POLITECNICO DI TORINO
Repository ISTITUZIONALE

An exact approach for the bilevel knapsack problem with interdiction constraints and extensions

Original
An exact approach for the bilevel knapsack problem with interdiction constraints and extensions / Della Croce, F., Scatamacchia, R.. - In: MATHEMATICAL PROGRAMMING. - ISSN 0025-5610. - 183:1-2(2020), pp. 249-281. [10.1007/s10107-020-01482-5]

Availability:
This version is available at: 11583/2876215 since: 2021-03-24T12:53:05Z

Publisher:
Springer

Published
DOI:10.1007/s10107-020-01482-5

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
Springer postprint/Author's Accepted Manuscript

This version of the article has been accepted for publication, after peer review (when applicable) and is subject to Springer Nature's AM terms of use, but is not the Version of Record and does not reflect post-acceptance improvements, or any corrections. The Version of Record is available online at: http://dx.doi.org/10.1007/s10107-020-01482-5

(Article begins on next page)

15 October 2026

Noname manuscript No.
(will be inserted by the editor)

An exact approach for the Bilevel Knapsack Problem with Interdiction Constraints and extensions

Federico Della Croce · Rosario Scatamacchia

the date of receipt and acceptance should be inserted later

Abstract We consider the Bilevel Knapsack Problem with Interdiction Constraints, an extension of the classic 0-1 knapsack problem formulated as a Stackelberg game with two agents, a leader and a follower, that choose items from a common set and hold their own private knapsacks. First, the leader selects some items to be interdicted for the follower while satisfying a capacity constraint. Then the follower packs a set of the remaining items according to his knapsack constraint in order to maximize the profits. The goal of the leader is to minimize the follower's total profit. We derive effective lower bounds for the Bilevel Knapsack Problem and present an exact method that exploits the structure of the induced follower's problem. The approach strongly outperforms the current state-of-the-art algorithms designed for the problem. We extend the same algorithmic framework to the Interval Min-Max Regret Knapsack Problem after providing a novel bilevel programming reformulation. Also for this problem, the proposed approach outperforms the exact algorithms available in the literature.

Keywords Bilevel programming, Exact approach, Bilevel Knapsack with Interdiction Constraints, Min-Max Regret Knapsack Problem

1 Introduction

Recently, growing attention has been centered to multilevel programming. This emerging field considers optimization problems with a hierarchal structure where many decision makers sequentially operate to reach conflicting objectives. Each agent takes decisions that may affect objectives and decisions of the agents at

Federico Della Croce ✉
1) Dipartimento di Ingegneria Gestionale e della Produzione, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Torino, Italy. 2) CNR, IEIIT, Torino, Italy.
E-mail: federico.dellacroce@polito.it

Rosario Scatamacchia
Dipartimento di Ingegneria Gestionale e della Produzione, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Torino, Italy.
E-mail: rosario.scatamacchia@polito.it

lower levels. At the same time, the latter decisions impact on the objectives of the agents at upper levels. Hierarchal contexts arise in many real-life applications in supply chains, energy sector, logistics and telecommunication networks among others. The presence of many decision levels makes these problems very challenging to solve.
The most relevant research in the field has been pursued for bilevel optimization where two agents, denoted as a leader and a follower, play a Stackelberg game ([34]). In this game, the leader takes the first decision and then the follower reacts taking into account the leader's strategy. Eventually, the agents receive a pay-off which depends on both leader's and follower's choices. The goal is typically to find a strategy for the leader that optimizes his own objective. Two standard assumptions are considered in a Stackelberg game: *complete information*, that is each agent knows the problem solved by the other agent; *rationale behavior*, namely each agent has no interest in deviating from his own objective.
Bilevel optimization considers Mixed-Integer Bilevel Linear Programs (MIBLP) where both the leader and the follower solve a combinatorial optimization problem with linear objective function and constraints and with either continuous or integer variables. The first generic Branch and Bound approach for MIBLP was provided in [27]. Branch and Cut schemes were introduced in [15], [14]. Further approaches were proposed in [5], [18] , [37] and [36]. An improved generic MIBLP solver has been proposed in [16]. We refer to [16] and the references therein for an overview on MIBLP solvers and related applications.

In this paper, we consider the Bilevel Knapsack with Interdiction Constraints (BKP), as introduced in [14]. The problem is an extension of the classic 0-1 Knapsack Problem (KP) (see monographs [23] and [26]) to a Stackelberg game where the leader and the follower choose items from a common set and hold their own private knapsacks. First, the leader selects some items to be interdicted for the follower while satisfying a capacity constraint. Then the follower packs a set of the remaining items according to his knapsack constraint in order to maximize the profits. The goal of the leader is to minimize the follower's total profit.
In [3] it is shown that BKP is Σ_2^p-complete in the polynomial hierarchy complexity. Essentially, BKP cannot be formulated as an ILP model of polynomial size unless the polynomial hierarchy collapses (also pointed out in [4]). This makes the problem even more difficult to solve than an NP-Complete problem. We refer to [22] for an introduction on polynomial hierarchy.
One of the best performing algorithms for BKP is given in [4]. The algorithm, denoted as CCLW, relies on the dualization of the continuous relaxation of the follower's problem and on iteratively computing upper bounds for the problem until a stopping criterion applies. The approach is motivated by the lack of significant lower bounds for the problem. Algorithm CCLW solves to optimality instances with 50 items within a CPU time limit of 3600 s, running out of time in instances with 55 items only. Another previous exact solution approach was proposed in [35]. Very recently, an improved branch-and-cut algorithm has been given in [17]. The proposed approach manages to solve to optimality all benchmark instances in [4], requiring at most a computation time of about 85 s in an instance with 55 items. The algorithm in [17] was shown to be superior to the other approaches also on the medium size instances with up to 50 items considered in [14] and [35]. We also

