J Optim Theory Appl
DOI 10.1007/s10957-013-0319-3

Complexity Results and Exact Algorithms for Robust Knapsack Problems

Fabrice Talla Nobibon · Roel Leus

Received: 22 July 2011 / Accepted: 24 April 2013

Abstract This paper studies the robust knapsack problem, for which solutions are, up to a certain point, immune from 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-hardness · 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

F. Talla Nobibon
PostDoc researcher for Research Foundation—Flanders, Brussels, Belgium

F. Talla Nobibon
QuantOM, HEC-Management School, University of Liège, Liège, Belgium

F. Talla Nobibon (✉) · R. Leus
ORSTAT, Faculty of Business and Economics, KU Leuven, Leuven, Belgium
e-mail: Fabrice.TallaNobibon@kuleuven.be

Published online: 11 May 2013

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 from 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 next section, we will provide formal definitions for these three types of objective functions.

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 Sect. 2. In Sect. 3, we survey the existing literature. Section 4 looks into the knapsack problem with absolute robustness criterion, Sect. 5 is devoted to the study of the min–max regret criterion and, in Sect. 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 experiments in Sect. 8 and we conclude in Sect. 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:

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

Let F_s^* be the optimal objective value of KP_s. Given a 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 $\sum_{i=1}^n a_i^s x_i > b$ then we adopt the convention $F_s(X) := -\infty$. Given $X \in \{0,1\}^n$, we define the *maximum regret* $Z(X) := \max_{s\in S}\{F_s^* - F_s(X)\}$. If there exists $s \in S$ with $\sum_{i=1}^n a_i^s x_i > b$ then $Z(X) = +\infty$.

Let $K := \{X \in \{0,1\}^n : \sum_{i=1}^n a_i^s x_i \le b,\ \forall s \in S\}$ be the set of feasible solutions to all scenarios. 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 $\min_{X\in K} Z(X) := \min_{X\in K} \max_{s\in S}\{F_s^* - F_s(X)\}$; and
(3) minimization of the *maximum relative regret* (or *min–max relative regret*), which is defined by: $\min_{X\in K} \max_{s\in S}\{\frac{F_s^* - F_s(X)}{F_s^*}\}$.

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. The theoretical results of this paper are stated without proof; the proofs can be found in the online[1] available working paper [19].

3 Literature Review

The deterministic knapsack problem is a well studied problem; we refer to [1, 2, 20, 21] 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 [22] 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

[1] See http://www.econ.kuleuven.be/public/ndbac96/robustKP.htm.

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 to 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. [23] consider a robust multi-dimensional knapsack problem where the goal is to select a subset of scenarios for which the sum of the weights (of the scenarios) exceeds a given threshold, and that minimizes the sum of the componentwise maxima of the vectors representing the scenarios. They show that the problem is NP-hard and develop a practically efficient algorithm for solving it.

Kalai and Vanderpooten [24] 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.

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:

$$\begin{aligned}
\text{(AbKP)} \qquad & \max_{X} \min_{s\in S} \sum_{i=1}^{n} v_i^s x_i \\
\text{s.t.} \quad & \sum_{i=1}^{n} a_i^s x_i \le b \qquad \forall s \in S, \\
& x_i \in \{0, 1\} \qquad i = 1, \dots, n,
\end{aligned}$$

with S a discrete set of distinct scenarios. The results below reduce the set S to be considered. Given two scenarios $s, u \in S$, we say that scenario s *dominates* scenario u iff both of the following sets of inequalities are satisfied: (i) $a_i^s \geq a_i^u, i = 1, \ldots, n$; and (ii) $v_i^s \leq v_i^u, i = 1, \ldots, n$. 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 4.1 *An optimal solution to AbKP can be obtained by solving a reduced problem in which the set of scenarios S is replaced by $\bar{S}$.*

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

First, we consider the setting where 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 not difficult to see that $\bar{s}$ is the only maximal scenario of Π.

Lemma 4.2 *If $\bar{s} \in S$ then it is the only maximal scenario of S.*

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

Lemma 4.3 *If $\bar{s} \in S$ then AbKP can be solved in pseudo-polynomial time.*

Second, we assume that 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 [22] 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.

