Exact and heuristic algorithms for the interval min-max regret generalized assignment problem

Heuristic and exact algorithms for the interval min-max regret knapsack problem

Fabio Furini[(1)], Manuel Iori[(2)], Silvano Martello[(3)], Mutsunori Yagiura[(4)]

[(1)] LIPN, Paris 13 University, 93430 Villetaneuse, France
fabio.furini@lipn.univ-paris13.fr

[(2)] DISMI, University of Modena and Reggio Emilia,
Via Amendola 2, 42122 Reggio Emilia, Italy
manuel.iori@unimore.it

[(3)] DEI "Guglielmo Marconi", University of Bologna, Viale Risorgimento 2, 40136 Bologna, Italy
silvano.martello@unibo.it

[(4)] Department of Computer Science and Mathematical Informatics,
Nagoya University, Nagoya 464-8603, Japan
yagiura@nagoya-u.jp

Abstract

We consider a generalization of the 0-1 knapsack problem in which the profit of each item can take any value in a range characterized by a minimum and a maximum possible profit. A set of specific profits is called a scenario. Each feasible solution associated with a scenario has a *regret*, given by the difference between the optimal solution value for such scenario and the value of the considered solution. The *interval min-max regret knapsack problem* (MRKP) is then to find a feasible solution such that the maximum regret over all scenarios is minimized. The problem is extremely challenging both from a theoretical and a practical point of view. Its decision version is complete for the complexity class Σ_2^p hence it is most probably not in $\mathcal{NP}$. In addition, even computing the regret of a solution with respect to a scenario requires the solution of an $\mathcal{NP}$-hard problem. We examine the behavior of classical combinatorial optimization approaches when adapted to the solution of the MRKP. We introduce an iterated local search approach and a Lagrangian-based branch-and-cut algorithm, and evaluate their performance through extensive computational experiments.

Keywords: robust optimization, knapsack problem, interval min-max regret, local search, Lagrangian relaxation, branch-and-cut.

1 Introduction

Consider an investor who wants to select the best way to invest a certain capital c, among a number of financial products, each requiring a given amount of money w_j, and ensuring a fixed

return p_j. This problem is easily formalized by the classical 0-1 knapsack problem. Given n *items*, each having an associated *profit* p_j and *weight* w_j $(j = 1, 2, \ldots, n)$, and a *capacity* c, the *0-1 knapsack problem* (01KP) is to select a subset of items which has maximum total profit, and a total weight not exceeding c. Formally,

$$\text{(01KP)} \quad \max \sum_{j=1}^{n} p_j x_j \tag{1}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} w_j x_j \leq c \tag{2}$$

$$x_j \in \{0, 1\} \quad (j = 1, 2, \ldots, n), \tag{3}$$

where x_j is a binary variable taking the value one if and only if item j is selected. This problem has been the subject of intensive research during the last decades, see, e.g., the books by Martello and Toth [17], and Kellerer, Pisinger and Pferschy [14].

While the above model concerns, e.g., financial products such as repurchase agreements (repos), in a more general situation the return of product j can be a priori unknown, and may be expected to vary within a given range $[p_j^-, p_j^+]$. By assuming that we are dealing with a prudent investor (which can quite frequently be the case in times of financial crisis) s/he could aim at minimizing the a posteriori regret of the selected choice with respect to any possible profit scenario. This problem can be formalized by the following "robust" generalization of the 01KP. The profit of each item j can take any value in a range characterized by two values p_j^-, p_j^+ $(j = 1, 2, \ldots, n)$. A set s of n profits p_j^s satisfying $p_j^s \in [p_j^-, p_j^+]$ for $j = 1, 2, \ldots, n$ is called a *scenario*.

Let X be the set of all feasible solutions, i.e.,

$$X = \left\{ x = (x_1, x_2, \ldots, x_n) : \sum_{j=1}^{n} w_j x_j \leq c, \; x_j \in \{0, 1\} (j = 1, 2, \ldots, n) \right\}. \tag{4}$$

We denote by $z^s(x)$ the solution value given, for scenario s, by a solution vector $x \in X$, i.e.,

$$z^s(x) = \sum_{j=1}^{n} p_j^s x_j. \tag{5}$$

Let $z_\star^s$ be the optimal solution value under scenario s, i.e., the solution value of the 01KP (1)-(3) when $p_j = p_j^s$ $(j = 1, 2, \ldots, n)$. The *regret* associated with a solution x, for a scenario s, is then

$$r^s(x) = z_\star^s - z^s(x). \tag{6}$$

Let S_0 denote the set of all possible scenarios s, i.e., $S_0 = \{s : p_j^s \in [p_j^-, p_j^+] (j = 1, 2, \ldots, n)\}$. The *maximum regret* $r(x)$ of a solution x is then the maximum $r^s(x)$ value over all scenarios, i.e., $r(x) = \max_{s \in S_0} r^s(x)$. The *interval min-max regret knapsack problem* (MRKP) is to find

a feasible solution vector x such that the maximum regret is minimized. Formally,

$$\text{(MRKP)} \quad \min \max_{s \in S_0} r^s(x) \tag{7}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} w_j x_j \leq c \tag{8}$$

$$x_j \in \{0,1\} \quad (j = 1, 2, \ldots, n). \tag{9}$$

For the 01KP it is usual to assume that all profits are non-negative, as cases with negative values can be easily handled through a preprocessing (see [17], Section 2.1). For the MRKP, while items j (financial products) with both p_j^- and p_j^+ less than 0 can obviously be removed from the instance, items with $p_j^- \leq 0$ and $p_j^+ \geq 0$ need a specific handling, which will be discussed in Section 3. Also, from now on, we will assume that capacity and weights are positive integers.

The reminder of the paper is organized as follows. In the next section we briefly review prior works in this area. In Section 3 we discuss a basic result from the literature, and derive a mixed-integer linear model for the problem. In Section 4 we present standard exact approaches (a Benders-like decomposition and a branch-and-cut algorithm) for the MRKP, and provide an effective method for their initialization. Heuristic and metaheuristic algorithms are introduced in Section 5, while an effective Lagrangian-based branch-and-cut approach is developed in Section 6. The outcome of extensive computational experiments on exact and approximate algorithms is discussed in Section 7, and conclusions follow in Section 8.

2 Complexity issues and literature review

The 01KP is the special case of the MRKP that arises when $p_j^- = p_j^+$ $(j = 1, 2, \ldots, n)$, which implies that the MRKP is $\mathcal{NP}$-hard in the ordinary sense. It is an open question whether it is $\mathcal{NP}$-hard in the strong sense. Deineko and Woeginger [10] recently proved that the decision version of the problem is complete for the complexity class Σ_2^p (see Garey and Johnson [11], Chapter 7, or Papadimitriou [21], Chapter 17) and hence is most probably not in $\mathcal{NP}$.

To our knowledge, the only paper devoted to the MRKP is the aforementioned study of its complexity [10]. Other robust versions of the 01KP have however been considered in the literature. The main variants concern: (i) the scenarios, and (ii) the objective function.

The profit scenarios can be given

- by the profit intervals, as in our case (*interval scenario case* or, alternatively, *interval data case*), or
- as an explicit list $\mathcal{L}$ of scenarios, i.e., of profit vectors (p_j^s) $(s \in \mathcal{L})$ (*discrete scenario case*), see, e.g., Yu [26] for an exact approach, or Aissi, Bazgan and Vanderpooten [2] for approximation properties.

Cases with an unbounded number of scenarios, whose interest is mostly theoretical, have also been considered (see, e.g., Kouvelis and Yu [15]).

Let S denote the set of all possible scenarios, however defined. The three main objective functions used in scenario-based robust optimization are:

- *max min*: maximize the worst-case solution over all scenarios, i.e., $\max_{x \in X} \min_{s \in S} z^s(x)$. For the interval scenario case of the 01KP, this is immediately obtained by solving (1)-(3) with $p_j = p_j^-$ for $j = 1, 2, \ldots, n$;
- *min max* (*absolute*) *regret*: minimize the maximum regret (our case);
- *min max relative regret*: minimize $\max_{s \in S} r_s(x)/z_\star^s$ over all feasible solutions $x \in X$.

For practical cases in which the min max regret is a useful measure see, e.g., the applications to the energy crop supply and to investment choices, discussed, respectively, by Kazakci, Rozakis and Vanderpooten [13] and Michenaudc and Solnik [18]. A thorough analysis of the huge literature on scenario-based robust optimization is beyond the scope of this article. The interested reader is referred to the classical books by Kouvelis and Yu [15] and Kasperski [12], as well as to the paper by Averbakh [3] and to the recent surveys by Aissi, Bazgan and Vanderpooten [1] and Candia-Véjar, Álvarez-Miranda and Maculan [9]. We mention here that, in recent years, the interval scenario min-max regret version of other relevant combinatorial optimization problems has been studied, such as the shortest path problem (Montemanni and Gambardella [20]), the traveling salesman problem (Montemanni, Barta, Mastrolilli and Gambardella [19]), the assignment problem (Pereira and Averbakh [22]), and the set covering problem (Pereira and Averbakh [23]).