mention the work of [19] where a heuristic approach is proposed for BKP and for other interdiction games.
Other bilevel knapsack problems have been tackled in the literature. In [2] the leader cannot interdict items but modifies the follower's capacity. In [7], the leader can modify the follower's objective function only. In [33], a variant of the problem is considered in which an item can be taken by both the leader and the follower, in which case its profit changes (either increasing or decreasing). As discussed in [4], these knapsack problems are easier to handle than BKP. A polynomial algorithm has been provided in [6] for the BKP variation where the follower solves a continuous knapsack problem. Finally, a bilevel knapsack problem where the leader controls the weights of a subset of the follower's items has been recently tackled in [28].

Our contribution for BKP is twofold. First, we derive effective lower bounds based on mathematical programming. Second, we present a new exact approach that exploits the induced follower's problem and the derived lower bounds. The proposed approach shows up to be very effective successfully solving all benchmark literature instances provided in [4,15,35] within few seconds of computation. Our algorithm manages to solve to optimality larger instances generated according to the same generation scheme of [4] with up to 500 items requiring in the worst-case instance less than 14 seconds of CPU time. An extended computational campaign is also applied to the instances considered in [19] reaching very good results. A preliminary conference version of this work appeared in [13].
Further, we managed to extend the proposed approach to the Interval Min-Max Regret Knapsack Problem (MRKP), as introduced in [24]. This problem is a generalization of the KP where the profit of each item ranges between a minimum and a maximum value. A given assignment of the items profit levels defines a scenario which corresponds to a standard knapsack instance. The problem calls for finding a feasible solution that minimizes the maximum regret over all possible scenarios, where the regret represents the difference between the optimal solution value of a scenario and the value given by the selected solution. The authors of [9] show that the decision version of the MRKP is Σ_2^p-complete and thus very challenging to solve. In [20] different heuristic and exact algorithms are proposed for the problem based on classical optimization approaches such as Benders decomposition and branch and cut methods. We propose a bilevel programming reformulation of the MRKP and correspondingly apply the algorithmic framework proposed for BKP with proper integrations. The approach significantly outperforms the best performing exact algorithms given in [20].

2 Notation and problem formulation

In BKP a set of n items and two knapsacks are given. Each item i $(= 1, \ldots, n)$ has associated a profit $p_i > 0$ and a weight $w_i > 0$ for the follower's knapsack and a weight $v_i > 0$ for the leader's knapsack. Leader and follower have different knapsack capacities denoted by C_u and C_l, respectively. Quantities p_i, v_i, w_i $(i = 1, \ldots, n)$, C_u, C_l are assumed to be integer, with $v_i \leq C_u$ and $w_i \leq C_l$ for all i. To avoid trivial instances, it is also assumed that $\sum_{i=1}^{n} v_i > C_u$ and $\sum_{i=1}^{n} w_i > C_l$. We introduce

0/1 variables x_i $(i = 1, \ldots, n)$ equal to one if the leader selects item i and 0/1 variables y_i equal to one if item i is chosen by the follower. BKP can be modeled as follows:

$$\min \quad \sum_{i=1}^{n} p_i y_i \tag{1}$$

$$\text{subject to} \quad \sum_{i=1}^{n} v_i x_i \leq C_u \tag{2}$$

$$x_i \in \{0,1\} \qquad i = 1, \ldots, n \tag{3}$$

where $y_1, \ldots, y_n$ solve

$$\text{the follower's problem:} \quad \max \quad \sum_{i=1}^{n} p_i y_i \tag{4}$$

$$\text{subject to} \quad \sum_{i=1}^{n} w_i y_i \leq C_l \tag{5}$$

$$y_i \leq 1 - x_i \qquad i = 1, \ldots, n \tag{6}$$

$$y_i \in \{0,1\} \qquad i = 1, \ldots, n \tag{7}$$

The leader's objective function (1) minimizes the profits of the follower through the interdiction constraints (6). These constraints ensure that each item i can be selected by the follower only if the item is not interdicted by the leader, i.e., if $x_i = 0$. Constraint (2) represents the leader's capacity constraint. The objective function (4) maximizes the follower's total profit and constraint (5) represents the follower's capacity constraint. Constraints (3) and (7) define the domain of the variables.

The optimal solution value of model (1)-(7) is denoted by z^*. The optimal solution vectors of variables x_i and y_i are respectively denoted by x^* and y^*. Notice that in model (1)-(7) there always exists an optimal solution for the leader which is maximal, namely where items are included in the leader's knapsack as long as there is enough capacity left.