Third, we assume that uncertainty affects only the weights of items. In this case, $V^s = V$ for each scenario $s \in S$ and we denote by AbKP_W the corresponding variant of the problem AbKP. If the size of the scenario set S is bounded by a constant then 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 problem is solved in pseudo-polynomial time. We obtain the following approximation result.

Proposition 4.4 *When the set S of scenarios is bounded, the problem AbKP_W has a PTAS but does not have an FPTAS, unless $P = NP$.*

When S is unbounded, on the other hand, AbKP_W is strongly NP-hard.

Theorem 4.5 *AbKP_W is strongly NP-hard for an unbounded scenario set S.*

Finally, 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 $T^s := (t_1^s, \dots, t_n^s, t_{n+1}^s, \dots, t_{2n}^s)$ 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 Chen et al. [25], for instance. The next result states that the problem can be solved in pseudo-polynomial time when S is bounded.

Lemma 4.6 *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 value function

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

i.e., $F_k(\alpha_1, \dots, \alpha_{|S|}; b_1, \dots, b_{|S|})$ 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 items $1, 2, \dots, k-1$. The initial condition is: $F_n(\alpha_1, \dots, \alpha_{|S|}; b_1, \dots, b_{|S|}) := \min\{\alpha_1 + v_n^1, \dots, \alpha_{|S|} + v_n^{|S|}\}$, if $b_s \ge a_n^s, \forall\, s \in S$; and $F_n(\alpha_1, \dots, \alpha_{|S|}; b_1, \dots, b_{|S|}) := \min\{\alpha_1, \dots, \alpha_{|S|}\}$, otherwise. We have the recursive relation

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

for $k = n-1, n-2, \dots, 2$. The optimal objective value is $F_1(0, \dots, 0; b, \dots, b)$. The time complexity of the DP algorithm is $O(nb^{|S|}L^{|S|})$, 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.

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 when the profit and the weight of the items are small, and there are few scenarios. Because our experiments in Sect. 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.

Proposition 4.7 *If S is bounded then AbKP has a PTAS but not an FPTAS.*

We next study the case where $|S|$ is unbounded. In this case, the problem is strongly NP-hard because it contains the special cases studied above. 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:

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

As mentioned above, we may assume that $S := \bar{S}$, where $\bar{S}$ is the set of maximal scenarios. This assumption, however, is not necessary for the application of Algorithm 1 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, 26] and propose a *scenario-relaxation algorithm* for solving AbKP. We distinguish two (not necessarily disjoint) subsets of S: $S^1 \subseteq S$ is the set of scenarios required to ensure that a given solution is feasible for all possible scenarios, and $S^2 \subseteq S$ contains the scenarios required to establish that a given feasible solution is optimal.

Given two scenarios s and u in S, u is *weight-dominated* by s iff $a_i^u \le a_i^s$ for $i = 1, \dots, n$; and u is *value-dominated* by s iff $v_i^u \ge v_i^s$ for $i = 1, \dots, n$. Using these notions, 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 4.8 *The set $\bar{S}$ of maximal scenarios is the union of S^1 and S^2; that is, $\bar{S} = S^1 \cup S^2$.*

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** stop
6:  **else**
7:      $W_1 := \{s \in S^1 \setminus \Omega \mid \sum_{i=1}^n a_i^s x_i^\Omega > b\}$
8:      **if** $W_1 \neq \emptyset$ **then**
9:          Select a non-empty subset $W_1' \subseteq W_1$ and update $\Omega \leftarrow \Omega \cup W_1'$; goto 2
10:     **else**
11:         For all $s \in S^2 \setminus \Omega$, compute $\delta^s := \sum_{i=1}^n v_i^s x_i^\Omega$; $W_2 := \{s \in S^2 \setminus \Omega \mid \delta^s < y^\star\}$
12:         **if** $W_2 = \emptyset$ **then** $LB := y^\star$, stop
13:         **else**
14:             $\delta := \min_{s \in W_2} \delta^s$, $LB := \max\{LB, \delta\}$
15:             **if** $|UB - LB| < \varepsilon$ **then** stop
16:             **else**
17:                 Select a non-empty subset $W_2' \subseteq W_2$ and update $\Omega \leftarrow \Omega \cup W_2'$; goto 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 with 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 4.1 and Lemma 4.8.

