly solve M_1 using any mixed-integer programming (MIP) solver. If the cardinality of S is large, on the other hand, we follow the idea of Assavapokee et al. [15, 27] and propose a *scenario-relaxation algorithm*

Algorithm 1 Scenario-relaxation algorithm for AbKP

```
1: Choose a subset $\Omega \subseteq S$ and set $UB = +\infty$, $LB = 0$ and $\varepsilon = \min_{s,i} v_i^s$
2: Solve the relaxation of model $(\mathrm{M}_1)$ by considering only the scenario set $\Omega$ instead of $S$
3: Let $x^\star$ and $y^\star$ be an optimal solution to the relaxation
4: $x^\Omega := x^\star$ and $UB := y^\star$
5: if $|UB - LB| < \varepsilon$ then
6:     stop
7: else
8:     $W_1 := \{s \in S^1 \setminus \Omega \mid \sum_{i=1}^n a_i^s x_i^\Omega > b\}$
9:     if $W_1 \neq \emptyset$ then
10:        Select a non-empty subset $W_1' \subseteq W_1$ and update $\Omega \leftarrow \Omega \cup W_1'$; goto 2
11:    else
12:        For all $s \in S^2 \setminus \Omega$, compute $\delta^s := \sum_{i=1}^n v_i^s x_i^\Omega$; $W_2 := \{s \in S^2 \setminus \Omega \mid \delta^s < y^\star\}$
13:        if $W_2 = \emptyset$ then
14:            $LB := y^\star$, stop
15:        else
16:            $\delta := \min_{s \in W_2} \delta^s$, $LB := \max\{LB, \delta\}$
17:            if $|UB - LB| < \varepsilon$ then
18:                stop
19:            else
20:                Select a non-empty subset $W_2' \subseteq W_2$ and update $\Omega \leftarrow \Omega \cup W_2'$; goto 2
21:            end if
22:        end if
23:    end if
24: end if
```

for solving AbKP. We distinguish two (not necessarily disjoint) subsets of S: $S^1 \subset S$ is the set of scenarios required to ensure that a given solution is feasible for all possible scenarios and $S^2 \subset S$ contains the scenarios required to establish that a given feasible solution is optimal.

Definition 4.9. *Given two scenarios s and u in S, u is weight-dominated by s if $a_i^u \leq a_i^s$ for $i = 1, \dots, n$; and u is value-dominated by s if $v_i^u \geq v_i^s$ for $i = 1, \dots, n$.*

Using Definition 4.9, S^1 and S^2 are explicitly defined as follows: $S^1 = \{s \in S \mid s \text{ is not weight-dominated}\}$ and $S^2 = \{s \in S \mid s \text{is not value-dominated}\}$ and we have the following straightforward result.

Lemma 4.10. *The set $\bar{S}$ of maximal scenarios is the union of S^1 and S^2; that is $\bar{S} = S^1 \cup S^2$.*

A scenario-relaxation algorithm for solving (M_1) follows the structure of Algorithm 1. This algorithm solves a relaxed version of (M_1) that contains only the constraints corresponding to a subset $\Omega \subseteq S^1 \cup S^2$ of scenarios and iteratively adds scenarios until it is guaranteed that the solution obtained is (feasible and) optimal to the full model (M_1). Notice that adding one scenario involves adding two constraints to the restricted version of (M_1), one constraint for the feasibility and the other to enforce the optimality. Adding a single constraint is an option that we have not considered because the scenario corresponding with that constraint may have to be generated again later in the course of the algorithm to ensure either the feasibility or the optimality. The correctness of Algorithm 1 follows from Proposition 4.2 and Lemma 4.10.

5. Min-max regret robust knapsack problem

This section is devoted to the study of the robust knapsack problem with the min-max regret criterion. We still consider the set S of scenarios to be discrete. The problem formulation is given by:

$$\begin{aligned}
(\mathrm{RgKP}) \qquad \min_{X} \max_{s \in S} \qquad & F_s^* - \sum_{i=1}^{n} v_i^s x_i \\
s.t. \qquad & \sum_{i=1}^{n} a_i^s x_i \leq b \quad \forall s \in S, \\
& x_i \in \{0, 1\} \quad i = 1, \dots, n.
\end{aligned}$$

We first study two special cases, namely the case where uncertainty affects only the profits of items and the case where it affects only the weights. Subsequently, we look at the general setting with uncertainty about both the profits and the weights.

5.1 Uncertainty affects only the profits of items

This special case occurs when $A^s = A$ for each scenario $s \in S$. Kouvelis and Yu [4] study this problem when the size of S is bounded and provide a pseudo-polynomial-time algorithm based on DP by weight for solving the problem. Aissi et al. [18] show that it is impossible that an approximation scheme exists for that problem when S is bounded, unless $P = NP$. When $|S|$ is unbounded, the problem is strongly NP-hard and does not have any approximation scheme unless $P = NP$ [16, 18].

5.2 Uncertainty affects only the weights of items

In this case, $V^s = V$ for each scenario $s \in S$. The problem RgKP reduces to:

$$\begin{aligned} \max_X \quad & \sum_{i=1}^{n} v_i x_i - F^* \\ s.t. \quad & \sum_{i=1}^{n} a_i^s x_i \le b \quad \forall s \in S, \\ & x_i \in \{0,1\} \quad i = 1, \ldots, n, \end{aligned}$$

where $F^* = \max_{s \in S} F_s^*$; this value F^* is a constant and can therefore be removed from the objective function. This leaves us with the problem AbKP_W studied in Section 4.3.

5.3 General case

We now consider the general problem RgKP with uncertainty both on the weights and on the profits. If the size of the set S of scenarios is bounded by a constant then the problem can be solve in pseudo-polynomial time since the $|S|$ values F_s^* for $s \in S$ are computed in pseudo-polynomial time and an adapted version of the DP algorithm devised for the case of the absolute robustness criterion (see Section 4.4) can be applied to find an optimal solution. The negative results obtained for the special cases imply that there is no approximation scheme for the general case even when the set S is bounded, unless $P = NP$.