Let us now recall the optimal solution of the continuous relaxation of a standard KP, namely the follower's model (4)-(7) without constraints (6) and constraints (7) replaced by inclusion in $[0, 1]$. Under the assumption $\sum_{i=1}^{n} w_i > C_l$, this solution has the following structure. Consider the sorting of the items by non-increasing ratios of profits over follower's weights:

$$\frac{p_1}{w_1} \geq \frac{p_2}{w_2} \geq \cdots \geq \frac{p_n}{w_n}. \tag{8}$$

According to this order, items $j = 1, 2, \ldots$ are inserted into the knapsack as long as $\sum_{k=1}^{j} w_k \leq C_l$. The first item s which cannot be fully packed is commonly denoted in the knapsack literature as the *split* item (or *break/critical* item). The optimal solution of the KP linear relaxation is given by setting $y_j = 1$ for $j = 1, \ldots, s-1$, $y_j = 0$ for $j = s+1, \ldots, n$ and $y_s = (C_l - \sum_{j=1}^{s-1} w_j)/w_s$. The solution with items

$1, \ldots, (s-1)$ is a feasible solution for KP and is commonly denoted as the *split solution.*

In the remainder of the paper, we assume the ordering of the items (8). We denote by $KP(x)$ the follower's knapsack problem induced by a leader's strategy encoded in vector x, i.e. a knapsack problem with item set

$$S := \{i : x_i = 0, x_i \in x\}.$$

We also denote by $KP^{LP}(x)$ the corresponding Linear Programming (LP) relaxation. If $\sum_{i \in S} w_i > C_l$, we define the *critical* item c of $KP^{LP}(x)$ as the last item with a strictly positive value in its optimal solution. Thus, we have $y_c \in (0, 1]$ and a corresponding split solution with profit

$$\sum_{i \in S : i < c} p_i = \sum_{i=1}^{c-1} p_i (1 - x_i) \tag{9}$$

which constitutes a feasible solution for $KP(x)$. Notice that we denote by $z(M)$ the optimal solution value of any given mathematical model M.

3 Computing lower bounds on BKP

Consider the optimal solution vector x^*. In the induced follower's knapsack problem $KP(x^*)$ with item set S, two cases may occur: either there is no critical item in $KP^{LP}(x^*)$, namely $\sum_{i \in S} w_i \leq C_l$, or one critical item exists, namely $\sum_{i \in S} w_i > C_l$. The first case can be easily handled by considering that the follower will pack all items not interdicted by the leader. This case is discussed in Section 4.2.1.
In the second case, we derive effective lower bounds on BKP that constitute the main ingredient of the exact approach presented in Section 4. Since we don't know a priori the leader's optimal solution x^*, we proceed by guessing the critical item of $KP^{LP}(x^*)$, namely we formulate an Integer Linear Programming (ILP) model where we impose that a given item c must be critical and evaluate the profit of the corresponding split solution in the objective function. We consider binary variables k_h $(h = 1, \ldots, w_c)$ associated with the weight contribution of the critical item and introduce the following model (denoted as $CRIT_1(c)$).

$CRIT_1(c)$:

$$\min \quad \sum_{i=1}^{c-1} p_i(1-x_i) \tag{10}$$

$$\text{subject to} \quad \sum_{i=1}^{n} v_i x_i \leq C_u \tag{11}$$

$$\sum_{i=1}^{c-1} w_i(1-x_i) + \sum_{h=1}^{w_c} h k_h = C_l \tag{12}$$

$$\sum_{h=1}^{w_c} k_h = 1 \tag{13}$$

$$x_c = 0 \tag{14}$$

$$x_i \in \{0,1\} \qquad i = 1,\ldots,n \tag{15}$$

$$k_h \in \{0,1\} \qquad h = 1,\ldots,w_c \tag{16}$$

The objective function (10) minimizes the value of the split solution. Constraint (11) represents the leader's capacity constraint. Constraints (12) and (13) ensure that item c is critical as it is the last item packed, with a weight in the interval $[1, w_c]$. Constraint (14) indicates that item c can be critical only if it is not interdicted by the leader. Constraints (15) and (16) indicate that all variables are binary. Admittedly, instead of using w_c binary variables k_h, an alternative model could be obtained by using a single integer positive variable in constraint (12). For sake of exposition, this issue is discussed at the end of this section.

We can state the following proposition.

Proposition 1 *If c is the critical item in $KP^{LP}(x^*)$, then $z(CRIT_1(c))$ is a valid lower bound on z^*.*

Proof If c is the critical item in $KP^{LP}(x^*)$, an optimal BKP solution x^* constitutes a feasible solution for model $CRIT_1(c)$. Let denote by z_1 the solution value of the split solution in $KP(x^*)$. Since the follower maximizes the profits in $KP(x^*)$, the value of z^* is greater than (or equal to) the one of the related split solution, that is $z_1 \leq z^*$. But then, as model $CRIT_1(c)$ searches for an interdiction of items by the leader inducing a minimum cost split solution (among all feasible split solutions for a given critical item c), we get $z(CRIT_1(c)) \leq z_1 \leq z^*$. □

The previous proposition already provides a first significant lower bound for the problem. However, following the reasoning in the proof of Proposition 1, we remark that improved bounds on z^* can be derived by considering any feasible solution for $KP(x^*)$ that might be obtained by removing (adding) items that were not interdicted by the leader and that were selected (not selected) by the split solution, provided that the follower capacity is not exceeded. Indeed, this corresponds to considering only items i not interdicted by the leader and to removing tuples of items $i \in [1, c-1]$ and/or to adding tuples of items $i \in [c, n]$ from/to the split solution without exceeding the follower's capacity.