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 is given by:

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

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.

First, we assume that 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 unlikely 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].

Second, we assume that uncertainty affects only the weights of items. In this case, $V^s := V$ for each scenario $s \in S$. Apart from a constant in the objective, the problem RgKP can be seen to reduce to problem AbKP_W studied in Sect. 4.

Finally, we 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 solved in pseudo-polynomial time, because 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 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:

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

A scenario-relaxation algorithm for solving (M_2) follows the same structure as Algorithm 1. The main modifications needed are the following: at the start of the algorithm (line 1), we additionally compute F_s^* by solving KP_s for every scenario; in line 4, each solution to the relaxation now yields a lower bound (meaning that *UB* in line 4 becomes *LB*), whereas in line 12 each feasible solution evaluated against all scenarios produces an upper bound; that is, *LB* in line 12 and line 14 becomes *UB*. Finally, in line 11 the quantity δ^s is computed as:

$$\delta^s := F_s^* - \sum_{i=1}^{n} v_i^s x_i^{\Omega}$$

and δ in line 14 is given by:

$$\delta := \max_{s \in W_2} \delta^s,$$

whereas $UB := \min\{UB, \delta\}$ in that same line.

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, which is defined by:

$$(\mathrm{ReKP}_0) \qquad \min_X \max_{s\in S} \frac{F_s^* - \sum_{i=1}^n v_i^s x_i}{F_s^*}$$

$$\text{s.t.} \quad \sum_{i=1}^n a_i^s x_i \le b \qquad \forall s \in S,$$

$$x_i \in \{0,1\} \qquad i = 1,\dots,n.$$

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.

$$(\mathrm{ReKP}) \qquad \max_X \min_s \frac{1}{F_s^*}\sum_{i=1}^n v_i^s x_i$$

$$\text{s.t.} \quad \sum_{i=1}^n a_i^s x_i \le b \qquad \forall s \in S,$$

$$x_i \in \{0,1\} \qquad i = 1,\dots,n.$$

First, we assume that 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:

$$\max_X \min_{s\in S} \sum_{i=1}^n \delta_s v_i^s x_i$$

$$\text{s.t.} \quad \sum_{i=1}^n a_i x_i \le b$$

$$x_i \in \{0,1\} \qquad i = 1,\dots,n.$$

The latter problem is a robust knapsack problem with absolute robustness criterion as presented in Sect. 4, and can be solved in pseudo-polynomial time. We have the following results.

Theorem 6.1 *The problem ReKP$_0$ has no approximation algorithm, not even with only two scenarios, unless $P = NP$.*

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

Next, we assume that uncertainty affects only the weights of items. In this case we have $V^s := V$ for each $s \in S$, and the robust knapsack problem with min–max relative regret criterion reduces to the problem AbKP$_-$W studied in Sect. 4. Finally, we consider the min–max relative regret knapsack problem with uncertainty both on the weights and on the profits. If $|S|$ is bounded, the problem is solvable in pseudo-polynomial time because 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:

$$
\begin{aligned}
(\mathrm{M}_3)\qquad & \min\ y \\
& \text{s.t.}\quad F_s^* - \sum_{i=1}^{n} v_i^s x_i \le F_s^* y \qquad \forall s \in S, \\
& \qquad \sum_{i=1}^{n} a_i^s x_i \le b \qquad \forall s \in S, \\
& \qquad x_i \in \{0, 1\} \quad i = 1, \ldots, n, \\
& \qquad y \ge 0.
\end{aligned}
$$

The scenario-relaxation algorithm previously described for absolute robustness and min–max regret can be modified to also solve this problem. The main adaptations that have to be embedded in Algorithm 1 are those already listed for the min–max regret criterion. In addition, after computing F_s^* for all the scenarios, if there exists a scenario s with $F_s^* = 0$ then the algorithm stops and declares the problem ‘ill-posed’ for this objective function. Tables 1 and 2 summarize the complexity and the approximability results obtained for the robust knapsack problem with discrete scenario set.

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 Sect. 4 lead us to conclude that the absolute robust knapsack problem with interval scenarios is equivalent to the deterministic knapsack problem.