Let us now assume that S is unbounded. Clearly, the problem is strongly NP-hard and does not have an approximation scheme. To solve the problem, we again develop a scenario-relaxation algorithm. A linear formulation associated with RgKP is:

$$\begin{aligned} (\mathrm{M}_2) \quad \min \quad & y \\ s.t. \quad & F_s^* - \sum_{i=1}^{n} v_i^s x_i \le y \quad \forall s \in S, \\ & \sum_{i=1}^{n} a_i^s x_i \le b \quad \forall s \in S, \\ & x_i \in \{0,1\} \quad i = 1, \ldots, n, \\ & y \ge 0. \end{aligned}$$

A scenario-relaxation algorithm for solving (M_2) can follow the same structure as Algorithm 1. The main differences with the previous algorithm are threefold: (1) at the start of the algorithm, we need to compute F_s^* by solving KP_s for every scenario s in Ω; (2) each solution of the relaxation now yields a lower bound, while each feasible solution evaluated against all scenarios produces an upper bound; (3) $\delta^s := F_s^* - \sum_{i=1}^{n} v_i^s x_i^\Omega$ and $\delta := \max_{s \in W_2} \delta^s$.

6. Min-max relative regret robust knapsack problem

This section is devoted to the study of the robust knapsack problem with the min-max relative regret criterion with discrete scenario set, defined by:

$$\begin{array}{llll} (\mathrm{ReKP}_0) & \min\limits_{X} \max\limits_{s \in S} & \dfrac{F_s^* - \sum_{i=1}^n v_i^s x_i}{F_s^*} & \\ & s.t. & \displaystyle\sum_{i=1}^n a_i^s x_i \le b \quad \forall s \in S, & \\ & & x_i \in \{0,1\} \quad i = 1, \ldots, n. & \end{array}$$

We note that the problem is well defined only if $F_s^* \neq 0$ for every $s \in S$; throughout this section we assume this to be true. The objective function can be rewritten as follows:

$$\min_X \max_s \frac{F_s^* - \sum_{i=1}^n v_i^s x_i}{F_s^*} = 1 + \min_X \max_s -\frac{1}{F_s^*} \sum_{i=1}^n v_i^s x_i = 1 - \max_X \min_s \frac{1}{F_s^*} \sum_{i=1}^n v_i^s x_i \, .$$

Therefore, solving the robust knapsack problem with the min-max relative regret criterion is equivalent to solving the following problem.

$$\begin{array}{llll} (\mathrm{ReKP}) & \max\limits_{X} \min\limits_{s} & \dfrac{1}{F_s^*} \displaystyle\sum_{i=1}^n v_i^s x_i & \\ & s.t. & \displaystyle\sum_{i=1}^n a_i^s x_i \le b \quad \forall s \in S, & \\ & & x_i \in \{0,1\} \quad i = 1, \ldots, n. & \end{array}$$

In the rest of this section, we study the complexity of and algorithms for ReKP and resort to ReKP_0 only for the study of approximation. As before, we distinguish two special cases before proceeding with the general setting.

6.1 Uncertainty affects only the profits of items

In this case, $A^s = A$ for each scenario $s \in S$. If S is bounded then the $|S|$ values F_s^* for $s \in S$ can be computed in pseudo-polynomial time. Let δ be the least common multiple of the $|S|$ values F_s^* for $s \in S$ and $\delta_s = \frac{\delta}{F_s^*}$ for $s \in S$. ReKP is then equivalent to the following problem:

$$\begin{array}{lll} \max\limits_{X} \min\limits_{s \in S} & \displaystyle\sum_{i=1}^n \delta_s v_i^s x_i & \\ s.t. & \displaystyle\sum_{i=1}^n a_i x_i \le b & \\ & x_i \in \{0,1\} \quad i = 1, \ldots, n. & \end{array}$$

The latter problem is a robust knapsack problem with absolute robustness criterion as presented in Section 4.2, and can be solved in pseudo-polynomial time. The following result shows that there is no approximation algorithm for (ReKP_0).

Theorem 6.1. *The problem (ReKP$_0$) has no approximation algorithm even for two scenarios, unless $P = NP$.*

Proof: The proof of Theorem 6 in [18] for the min-max regret objective, which uses a gap-introducing reduction from PARTITION, is valid for proving this result. $\square$

When the set S of scenarios is unbounded, we have the following result.

Theorem 6.2. *The problem (ReKP$_0$) with an unbounded set S of scenarios is strongly NP-hard.*

Proof: The proof of Theorem 7 in [18], which uses a reduction from VERTEX COVER, is easily adapted to prove this result. $\square$

6.2 Uncertainty affects only the weights of items

When only the weights are uncertain then we have $V^s = V$ for each $s \in S$. In this case, the robust knapsack problem with min-max relative regret criterion reduces to the problem AbKP_W studied in Section 4.3.

6.3 General case

In this subsection, we consider the min-max relative regret knapsack problem with uncertainty both on the weights and the profits. If $|S|$ is bounded, the problem is solvable in pseudo-polynomial time since the $|S|$ values F_s^* are computed in pseudo-polynomial time and an adapted version of the DP algorithm for the case of absolute robustness can be used to find an optimal solution. The negative results obtained for the special cases imply that there is no approximation scheme for the general case even when the set S is bounded.

Let us now assume that S is unbounded. Clearly, the problem is strongly NP-hard and does not have an approximation scheme. An appropriate linear formulation is:

$$\begin{array}{lll} (\mathrm{M}_3) & \min & y \\ & s.t. & F_s^* - \displaystyle\sum_{i=1}^{n} v_i^s x_i \leq F_s^* y \quad \forall s \in S, \\ & & \displaystyle\sum_{i=1}^{n} a_i^s x_i \leq b \quad \forall s \in S, \\ & & x_i \in \{0,1\} \quad i = 1, \dots, n, \\ & & y \geq 0. \end{array}$$

The scenario-relaxation algorithm previously described for absolute robustness and min-max regret can be modified to also solve this problem.

Tables 1 and 2 summarize the complexity and the approximability results obtained for the robust knapsack problem with discrete scenario set.

Uncertainty	profit	weight	both
Absolute robustness			
Bounded	NP-hard [4]	NP-hard [1]	NP-hard
Unbounded	Strongly NP-hard [4]	Strongly NP-hard	Strongly NP-hard
Maximum regret			
bounded	NP-hard [4]	NP-hard	NP-hard
Unbounded	Strongly NP-hard [18]	Strongly NP-hard	Strongly NP-hard
Maximum relative regret			
bounded	NP-hard	NP-hard	NP-hard
Unbounded	Strongly NP-hard	Strongly NP-hard	Strongly NP-hard

Table 1: Summary of the complexity results of the robust knapsack problem with discrete set of scenarios.