We also mention that different (probabilistic) robustness paradigms have been widely treated in the literature. Such approaches model optimization under uncertainty through probability distributions over the space of all possible scenarios. For this research area, which is closely related to the classical field of stochastic programming (see, e.g., the recent books by Shapiro, Dentcheva, and Ruszczyńskiwe [24] and Birge and Louveaux [8]), we refer the reader to the basic works by Ben-Tal and Nemirovski [4] and Bertsimas and Sim [6, 7].

As recently observed by Deineko and Woeginger [10], the solution of even moderately sized instances of the interval min-max regret knapsack is challenging and seems to require innovative approaches. Also note that even computing the regret of a single solution with respect to a single scenario requires the solution of an $\mathcal{NP}$-hard problem (a 01KP).

3 A mixed-integer linear model

The following basic result (whose roots are in Yaman, Karaşan and Pinar [25]) has been explicitly proved for the MRKP (in the more general context of binary interval min-max regret problems) by Aissi, Bazgan and Vanderpooten [1]:

Lemma 1 *For any solution $x \in X$, its worst case scenario is $\sigma(x)$, defined by*

$$p_j^{\sigma(x)} = \begin{cases} p_j^- & \text{if } x_j = 1; \\ p_j^+ & \text{otherwise.} \end{cases} \tag{10}$$

(Intuitively, the scenario inducing the maximum regret has the worst profits for the selected items, and the best profits for the non-selected items.) Hence, from now on, we will restrict our attention, without loss of generality to the subset of scenarios $S \subseteq S_0$ induced by Lemma 1, i.e., $S = \{\sigma(x) : x \in X\}$. In addition, by observing that there is a unique scenario $\sigma(x) \in S$ for each solution $x \in X$, when no confusion arises we will use σ instead of $\sigma(x)$.

From Lemma 1 and equations (4), (6) and (7), by considering the worst-case scenario of x, the MRKP can be re-written as

$$\min_{x\in X}(\max_{s\in S}(\max_{y\in X}\sum_{j=1}^{n}p_j^s y_j-\sum_{j=1}^{n}p_j^s x_j)) \;=\; \min_{x\in X}(\max_{y\in X}\sum_{j=1}^{n}p_j^{\sigma(x)} y_j-\sum_{j=1}^{n}p_j^{\sigma(x)} x_j). \tag{11}$$

By further observing that the last summation of (11) can be developed without including the p_j^+ terms (which disappear when $x_j=0$), we get our simplest formulation of the MRKP:

$$\min_{x\in X}(z_\star^\sigma-\sum_{j=1}^{n}p_j^- x_j), \tag{12}$$

where (recall that we use σ for $\sigma(x)$)

$$z_\star^\sigma=\max_{y\in X}\sum_{j=1}^{n}p_j^\sigma y_j=\sum_{j=1}^{n}p_j^\sigma \tilde{y}_j^\sigma \tag{13}$$

and $\tilde{y}_j^\sigma$ $(j=1,2,\ldots,n)$ denotes an optimal solution vector for the 01KP under scenario σ.

By observing that, from Lemma 1, for $x\in X$ we can write p_j^σ as

$$p_j^\sigma=p_j^+ +(p_j^- -p_j^+)x_j, \tag{14}$$

and introducing a new (non integer) variable ϑ, along with a constraint that forces ϑ to satisfy $\vartheta\geq z_\star^\sigma\ \forall\ \sigma\in S$, the MRKP is expressed by the mixed integer linear model

$$\mathrm{M}(S)\quad \min\ \vartheta-\sum_{j=1}^{n}p_j^- x_j \tag{15}$$

$$\text{s.t.}\qquad \vartheta\ \geq\ \sum_{j=1}^{n}p_j^+\tilde{y}_j^\sigma+\sum_{j=1}^{n}(p_j^- -p_j^+)\tilde{y}_j^\sigma x_j \qquad \forall\ \sigma\in S \tag{16}$$

$$\sum_{j=1}^{n}w_j x_j\ \leq\ c \tag{17}$$

$$x_j\ \in\ \{0,1\}\quad (j=1,2,\ldots,n), \tag{18}$$

because an optimal solution $(x^\star,\ \vartheta^\star)$ satisfies $\vartheta^\star\geq z_\star^s$ for all $s\in S$.

We will denote by $\hat{\mathrm{M}}(S)$ the continuous relaxation of $\mathrm{M}(S)$, obtained by relaxing (18) to

$$0\leq x_j\leq 1\quad (j=1,2,\ldots,n), \tag{19}$$

Note that the model has $n+1$ variables, but an exponential number of constraints (16) (one per scenario), each of which would require the solution of a 01KP. In the next section we discuss how to deal with such inconvenience.

Lemma 1 can also be used to see how to handle cases where $p_j^-\leq 0$ holds for some item j.

Property 1 *Let j be an item for which $p_j^-\leq 0$. If $p_j^+\leq|p_j^-|$ then item j can be removed from the instance.*

Proof If $p_j^- = 0$, then $p_j^+ = 0$ and the claim follows. Hence assume $p_j^- < 0$. We show that the regret of a solution x with $x_j = 1$ is never smaller than that of a solution x' with $x'_j = 0$ and $x'_i = x_i$ for all $i \neq j$. From Lemma 1, we have $p_j^{\sigma(x)} = p_j^-$, $p_j^{\sigma(x')} = p_j^+$ and $p_i^{\sigma(x)} = p_i^{\sigma(x')}$ for all $i \neq j$. Hence

$$z^{\sigma(x)}(x) = z^{\sigma(x')}(x') - |p_j^-|. \tag{20}$$

The optimal solutions under scenarios $\sigma(x)$ and $\sigma(x')$ satisfy

$$z_\star^{\sigma(x)} \geq z_\star^{\sigma(x')} - \max\{0, p_j^+\}, \tag{21}$$

because by removing item j from the latter optimal solution (if present) we obtain a solution that is feasible for the former. From (20) and (21), the regrets for the two cases satisfy

$$\begin{aligned} r^{\sigma(x)}(x) &= z_\star^{\sigma(x)} - z^{\sigma(x)}(x) \geq z_\star^{\sigma(x')} - \max\{0, p_j^+\} - z^{\sigma(x')}(x') + |p_j^-| = \\ & r^{\sigma(x')}(x') - \max\{0, p_j^+\} + |p_j^-| \geq r^{\sigma(x')}(x'). \ \square \end{aligned}$$

Additionally observe that an item with $p_j^- \leq 0$ but $p_j^+ > |p_j^-|$ cannot be removed from the instance, since it could be included in an optimal solution. Consider indeed the family of instances with $n = 2$, $w_1 = w_2 = c$, $p_1^- < 0$, $p_1^+ > |p_1^-|$, and $p_2^- = p_2^+ = \varepsilon$ (where ε is a small positive value). The regrets of the three feasible solutions $x^{(1)} = (0, 0)$, $x^{(2)} = (1, 0)$ and $x^{(3)} = (0, 1)$ are, respectively, $p_1^+ - 0$, $\varepsilon + |p_1^-|$ and $p_1^+ - \varepsilon$. For a sufficiently small ε, the regret of $x^{(2)}$ is smaller than the other two, i.e., the unique optimal solution includes item 1.

3.1 Evaluating the maximum regret

As already observed, evaluating the regret $r(x)$ of a solution $x \in X$ through Lemma 1 requires the solution to an $\mathcal{NP}$-hard problem. The exact and approximation algorithms introduced in the next sections will try to limit the number of such solutions through a standard approach that assumes that a feasible solution providing an upper bound U^r on the optimal regret is known. Let 01KP(σ) denote the instance of 01KP in which $p_j = p_j^\sigma$ for $j = 1, 2, \ldots, n$. Recall that $z^s(x)$ is the solution value given, for scenario s, by solution x (see (5)). The approach, outlined in Algorithm 1, avoids the computation of the regret when a lower bound based evaluation shows that $r(x) \geq U^r$.

input: a solution x and an upper bound value U^r;
$L :=$ heuristic solution value for 01KP(σ);
if $L - z^\sigma(x) \geq U^r$ **then return** $r(x) := +\infty$
else
  $U :=$ upper bound for 01KP(σ);
  if $L = U$ **then** $z_\star^\sigma := L$
  else compute $z_\star^\sigma :=$ optimal solution value for 01KP(σ)
end if;
return $r(x) := z_\star^\sigma - z^\sigma(x)$.

Algorithm 1: Evaluation of the maximum regret of a solution x.

4 Standard exact algorithms

In this section we discuss two solution approaches (a Benders-like decomposition and a branch-and-cut algorithm) that have been frequently used for the exact solution of problems of this kind. The former approach is not the classical Benders' decomposition [5], as the slave problem is not a linear program. However the term is widely used in the literature, and specifically in the min-max regret literature. We thus decided to adopt it for better clarity with respect to previous similar approaches like, e.g., those in Montemanni, Barta, Mastrolilli and Gambardella [19] or Pereira and Averbakh [23]. Also note that the results presented so far in the literature on robust versions of combinatorial optimization problems do not indicate a clear winner among such approaches. For example, the computational results presented by Montemanni, Barta, Mastrolilli and Gambardella [19] for the traveling salesman problem show that Benders' decomposition is more effective, while those in Pereira and Averbakh [23] for the set covering problem indicate a superiority of branch-and-cut.