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

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

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

For the min–max regret and relative regret criteria, on the other hand, we consider the feasible set given by: $K := \{X \in \{0,1\}^n : \sum_{i=1}^n a_i x_i \leq b,\ \forall a_i \in [a_i^L, a_i^U] \text{ for } i = 1, \dots, n\}$. 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'.*

Results similar to Lemma 7.1 are proved by Averbakh [27] for subset-type combinatorial optimization, when there is uncertainty only in the objective function. The above result implies that the scenario-relaxation algorithm derived earlier can be used to solve the problem with interval scenarios. This, however, may not the best approach for solving interval-scenario problems: dedicated algorithms might obtain better results. It may also be noted that Lemma 7.1 does not imply straightforward complexity results for the interval-scenario case, because the scenario set in the latter case 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 data sets and subsequently, we discuss the computational results. Computation time is referred to as *Time* and is expressed in seconds.

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.[2] The problem instances are generated as follows: for each item $i = 1, \dots, n$, 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 $[(1-\sigma_a)a_i^0, (1+\sigma_a)a_i^0]$ (uniformly distributed), where $\sigma_a \in \{0.3, 0.6, 0.9\}$ is a parameter that determines the variability level of the weights in the different scenarios. For the value v_i^s, we apply the same process with interval $[(1-\sigma_v)v_i^0, (1+\sigma_v)v_i^0]$ 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 $b := \lfloor \theta \min_{s \in S} \sum_{1 \le i \le n} a_i^s \rfloor$ with $\theta \in \{0.4, 0.8\}$, where $\lfloor \cdot \rfloor$ is the floor. The parameter θ indicates whether the capacity b is tight or rather loose. For each value of n, θ, and $|S|$, we generate nine instances, each corresponding with a unique combination of σ_a and σ_v. In total, we have $3 \times 2 \times 3 \times 3 \times 3 = 162$ instances. Throughout this section, we provide results only for instances with discrete scenario set. We first consider the application of the scenario-relaxation algorithm for solving

[2]The website http://www.econ.kuleuven.be/public/ndbac96/robustKP.htm contains the instances for $n = 1\,000$; the remaining ones can be obtained from the authors.

the generated instances with the absolute robustness criterion. Next, we focus on the min–max regret criterion and subsequently, we very briefly comment the problem with the min–max relative regret criterion. For every objective function, we have also implemented a heuristic based on the scenario-relaxation algorithm.

For the absolute robustness objective, we have considered various implementations of the scenario-relaxation algorithm; the best results are obtained by following the main steps of Algorithm 1, but 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 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, 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. If the obtained 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.

The scenario-relaxation algorithm is converted into a heuristic, denoted *Heur*, as follows. We 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 one 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. We compare all these algorithms to CPLEX used as a MIP solver, applied to formulation (M_1). Furthermore, we evaluate the effect of the choices for σ_a, σ_v, and θ, on the performance of our algorithms.

In Table 3, in addition to the results of the best implementation of the scenario-relaxation algorithm 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 row *Stat* contains a pair of numbers between 0 and 9; the first (respectively the second) value represents 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 to four instances out of nine; by 'non-trivial solution,' we mean a feasible solution that selects at least one item. The row *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 to at least one instance out of nine associated with a given cell, we write $--$ in the cell corresponding with *Gap*, to express the fact that the gap is not computed. The last column (*aver.*) reports the average values for *Gap* and for *Time*, and the total values for *Stat*. Before solving the MIP formulation (M_1) with CPLEX (*Full*), the result of Proposition 4.1 is used in an attempt to

Table 3 Comparison of the results for the robust knapsack problem with absolute robustness criterion