Uncertainty	profit	weight	both
Absolute robustness			
bounded	FPTAS [18]	PTAS and no FPTAS	PTAS and no FPTAS
Unbounded	No approx.* [18]	No FPTAS	No approx.
Maximum regret			
bounded	No approx. [18]	PTAS	No approx.
Unbounded	No approx. [18]	No FPTAS	No approx.
Maximum relative regret			
bounded	No approx.	PTAS	No approx.
Unbounded	No approx.	No FPTAS	No approx.

* 'No approx.' means that there is no constant-factor approximation algorithm.

Table 2: Summary of the approximability results of the robust knapsack problem with discrete set of scenarios, assuming $P \neq NP$.

7. Interval scenarios

In this section we briefly study the robust knapsack problem with interval scenarios. For each item $i \in \{1, \ldots, n\}$, the profit (respectively weight) of i can take any value between a lower bound v_i^L (respectively a_i^L) and an upper bound v_i^U (respectively a_i^U). Our findings in Section 4.1 lead us to conclude that the absolute robust knapsack problem with interval scenarios is equivalent to the deterministic knapsack problem.

For the min-max regret and relative regret criteria, on the other hand, we consider the feasible set given by: $K = \{X \in \{0,1\}^n : \sum_{i=1}^n a_i x_i \leq b,\ \forall a_i \in [a_i^L, a_i^U] \text{ for } i = 1, \ldots, n\}$. Let $K^U = \{X \in \{0,1\}^n : \sum_{i=1}^n a_i^U x_i \leq b\}$. It is not difficult to see that $K = K^U$. In the remainder of this section, we replace K by K^U and use a_i instead of a_i^U. Therefore, we can consider that uncertainty affects only the profit of items. We define a discrete set S' of scenarios as the set of all 'extreme' scenarios for the profits: for all $s \in S'$ and for all item i, we have $v_i^s \in \{v_i^L, v_i^U\}$. Notice that S' contains at most 2^n scenarios. We obtain the following outcome:

Lemma 7.1. *The interval-scenario robust knapsack problem with the min-max (relative) regret criterion is equivalent to the robust knapsack problem with the same objective for the* *discrete set of scenarios S'.*

Proof: **1. min-max regret criterion**

Let $X \in K^U$. By definition, we have $Z(X) = \max\{F_s^* - F_s(X) \,:\, s \in S\}$ and we want to prove that $Z(X) = Z'(X) = \max\{F_s^* - F_s(X) \,:\, s \in S'\}$.

Clearly, we have $Z(X) \geq Z'(X)$ since $S' \subseteq S$. Let $s_0 \in S$ such that $Z(X) = F_{s_0}^* - F_{s_0}(X)$. If $s_0 \in S'$ then $Z(X) = Z'(X)$; otherwise ($s_0 \notin S'$) let $X^0 \in K^U$ such that $F_{s_0}^* = \sum_{i=1}^n v_i^{s_0} x_i^0$; then consider the scenario $s_1 \in S'$ defined as follows. If $x_i^0 = 1$ then $v_i^{s_1} = v_i^U$, else $v_i^{s_1} = v_i^L$. We have $F_{s_0}^* \leq \sum_{i=1}^n v_i^{s_1} x_i^0 \leq F_{s_1}^*$. Moreover,

$$\begin{aligned} Z'(X) \geq F_{s_1}^* - \sum_{i=1}^n v_i^{s_1} x_i \;&\geq\; \sum_{i=1}^n v_i^{s_1} x_i^0 - \sum_{i=1}^n v_i^{s_1} x_i \\ &\geq\; \sum_{i=1}^n v_i^{s_1}\left(x_i^0 - x_i\right) \\ &\geq\; \sum_{i=1}^n v_i^{s_0}\left(x_i^0 - x_i\right) \\ &\geq\; \sum_{i=1}^n v_i^{s_0} x_i^0 - \sum_{i=1}^n v_i^{s_0} x_i = Z(X). \end{aligned}$$

We conclude that $Z(X) = Z'(X)$. By taking the minimum over $X \in K^U$ we obtain the equivalence result for the min-max regret criterion.

2. min-max relative regret criterion

Let $X \in K^U$ and

$$Z''(X) = \max_{s \in S} \frac{F_s^* - F_s(X)}{F_s^*} = \max_{s \in S} \left\{1 - \frac{F_s(X)}{F_s^*}\right\} = 1 - \min_{s \in S} \frac{F_s(X)}{F_s^*} = 1 - \tilde{Z}(X).$$

Notice that given $\tilde{Z}(X)$ we can easily compute $Z''(X)$. Further, for any $X \in K^U$, we have $\tilde{Z}(X) \leq 1$ and

$$\tilde{Z}(X) = \min_{s \in S} \frac{F_s(X)}{F_s^*} = \min_{s \in S} \frac{F_s(X)}{\max_{Y \in K^U} F_s(Y)} = \min_{s \in S} \min_{Y \in K^U} \frac{F_s(X)}{F_s(Y)}.$$

Observe that to compute $\tilde{Z}(X)$, we can consider only the sets $Y \in K^U$ satisfying $\frac{F_s(X)}{F_s(Y)} \leq 1$. For any $X, Y \in K^U$, we define the scenario $s(X,Y)$ as follows: the i^{th} value component of $s(X,Y)$ is v_i^U if $i \in Y \setminus X$; otherwise, it is v_i^L.

Notice that $s(X,Y) \in S'$ for any $X, Y \in K^U$. Let $s^* \in S$, $Y^* \in K^U$ such that $\tilde{Z}(X) = \frac{F_{s^*}(X)}{F_{s^*}(Y^*)}$. We want to show that $\tilde{Z}(X) = \frac{F_{s(X,Y^*)}(X)}{F_{s(X,Y^*)}(Y^*)}$. In fact, we have

$$\frac{F_{s(X,Y^*)}(X)}{F_{s(X,Y^*)}(Y^*)} = \frac{\sum_{i \in X} v_i^L}{\sum_{i \in Y^* \setminus X} v_i^U + \sum_{i \in X \cap Y^*} v_i^L} \leq \frac{F_s(X)}{F_s(Y^*)}$$

for all scenarios $s \in S$; the last inequality follows from the fact that $\frac{F_s(X)}{F_s(Y)} \leq 1$. In particular for $s = s^*$ we obtain $\tilde{Z}(X) = \frac{F_{s(X,Y^*)}(X)}{F_{s(X,Y^*)}(Y^*)}$. □

Results similar to Lemma 7.1 are shown in [28] for subset-type combinatorial optimization when there is uncertainty only in the objective function. Observe that the above result implies that the scenario-relaxation algorithm derived earlier can be used to solve the problem with interval scenarios. Note, however, that Lemma 7.1 does not imply straightforward complexity results for the interval-scenario case because its scenario set is always a Cartesian product.