We will use model M(S) by iteratively solving instances M(R) in which only a subset, $R \subseteq S$, of scenarios (i.e., of constraints (16)) is considered, and progressively adding scenarios to R until an optimal solution is found. A fundamental tool for such approaches is a method to separate violated constraints from non-violated ones.

4.1 Separation

Given a solution $(\tilde{x}, \tilde{\vartheta})$ satisfying (17)-(18), we want to check if it also satisfies constraints (16). In order to determine if there exists a scenario $\sigma \in S$ for which

$$\sum_{j=1}^{n} \left(p_j^+ + (p_j^- - p_j^+)\tilde{x}_j\right) \tilde{y}_j^\sigma > \tilde{\vartheta} \tag{22}$$

holds, we can define, for $j = 1, 2, \ldots, n$, $p'_j = p_j^+ + (p_j^- - p_j^+)\tilde{x}_j$, and solve the 01KP

$$\max \sum_{j=1}^{n} p'_j y_j : \sum_{j=1}^{n} w_j y_j \leq c, y_j \in \{0,1\} \ (j = 1, 2, \ldots, n). \tag{23}$$

If the optimal solution value is greater than $\tilde{\vartheta}$, then a violated constraint has been found.

It is interesting to observe that the above procedure does not exploit the integrality of $\tilde{x}$, and hence the separation also holds for fractional solutions $(\tilde{x}, \tilde{\vartheta})$ satisfying (17) and (19). However, for a fractional $\tilde{x}$, p'_j can take any value in the continuous interval $[p_j^-, p_j^+]$ so the set of scenarios extends from S to S_0.

4.2 Algorithms

The above separation method can be used for two classical algorithmic approaches which iteratively solve problem instances by only considering a subset of scenarios.

Benders' decomposition

We tested a first approach, based on *Benders' decomposition*, which solves, at each iteration, a *master problem* M(R) defined by a relaxation of (15)-(18) in which S is replaced a by a subset $R \subset S$ of constraints. Let $(\tilde{x}, \tilde{\vartheta})$ be the optimal solution for the current master. The *slave problem* (23) is then used to find a violated constraint, if any: if such a constraint is found, the corresponding scenario is added to R. The process is iterated until a solution is found that violates no constraint, and hence is feasible. Since each solution to the master problem provides a valid lower bound on the optimal solution value, the first feasible solution encountered is optimal.

Branch-and-cut

The second approach we tested was a branch-and-cut algorithm. At each node of the branch-decision tree, we solve the continuous relaxation, $\hat{\text{M}}(R)$, of the master problem above, i.e., (15)-(17) and (19) with $R \subset S_0$. If its value is not smaller than that of the incumbent solution, then the node is fathomed. Otherwise, the (possibly fractional) current solution is tested, as above, to find violated constraints (*cuts*) to be added to the current set R. When no constraint is violated, if the current solution is integer, the incumbent is possibly updated. If instead it is fractional, a branching follows. The general framework of branch-and-cut algorithms is very flexible, and various implementations are possible. As will be seen in Section 7.2, our approach was implemented using the default branch-and-cut framework of Cplex.

The former approach usually requires a smaller number of 01KP solutions, but has the disadvantage of only providing a feasible solution upon completion. Although branch-and-cut tends to solve a (much) larger number of 01KPs, it can be more efficient as the produced cuts can fathom branch-decision nodes at an earlier stage hence accelerating the overall convergence. For a case such as the considered one, in which the separation is provided by a problem which (although $\mathcal{NP}$-hard) is relatively easy to solve in practice, the latter approach tends to be faster. This is confirmed by the computational experiments that will be discussed in Section 7.

A relevant aspect for a practically efficient implementation of the above algorithms is the determination of an effective initial set of constraints, which is discussed in the next section.

4.3 Initialization

For algorithms based on Benders' decomposition and branch-and-cut, it is convenient to initialize the restricted scenario set R with some scenario, as an initial empty set R would produce a negative unbounded solution value. The following property gives useful indication on a possible effective initialization of the restricted scenario set.

Property 2 *Let x^- and $\hat{x}^-$ be, respectively, the optimal solutions to the 01KP* (1)-(3) *with $p_j = p_j^-$ for $j = 1, 2, \ldots, n$ and to its* continuous relaxation*, given by* (1)-(2) *and* (19)*. Similarly define x^+ and $\hat{x}^+$ for the case $p_j = p_j^+$.*

1. *If R contains a constraint* (16)*, for a certain $\sigma \in S$, in which $\tilde{y}_j^\sigma = x_j^-$ $(j = 1, 2, \ldots, n)$, then the solution value of $M(R)$ is non-negative, and the solution value of $\hat{M}(R)$ is at least $\sum_{j=1}^n p_j^-(x_j^- - \hat{x}_j^-)$.*

2. *If R contains a constraint* (16)*, for a certain $\sigma \in S$, in which $\tilde{y}_j^\sigma = x_j^+$ $(j = 1, 2, \ldots, n)$, then the solution value of $M(R)$ is non-negative, and the solution value of $\hat{M}(R)$ is at least $\sum_{j=1}^n p_j^+(x_j^+ - \hat{x}_j^+)$.*

Proof To prove point *1.*, preliminary observe that the constraint implies

$$\vartheta \geq \sum_{j=1}^n p_j^+ x_j^- + \sum_{j=1}^n (p_j^- - p_j^+) x_j^- x_j \geq \sum_{j=1}^n p_j^+ x_j^- + \sum_{j=1}^n (p_j^- - p_j^+) x_j^- = \sum_{j=1}^n p_j^- x_j^-. \tag{24}$$

Since x^- is an optimal 01KP solution for the case $p_j = p_j^-$, we have

$$\sum_{j=1}^n p_j^- x_j \leq \sum_{j=1}^n p_j^- x_j^- \tag{25}$$

for all feasible solutions x to $\mathrm{M}(R)$. It follows, from (24) and (25), that

$$\vartheta - \sum_{j=1}^n p_j^- x_j \geq \sum_{j=1}^n p_j^- x_j^- - \sum_{j=1}^n p_j^- x_j^- = 0, \tag{26}$$

and hence the solution value of $\mathrm{M}(R)$ is non-negative. In addition, for any feasible solution x to $\hat{\mathrm{M}}(R)$, we have

$$\sum_{j=1}^n p_j^- x_j \leq \sum_{j=1}^n p_j^- \hat{x}_j^-. \tag{27}$$

Hence, from (24) and (27), the solution value to $\hat{\mathrm{M}}(R)$ is bounded as

$$\vartheta - \sum_{j=1}^n p_j^- x_j \geq \sum_{j=1}^n p_j^- x_j^- - \sum_{j=1}^n p_j^- \hat{x}_j^-. \tag{28}$$

Coming to point *2.*, the constraint implies, for a feasible solution x,

$$\vartheta \geq \sum_{j=1}^n p_j^+ x_j^+ + \sum_{j=1}^n (p_j^- - p_j^+) x_j^+ x_j. \tag{29}$$

It follows that the objective function values of both $\mathrm{M}(R)$ and $\hat{\mathrm{M}}(R)$ satisfy

$$\begin{aligned} \vartheta - \sum_{j=1}^n p_j^- x_j &\geq \sum_{j=1}^n p_j^+ x_j^+ + \sum_{j=1}^n (p_j^- - p_j^+) x_j^+ x_j - \sum_{j=1}^n p_j^- x_j \\ &= \sum_{j=1}^n p_j^+ x_j^+ + \sum_{j=1}^n \left((p_j^- - p_j^+) x_j^+ - p_j^- \right) x_j \geq \sum_{j=1}^n p_j^+ x_j^+ - \sum_{j=1}^n p_j^+ x_j. \end{aligned} \tag{30}$$

For M(R), the last quantity in (30) is non-negative, because x^+ is an optimal 01KP solution for the case $p_j = p_j^+$. For $\hat{\text{M}}(R)$, the last quantity in (30) is not less than

$$\sum_{j=1}^{n} p_j^+ x_j^+ - \sum_{j=1}^{n} p_j^+ \hat{x}_j^+ . \ \square \tag{31}$$

Following Property 2, in our computational experiments (see Section 7.2) the set R of all algorithms was initialized with the two constraints corresponding to solutions x^+ and x^-, very often obtaining a positive (possibly relatively high) initial lower bound value. On the other hand, an initialization including just one scenario would never produce a positive lower bound. Indeed, the following property holds.

Property 3 *If $|R| = 1$ then the optimal value of both $M(R)$ and $\hat{M}(R)$ cannot be positive.*

Proof Let $R = \{\sigma\}$. In an optimal solution (both to M(R) and $\hat{\text{M}}(R)$) we have $\vartheta = \sum_{j=1}^{n} p_j^+ \tilde{y}_j^\sigma + \sum_{j=1}^{n} (p_j^- - p_j^+) \tilde{y}_j^\sigma x_j$, hence the objective function can be written as

$$\sum_{j=1}^{n} p_j^+ \tilde{y}_j^\sigma + \sum_{j=1}^{n} \left((p_j^- - p_j^+) \tilde{y}_j^\sigma - p_j^- \right) x_j = \sum_{j:\tilde{y}_j^\sigma = 1} p_j^+ (1 - x_j) - \sum_{j:\tilde{y}_j^\sigma = 0} p_j^- x_j,$$

which takes the value zero if $x_j = \tilde{y}_j^\sigma$ for $j = 1, 2, \ldots, n$. $\square$

5 Heuristic algorithms

In this section we describe a heuristic approach obtained by constructing an initial solution (using greedy or ILP-based heuristics), and refining it through metaheuristic search.