		$\theta = 0.4$			$\theta = 0.8$			Aver.
	$\lvert S\rvert =$	100	1 000	10 000	100	1 000	10 000	
$n = 1\,000$								
Full	Time (*sec*)	1801.16	1801.03	1805.09	55.79	679.42	1806.40	1324.82
	Stat (9 / 9)	1 / 9	0 / 9	0 / 0	9 / 9	9 / 9	0 / 0	19 / 36
	Gap (%)	0.04	0.28	−−	0.00	0.00	−−	0.08
SR	Time (*sec*)	1176.83	1954.75	2057.89	30.68	33.22	236.93	915.05
	Stat (9 / 9)	5 / 9	1 / 9	1 / 9	9 / 9	9 / 9	8 / 9	33 / 54
	Gap (%)	1.15	2.69	3.12	0.00	0.00	0.02	1.16
Heur	Time (*sec*)	0.53	2.88	14.99	0.13	0.52	3.06	3.69
	Stat (9 / 9)	0 / 9	0 / 9	0 / 9	0 / 9	0 / 9	0 / 9	0 / 54
	Gap (%)	2.71	3.85	15.23	0.29	1.46	1.50	4.17
$n = 5\,000$								
Full	Time (*sec*)	1628.75	1801.88	1808.89	60.58	861.29	1811.12	1328.75
	Stat (9 / 9)	1 / 9	0 / 9	0 / 0	9 / 9	7 / 9	0 / 0	17 / 36
	Gap (%)	17.94	43.25	−−	0.00	0.45	−−	15.41
SR	Time (*sec*)	1459.17	1875.73	2167.18	34.13	135.13	362.13	1005.58
	Stat (9 / 9)	1 / 9	0 / 9	0 / 9	9 / 9	8 / 9	8 / 9	26 / 54
	Gap (%)	0.86	1.25	12.43	0.00	0.02	0.03	2.43
Heur	Time (*sec*)	41.28	52.32	104.99	3.46	40.30	43.86	47.70
	Stat (9 / 9)	0 / 9	0 / 9	0 / 9	0 / 9	0 / 9	0 / 9	0 / 54
	Gap (%)	1.44	1.49	13.39	0.30	0.30	0.32	2.87
$n = 10\,000$								
Full	Time (*sec*)	1640.76	1805.02	1809.24	86.55	1819.06	1894.52	1509.19
	Stat (9 / 9)	1 / 9	0 / 0	0 / 0	9 / 9	0 / 0	0 / 0	10 / 18
	Gap (%)	57.11	−−	−−	0.00	−−	−−	28.56
SR	Time (*sec*)	1685.41	2152.42	2274.11	37.02	228.38	321.57	1116.48
	Stat (9 / 9)	1 / 9	0 / 9	0 / 9	9 / 9	9 / 9	9 / 9	28 / 54
	Gap (%)	0.62	0.95	12.83	0.00	0.00	0.00	2.40
Heur	Time (*sec*)	42.98	344.21	466.29	6.01	20.38	84.40	160.71
	Stat (9 / 9)	0 / 9	0 / 9	0 / 9	0 / 9	0 / 9	0 / 9	0 / 54
	Gap (%)	0.96	2.12	15.46	0.05	0.06	0.08	3.12

reduce the number of scenarios to consider. For the instances that we have generated, however, no scenario can be removed based on this result.

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. When looking at the results produced by the scenario-relaxation algorithm (*SR*), we find that for each group of instances, it optimally solves at least as many instances as the full model, using less average CPU time. Finally, although the

Table 4 Behavior of our algorithms as function of σ_a and σ_v, for $n = 1\,000$ and $|S| = 1\,000$

		$\sigma_v = 0.3$			$\sigma_v = 0.6$			$\sigma_v = 0.9$		
	$\sigma_a =$	0.3	0.6	0.9	0.3	0.6	0.9	0.3	0.6	0.9
Full	Time	720.11	944.21	1520.68	911.71	1016.69	1800.41	1040.00	1390.11	1800.30
	Stat	4 / 6	3 / 6	1 / 6	3 / 6	3 / 6	0 / 6	3 / 6	2 / 6	0 / 6
	Gap	0.21	0.80	1.45	0.73	1.41	2.30	1.67	2.11	2.98
SR	Time	94.15	1074.26	1286.18	414.87	917.36	1025.67	906.76	919.28	1192.68
	Stat	5 / 6	4 / 6	2 / 6	4 / 6	3 / 6	1 / 6	4 / 6	3 / 6	0 / 6
	Gap	0.01	0.52	0.99	0.37	1.26	1.60	1.45	1.87	2.12
Heur	Time	0.55	1.09	0.78	1.62	1.58	1.86	1.80	1.88	2.44
	Stat	0 / 6	0 / 6	0 / 6	0 / 6	0 / 6	0 / 6	0 / 6	0 / 6	0 / 6
	Gap	0.59	0.95	1.23	1.26	1.48	1.90	1.85	2.09	2.85

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 average gap of 15.46 %, obtained for $n = 10\,000$ when the budget is restricted. The fact that *Heur* does not achieve the optimal solution value may be due to the stopping criterion: the algorithm is halted after the first time that the LP relaxation is solved to optimality. This stopping rule aims to reduce the time needed to arrive at a good solution.