Notice that, the state-of-the-art algorithms for KP, *Minknap* ([31]) and *Combo* ([25]) consider that in general only few items with ratio p_i/w_i close to that of the critical item change their values in an optimal solution with respect to the

values taken in the split solution. These items constitute the so-called *core* of the knapsack. *Minknap* and *Combo* start with the computation of the split solution and define a core initialized with the critical item only. Then, the algorithms iteratively enlarge the core by evaluating both the removal of items from the split solution and the addition of items after the critical item. The empirical evidence illustrates that an optimal (or close to be optimal) KP solution is typically found after few iterations.

We cannot precisely characterize the features of these exact algorithms by a set of constraints within an ILP model, but we can mimic the same algorithmic reasoning by considering subsets of items $c-\Delta, ..., c+\Delta$ provided that these items are not interdicted by the leader. In each subset, the items $i : i \leq c-1$ are removed from the split solution, while the items $j : j \geq c$ are added to the solution. Whenever this operation does not violate the capacity constraint, a new feasible solution is reached. Correspondingly, the initial profit and weight of the split solution are modified by subtracting the profits and the weights of the removed items and by summing up the profits and the weights of the added items.

Then, for any given subset τ of items $c-\Delta, ..., c+\Delta$, let p^τ and w^τ be the related profit and weight contributions, namely:

$$p^\tau = -\sum_{i\in\tau: i<c} p_i + \sum_{j\in\tau: j\geq c} p_j; \tag{17}$$

$$w^\tau = -\sum_{i\in\tau: i<c} w_i + \sum_{j\in\tau: j\geq c} w_j. \tag{18}$$

A subset τ with $p^\tau \leq 0$ is not considered since it does not improve upon the split solution (whatever is the split solution originated by the relevant ILP model) associated with the critical item c. Instead, an improving subset with $p^\tau > 0$ is of interest only if it is feasible. Feasibility is obtained if w^τ does not exceed the residual capacity, that is if $w^\tau \leq \sum_{h=1}^{w_c} hk_h$ which implies $\sum_{h=w^\tau}^{w_c} k_h = 1$. Also, only items not interdicted by the leader can be considered, that is $x_i = 0\ \forall i \in \tau$ must hold. Correspondingly, an improving subset τ is detected inducing an additional profit p^τ, if and only if the following expression holds:

$$\sum_{h=w^\tau}^{w_c} k_h - \sum_{i\in\tau} x_i = 1. \tag{19}$$

A new model can then be generated by introducing a non-negative variable π that carries the maximum additional profit to the split solution value provided by any subset τ and related additional constraints of the type

$$\pi \geq p^\tau \Big(\sum_{h=w^\tau}^{w_c} k_h - \sum_{i\in\tau} x_i \Big). \tag{20}$$

Notice that the right-hand side of (20) is not larger than 0 whenever $\sum_{h=w^\tau}^{w_c} k_h = 0$ or $\sum_{i\in\tau} x_i \geq 1$. Correspondingly, it equals p^τ if and only if expression (19) holds.

We can then consider a set of constraints (each corresponding to a subset of items), denoted (with a little abuse of notation) as $\mathcal{F}(\pi, x, k, \tau)$, linking variable

π to variables x_i and k_h for each considered subset τ. The model (denoted as $CRIT_2(c)$) is as follows.

$CRIT_2(c)$:

$$\min \quad \sum_{i=1}^{c-1} p_i(1-x_i) + \pi \tag{21}$$

$$\text{subject to} \quad \mathcal{F}(\pi, x, k, \tau) \tag{22}$$

$$(11) - (16)$$

$$\pi \geq 0 \tag{23}$$

Due to the addition of any set of constraints $\mathcal{F}(\pi, x, k, \tau)$, for every c we have $z(CRIT_1(c)) \leq z(CRIT_2(c))$ as in $CRIT_2(c)$ there are more constraints and the objective function contains an additional positive term π. Notice that, every additional constraint contains only items not interdicted by the leader (I) and does not violate the follower's capacity (II). Every set $\mathcal{F}(\pi, x, k, \tau)$ of constraints satisfying both requirements (I) and (II) is denoted as *proper*. Once set $\mathcal{F}(\pi, x, k, \tau)$ is built, variable π represents in the objective function the largest profit induced by the tuples of items that can be added to the profit of the split solution.