5.1 Greedy heuristics

The most classical tool for constructing an initial feasible solution to knapsack problems is the well-known *greedy* algorithm, which consists of: (i) examining the items according to a pre-specified order, and, at each iteration, (ii) adding the current item to the solution iff its weight does not exceed the current residual capacity.

The greedy algorithm for the traditional 01KP sorts the items according to non-increasing p_j/w_j values. For the MRKP, many different orderings are possible. On the basis of extensive computational experiments with the different algorithmic approaches, four sorting criteria turned out to be the best ones, namely

$$p_j^+/w_j; \qquad (p_j^+ + p_j^-)/w_j; \qquad (p_j^- p_j^+)/w_j; \qquad p_j^-/w_j. \tag{32}$$

Other sorting criteria (e.g., p_j^+, $1/w_j$, $(p_j^+ - p_j^-)/w_j$) turned out to be worse than the ones above. A possible explanation is that, provided all profits are non-negative, criteria (32) tend to avoid producing solutions violating the following *dominance relation*.

Property 4 *If $p_j^- \geq p_k^-$, $p_j^+ \geq p_k^+$ and $w_j \leq w_k$, then item j* dominates *item k in the following sense: If an optimal solution has $x_k = 1$ and $x_j = 0$, then the solution obtained from it by setting $x_k = 0$ and $x_j = 1$ is also optimal.*

Proof. Let $x^{\rm A}$ and $x^{\rm B}$ be two solutions such that $x^{\rm A}_j = 0, x^{\rm A}_k = 1$, $x^{\rm B}_j = 1, x^{\rm B}_k = 0$, and $x^{\rm A}_i = x^{\rm B}_i \ \ \forall i \neq j,k$. The maximum regret $r(x)$ of a solution x can be expressed as

$$r(x) = \max_{y \in X} \sum_{i=1}^{n} p_i^{\sigma(x)} y_i - \sum_{i=1}^{n} p_i^{\sigma(x)} x_i = \max_{y \in X} d(x,y), \tag{33}$$

where

$$d(x,y) = \sum_{i=1}^{n} p_i^{\sigma(x)} (y_i - x_i). \tag{34}$$

Let y^* be an optimal solution to the 01KP under the worst-case scenario $\sigma(x^{\rm B})$ induced by solution $x^{\rm B}$ (i.e., $y^* = \arg\max_{y' \in X} d(x^{\rm B}, y')$). For every solution $y \in X$, we have, from (33) and (34),

$$r(x^{\rm B}) - r(x^{\rm A}) = \max_{y' \in X} d(x^{\rm B}, y') - \max_{y'' \in X} d(x^{\rm A}, y'') \leq d(x^{\rm B}, y^*) - d(x^{\rm A}, y). \tag{35}$$

We will show that, for any optimal solution y^*, there is a solution $\tilde{y} \in X$ such that $d(x^{\rm B}, y^*) - d(x^{\rm A}, \tilde{y}) \leq 0$, i.e., from (35), $r(x^{\rm B}) \leq r(x^{\rm A})$. Consider indeed solutions $\tilde{y}$ such that $\tilde{y}_i = y^*_i$ for all $i \neq j,k$, and recall that $p_j^{\sigma(x^{\rm A})} = p_j^+$, $p_k^{\sigma(x^{\rm A})} = p_k^-$, $p_j^{\sigma(x^{\rm B})} = p_j^-$, $p_k^{\sigma(x^{\rm B})} = p_k^+$, and $p_i^{\sigma(x^{\rm A})} = p_i^{\sigma(x^{\rm B})}$ for all $i \neq j,k$. Then

$$\begin{aligned} \Delta &= d(x^{\rm B}, y^*) - d(x^{\rm A}, \tilde{y}) = p_j^-(y^*_j - x^{\rm B}_j) + p_k^+(y^*_k - x^{\rm B}_k) - p_j^+(\tilde{y}_j - x^{\rm A}_j) - p_k^-(\tilde{y}_k - x^{\rm A}_k) \\ &= p_j^-(y^*_j - 1) + p_k^+ y^*_k - p_j^+ \tilde{y}_j - p_k^-(\tilde{y}_k - 1). \end{aligned}$$

There are the following four cases for the configuration of y^*.

1. $y^*_j = 0$ and $y^*_k = 0$: For $\tilde{y} = y^*$ we have $\Delta = -p_j^- + p_k^- \leq 0$.
2. $y^*_j = 1$ and $y^*_k = 1$: For $\tilde{y} = y^*$ we have $\Delta = p_k^+ - p_j^+ \leq 0$.
3. $y^*_j = 1$ and $y^*_k = 0$: For $\tilde{y} = y^*$ we have $\Delta = -p_j^+ + p_k^- \leq -p_j^+ + p_k^+ \leq 0$.
4. $y^*_j = 0$ and $y^*_k = 1$: For $\tilde{y}_j = 1$ and $\tilde{y}_k = 0$ (which is feasible because y^* is feasible and $w_j \leq w_k$), we have $\Delta = -p_j^- + p_k^+ - p_j^+ + p_k^- \leq 0$. $\Box$

A greedy algorithm, with any of the above orderings, finds a feasible solution in $O(n \log n)$ time. Note however that, if one is also interested in the corresponding maximum regret, this would require the solution of an $\mathcal{NP}$-hard problem (the 01KP for the worst scenario).

5.2 ILP-based heuristics

As we will see in Section 7, the quality of the greedy solutions is not sufficiently good. Better quality solutions were obtained, at the expenses of higher CPU times, by heuristically solving problem M(S) through the branch-and-cut algorithm discussed in Section 4.2, halted after a small time limit.

A more sophisticated approach is provided by the following ILP-based heuristic, coming from a technique developed by Kasperski [12] for min-max regret versions of interval scenario cases with a zero duality gap. From (12), we can write the MRKP as

$$\min_{x\in X}\left(\max_{y\in X}\sum_{j=1}^{n}p_j^{\sigma}y_j-\sum_{j=1}^{n}p_j^{-}x_j\right). \tag{36}$$

Consider the continuous relaxation of $\max_{y\in X}\sum_{j=1}^{n}p_j^{\sigma}y_j$:

$$\max\sum_{j=1}^{n}p_j^{\sigma}y_j:\sum_{j=1}^{n}w_jy_j\le c, 0\le y_j\le 1\ (j=1,2,\ldots,n). \tag{37}$$

By introducing dual variables α for the capacity constraint and β_j for constraints $y_j\le 1$, we obtain the dual of (37) as

$$\min \alpha c+\sum_{j=1}^{n}\beta_j:\alpha w_j+\beta_j\ge p_j^{\sigma}\ (j=1,2,\ldots,n),\alpha\ge 0,\beta_j\ge 0\ (j=1,2,\ldots,n). \tag{38}$$

By embedding (38) in (36), using (14) for p_j^{σ}, and replacing $x\in X$ with (2)-(3), we finally obtain problem U-MRKP:

$$\text{(U-MRKP)}\quad \min\quad \alpha c+\sum_{j=1}^{n}\beta_j-\sum_{j=1}^{n}p_j^{-}x_j \tag{39}$$

$$\text{s.t.}\qquad \sum_{j=1}^{n}w_jx_j\le c \tag{40}$$

$$\alpha w_j+\beta_j\ge p_j^{+}+(p_j^{-}-p_j^{+})x_j\quad (j=1,2,\ldots,n) \tag{41}$$

$$x_j\in\{0,1\}\quad (j=1,2,\ldots,n) \tag{42}$$

$$\alpha\ge 0 \tag{43}$$

$$\beta_j\ge 0\quad (j=1,2,\ldots,n) \tag{44}$$

The U-MRKP is considerably simpler than the M(S) formulation of the MRKP, as it replaces the exponentially many constraints (16) with a polynomial number of constraints and variables. However, the two problems are not equivalent: the solution, say $\tilde{x}$, provided by the U-MRKP is indeed feasible for the MRKP, but (because of the integrality gap introduced by (37)) it is not necessarily optimal with respect to (36). On the other hand, the problem remains theoretically difficult:

Property 5 *Problem* U-MRKP *is* $\mathcal{NP}$*-hard.*

Proof We show that the 01KP polynomially transforms to the U-MRKP. Given an instance of 01KP, define an instance of U-MRKP with $p_j^{-}=p_j^{+}=p_j$ $(j=1,2,\ldots,n)$. Such an instance decomposes into problems

(i) $\{\min \alpha c+\sum_{j=1}^{n}\beta_j:\alpha w_j+\beta_j\ge p_j\ (j=1,2,\ldots,n),\alpha\ge 0,\beta_j\ge 0\ (j=1,2,\ldots,n)\}$, and

(ii) $\{\max\sum_{j=1}^{n}p_jx_j:\sum_{j=1}^{n}w_jx_j\le c, x_j\in\{0,1\}\ (j=1,2,\ldots,n)\}$.