8. Computational results

All algorithms have been coded in C using Visual Studio C++ 2005; all the experiments were run on a Dell Optiplex 760 personal computer with Pentium R processor with 3.16 GHz clock speed and 3.21 GB RAM, equipped with Windows XP. CPLEX 12.2 was used for solving the linear formulations. Below, we first provide some details on the generation of the datasets and subsequently, we discuss the computational results.

8.1 Data generation

The scenario-relaxation algorithms developed in this paper are tested on randomly generated instances with n items, for $n = 1\,000$, $5\,000$ and $10\,000$. For each item i and each scenario s, an integer profit v_i^s and an integer weight a_i^s are generated.

We extend the generation process used by Sbihi [12] and Taniguchi et al. [13] to generate instances with scenarios affecting both the profits and the weights of the items. To the best of our knowledge, there is no library of robust knapsack instances and we have tried to access the instances used by the above mentioned authors without success. Furthermore, their instances are generated with uncertainty affecting only the profit of the items and contain few scenarios. For these reasons, we have chosen to generate our own instances[1]. The problem instances are generated as follows: for each item $i = 1, \ldots, n$, the initial weight a_i^0 and the initial value v_i^0 are integers randomly generated from the interval $[1, 100]$ (assuming independent uniform distributions). For a given scenario s, the weight a_i^s of each item i is a random integer selected from the interval $\left[(1-\sigma_a)a_i^0, (1+\sigma_a)a_i^0\right]$ (uniformly distributed), where $\sigma_a \in \{0.3, 0.6, 0.9\}$ is a parameter to determine the variability level of the weights in the different scenarios. For the value v_i^s, we apply the same process with interval $\left[(1-\sigma_v)v_i^0, (1+\sigma_v)v_i^0\right]$ and $\sigma_v \in \{0.3, 0.6, 0.9\}$. The closer σ_a and σ_v are to 0, the more the scenarios (and consequently the profits and/or the weights) are similar to one another. The knapsack capacity is set to $b = \theta \min_{s \in S} \sum_{1 \leq i \leq n} a_i^s$ with $\theta \in \{0.4, 0.8\}$. The parameter θ indicates whether the capacity b is tight or rather loose.

For each value of n, b, σ_a and σ_v, we generate nine instances with $|S| = 100$, $1\,000$ and $10\,000$ scenarios. In total, we have $3 \times 2 \times 3 \times 9 = 162$ instances.

8.2 Computational results

In this section, computation time is referred to as *Time* and is expressed in seconds. We first consider the application of the scenario-relaxation algorithm for solving the generated

[1]The instances can be found at http://www.econ.kuleuven.be/public/ndbac96/robustKP.htm

instances with the absolute robustness criterion. Next, we focus on the min-max regret criterion and subsequently, we consider the problem with the min-max relative regret criterion. For every objective function, we also present heuristics based on the scenario-relaxation algorithm.

8.2.1 Absolute robustness

We first present different implementations of the scenario-relaxation algorithm, followed by the description of a heuristic based on the scenario-relaxation algorithm. We compare all these algorithms to CPLEX used as a MIP solver, applied to formulation (M_1).

Different implementations and heuristics

We have investigated three different implementations of the scenario-relaxation algorithm. The first implementation, identified by *SC-Ab1*, is the implementation as described by the pseudocode of Algorithm 1. We start the algorithm with a set Ω containing two scenarios in S^1 as follows. The two initial scenarios are selected such that the set of items for which there is at least one scenario with the highest weight is maximal. At each iteration after solving the restricted MIP, two additional scenarios are added if necessary (cf. lines 10 and line 20 in the pseudocode); this number (two) was chosen after preliminary experiments. The two scenarios added are chosen among the scenarios under which the current solution performs worst. The second implementation, *SC-Ab2*, follows the main steps of Algorithm 1, but here at each iteration, we solve the LP relaxation of the restricted problem instead of solving the MIP formulation. We use the variables whose values equal one to update the lower bound. To ensure the optimality, the stopping criteria are adapted as follows. When the set of scenarios to be added is empty, the LP relaxation of the full problem is solved to optimality and we solve the restricted MIP obtained by considering the constraints added so far. If its solution does not lead to the identification of new scenarios to be added to the restricted problem, then the algorithm stops. Otherwise, two scenarios are added and the algorithm continues by solving the LP relaxation of the current problem. This procedure is executed until the algorithm stops. The rationale behind the second implementation is the following empirical observation: the time spent at each iteration of *SC-Ab1* is dominated by the time CPLEX needs to solve the restricted MIP problem.

The third implementation, *SC-Ab3*, is similar to *SC-Ab2*; the main difference occurs when there is no scenario to be added after the LP relaxation is solved to optimality. In this implementation, we consider all the variables whose values equal one in the current LP solution as fixed and we solve the corresponding constrained MIP problem (notice that this may substantially reduce the number of variables in the problem). Hence, the full restricted MIP problem is solved only when the solution to the constrained MIP does not lead to an improvement over the current best solution. The rationale for this implementation is again the desire to reduce the number of times that the full MIP problem is solved throughout the algorithm. For each of these implementations, we use a time limit of 30 minutes to interrupt the algorithm. When this happens, we say that the instance was not solved to optimality. A fourth implementation has also been investigated. In that implementation, once we have solved the LP relaxation to optimality, instead of solving the (constrained) MIP problem, we add a cut that forbids the variables whose values are one in the LP solution to be all selected in any subsequent solution. The results of this last implementation are not reported because they were not better than the results obtained using previous implementations.