To evaluate the effect of σ_a and σ_v on the performance of our algorithms, we have generated, for each combination of σ_a and σ_v, three instances with $n = 1\,000$ and $|S| = 1\,000$ (In Table 3, there was only one such instance), and we combine this with each of the two values of the budget ($\theta = 0.4$ and $\theta = 0.8$). The results are reported in Table 4; each cell represents the analysis of six instances. The main observation is that when σ_a and σ_v increase, the instances become more difficult: the gap increases and the number of instances solved to optimality within the time limit decreases, when σ_a or σ_v increases.

For a closer study of the sensitivity of our algorithms to correlation and the budget, we have generated additional instances with $n = 1\,000$ and $|S| = 1\,000$ following the description of Martello et al. [20]. We use the range $R = 100$ and for each value of $\theta = 0.2$, 0.4, 0.5, 0.6, and 0.8, we generate three instances, where the value and the weight of each item is an integer uniformly chosen in the interval $[1, R]$. These 15 instances constitute the set of uncorrelated instances. With each such instance, we associate one *weakly* and one *strongly* correlated instance, derived as follows. Given an uncorrelated instance, we keep the weights of items as well as the budget. For the associated weakly correlated instance, for each item i and scenario s, we randomly generate the value v_i^s from the interval $[a_i^s - R/10, a_i^s + R/10]$, whereas for the strongly correlated instance, we set $v_i^s = a_i^s + R/10$. In total, we have 15 uncorrelated instances, 15 weakly correlated instances, and 15 strongly correlated instances. Table 5 summarizes the results of the application of our algorithms to these instances. We find that hard instances with respect to θ are obtained when θ is close to 0.5. Indeed, within the time limit, the gap increases with θ up to $\theta = 0.5$, after

Table 5 Behavior of the algorithms when θ varies and when dealing with *weakly* and *strongly* correlated instances, for $n = 1\,000$ and $|S| = 1\,000$

		θ					Uncorr.	Weakly	Strongly
		0.2	0.4	0.5	0.6	0.8			
Full	Time	1801.23	1800.64	1846.93	1800.48	1800.38	1809.93	1800.77	1800.65
	Stat	0 / 3	0 / 2	0 / 3	0 / 3	0 / 3	0 / 14	0 / 15	0 / 15
	Gap	39.40	9.66	50.17	26.32	13.41	29.08	31.70	35.83
SR	Time	1831.05	1807.34	1843.21	1808.68	2304.44	1918.94	1905.65	1893.16
	Stat	0 / 3	0 / 3	0 / 3	0 / 3	0 / 3	0 / 15	0 / 15	0 / 15
	Gap	23.33	27.88	31.86	20.72	8.69	22.50	24.49	27.98
Heur	Time	42.87	56.99	85.02	50.45	30.58	53.18	83.09	96.83
	Stat	0 / 3	0 / 3	0 / 3	0 / 3	0 / 3	0 / 15	0 / 15	0 / 15
	Gap	28.75	35.88	43.81	29.13	17.94	31.10	33.66	38.39

which it starts to decrease; this phenomenon is observed for the three algorithms. Also, strongly correlated instances are more difficult to solve than weakly correlated ones: the gap is higher. Similarly, weakly correlated instances are at least as difficult to solve as uncorrelated instances. These observations are in line with experiments for the classic knapsack problem [21].

The results for the min–max regret objective function are reported in Table 6 using the same notation 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 among the lower bounds produced by the four algorithms. *Full* refers to the use of CPLEX for solving (M_2). Table 6 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. Scenario relaxation 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, *SR* 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 trade-off between the CPU time and the gap.