Proposition 2 *If $KP^{LP}(x^*)$ admits a critical item c and model $CRIT_2(c)$ has a proper set $\mathcal{F}(\pi, x, k, \tau)$, then $z(CRIT_2(c)) \leq z^*$.*

Proof Since model $CRIT_2(c)$ considers feasible solutions for $KP(x^*)$, the inequality holds by applying the same argument of Proposition 1. □

We remark that models $CRIT_1(c)$ and $CRIT_2(c)$ contain a pseudo polynomial number of binary variables k_h depending on the magnitude of the follower's weights. Hence, the hardness of these ILP models may increase with the size increase of such input entries. At the same time, a different handling of models $CRIT_1(c)$ and $CRIT_2(c)$ is possible in presence of large follower's weights. We can avoid to use a pseudo polynomial number of variables by replacing term $\sum_{h=1}^{w_c} hk_h$ in constraint (12) with one integer variable and by introducing a binary variable and two constraints for each tuple considered in $\mathcal{F}(\pi, x, k, \tau)$. For the sake of exposition, we show this alternative treatment of the ILP models only for the MRKP in Section 8.2. For BKP, after computational tests on benchmark instances, we noticed that the use of a pseudo polynomial number of variables k_h appears to be more effective. This is because the data generation from the literature chooses rather small follower's weights in the interval $[1, 100]$ (see Section 5) thus limiting the number of variables k_h in models $CRIT_1(c)$ and $CRIT_2(c)$.

4 A new exact approach for BKP

4.1 Overview

We propose an exact algorithm for BKP that considers the possible existence of a critical item in $KP^{LP}(x^*)$ and exploits the bounds provided by model $CRIT_2(c)$.

The key idea of the algorithm is to compute appropriate leader's solutions by exploring the most promising subproblems of the follower in terms of lower bounds. This strategy considerably speeds up in practice both the identification and certification of an optimal interdiction structure. The approach involves two main steps. In the first step, the possible non-existence of a critical item is first evaluated. Then, the approach assumes the existence of a critical item and identifies a set of possible candidate items. For each candidate item c and value of a parameter Δ (that controls the core size), model $CRIT_2(c)$ is built by considering several subsets of items and related additional constraints (20). Then the linear relaxation $CRIT_2^{LP}(c)$ is solved, where the integrality constraints (15) and (16) are replaced by inclusion in $[0, 1]$.

The feasible problems $CRIT_2^{LP}(c)$ are sorted by increasing optimal value so as to identify an order of the most promising subproblems to explore. A limited number of feasible BKP solutions is also computed in this step.

In the second step, each relevant subproblem is explored by constraint generation until the subproblem can be pruned. An optimal BKP solution is eventually returned. The approach takes as input six parameters α, β, Δ, μ, γ, ω and relies on an ILP solver along its steps. We discuss the steps of the algorithm in the following. The corresponding pseudo code is then provided.

4.2 Step 1

4.2.1 Handling the possible non-existence of a critical item

We first consider the case where there does not exist a critical item in $KP^{LP}(x^*)$. Thus, the follower will select all available items which are not interdicted by the leader and an optimal solution of BKP is found by solving the following problem NCR.

NCR:

$$\min \quad \sum_{i=1}^{n} p_i(1 - x_i) \tag{24}$$

$$\text{subject to} \quad \sum_{i=1}^{n} v_i x_i \leq C_u \tag{25}$$

$$\sum_{i=1}^{n} w_i(1 - x_i) \leq C_l \tag{26}$$

$$x_i \in \{0, 1\} \qquad i = 1, \ldots, n \tag{27}$$

If problem NCR is feasible, let denote by x' the related optimal solution representing the leader's strategy. The corresponding follower's solution is denoted by y', with $y'_i = 1 - x'_i$ $(i = 1, \ldots, n)$. The current best solution (x^*, y^*) with value z^* (which will be optimal at the end of the algorithm) is initialized accordingly (Lines 3-4 of the pseudo code).

4.2.2 Identifying the relevant critical items

We now assume that there exists a critical item c in $KP^{LP}(x^*)$ (Lines 5-13) and estimate the first and last possible items q and r that can be critical according to ordering (8). For item q we have

$$q := \min\{j : \sum_{i=1}^{j} w_i \geq C_l\}. \tag{28}$$

All items $1, \ldots, (q-1)$ cannot in fact be critical even without the leader's interdiction. For the last item r, we first compute the maximum weight of the follower that can be interdicted by the leader (similarly to [4]) by solving the following problem (denoted by LW).

LW :

$$\max \quad \sum_{i=1}^{n} w_i x_i \tag{29}$$

$$\text{subject to} \quad \sum_{i=1}^{n} v_i x_i \leq C_u \tag{30}$$

$$x_i \in \{0,1\} \qquad i = 1, \ldots, n \tag{31}$$

Item r is defined as

$$r := \min\{j : \sum_{i=1}^{j} w_i \geq C_l + z(LW)\}. \tag{32}$$

Since from (32) we have $\sum_{i=1}^{r} w_i(1-x_i) \geq C_l$ for any leader's strategy, all items from $(r+1)$ to n cannot be critical.

4.2.3 Building models $CRIT_2(c)$

For each candidate critical item $c \in [q, r]$, we formulate model $CRIT_2(c)$ by constructing a proper set $\mathcal{F}(\pi, x, k, \tau)$ as follows. Consider the subsets involving items in the interval $[c-\Delta, c+\Delta]$. Even for small value of Δ, the number of subsets can be very large. Hence, in order to limit the number of constraints in $\mathcal{F}(\pi, x, k, \tau)$, we propose a different strategy that greedily selects the subsets according to the procedure denoted as $DefineTuples$ and sketched below. We remark that the adopted strategy slightly differs from the $ComputeTuples$ procedure we used in [13] and contributes to further improving the performance of the proposed approach (see Section 5).