The scenario-relaxation algorithm is converted into a heuristic, denoted *Heur*, as follows. We consider the implementation *SC-Ab2* and halt the algorithm when the LP relaxation of the problem is solved to optimality for the first time. At each iteration, the solution to the restricted LP is used to derive an integer solution by selecting only items whose corresponding variables take the value 1 in the LP solution. This solution is kept only if it is better than the current best solution. The best integer solution is then output by the heuristic.

Comparison of the efficiency of the algorithms

In Table 3, in addition to the results of the scenario-relaxation algorithms and the heuristic described above, we also report the results obtained using CPLEX (see *Full*) to solve the MIP formulation (M_1). Each cell in the column *Stat* contains a pair of numbers between 0 and 9, the first (respectively the second) one representing the number of instances solved to optimality (respectively the number of instances for which the algorithm produces a non-trivial solution) within the time limit. The pair 2 / 4, for instance, means that the considered algorithm solves two instances out of nine to optimality and produces a non-trivial solution for four instances out of nine; by 'non-trivial solution' we mean a feasible solution that selects at least one item. The column *Gap* reports the average gap (expressed in %) computed with respect to the lower bound produced by the algorithm and the best (smallest) upper bound amongst the upper bounds produced by the four algorithms, this by considering only instances for which the algorithm produces non-trivial solutions. Each cell in the table corresponds with the analysis of nine instances. When the algorithm fails to produce a non-trivial solution for at least one instance out of nine associated with a given cell, we use $--$ in the cell corresponding with *Gap* to express the fact that the gap is not computed.

Table 3 shows that instances with restricted budget ($\theta = 0.4$) are substantially more difficult to solve than those with large budget ($\theta = 0.8$); the difficulty also increases with the number of scenarios. The full model (M_1) solved using CPLEX (see *Full*) has a good performance when there are few scenarios ($|S| = 100$) and a large budget. In this case, CPLEX always solves all the instances regardless of the value of n. When the budget remains large and there are more scenarios ($|S| = 1\,000$), CPLEX solves all the instances when $n = 1\,000$, seven out of nine when $n = 5\,000$ and none when $n = 10\,000$ while when $|S| = 10\,000$, CPLEX fails to produce any non-trivial solution within the time limit. The results obtained using the full model (M_1) for restricted budget ($\theta = 0.4$) show that three instances are solved optimally, all with few scenarios ($|S| = 100$). The gap in this case increases with n from 0.04% to more than 57% when $n = 10\,000$. For $|S| = 1\,000$, the gap is 0.28% for $n = 1\,000$ and 43.25% for $n = 5\,000$, while no feasible solution with non-zero objective value is found when $n = 10\,000$. Finally for $|S| = 10\,000$, the full model does not find a non-trivial solution to any instance.

When looking at the results produced by the three implementations of the scenario-relaxation algorithm, we find that the third implementation *SC-Ab3* performs better than the two others. Indeed, for any group of instances, *SC-Ab3* optimally solves at least as many instances (and usually more) as any other exact algorithm (including the full model),

			Full			*SC-Ab1*			*SC-Ab2*			*SC-Ab3*			*Heur*		
n	θ	$\|S\|$	Time (*sec*)	Stat (9 / 9)	Gap (%)	Time (*sec*)	Stat (9 / 9)	Gap (%)	Time (*sec*)	Stat (9 / 9)	Gap (%)	Time (*sec*)	Stat (9 / 9)	Gap (%)	Time (*sec*)	Stat (9 / 9)	Gap (%)
1 000	0.4	100	1801.16	1 / 9	0.04	2426.63	0 / 9	78.31	2391.93	0 / 9	2.20	1176.83	5 / 9	1.15	0.53	0 / 9	2.71
		1 000	1801.03	0 / 9	0.28	2555.32	0 / 9	88.92	1851.78	0 / 9	2.98	1954.75	1 / 9	2.69	2.88	0 / 9	3.85
		10 000	1805.09	0 / 0	−−	2432.50	0 / 0	−−	1815.25	0 / 9	14.30	2057.89	1 / 9	3.12	14.99	0 / 9	15.23
	0.8	100	55.79	9 / 9	0.00	32.79	9 / 9	0.00	32.89	9 / 9	0.00	30.68	9 / 9	0.00	0.13	0 / 9	0.29
		1 000	679.42	9 / 9	0.00	555.36	8 / 8	0.00	551.2	8 / 9	0.12	33.22	9 / 9	0.00	0.52	0 / 9	1.46
		10 000	1806.40	0 / 0	−−	955.48	5 / 9	22.67	1015.35	5 / 9	0.36	236.93	8 / 9	0.02	3.06	0 / 9	1.50
5 000	0.4	100	1628.75	1 / 9	17.94	2531.39	1 / 9	78.02	1691.97	1 / 9	0.86	1459.17	1 / 9	0.86	41.28	0 / 9	1.44
		1 000	1801.88	0 / 9	43.25	2595.76	0 / 0	−−	1878.44	0 / 9	1.25	1875.73	0 / 9	1.25	52.32	0 / 9	1.49
		10 000	1808.89	0 / 0	−−	2627.38	0 / 0	−−	2218.95	0 / 9	12.43	2167.18	0 / 9	12.43	104.99	0 / 9	13.39
	0.8	100	60.58	9 / 9	0.00	35.07	9 / 9	0.00	37.65	9 / 9	0.00	34.13	9 / 9	0.00	3.46	0 / 9	0.30
		1 000	861.29	7 / 9	0.45	616.89	9 / 9	0.00	693.28	8 / 9	0.02	135.13	8 / 9	0.02	40.30	0 / 9	0.30
		10 000	1811.12	0 / 0	−−	933.81	8 / 8	0.00	970.19	8 / 9	0.04	362.13	8 / 9	0.03	43.86	0 / 9	0.32
10 000	0.4	100	1640.76	1 / 9	57.11	2442.29	1 / 1	0.00	1676.75	1 / 9	0.62	1685.41	1 / 9	0.62	42.98	0 / 9	0.96
		1 000	1805.02	0 / 0	−−	2351.42	0 / 0	−−	2192.85	0 / 9	0.96	2152.42	0 / 9	0.95	344.21	0 / 9	2.12
		10 000	1809.24	0 / 0	−−	2785.44	0 / 0	−−	2291.57	0 / 9	10.20	2274.11	0 / 9	12.83	466.29	0 / 9	15.46
	0.8	100	86.55	9 / 9	0.00	41.33	9 / 9	0.00	39.66	9 / 9	0.00	37.02	9 / 9	0.00	6.01	0 / 9	0.05
		1 000	1819.06	0 / 0	−−	664.42	9 / 9	0.00	641.69	9 / 9	0.00	228.38	9 / 9	0.00	20.38	0 / 9	0.06
		10 000	1894.52	0 / 0	−−	979.98	9 / 9	0.00	970.68	9 / 9	0.00	321.57	9 / 9	0.00	84.40	0 / 9	0.08