The former problem is the dual of the continuous relaxation of a 01KP, while the latter is a

regular 01KP. □

As we will see in Section 7, good approximate solutions to the MRKP are produced by the following heuristic approach:

(i) find an optimal solution $\tilde{x}$ to the U-MRKP;

(ii) compute the actual regret produced by $\tilde{x}$ through Algorithm 1.

Although this approach requires the solution to two $\mathcal{NP}$-hard problems, the computational results of Section 7 show that high quality solutions are achieved very quickly in practice.

5.3 Iterated local search

The initial heuristic solutions were improved through iterated local search. Recall that $r(x)$ is the maximum regret associated with a solution x. We will denote by $x_\star$ the incumbent MRKP solution. The iterated local search makes use of two procedures which, respectively, perturb $x_\star$, and improve it through local search.

Perturbation

Procedure `Perturbation`(x, k) removes k randomly selected items from the given solution x, and fills the resulting residual capacity by executing the greedy algorithm on the set of items originally having $x_j = 0$. The resulting overall solution is returned.

Local Search

Recall that, given a feasible solution x to an instance of MRKP, 01KP$(\sigma(x))$ denotes the corresponding instance of 01KP in which $p_j = p_j^{\sigma(x)}$ for $j = 1, 2, \ldots, n$ (see Lemma 1).

Procedure `Local_Search`(x) considers every item pair (i, j) with $x_i = 1$ and $x_j = 0$, and produces a new solution by swapping them, provided that the new solution satisfies the capacity constraint. The resulting residual capacity is then filled through the greedy algorithm applied to those items k for which $x_k = 0$, $k \neq i$ and $k > j$ (in order to avoid duplicated solutions). Let x' denote the resulting solution. For each feasible swap, the regret $r(x')$ is computed through Algorithm 1.

The local search follows a *best improvement* policy: once all new solutions x' have been evaluated, the one producing the smallest regret is selected. If it improves on the regret of the input solution x, then x is replaced by x' and the procedure is iterated. When no improvement occurs, the procedure terminates returning the current solution x.

Iterated Local Search

The overall iterated local search procedure is outlined in Algorithm 2. In principle, it can be terminated through any termination condition. In our experiments it was halted by a time limit.

input: a heuristic solution x
$x_\star := x$;
$k := 1$;
repeat
  $x :=$ `Perturbation`$(x_\star, k)$;
  $\bar{x} :=$ `Local_Search`(x);
  if $r(\bar{x}) < r(x_\star)$ **then**
    $x_\star := \bar{x}$;
    $k := 1$;
  else
    $k := k + 1$;
    if $k > k_{\max}$ **then** $k := 1$
  end if
until `Termination Condition`
return $x_\star$

Algorithm 2: Procedure *Iterated Local Search*.

6 A Lagrangian-based branch-and-cut algorithm

As we will see in Section 7, the computational experimentation of the standard exact approaches outlined in Section 4 did not produce satisfactory results. In this section we describe the specifically tailored branch-and-cut algorithm we developed, which proved to be computationally more effective. In the following, we denote such algorithm as FIMY.

We will first introduce the general structure of FIMY, and the main steps that are performed at each decision node, and then describe the enhancements adopted to speed up its performance. Let $x_\star$ denote the incumbent solution, and $r(x_\star)$ its value. The first incumbent solution is obtained by improving through Algorithm 2, the solution to U-MRKP (see Section 5).

The algorithm starts by only considering a very small subset of the exponentially many constraints (16), i.e., problem (15)-(18) with S replaced by $R \subset S$. The initial R includes the two scenarios discussed in Property 2 ($p_j = p_j^-$ and $p_j = p_j^+$ for all j), and the one induced by the incumbent ($p_j = p_j^{\sigma(x_\star)}$ for all j). At each iteration the current set R is possibly augmented.

The branch-decision tree follows the strategy commonly adopted for the 01KP: after having sorted the items according to one of the criteria considered in (32), at each level two child nodes are generated by setting the next variable x_j to 1 (if the current residual capacity allows insertion of item j) and to 0. The tree is searched in a depth-first fashion by first exploring the node, if any, generated by condition $x_j = 1$. On the basis of preliminary computational experiments, the values $(p_j^+ p_j^-)/w_j$ were used for sorting the items.

At each iteration, a first attempt to fathom the current node is performed by solving the continuous relaxation, $\hat{\mathrm{M}}(R)$, of the current problem $\mathrm{M}(R)$ (see Section 3). If the resulting lower bound is lower than $r(x_\star)$, a second attempt is performed through the Lagrangian relaxation of those constraints (16) that have been generated so far. Namely, we relax the

current set R, obtaining:

$$L(R,\lambda) \quad \min\left(\vartheta - \sum_{j=1}^{n} p_j^- x_j - \sum_{\sigma\in R}\lambda_\sigma\left(\vartheta - \sum_{j=1}^{n} p_j^+ \tilde{y}_j^\sigma - \sum_{j=1}^{n}(p_j^- - p_j^+)\tilde{y}_j^\sigma x_j\right)\right)$$

$$= \sum_{\sigma\in R}\lambda_\sigma \sum_{j=1}^{n} p_j^+ \tilde{y}_j^\sigma + \min\left(\vartheta\left(1 - \sum_{\sigma\in R}\lambda_\sigma\right) - \sum_{j=1}^{n}\hat{p}_j x_j\right) \tag{45}$$

$$\text{s.t.} \sum_{j=1}^{n} w_j x_j \le c \tag{46}$$

$$x_j \in \{0,1\} \quad (j = 1,2,\ldots,n), \tag{47}$$

where $\hat{p}_j = p_j^- - \sum_{\sigma\in R}\lambda_\sigma(p_j^- - p_j^+)\tilde{y}_j^\sigma$.

Solving the Lagrangian dual problem, i.e., finding the multipliers that provide the highest possible value of $L(R,\lambda)$, could be computationally intractable. Instead, as the solution to $\hat{\mathrm{M}}(R)$ also provides an optimal solution to its dual, we use the optimal values of the dual variables associated with (16) (i.e., optimal multipliers for the continuous relaxation of $L(R,\lambda)$) as Lagrangian multipliers λ_σ. Since ϑ is not restricted in sign, the dual of $\hat{\mathrm{M}}(R)$ forces

$$\sum_{\sigma\in R}\lambda_\sigma = 1. \tag{48}$$

It follows that, for this specific choice of λ, the optimal solution to $L(R,\lambda)$ is independent of ϑ, and hence all Lagrangian bounds can be obtained by solving the 01KP $\max \sum_{j=1}^{n}\hat{p}_j x_j : \sum_{j=1}^{n} w_j x_j \le c, x_j \in \{0,1\}$ $(j = 1,2,\ldots,n)$.

When the node is not fathomed by the Lagrangian bound, we invoke the separation procedure of Section 4.1 on an optimal solution $(\hat{x},\hat{\vartheta})$ to $\hat{\mathrm{M}}(R)$, in order to determine a violated constraint (if any), which is then added to R. The two attempts are iterated until either no violated constraint is found, or a pre-specified maximum number of attempts has been performed. Whenever the Lagrangian bound is computed, the actual value of the integer solution it provides (i.e., its regret with respect to all scenarios) is evaluated through Algorithm 1, and the incumbent solution is possibly updated. The overall node processing is outlined in Algorithm 3.

Note that the algorithm does not increase the iteration counter h when the solution $\hat{x}$ to $\hat{\mathrm{M}}(R)$ is integer. In such cases it is indeed convenient to keep separating: if (possibly after some iterations) no violated cut is found, we know that the solution is optimal for the current node, which can then be fathomed.

The cuts generated at each node are added to the model, which is passed to the descendant nodes, without storing them in a general pool. Hence, although such cuts are globally valid, they are removed from the model when backtracking, i.e., the same cut can be generated again at another node. Indeed, the separation procedure heavily depends on the solution obtained for the current node, and the computational experiments showed that cuts are rarely effective for non-descendant nodes.

The performance of Algorithm FIMY was improved through other enhancements, namely:

- when a node is generated by condition $x_j = 0$, we also set $x_k = 0$ for all subsequent items that are dominated by item j (see Property 4);

comment: $x_\star$ is the incumbent solution, and has value $r(x_\star)$;
$h := 1$;
repeat
  solve $\hat{\mathrm{M}}(R)$, getting a solution $(\hat{x}, \hat{\vartheta})$ of value LB^c, and a dual solution λ;
  if $LB^c \geq r(x_\star)$ **then** fathom the current node
  else
    solve the Lagrangian relaxation $L(R, \lambda)$, getting a solution x^λ of value LB^λ;
    if $LB^\lambda \geq r(x_\star)$ **then** fathom the current node
    else
      compute the regret $r(x^\lambda)$ (see Section 3.1), and possibly update $x_\star$ and $r(x_\star)$;
      look for a cut violated by $(\hat{x}, \hat{\vartheta})$ through the separation method of Section 4.1;
      if $\hat{x}$ is not integer **then** $h := h + 1$
    end if
  end if
until $h > h_{\max}$ **or** no violated cut has been found.