For a given value of Δ, we consider the interval of items $[a, b]$, with $a = \max\{1; c-\Delta\}$ and $b = \min\{c+\Delta; n\}$. Starting with the empty set, we enumerate all "backward" sets with at most α items among items $a, \ldots, (c-1)$. Each set has a profit and weight equal to the sum of profits and weights of the included items. We also compute all "forward" sets with items $c, \ldots, b$ and at most β items. Then we combine each backward set with a forward set and generate a tuple τ. If

$p^\tau > 0$ and $w^\tau \leq w_c$, we store the tuple for possible insertion in set $\mathcal{F}(\pi, x, k, \tau)$. Then we sort all the stored tuples by nondecreasing cardinality with ties broken in favor of tuples with a higher ratio p^τ/w^τ. The rationale is to fill set $\mathcal{F}(\pi, x, k, \tau)$ by considering tuples with a limited number of items (and so expected to be more difficult to interdict) while ensuring the insertion of tuples with high efficiency in terms of ratio profit over weight. Following the ordering of the tuples, we add the related constraints (20) to $\mathcal{F}(\pi, x, k, \tau)$ until their number is larger than an input parameter μ. If not previously included, we also add to set $\mathcal{F}(\pi, x, k, \tau)$ the constraint $\pi \geq p_c k_{w_c}$ which handles the possible adding of the critical item to the split solution if the residual capacity is equal to w_c.

DefineTuples(c, α, β, Δ, μ)

1: Consider items in the interval $[a, b]$ with $a := \max\{c-\Delta; 1\}$, $b := \min\{c+\Delta; n\}$.
2: Starting from the empty set, enumerate all backward sets with at most α items among items $a, \ldots, (c-1)$.
3: Enumerate all forward sets with at most β items $c, \ldots, b$.
4: Generate tuples by merging each backward set with each forward set and store any tuple τ with $p^\tau > 0$ and $w^\tau \leq w_c$.
5: Sort the stored tuples by nondecreasing cardinality with ties broken in favor of tuples with higher ratio p^τ/w^τ.
6: Following the order of the tuples, add the related constraints (20) to $\mathcal{F}(\pi, x, k, \tau)$ as long as $|\mathcal{F}(\pi, x, k, \tau)| \leq \mu$.
7: If not already included, add to $\mathcal{F}(\pi, x, k, \tau)$ constraint $\pi \geq p_c k_{w_c}$.

Then we solve models $CRIT_2^{LP}(c)$ for each $c \in [q, r]$ and order the models by increasing optimal value so as to have an order of most promising subproblems to explore. If for the first subproblem we have $z(CRIT_2^{LP}(c)) \geq z^*$, an optimal BKP solution is already certified (Line 13 of the pseudo code).

4.2.4 Computing feasible BKP solutions

According to the previous order of subproblems, we compute BKP feasible solutions by considering the first γ subproblems (Lines 15-22). For a given item c, we solve model $CRIT_2(c)$ obtaining a solution $\hat{x}$.
If $z(CRIT_2(c)) < z^*$, we solve the induced follower's problem $KP(\hat{x})$ with optimal solution $\hat{y}$ and update the current best solution if $z(KP(\hat{x})) < z^*$.

4.3 Step 2

This step considers all relevant (ordered) suproblems $CRIT_2(c)$. For each subproblem, we first test for standard variables fixing and then each subproblem is explored by means of a constraint generation approach (Lines 24-42).

4.3.1 Fixing variables in subproblems

For a given problem $CRIT_2^{LP}(c)$, denote the optimal values of variables x_i and k_h by x_i^{LP} and k_h^{LP} respectively. Let r_{x_i} and r_{k_h} be the reduced costs of non basic variables in the optimal solution of $CRIT_2^{LP}(c)$. We apply then standard variable-fixing techniques from Integer Linear Programming: if the gap between the best feasible solution available and the optimal solution value of the continuous relaxation solution is not greater than the absolute value of a non basic variable reduced cost, then the related variable can be fixed to its value in the continuous relaxation solution. Thus, the following constraints are added to $CRIT_2(c)$:

$$x_i = x_i^{LP} \quad \forall\, i : |r_{x_i}| \geq z^* - z(CRIT_2^{LP}(c)); \tag{33}$$

$$k_h = k_h^{LP} \quad \forall\, h : |r_{k_h}| \geq z^* - z(CRIT_2^{LP}(c)). \tag{34}$$

4.3.2 Solving subproblems

For each open subproblem induced by a critical item c, we first solve $CRIT_2(c)$ (Line 27) obtaining a solution $\bar{x}$. If the corresponding objective value is lower than the current best feasible solution value, we solve $KP(\bar{x})$ with solution $\bar{y}$ and if an improving solution is found, the current best solution is updated, as in Section 4.2.4. In [13], the following straightforward cut was added to $CRIT_2(c)$ in order to impose that at least one item selected by the follower in solution $\bar{y}$ must be interdicted:

$$\sum_{i:\bar{y}_i=1}^{n} x_i \geq 1. \tag{35}$$

Model $CRIT_2(c)$ is then solved with one more constraint and then the same procedure is applied until $z(CRIT_2(c)) \geq z^*$ or the problem becomes infeasible. Taking a closer look to the structure of the subproblems, at each iteration we can replace when possible constraint (35) with a new additional constraint (20) to induce more targeted changes of the interdiction structure in the leader's solutions and speed up the exploration process of the subproblems.
Given solutions $\bar{x}$ and $\bar{y}$, let $\bar{\tau}$ be the subset of items $1, ..., c-1$ not included in $\bar{y}$ and items $c, ..., n$ included in $\bar{y}$. If $\bar{\tau}$ is not empty, we compute quantities $p^{\bar{\tau}}$, $w^{\bar{\tau}}$ according to (17), (18) respectively and add the corresponding constraint (20) to set $\mathcal{F}(\pi, x, k, \tau)$ in model $CRIT_2(c)$. Two cases may occur when model $CRIT_2(c)$ is solved in the next iteration: either the same solution $\bar{x}$ is obtained but then we would have $z(CRIT_2(c)) = z(KP(\bar{x})) \geq z^*$, or a solution different from $\bar{x}$ is obtained. In this second case, either at least one item in $\bar{\tau}$ is interdicted and/or the weight contribution of the critical item has a value smaller than $w^{\bar{\tau}}$ (corresponding to a different choice of a variable k_h in constraint (12)).
Thus at each iteration we add a valid cut for subproblem $CRIT_2(c)$ when subset $\bar{\tau}$ is not empty or else we add constraint (35) as indicated in the pseudocode (Lines 31-35). In addition, if the value of $z(CRIT_2(c))$ stagnates for ω iterations, we add up to μ tuples with limited weight (Lines 36-39). At the end of Step 2, the optimal BKP solution (x^*, y^*) is returned (Line 43).

Exact solution approach

Input: BKP instance, parameters α, β, Δ, μ, γ, ω.

▷ Step 1

Handle the absence of a critical item:
solve NCR; $z^* \leftarrow +\infty$;
if *NCR has a feasible solution* **then** $x^* = x'$, $y^* = y'$, $z^* = z(NCR)$; **end if**
Identify the candidate critical items and build models $CRIT_2(c)$:
Compute the interval of critical items $[q, r]$: $q \leftarrow$ apply (28), $r \leftarrow$ apply (32);
for all c in $[q, r]$ **do**
    Build model $CRIT_2(c)$ by procedure *DefineTuples(c, α, β, Δ, μ)*;
    Solve model $CRIT_2^{LP}(c)$;
end for
Sort models $CRIT_2(c)$ by increasing $z(CRIT_2^{LP}(c))$.
$\Longrightarrow$ Create a list of ordered critical items $L = \{c_1, c_2, \dots\}$;
if $z(CRIT_2^{LP}(c_1)) \geq z^*$ **then return** (x^*, y^*); **end if**
Compute feasible BKP solutions:
for $i = 1, \dots, \gamma$ **do**
    if $z(CRIT_2^{LP}(c_i)) < z^*$ **then** $\hat{x} \leftarrow$ solve $CRIT_2(c_i)$;
        if $z(CRIT_2(c_i)) < z^*$ **then** $\hat{y} \leftarrow$ solve $KP(\hat{x})$;
            if $z(KP(\hat{x})) < z^*$ **then** $x^* = \hat{x}$, $y^* = \hat{y}$, $z^* = z(KP(\hat{x}))$;
            end if
        end if
    end if
end for

▷ Step 2

Solve subproblems:
for all c in list L **do**
    if $z(CRIT_2^{LP}(c)) \geq z^*$ **then return** (x^*, y^*); **end if**
    Apply (33), (34) and fix variables in $CRIT_2(c)$;
    $\bar{x} \leftarrow$ solve $CRIT_2(c)$;
    while $z(CRIT_2(c)) < z^*$ **do**
        $\bar{y} \leftarrow$ solve $KP(\bar{x})$;
        if $z(KP(\bar{x}) < z^*$ **then** $x^* = \bar{x}$, $y^* = \bar{y}$, $z^* = z(KP(\bar{x}))$; **end if**
        Compute subset $\bar{\tau}$;
        if *$\bar{\tau}$ is not empty* **then**
        add constraint (20), as indicated in section 4.3.2, to $CRIT_2(c)$; **end if**
        if *$\bar{\tau}$ is empty* **then**
        add constraint (35) to $CRIT_2(c)$; **end if**
        if *$z(CRIT_2(c))$ does not increase for ω iterations* **then**
        Consider the remaining tuples computed by *DefineTuples(c, α, β, Δ, μ)*
        with weight $\leq C_l - \sum_{i=1}^{c-1} w_i(1 - \bar{x_i})$;
        Add μ constraints (20) to $CRIT_2(c)$; **end if**
        $\bar{x} \leftarrow$ solve $CRIT_2(c)$;
    end while
end for
return (x^*, y^*).

5 Computational testing

All tests were performed on an Intel i5 CPU @ 3.0 GHz with 16 GB of RAM. The code was implemented in the C++ programming language. The ILP solver used along the steps of the algorithm is CPLEX 12.9. The parameters of the ILP solver were set to their default values. We considered first the BKP instances with $n = 35, 40, 45, 50, 55$ that were generated in [4] as follows. Profits p_i and weights w_i of the follower and weights v_i of the leader are integers randomly distributed in $[1, 100]$: 10 instances are generated for each value of n. The follower's capacity C_l is set to $\lceil (INS/11) \sum_i^n w_i \rceil$ where INS $(= 1, \ldots, 10)$ denotes the instance identifier. The leader's capacity is randomly selected in the interval $[C_l - 10; C_l + 10]$.