Table 3: Comparison of the results for robust knapsack problems with absolute robustness criterion.

using the lowest average CPU time. *SC-Ab3* also solves all the instances with few scenarios ($|S| = 100$) and a large budget. When the budget remains large and there are more scenarios ($|S| = 1000$), *SC-Ab3* solves all instances except one (with 5000 items) and when $|S| = 10000$ with large budget, there are only two that are not solved to optimality within the time limit (one with 1000 items and one with 5000 items). For restricted budget ($\theta = 0.4$), *SC-Ab3* optimally solves nine instances out of 81. For the remaining instances, *SC-Ab3* always produces solutions with non-zero objective value and achieves a gap of at most 12.83%.

Finally, although the scenario-relaxation-based heuristic (*Heur*) does not solve any instance to optimality, the CPU time is very low compared to the exact algorithms. *Heur* always outputs solutions with non-zero objective value that are close to optimal, with a maximum gap of 15.46%, obtained for $n = 10\,000$ when the budget is restricted. The differences between the heuristic and the exact algorithms (both with respect to CPU times as well as to the gap) highlight that the scenario-relaxation framework quickly produces a good non-trivial solution, but subsequently spends a considerable amount of time to find an optimal solution.

To summarize the comparison between the four exact algorithms and the heuristic reported in Table 3, we formulate the following advice. If the solution quality is the only criterion to be taken into account, the use of the scenario-relaxation algorithm with the third implementation (*SC-Ab3*) is advised. If both the computation time and the solution quality are relevant, however, we strongly recommend the use of the scenario-relaxation-based heuristic.

8.2.2 Min-max regret

For this objective function, we have considered the three implementations of the scenario-relaxation algorithm described in the previous section as well as the use of CPLEX for solving (M_2) and the heuristic based on the scenario-relaxation algorithm. The results are reported in Table 4 using the same notations as in Table 3 with the same definitions, except for *Gap*, which is still the average gap but now computed with respect to the upper bound produced by the considered algorithm and the best (highest) lower bound amongst the lower bounds produced by the four algorithms.

Table 4 confirms that also for the min-max regret objective, the instances with restricted budget ($\theta = 0.4$) are more difficult to solve than those with large budget ($\theta = 0.8$) and the difficulty increases with n and with the cardinality of S. CPLEX (see *Full*) produces non-trivial solutions only for 11 instances (out of 162), all with large budget for $n = 1\,000$. Six (respectively five) instances are solved optimally when $|S| = 100$ (respectively $|S| = 1\,000$).

Amongst the three implementations of the scenario-relaxation algorithm, similarly to the case of absolute robustness, *SC-Ab3* performs better than *SC-Ab1* and *SC-Ab2*. It provides a non-trivial solution to each instance and optimally solves 38 instances out of 162, which is more than any other exact algorithm. Also, *SC-Ab3* always produces the smallest gap, ranging from 0.06% to at most 14.47%, regardless of the value of n and the cardinality of S. The heuristic produces non-trivial solutions with good objective value to each instance within a low CPU time. Indeed, the average CPU time is less than ten minutes and the average gap is at most 26.06%. We conclude that the heuristic strikes a convenient tradeoff between the CPU time and the gap.

			Full			*SC-Ab1*			*SC-Ab2*			*SC-Ab3*			*Heur*		
n	θ	$\|S\|$	Time (*sec*)	Stat (9 / 9)	Gap (%)	Time (*sec*)	Stat (9 / 9)	Gap (%)	Time (*sec*)	Stat (9 / 9)	Gap (%)	Time (*sec*)	Stat (9 / 9)	Gap (%)	Time (*sec*)	Stat (9 / 9)	Gap (%)
1 000	0.4	1 000	1801.48	0 / 0	−−	2140.07	0 / 0	−−	1801.67	0 / 9	10.37	1861.43	2 / 9	8.70	1.67	0 / 9	11.73
		5 000	1816.78	0 / 0	−−	2191.30	0 / 0	−−	1819.48	0 / 9	12.49	1927.06	0 / 9	9.87	19.43	0 / 9	14.93
		10 000	1894.35	0 / 9	−−	2209.95	0 / 1	16.96	1900.90	0 / 9	13.81	1901.01	0 / 9	13.81	100.92	0 / 9	18.18
	0.8	1 000	468.30	6 / 8	12.40	552.01	7 / 8	2.10	400.26	8 / 9	2.17	311.37	8 / 9	0.06	0.54	0 / 9	5.14
		5 000	1297.00	5 / 5	0.00	1342.63	5 / 8	3.19	823.28	6 / 9	4.87	663.20	7 / 9	3.29	3.46	0 / 9	8.51
		10 000	1530.86	0 / 0	−−	1514.22	3 / 6	2.79	999.90	5 / 9	7.74	1124.31	5 / 9	6.11	18.64	0 / 9	10.61
5 000	0.4	1 000	1807.57	0 / 0	−−	2339.85	0 / 0	−−	1829.41	0 / 9	10.23	1774.60	1 / 9	5.32	28.73	0 / 9	12.03
		5 000	1888.82	0 / 0	−−	2351.82	0 / 0	−−	2377.30	0 / 9	10.20	2379.04	0 / 9	10.20	576.71	0 / 9	13.02
		10 000	1873.45	0 / 0	−−	2138.03	0 / 0	−−	2029.52	0 / 9	14.41	2024.18	0 / 9	14.41	218.27	0 / 9	19.38
	0.8	1 000	1807.56	0 / 0	−−	2649.67	0 / 1	19.23	1569.31	3 / 9	6.19	1624.97	3 / 9	5.15	7.41	0 / 9	10.66
		5 000	1874.81	0 / 0	−−	2062.17	1 / 1	0.00	1856.48	0 / 9	12.33	1970.05	1 / 9	9.95	55.47	0 / 9	13.26
		10 000	1881.00	0 / 0	−−	2246.93	0 / 1	8.08	2125.12	0 / 9	14.41	2127.75	0 / 9	12.78	321.10	0 / 9	16.14
10 000	0.4	1 000	1820.94	0 / 0	−−	2333.83	0 / 0	−−	1901.51	0 / 9	10.39	1902.60	0 / 9	8.99	101.74	0 / 9	13.90
		5 000	1839.22	0 / 0	−−	2289.14	0 / 0	−−	2044.93	0 / 9	16.81	2043.69	0 / 9	10.90	233.70	0 / 9	18.23
		10 000	1803.07	0 / 0	−−	2368.23	0 / 0	−−	2001.87	0 / 9	24.74	2001.61	0 / 9	14.47	192.25	0 / 9	26.06
	0.8	1 000	1818.80	0 / 0	−−	1994.52	0 / 1	19.83	1764.33	2 / 9	7.89	1538.58	5 / 9	6.01	21.79	0 / 9	9.87
		5 000	1869.63	0 / 0	−−	2169.37	1 / 1	0.00	1994.10	0 / 9	9.76	1906.76	3 / 9	8.23	192.15	0 / 9	10.79
		10 000	1885.81	0 / 0	−−	1939.78	3 / 3	0.00	2216.52	0 / 9	11.48	2214.82	3 / 9	10.84	413.89	0 / 9	13.82