Algorithm 3: Processing at each decision node in FIMY.

- when starting the processing of a new node, say generated by condition $x_j = a$ ($a \in \{0, 1\}$), let $\hat{x}^p$ be the solution to $\hat{\mathrm{M}}(R)$ found by the parent node. If $\hat{x}^p_j = a$, then we avoid computing the solution of the new $\hat{\mathrm{M}}(R)$, and just "inherit" $\hat{x}^p$ in the first iteration of Algorithm 3;

- if the current residual capacity is not smaller than the total weight of the residual items plus the smallest weight of an item, say k, such that $p^-_k \geq 0$ and an ancestor node set $x_k = 0$, then the node can be fathomed, because the current solution is dominated by the twin node with $x_k = 1$. (Note indeed that two solutions x and x', only differing in $x_k = 1$ and $x'_k = 0$, satisfy, for any scenario $s \in S_0$, $z^s(x) - z^s(x') = p^s_k \geq p^-_k \geq 0$, and hence $r^s(x) - r^s(x') = (z^s_\star - z^s(x)) - (z^s_\star - z^s(x')) \leq 0$, i.e., $r(x) \leq r(x')$.)

Recall that λ_σ is not an optimal solution to the Lagrangian dual problem. Attempts to improve the Lagrangian bound LB^λ through subgradient optimization did not improve the computational performance of the algorithm. It indeed decreased the overall number of explored decision nodes, but at the expenses of an increased CPU time.

7 Computational experiments

We performed an extensive computational evaluation of the heuristic algorithms of Section 5, and of the exact algorithms of Sections 4 and 6. All algorithms were coded in C++, and run on a PC with a Pentium 4 at 3.2 GHz and 1 GB RAM memory, under Linux Ubuntu 11.

All the approaches require the solution to 01KP sub-instances, which were produced using algorithm `combo` by Martello, Pisinger and Toth [16] (a C code available at `http://www.diku.dk/hjemmesider/ansatte/pisinger/codes.html`). The values L and U needed by Algorithm 1 of Section 3.1, were obtained through the classical greedy algorithm for the 01KP (with items sorted by non-increasing p^σ_j/w_j values, see Section 5.1) and through the upper bound U_2 by Martello and Toth [17], respectively.

The ILP solutions needed by the Benders' decomposition (Section 4) and by the U-MRKP heuristic (Section 5.2) were obtained through Cplex 12.3 single thread. The branch-and-cut approach of Section 4 was implemented using the Cplex `callback` framework. In algorithm FIMY, Cplex was only used to solve the LP relaxations.

We randomly generated nine classes of MRKP instances by defining profit intervals for nine standard classes of 01KP instances from the literature, obtained through the generator used in [16] (available at the same URL as `combo`). For each class, two values were considered for a parameter $\overline{R}$: $\overline{R} = 1000$ and $\overline{R} = 10000$. The 01KP instance classes are (u.r. stands for "uniformly random integer"):

1. *Uncorrelated*: w_j u.r. in $[1, \overline{R}]$, p_j u.r. in $[1, \overline{R}]$.
2. *Weakly correlated*: w_j u.r. in $[1, \overline{R}]$, p_j u.r. in $[\max\{1, w_j - \overline{R}/10\}, w_j + \overline{R}/10]$.
3. *Strongly correlated*: w_j u.r. in $[1, \overline{R}]$, $p_j = w_j + \overline{R}/10$.
4. *Inverse strongly correlated*: p_j u.r. in $[1, \overline{R}]$, $w_j = p_j + \overline{R}/10$.
5. *Almost strongly correlated*: w_j u.r. in $[1, \overline{R}]$, p_j u.r. in $[w_j + \overline{R}/10 - \overline{R}/500, w_j + \overline{R}/10 + \overline{R}/500]$.
6. *Subset-sum*: w_j u.r. in $[1, \overline{R}]$, $p_j = w_j$.
7. *Even-odd subset-sum*: w_j even value u.r. in $[1, \overline{R}]$, $p_j = w_j$, c odd.
8. *Even-odd strongly correlated:* w_j even value u.r. in $[1, \overline{R}]$, $p_j = w_j + \overline{R}/10$, c odd.
9. *Uncorrelated with similar weights:* w_j u.r. in $[100\,\overline{R}, 100\,\overline{R} + \overline{R}/10]$, p_j u.r. in $[1, \overline{R}]$.

For each class and value of $\overline{R}$, we generated 27 MRKP instances through all combinations of

- number of items $n \in \{50, 60, 70\}$;
- capacity $c \in \{\lfloor 0.45W \rfloor, \lfloor 0.50W \rfloor, \lfloor 0.55W \rfloor\}$, with $W = \sum_{j=1}^{n} w_j$ (and c increased by 1, if even, for classes 7 and 8);
- profit interval $[p_j^-, p_j^+]$, with p_j^- u.r. in $[(1-\delta)p_j, p_j]$, p_j^+ u.r. in $[p_j, (1+\delta)p_j]$, and $\delta \in \{0.1, 0.2, 0.3\}$,

thus obtaining 486 MRKP instances. The whole set of instances is available on the Internet at `http://www.or.deis.unibo.it/research_pages/ORinstances/MRKP_instances.zip`. In the next section we present the computational outcome of the experiments.

7.1 Heuristic algorithms

In Table 1 we compare the different approaches described in Section 5. The first group of lines refers to the classical greedy algorithm, with the four sorting criteria discussed in Section 4.3. The second group of lines refers to the two ILP-based approaches of Section 5.2: branch-and-cut algorithm of Section 4.2 (B&Cut in the following) halted after 10 and 60 CPU seconds, respectively; ILP solution of model U-MRKP (equations (39)-(44)) halted after 10 CPU seconds. The final group of lines refers to the improvement of the initial solutions

through Algorithm 2 (iterated local search, ILS) of Section 5.3, with iteration limit $k_{\max}$ experimentally fixed to $n/10$.

For each class, the entries give the average percentage gap with respect to the best known solution (usually obtained through the exact algorithms discussed in the next section), computed over the 54 instances of the class. The final column provides the overall average percentage gap with respect to the whole set of instances.

The greedy algorithms performed in general very poorly (with some exceptions occurring for greedy$[(p_j^+ + p_j^-)/w_j]$, especially on Classes 6, 7 and 9). Much better results were provided by the ILP-based approaches: the truncated B&Cut had a decent performance, but U-MRKP outperformed it in all cases but two. An interesting fact is that such two cases occurred for the instances of Classes 1 and 9, which are relatively "easy". As we will see in the next section, all instances of these two classes are quickly solved to optimality by all exact algorithms. The U-MRKP too was solved to optimality within the given time limit, but the solution found was not optimal for the MRKP: this experimentally confirms the integrality gap introduced by (37).

The last two lines in Table 1 give the results obtained by improving the initial solutions through the ILS of Section 5.3. We selected the "winners" of the two initialization algorithms for the starting solution, namely greedy algorithm with the fourth sorting criterion and U-MRKP. The results are very satisfactory, especially for the latter, which provides, within one minute (including 10 seconds for U-MRKP), solutions with an overall average error below 0.04% with respect to the best known solution. In the following, we will denote this winning configuration as ILS*.

The solution values produced by ILS* are very close to the optimal solution values produced by the exact algorithms. However, apart from the easy instances of Classes 1 and 9, the number of solutions that are provably optimal by comparison with the lower bound given by B&Cut halted after 60 seconds was very low (only 73 over 378 instances). In the next section we will see that FIMY is the only algorithm capable of obtaining a high number of provably optimal solutions.

Table 1: Average percentage gaps from best known solutions by different heuristic algorithms.

	Class									
Algorithm	1	2	3	4	5	6	7	8	9	avg
Greedy heuristics										
greedy$[(p_j^+ p_j^-)/w_j]$	39.581	59.346	67.129	15.062	69.974	28.383	27.964	67.534	0.029	41.667
greedy$[p_j^-/w_j]$	21.271	15.966	18.609	17.146	23.369	10.983	10.348	20.018	10.160	16.430
greedy$[p_j^+/w_j]$	22.857	9.527	18.309	13.944	22.733	12.910	13.092	18.072	5.368	15.202
greedy$[(p_j^+ + p_j^-)/w_j]$	20.209	4.015	13.405	4.589	14.144	1.042	1.105	12.786	0.029	7.925
ILP-based heuristics										
B&Cut, 10ṡ	0.000	1.348	1.317	2.819	1.729	1.520	1.105	1.837	0.000	1.297
B&Cut, 60ṡ	0.000	1.167	1.148	2.086	1.266	1.196	0.885	1.266	0.000	1.002
U-MRKP, 10ṡ	0.262	0.300	0.044	0.073	0.281	0.046	0.041	0.147	0.005	0.133
Iterated Local Search										
greedy$[(p_j^+ + p_j^-)/w_j]$+ILS, 60ṡ	0.000	1.106	0.004	0.312	0.065	0.241	0.213	0.090	0.000	0.226
ILS* = U-MRKP + ILS, 60ṡ	0.000	0.167	0.000	0.000	0.065	0.035	0.012	0.066	0.000	0.038

7.2 Exact algorithms

In this section we report the computational experiments performed on the exact algorithms for the MRKP. We compared the two standard approaches of Section 4.2 (Benders' and B&Cut) and the proposed Lagrangian-based algorithm of Section 6 (FIMY) on the nine classes of instances previously described. Tables 2, 3 and 4 provide the results for Classes 1-3, 4-6, and 7-9, respectively. The iteration limit $h_{\max}$ of Algorithm 3 was experimentally fixed to 2.