For the min–max relative regret objective, we have obtained results similar to those described above for the min–max regret objective function. We refer the reader to our working paper [19] for more details.

9 Conclusions

Our experiments demonstrate the efficiency of the scenario-relaxation algorithm on the case of a discrete set of scenarios. In our best implementation, the LP relaxation

Table 6 Comparison of the results for the robust knapsack problem with min–max regret criterion

		$\theta = 0.4$			$\theta = 0.8$			Aver.
	$\lvert S\rvert =$	100	1 000	10 000	100	1 000	10 000	
$n = 1\,000$								
Full	Time	1801.48	1816.78	1894.35	468.30	1297.00	1530.86	1468.13
	Stat	0 / 0	0 / 0	0 / 0	6 / 8	5 / 5	0 / 0	11 / 13
	Gap	−−	−−	−−	12.40	0.00	−−	7.63
SR	Time	1861.43	1927.06	1901.01	311.37	663.20	1124.31	1298.06
	Stat	2 / 9	0 / 9	0 / 9	8 / 9	7 / 9	5 / 9	22 / 54
	Gap	8.70	9.87	13.81	0.06	3.29	6.11	6.97
Heur	Time	1.67	19.43	100.92	0.54	3.46	18.64	24.11
	Stat	0 / 9	0 / 9	0 / 9	0 / 9	0 / 9	0 / 9	0 / 54
	Gap	11.73	14.93	18.18	5.14	8.51	10.61	11.52
$n = 5\,000$								
Full	Time	1807.57	1888.82	1873.45	1807.56	1874.81	1881.00	1855.53
	Stat	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0
	Gap	−−	−−	−−	−−	−−	−−	−−
SR	Time	1774.60	2379.04	2024.18	1624.97	1970.05	2127.75	1983.43
	Stat	1 / 9	0 / 9	0 / 9	3 / 9	1 / 9	0 / 9	5 / 54
	Gap	5.32	10.20	14.41	5.15	9.95	12.78	9.63
Heur	Time	28.73	576.71	218.27	7.41	55.47	321.10	201.28
	Stat	0 / 9	0 / 9	0 / 9	0 / 9	0 / 9	0 / 9	0 / 54
	Gap	12.03	13.02	19.38	10.66	13.26	16.14	14.08
$n = 10\,000$								
Full	Time	1820.94	1839.22	1803.07	1818.80	1869.63	1885.81	1839.41
	Stat	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0	0 / 0
	Gap	−−	−−	−−	−−	−−	−−	−−
SR	Time	1902.60	2043.69	2001.61	1538.58	1906.76	2214.82	1934.68
	Stat	0 / 9	0 / 9	0 / 9	5 / 9	3 / 9	3 / 9	11 / 54
	Gap	8.99	10.90	14.47	6.01	8.23	10.84	9.91
Heur	Time	101.74	233.70	192.25	21.79	192.15	413.89	192.59
	Stat	0 / 9	0 / 9	0 / 9	0 / 9	0 / 9	0 / 9	0 / 54
	Gap	13.90	18.23	26.06	9.87	10.79	13.82	15.44

of the restricted problem is solved first, in an attempt to identify new scenarios to be added. In case this does not lead to new scenarios, 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 trade-off 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 run 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.

As for perspectives for future work, we mention especially the validation of the proposed algorithms on instances derived from practical cases, since evidence of the practical use of robust variants of classic combinatorial problems seems to be lacking in the literature. A different avenue to pursue is also a further characterization of the planning settings in which either robust or stochastic optimization is more appropriate. We have also observed that different implementations of the scenario-relaxation framework can display quite different algorithmic performance, so the development of further improvements of our currently best performing variant for each of the robustness objectives is another open research opportunity.

Acknowledgements The authors thank Frits Spieksma (KU Leuven) and Reviewer 1 for useful comments on an earlier version of this paper.