Table 4: Comparison of the results for robust knapsack problems with min-max regret criterion.

			Full			*SC-Ab1*			*SC-Ab2*			*SC-Ab3*			*Heur*		
n	θ	$\|S\|$	Time (*sec*)	Stat (9 / 9)	Gap (%)	Time (*sec*)	Stat (9 / 9)	Gap (%)	Time (*sec*)	Stat (9 / 9)	Gap (%)	Time (*sec*)	Stat (9 / 9)	Gap (%)	Time (*sec*)	Stat (9 / 9)	Gap (%)
1 000	0.4	1 000	1776.01	0 / 0	−−	2276.42	0 / 0	−−	1879.97	0 / 9	18.83	1901.84	1 / 9	12.24	454.11	0 / 9	22.87
		5 000	1818.09	0 / 0	−−	2165.63	0 / 1	25.21	2407.19	0 / 9	19.34	2440.59	0 / 9	15.11	598.77	0 / 9	23.56
		10 000	1927.11	0 / 0	−−	2277.62	0 / 1	36.33	1805.77	0 / 9	25.36	1805.98	0 / 9	19.41	602.35	0 / 9	28.74
	0.8	1 000	1247.29	3 / 7	20.10	1369.00	3 / 9	39.26	474.62	6 / 9	9.72	502.95	7 / 9	5.76	98.24	0 / 9	14.29
		5 000	1259.69	1 / 2	34.08	1940.98	1 / 7	31.06	1642.58	2 / 9	11.97	1730.10	3 / 9	9.12	106.58	0 / 9	15.27
		10 000	1909.31	0 / 0	−−	2181.10	1 / 5	39.26	1807.47	1 / 9	15.23	1806.04	2 / 9	10.22	390.76	0 / 9	17.69
5 000	0.4	1 000	1808.90	0 / 0	−−	2201.07	0 / 0	−−	1847.25	0 / 9	15.82	1844.62	0 / 9	8.26	143.89	0 / 9	17.53
		5 000	1888.74	0 / 0	−−	2197.40	0 / 0	−−	1812.49	0 / 9	21.33	1812.60	0 / 9	12.98	364.31	0 / 9	24.04
		10 000	1878.37	0 / 0	−−	2164.55	0 / 0	−−	1804.27	0 / 9	28.13	1806.62	0 / 9	16.14	475.15	0 / 9	29.17
	0.8	1 000	1807.87	0 / 0	−−	2335.92	0 / 1	34.18	1824.36	1 / 9	12.94	1934.44	2 / 9	9.53	123.18	0 / 9	16.54
		5 000	1890.77	0 / 0	−−	2092.93	0 / 0	−−	1853.21	0 / 9	19.41	1852.76	1 / 9	13.82	365.72	0 / 9	21.66
		10 000	1841.76	0 / 0	−−	2015.94	0 / 0	−−	2025.45	0 / 9	27.10	2022.83	0 / 9	14.39	564.28	0 / 9	31.24
10 000	0.4	1 000	1824.87	0 / 0	−−	2384.91	0 / 0	−−	1954.92	0 / 9	18.71	1950.58	0 / 9	10.82	147.22	0 / 9	21.69
		5 000	1819.31	0 / 0	−−	2286.10	0 / 0	−−	1915.66	0 / 9	21.33	1913.71	0 / 9	17.88	326.47	0 / 9	23.96
		10 000	1803.27	0 / 0	−−	2199.05	0 / 0	−−	1819.22	0 / 8	27.31	1818.73	0 / 8	20.08	694.52	0 / 9	35.67
	0.8	1 000	1818.50	0 / 0	−−	2209.87	0 / 0	−−	1896.42	1 / 9	12.85	1911.94	2 / 9	10.92	195.81	0 / 9	18.59
		5 000	1807.16	0 / 0	−−	2260.49	0 / 0	−−	2007.45	0 / 9	14.82	2006.14	1 / 9	12.76	519.23	0 / 9	20.56
		10 000	1885.79	0 / 0	−−	2224.26	0 / 0	−−	1950.89	0 / 9	20.31	1944.64	1 / 9	15.87	741.98	0 / 9	23.67

Table 5: Comparison of the results for robust knapsack problems with min-max relative regret criterion.

8.2.3 Min-max relative regret

Using the definitions and notations of the previous section, we report in Table 5 the results of the three implementations of the scenario-relaxation algorithm as well as the use of CPLEX for solving (M_3) and the scenario-relaxation-based heuristic for solving the generated instances with the min-max relative regret objective function.

Table 5 confirms the observations made for Table 4 regarding the evolution of the difficulty of the problem. CPLEX is able to solve optimally only four instances out of 162, and provides non-trivial solutions to only nine instances. The third implementation of the scenario-relaxation algorithm (*SC-Ab3*) displays the best performance also for this objective. It provides a non-trivial solution to each instance and optimally solves 20 instances out of 162, with a largest average gap of 20.08%. The heuristic based on scenario relaxation outputs non-trivial solutions with good objective value to each instance after at most 12 minutes, with a largest average gap of 35.67%.