In each table we consider all values of n and, for each n, all values of δ. The pairs of entries in each line refer to the six MRKP instances generated, for the corresponding n and δ values, by varying the data range $\overline{R}$ (2 values) and the capacity c (3 values). Each pair of entries provides the average CPU time (*sec*) expressed in seconds, and the total number of failures (*#f*), i.e., the total number of instances not solved to proven optimality. For each group of 3 lines (with an identical value of n), an additional row provides the average and total values over the 18 instances, while the final line of each table reports the overall average and total values over the 54 instances.

All algorithms received on input, as incumbent solution, the one produced by ILS* (last line in Table 1). Each algorithm had a time limit of 1 CPU hour per single instance. An entry 't.l.' in the tables indicates that the algorithm reached the time limit for *all* six instances. For the other cases, the time limits (if any) were included in the average CPU time computation.

The Benders' decomposition approach was always dominated by B&Cut, both for what concerns CPU time and number of failures. In turn, FIMY dominated B&Cut, with few irrelevant exception mostly concerning the very easy instances of Classes 1 and 9, which were solved by all algorithms within very short CPU times. The hardest instances were clearly those in Classes 6 and 7: Benders' and B&Cut could not solve a single instance over 108, while FIMY solved 42 of them to proven optimality. The remaining five classes (2, 3, 4, 5, and 8) were also very hard for Benders' and B&Cut, which solved only a small portion of the instances (65 and 128, respectively, out of 270), whereas FIMY solved the majority of them (241 out of 270), frequently within short CPU times. Overall, FIMY turns out to be, by far, the most effective algorithm for the exact solution of the MRKP.

In Table 5 we show how CPU times and numbers of failures vary when increasing one of the four parameters: number of items n (first group of lines), capacity c (second group), profit interval width δ (third group), and range $\overline{R}$ (fourth group). Each pair of entries refers in this case to all instances generated for the considered parameter value (162 instances for the first three groups, 243 instances for the last group). The overall average and total values (obviously the same for each group) are provided in the final line.

As expected, the difficulty sharply increases when n grows. The most difficult capacity value is $0.5\,W$, which confirms a known property of the 01KP: instances in which about half the items are in the optimal solution tend to be more difficult than instances with solutions containing many or few items. An increase in the value of δ produces instances that are much more difficult for Benders' and B&Cut, while its effect is less remarkable for FIMY. The results in the final group show that instances with weights in a larger range are considerably easier to solve, while the 01KP usually exhibits an opposite behavior: indeed, the maximum regret value turned out to be, on average, relatively smaller for $\overline{R} = 10000$ ($11181 \simeq 1.12\overline{R}$) than for $\overline{R} = 1000$ ($1550 = 1.55\overline{R}$), i.e., the worst scenario has, in the former case, a smaller relative deviation between the two involved 01KP solutions.

Table 2: Exact algorithms, classes 1–3.

		class 1						class 2						class 3					
		Benders'		B&Cut		FIMY		Benders'		B&Cut		FIMY		Benders'		B&Cut		FIMY	
n	δ	sec	*#f*	sec	*#f*	sec	*#f*	sec	*#f*	sec	*#f*	sec	*#f*	sec	*#f*	sec	*#f*	sec	*#f*
50	0.1	0.02	*0*	0.00	*0*	0.02	*0*	2.77	*0*	0.15	*0*	0.08	*0*	1813.68	*3*	43.34	*0*	3.22	*0*
0	0.2	0.05	*0*	0.02	*0*	0.02	*0*	1781.57	*2*	11.13	*0*	1.23	*0*	t.l.	*6*	1925.64	*3*	235.65	*0*
0	0.3	0.22	*0*	0.04	*0*	0.04	*0*	1959.39	*3*	572.02	*0*	2.88	*0*	t.l.	*6*	3133.34	*5*	195.17	*0*
avg/*tot*		0.10	*0*	0.02	*0*	0.03	*0*	1247.91	*5*	194.43	*0*	1.40	*0*	3004.56	*15*	1700.77	*8*	144.68	*0*
60	0.1	0.03	*0*	0.02	*0*	0.02	*0*	4.18	*0*	0.23	*0*	0.16	*0*	2097.59	*3*	1458.59	*2*	31.78	*0*
0	0.2	0.06	*0*	0.04	*0*	0.03	*0*	2325.62	*3*	94.65	*0*	4.63	*0*	t.l.	*6*	3298.05	*5*	1835.69	*3*
0	0.3	0.48	*0*	0.06	*0*	0.04	*0*	2598.95	*3*	1812.97	*3*	11.64	*0*	t.l.	*6*	t.l.	*6*	1906.21	*2*
avg/*tot*		0.19	*0*	0.04	*0*	0.03	*0*	1642.92	*6*	635.95	*3*	5.47	*0*	3099.19	*15*	2785.36	*13*	1257.89	*5*
70	0.1	0.04	*0*	0.01	*0*	0.03	*0*	44.65	*0*	0.66	*0*	0.21	*0*	3522.97	*5*	1832.37	*3*	108.50	*0*
0	0.2	0.09	*0*	0.03	*0*	0.03	*0*	t.l.	*6*	1495.90	*0*	10.88	*0*	t.l.	*6*	t.l.	*6*	2029.31	*3*
0	0.3	1.26	*0*	0.10	*0*	0.06	*0*	t.l.	*6*	2628.37	*3*	53.70	*0*	t.l.	*6*	t.l.	*6*	3504.50	*4*
avg/*tot*		0.46	*0*	0.04	*0*	0.04	*0*	2414.88	*12*	1374.98	*3*	21.60	*0*	3574.32	*17*	3010.52	*15*	1880.77	*7*
overall		0.25	*0*	0.03	*0*	0.03	*0*	1768.57	*23*	735.12	*6*	9.49	*0*	3226.02	*47*	2498.88	*36*	1094.45	*12*

Table 3: Exact algorithms, classes 4–6.

		class 4						class 5						class 6					
		Benders'		B&Cut		FIMY		Benders'		B&Cut		FIMY		Benders'		B&Cut		FIMY	
n	δ	sec	*#f*	sec	*#f*	sec	*#f*	sec	*#f*	sec	*#f*	sec	*#f*	sec	*#f*	sec	*#f*	sec	*#f*
50	0.1	1827.99	*3*	224.54	*0*	2.92	*0*	1873.67	*3*	505.06	*0*	7.54	*0*	t.l.	*6*	t.l.	*6*	252.69	*0*
0	0.2	3520.07	*5*	1816.14	*3*	81.96	*0*	2510.63	*4*	1587.63	*2*	55.54	*0*	t.l.	*6*	t.l.	*6*	1161.07	*0*
0	0.3	t.l.	*6*	3145.86	*5*	30.36	*0*	t.l.	*6*	2205.25	*3*	72.12	*0*	t.l.	*6*	t.l.	*6*	315.12	*0*
avg/*tot*		2982.69	*14*	1728.85	*8*	38.41	*0*	2661.43	*13*	1432.65	*5*	45.07	*0*	t.l.	*18*	t.l.	*18*	576.29	*0*
60	0.1	1979.22	*3*	1470.83	*2*	21.45	*0*	2140.18	*3*	1803.50	*3*	46.62	*0*	t.l.	*6*	t.l.	*6*	3409.07	*5*
0	0.2	t.l.	*6*	3054.10	*5*	1383.02	*0*	2499.71	*4*	1805.44	*3*	348.63	*0*	t.l.	*6*	t.l.	*6*	3235.28	*4*
0	0.3	t.l.	*6*	t.l.	*6*	411.16	*0*	t.l.	*6*	t.l.	*6*	703.82	*0*	t.l.	*6*	t.l.	*6*	t.l.	*6*
avg/*tot*		3059.74	*15*	2708.18	*13*	605.21	*0*	2746.63	*13*	2402.73	*12*	366.36	*0*	t.l.	*18*	t.l.	*18*	3414.75	*15*
70	0.1	2906.09	*4*	1824.26	*3*	60.38	*0*	t.l.	*6*	1859.17	*3*	112.26	*0*	t.l.	*6*	t.l.	*6*	t.l.	*6*
0	0.2	t.l.	*6*	t.l.	*6*	1849.82	*3*	t.l.	*6*	2565.37	*4*	1723.33	*2*	t.l.	*6*	t.l.	*6*	t.l.	*6*
0	0.3	t.l.	*6*	t.l.	*6*	1662.88	*0*	t.l.	*6*	t.l.	*6*	2662.27	*3*	t.l.	*6*	t.l.	*6*	t.l.	*6*
avg/*tot*		3368.70	*16*	3007.66	*15*	1191.03	*3*	t.l.	*18*	2674.73	*13*	1499.29	*5*	t.l.	*18*	t.l.	*18*	t.l.	*18*
overall		3137.04	*45*	2481.56	*36*	611.55	*3*	3002.69	*44*	2170.03	*30*	636.90	*5*	t.l.	*54*	t.l.	*54*	2530.32	*33*