References

1. Kellerer, H., Pferschy, U., Pisinger, D.: Knapsack Problems. Springer, Berlin (2004)
2. Martello, S., Toth, P.: Knapsack Problems: Algorithms and Computer Implementations. Wiley, New York (1990)
3. Eilon, S.: Application of the knapsack model for budgeting. Omega Int. J. Manag. Sci. **15**(6), 489–494 (1987)
4. Kouvelis, P., Yu, G.: Robust Discrete Optimization and its Applications. Kluwer Academic, Norwell (1997)
5. Ben-Tal, A., Nemirovski, A.: Robust solutions of linear programming problems contaminated with uncertain data. Math. Program., Ser. A **88**, 411–424 (2000)
6. Lida, H.: A note on the max–min 0-1 knapsack problem. J. Comb. Optim. **3**, 94–99 (1999)
7. Kleywegt, A., Papastavrou, J.: The dynamic and stochastic knapsack problem. Oper. Res. **46**, 17–35 (1998)
8. Lin, G., Lu, Y., Yao, D.: The stochastic knapsack revisited: switch-over policies and dynamic pricing. Oper. Res. **56**, 945–957 (2008)
9. Averbakh, I.: On the complexity of a class of combinatorial optimization problems with uncertainty. Math. Program., Ser. A **90**, 263–272 (2001)
10. Bertsimas, D., Sim, M.: Robust discrete optimization and network flows. Math. Program., Ser. B **98**, 49–71 (2003)
11. Bertsimas, D., Sim, M.: The price of robustness. Oper. Res. **52**, 35–53 (2004)
12. Sbihi, A.: A cooperative local search-based algorithm for the multiple-scenario max–min knapsack problem. Eur. J. Oper. Res. **202**, 339–346 (2010)
13. Taniguchi, F., Yamada, T., Kataoka, S.: Heuristic and exact algorithms for the max–min optimization of the multi-scenario knapsack problem. Comput. Oper. Res. **35**, 2034–2048 (2008)
14. Klopfenstein, O., Nace, D.: A robust approach to the chance-constrained knapsack problem. Oper. Res. Lett. **36**, 628–632 (2008)
15. Assavapokee, T., Realff, M., Ammons, J.: A new min–max regret robust optimization approach for interval data uncertainty. J. Optim. Theory Appl. **137**, 297–316 (2008)
16. Aissi, H., Bazgan, C., Vanderpooten, D.: Min–max and min–max regret versions of combinatorial optimization problems: a survey. Eur. J. Oper. Res. **197**, 427–438 (2009)

17. Conde, E.: On the complexity of the continuous unbounded knapsack problem with uncertain coefficients. Oper. Res. Lett. **33**, 481–485 (2005)
18. Aissi, H., Bazgan, C., Vanderpooten, D.: Approximation of min–max and min–max regret versions of some combinatorial optimization problems. Eur. J. Oper. Res. **179**, 281–290 (2007)
19. Talla Nobibon, F., Leus, R.: Complexity results and exact algorithms for robust knapsack problems. Research report KBI 1118, Faculty of Business and Economics, KU, Leuven (2011)
20. Martello, S., Pisinger, D., Toth, P.: Dynamic programming and strong bounds for the 0–1 knapsack problem. Manag. Sci. **45**, 414–424 (1999)
21. Pisinger, D.: Where are the hard knapsack problems? Comput. Oper. Res. **32**, 2271–2284 (2005)
22. Yu, G.: On the max–min 0–1 knapsack problem with robust optimization applications. Oper. Res. **44**, 407–415 (1996)
23. Kress, M., Penn, M., Polukarov, M.: The minmax multidimensional knapsack problem with application to a chance-constrained problem. Nav. Res. Logist. **54**, 656–666 (2007)
24. Kalai, R., Vanderpooten, D.: Lexicographic α-robust knapsack problem: complexity results. Int. Trans. Oper. Res. **18**, 103–113 (2011)
25. Chen, G., Hwang, H., Tsai, T.: Efficient maxima-finding algorithms for random planar samples. Discrete Math. Theor. Comput. Sci. **6**, 107–122 (2003)
26. Assavapokee, T., Realff, M., Ammons, J., Hong, I.: Scenario relaxation algorithm for finite scenario-based min–max regret and min–max relative regret robust optimization. Comput. Oper. Res. **35**, 2093–2102 (2008)
27. Averbakh, I.: Computing and minimizing the relative regret in combinatorial optimization with interval data. Discrete Optim. **2**, 273–287 (2005)