9. Summary and conclusions

In this paper, we have studied the robust knapsack problem with three different criteria, namely the absolute robustness criterion, the min-max regret criterion and the min-max relative regret criterion. We consider the general case where uncertainty affects both the profit and the weight of the items and is represented either by a discrete set of scenarios or by interval scenarios. We have studied the complexity and the approximability of the problem and some of its subproblems, for each of the above objective functions when the set of scenarios is discrete. We have described a scenario-relaxation algorithm for solving the problem with a discrete set of scenarios and show that the solution to the problem with interval scenarios can be obtained by solving the problem with discrete scenario set where each item can take only one of the two values representing the bounds of its interval. We have also converted the scenario-relaxation algorithm into a heuristic for producing good solutions within reasonable running times.

Our experimental results demonstrate the efficiency of the scenario-relaxation algorithm. In our best implementation, the LP relaxation of the restricted problem is first solved in an attempt to identify new scenarios to be added. When no scenario can be identified, all the variables equal to one in the current LP solution are fixed and the corresponding constrained MIP problem is run. Only when this constrained MIP does not lead to new scenarios, the full MIP with all scenarios added up to that point, is computed. The full benefit of the scenario-relaxation algorithm is observed when the set of scenarios is very large. The scenario-relaxation-based heuristic presents a convenient tradeoff between computation time and solution quality. We have reported results for instances with up to 10 000 items and 10 000 scenarios, using both CPLEX, the scenario-relaxation algorithm and the heuristic. Globally, we observe that the difficulty of the instances increases with the number of items, with the number of scenarios and with the tightness of the budget. We have also found the necessary running times to be dependent on the particular objective function studied: the absolute robustness criterion is usually easier than the min-max regret criterion with the same parameters, which in turn is easier than the min-max relative regret.

Acknowledgements

The authors thank Frits Spieksma (KULeuven) for useful comments.

References

[1] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems.* Springer, 2004.

[2] S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations.* J. Wiley, 1990.

[3] S. Eilon. Application of the knapsack model for budgeting. *OMEGA the International Journal of Management Science*, 15(6):489–494, 1987.

[4] P. Kouvelis and G. Yu. *Robust Discrete Optimization and its Applications.* Kluwer Academic Publishers, Norwell, MA, 1997.

[5] A. Ben-Tal and A. Nemirovski. Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical Programming, Series A*, 88:411–424, 2000.

[6] H. Lida. A note on the max-min 0-1 knapsack problem. *Journal of Combinatorial Optimization*, 3:99–94, 1999.

[7] A.J. Kleywegt and J.D. Papastavrou. The dynamic and stochastic knapsack problem. *Operations Research*, 46:17–35, 1998.

[8] G.Y. Lin, Y. Lu, and D.D. Yao. The stochastic knapsack revisited: Switch-over policies and dynamic pricing. *Operations Research*, 56:945–957, 2008.

[9] I. Averbakh. On the complexity of a class of combinatorial optimization problems with uncertainty. *Mathematical Programming, Series A*, 90:263–272, 2001.

[10] D. Bertsimas and M. Sim. Robust discrete optimization and network flows. *Mathematical Programming, Series B*, 98:49–71, 2003.

[11] D. Bertsimas and M. Sim. The price of robustness. *Operations Research*, 52:35–53, 2004.

[12] A. Sbihi. A cooperative local search-based algorithm for the multiple-scenario max-min knapsack problem. *European Journal of Operational Research*, 202:339–346, 2010.

[13] F. Taniguchi, T. Yamada, and S. Kataoka. Heuristic and exact algorithms for the max-min optimization of the multi-scenario knapsack problem. *Computers & Operations Research*, 35:2034–2048, 2008.

[14] O. Klopfenstein and D. Nace. A robust approach to the chance-constrained knapsack problem. *Operations Research Letters*, 36:628–632, 2008.

[15] T. Assavapokee, M.J. Realff, and J.C. Ammons. A new min-max regret robust optimization approach for interval data uncertainty. *Journal of Optimization Theory and Applications*, 137:297–316, 2008.

[16] H. Aissi, C. Bazgan, and D. Vanderpooten. Min-max and min-max regret versions of combinatorial optimization problems: A survey. *European Journal of Operational Research*, 197:427–438, 2009.

[17] E. Conde. On the complexity of the continuous unbounded knapsack problem with uncertain coefficients. *Operations Research Letters*, 33:481–485, 2005.

[18] H. Aissi, C. Bazgan, and D. Vanderpooten. Approximation of min-max and min-max regret versions of some combinatorial optimization problems. *European Journal of Operational Research*, 179:281–290, 2007.

[19] G. Yu. On the max-min 0-1 knapsack problem with robust optimization applications. *Operations Research*, 44:407–415, 1996.

[20] M. Kress, M. Penn, and M. Polukarov. The minmax multidimensional knapsack problem with application to a chance-constrained problem. *Naval Research Logistics*, 54:656–666, 2007.

[21] R. Kalai and D. Vanderpooten. Lexicographic α-robust knapsack problem : Complexity results. *International Transactions in Operational Research*, 18:103–113, 2011.

[22] V.G. Deineko and G.J. Woeginger. Pinpointing the complexity of the interval minmax regret knapsack problem. *Discrete Optimization*, 7:191–196, 2010.

[23] S. Arora and B. Barak. *Computational Complexity: A Modern Approach.* Cambridge University Press, 2009.

[24] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman and Co., 1979.

[25] G. Chen, H. Hwang, and T. Tsai. Efficient maxima-finding algorithms for random planar samples. *Discrete Mathematics and Theoretical Computer Science*, 6:107–122, 2003.

[26] T. Erlebach, H. Kellerer, and U. Pferschy. Approximating multiobjective knapsack problems. *Management Science*, 12:1603–1612, 2002.

[27] T. Assavapokee, M.J. Realff, J.C. Ammons, and I.-H. Hong. Scenario relaxation algorithm for finite scenario-based minmax regret and minmax relative regret robust optimization. *Computers & Operations Research*, 35:2093–2102, 2008.

[28] I. Averbakh. Computing and minimizing the relative regret in combinatorial optimization with interval data. *Discrete Optimization*, 2:273–287, 2005.