Table 4: Exact algorithms, classes 7–9.

		class 7						class 8						class 9					
		Benders'		B&Cut		FIMY		Benders'		B&Cut		FIMY		Benders'		B&Cut		FIMY	
n	δ	sec	*#f*	sec	*#f*	sec	*#f*	sec	*#f*	sec	*#f*	sec	*#f*	sec	*#f*	sec	*#f*	sec	*#f*
50	0.1	t.l.	*6*	t.l.	*6*	234.40	*0*	1157.34	*0*	9.59	*0*	2.13	*0*	0.02	*0*	0.01	*0*	0.02	*0*
0	0.2	t.l.	*6*	t.l.	*6*	794.61	*0*	t.l.	*6*	1935.18	*3*	120.84	*0*	0.19	*0*	0.03	*0*	0.02	*0*
0	0.3	t.l.	*6*	t.l.	*6*	594.62	*0*	t.l.	*6*	2386.95	*3*	103.80	*0*	0.67	*0*	0.07	*0*	0.03	*0*
avg/*tot*		t.l.	*18*	t.l.	*18*	541.21	*0*	2785.78	*12*	1443.90	*6*	75.59	*0*	0.29	*0*	0.04	*0*	0.03	*0*
60	0.1	t.l.	*6*	t.l.	*6*	3088.63	*3*	2648.91	*4*	854.84	*1*	25.08	*0*	0.05	*0*	0.01	*0*	0.03	*0*
0	0.2	t.l.	*6*	t.l.	*6*	t.l.	*6*	t.l.	*6*	t.l.	*6*	1834.65	*3*	0.40	*0*	0.07	*0*	0.04	*0*
0	0.3	t.l.	*6*	t.l.	*6*	t.l.	*6*	t.l.	*6*	t.l.	*6*	900.30	*0*	3.69	*0*	0.15	*0*	0.07	*0*
avg/*tot*		t.l.	*18*	t.l.	*18*	3429.48	*15*	3282.97	*16*	2684.47	*13*	920.01	*3*	1.38	*0*	0.08	*0*	0.05	*0*
70	0.1	t.l.	*6*	t.l.	*6*	t.l.	*6*	t.l.	*6*	1887.72	*3*	63.11	*0*	0.05	*0*	0.02	*0*	0.03	*0*
0	0.2	t.l.	*6*	t.l.	*6*	t.l.	*6*	t.l.	*6*	t.l.	*6*	1997.70	*3*	0.39	*0*	0.06	*0*	0.03	*0*
0	0.3	t.l.	*6*	t.l.	*6*	t.l.	*6*	t.l.	*6*	t.l.	*6*	2620.39	*3*	3.46	*0*	0.15	*0*	0.06	*0*
avg/*tot*		t.l.	*18*	t.l.	*18*	t.l.	*18*	t.l.	*18*	3028.96	*15*	1560.40	*6*	1.30	*0*	0.08	*0*	0.04	*0*
overall		t.l.	*54*	t.l.	*54*	2523.53	*33*	3222.91	*46*	2385.78	*34*	852.00	*9*	0.99	*0*	0.06	*0*	0.04	*0*

Table 5: Exact algorithms, summary.

		Benders'		B&Cut		FIMY	
		sec	*#f*	sec	*#f*	sec	*#f*
n	50	2209.19	*95*	1522.18	*63*	158.08	*0*
	60	2337.00	*101*	2046.22	*90*	1111.03	*38*
	70	2639.96	*117*	2255.13	*97*	1483.66	*57*
c	$0.45\,W$	2407.00	*105*	1898.24	*81*	884.78	*30*
	$0.50\,W$	2366.93	*102*	1967.44	*85*	957.56	*34*
	$0.55\,W$	2412.23	*106*	1957.85	*84*	910.43	*31*
δ	0.1	1882.20	*79*	1310.08	*56*	543.34	*20*
	0.2	2601.43	*114*	2058.68	*88*	1092.73	*39*
	0.3	2702.52	*120*	2454.77	*106*	1116.70	*36*
$\overline{R}$	1000	2633.05	*175*	2307.74	*150*	1149.38	*64*
	10000	2157.73	*138*	1574.61	*100*	685.80	*31*
avg/*tot*		2395.39	*313*	1941.18	*250*	917.59	*95*

8 Conclusions

If an investor wants to determine the best way to invest her/his capital by selecting among a number of financial products, each requiring a given capital and ensuring a fixed return, then she/he faces the well-known 0-1 knapsack problem. Despite the theoretical difficulty of this problem, modern combinatorial algorithms solve knapsack instances with thousands of items within seconds. In this paper we studied a robust version of the knapsack problem, in which the return of a product does not take a fixed value but varies in a given interval, and the aim of the investor is to minimize the maximum regret of her/his investment. The resulting problem is known as the interval min-max regret knapsack problem. This problem is not only difficult from a theoretical point of view (to an extent which is still topic of research), but also very challenging in practice.

If a heuristic solution is to be found in a short time, then the investor should not use standard greedy techniques, because this could easily lead to gaps, from the best possible regret, ranging between 10 and 40%. We showed that the use of more elaborate ILP-based heuristics improves such gap, but still there are instances in which these algorithms behave poorly. In order to obtain good results, it is necessary to improve on the heuristic solutions through metaheuristic refinements, such as the iterated local search we introduced. We also studied the case in which a longer computation time is allowed, and the aim is to find a proven optimal regret. In such a case, the exact techniques normally used for this kind of robust problems, such as Benders' decomposition and branch-and-cut, show a poor performance. Through theoretical findings and algorithmic developments, we were able to solve to proven optimality all instances with up to 50 investments, and to outperform the standard approaches for larger instances.

References

[1] H. Aissi, C. Bazgan, and D. Vanderpooten. Minmax and minmax regret versions of combinatorial optimization problems: A survey. *European Journal of Operational Research*, 197:427–438, 2009.

[2] H. Aissi, C. Bazgan, and D. Vanderpooten. General approximation schemes for minmax (regret) versions of some (pseudo-)polynomial problems. *Discrete Optimization*, 7:136–148, 2010.

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

[4] A. Ben-Tal and A. Nemirovski. Robust solution of linear programming problems contaminated with uncertain data. *Mathematical Programming*, 88:411–4124, 2000.

[5] J.F. Benders. Partitioning procedures for solving mixed integer variables programming problems. *Numerische Mathematik*, 4:238–252, 1962.

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

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

[8] J. Birge and F. Louveaux. *Introduction to Stochastic Programming*. Springer Series in Operations Research and Financial Engineering. Springer, Berlin, 2nd edition, 2011.

[9] A. Candia-Véjar, E. Álvarez-Miranda, and N. Maculan. Min-max regret combinatorial optimization problems: An algorithmic perspective. *RAIRO - Operations Research*, 45:101–129, 2011.

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

[11] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W.H. Freeman, San Francisco, 1979.

[12] A. Kasperski. *Discrete optimization with interval data*. Springer, Berlin, 2008.

[13] A.O. Kazakci, S. Rozakis, and D. Vanderpooten. Energy crop supply in France: A minmax regret approach. *Journal of the Operational Research Society*, 58:1470–1479, 2007.

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

[15] P. Kouvelis and G. Yu. *Robust discrete optimization and its applications*. Kluwer AP, Dordrecht, 1997.

[16] S. Martello, D. Pisinger, and P. Toth. Dynamic programming and strong bounds for the 0-1 knapsack problem. *Management Science*, 45:414–424, 1999.

[17] S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons, Chichester, 1990.

[18] S. Michenaudc and B. Solnik. Applying regret theory to investment choices: Currency hedging decisions. *Journal of International Money and Finance*, 27:677–694, 2008.

[19] R. Montemanni, J. Barta, M. Mastrolilli, and L.M. Gambardella. The robust traveling salesman problem with interval data. *Transportation Science*, 41:366–381, 2011.

[20] R. Montemanni and L.M. Gambardella. The robust shortest path problem with interval data via Benders decomposition. *4OR: A Quarterly Journal of Operations Research*, 3:315–328, 2005.

[21] C.H. Papadimitriou. *Computational Complexity*. Addison-Wesley, Reading, MA, 1994.

[22] J. Pereira and I. Averbakh. Exact and heuristic algorithms for the interval data robust assignment problem. *Computers & Operations Research*, 38:1153–1163, 2011.

[23] J. Pereira and I. Averbakh. The robust set covering problem with interval data. *Annals of Operations Research*, 207:217–235, 2013.

[24] A. Shapiro, D. Dentcheva, and A. Ruszczyński. *Lectures on Stochastic Programming: Modeling and Theory*. MOS-SIAM Series on Optimization. SIAM, Philadelphia, PA, 2009.

[25] H. Yaman, O.E. Karaşan, and M.C. Pinar. The robust spanning tree problem with interval data. *Operations Research Letters*, 29:31–40, 2001.

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