These 50 benchmark instances were the most challenging ones solved to optimality in the literature. Indeed, the computational tests in both [4] and [17] were limited to instances with 55 items. After some preliminary computational tests, we chose the following parameter entries for our approach: $\alpha = 2$, $\beta = 2$, $\Delta = 10$, $\mu = 150$, $\gamma = 2$, $\omega = 5$. The corresponding results are presented in Table 1. For each instance, we report the CPU time to obtain an optimal solution and the number of subproblems explored in Step 2. The last column also reports the number of times model $CRIT_2(c)$ is solved along the two steps. Algorithm CCLW in [4] was executed on a Quad-Core Intel Xeon @ 2.66 GHz using solver Gurobi 5.5.0. This algorithm solves all instances with 50 items within a CPU time limit of 3600 seconds but runs out of time limit in instances 55-3, 55-4. Algorithm in [17] was executed on an Intel Xeon E3-1220V2 @ 3.1-GHz using solver CPLEX 12.6. This algorithm solves all benchmark instances, requiring at most a computation time of about 85 seconds for solving instance 55-3. As the results in Table 1 illustrate, the proposed exact approach successfully solves to optimality the whole batch of instances in approximately 5 seconds, that is 0.1 seconds on the average, requiring at most 0.34 seconds on instance 55-3. Also, the number of subproblems explored in Step 2 and the number of models $CRIT_2(c)$ solved are very limited. Even though the tests in [4] and in [17] were carried out on different machines and using different solvers, the improvement with respect to the current state of the art literature appears to be very significant.

n	INS	CPU Time	# Subprob. in Step 2	# $CRIT_2(\cdot)$ solved
35	1	0.08	3	5
	2	0.15	0	2
	3	0.15	1	3
	4	0.12	2	4
	5	0.07	2	4
	6	0.04	0	0
	7	0.03	0	0
	8	0.03	0	0
	9	0.02	0	0
	10	0.02	0	0
40	1	0.12	1	3
	2	0.16	1	3
	3	0.20	3	5
	4	0.11	0	2
	5	0.08	0	2
	6	0.04	0	0
	7	0.04	0	0
	8	0.03	0	0
	9	0.03	0	0
	10	0.01	0	0
45	1	0.14	3	5
	2	0.31	3	5
	3	0.19	2	4
	4	0.19	1	3
	5	0.12	2	4
	6	0.04	0	0
	7	0.04	0	0
	8	0.04	0	0
	9	0.03	0	0
	10	0.03	0	0
50	1	0.26	5	8
	2	0.25	2	4
	3	0.17	2	4
	4	0.13	0	2
	5	0.11	0	2
	6	0.05	0	0
	7	0.05	0	0
	8	0.04	0	0
	9	0.03	0	0
	10	0.02	0	0
55	1	0.22	3	5
	2	0.28	2	4
	3	0.34	8	10
	4	0.17	5	7
	5	0.07	0	0
	6	0.05	0	0
	7	0.05	0	0
	8	0.04	0	0
	9	0.04	0	0
	10	0.03	0	0

Table 1 BKP instances from [4].

We considered also other instances considered in the literature proposed in [14] (available at `http://coral.ise.lehigh.edu/data-sets/bilevel-instances/`) and [35] (available at `http://people.clemson.edu/~jcsmith/Test_Instances_files/BKPIns.zip`). There is a total of 160 instances in [14] with size not larger than $n = 50$ and there is a total of 210 instances in [35] with size not larger than $n = 30$. We do not provide the complete results on those instances but just mention that our exact algorithm with the above mentioned settings solved the whole batch of 160 instances in [14] in 9.87 seconds (less than 0.06 seconds on average for instance) and the whole batch of 210 instances in [35] in 18.1 seconds (less than

0.09 seconds on average for instance). We then generated and tested larger instances with $n = 100, 200, 300, 400, 500$ (available at `https://drive.google.com/drive/folders/1LTDEB3b8gFDXbRJ-9b3RRjJ978BpynYm`) according to the generation scheme in [4]. For each value of n and INS, we generated 10 instances for a total of 500 instances. For these large instances, we set the parameters of our algorithm to the following values: $\alpha = 2$, $\beta = 2$, $\Delta = 20$, $\mu = 1000$, $\gamma = 5$, $\omega = 5$. It is pointed out in [4] that in instances with $INS \geq 5$ the follower's capacity constraint is expected to be inactive for any maximal leader's interdiction strategy. This makes these instances easy to solve. Our computational experiments confirm this trend also on larger instances: the proposed algorithm solves each instance with n from 100 to 500 and $INS \geq 5$ in at most 6 seconds without ever invoking Step 2. In the light of this consideration, we report in the following Table 2 only the results for instances with $INS \leq 4$.

			CPU Time		# Subproblems in Step 2		# $CRIT_2(\cdot)$ solved	
n	INS	#Opt	Average	Max	Average	Max	Average	Max
100	1	10	0.9	1.1	0.3	1	4.3	6
	2	10	1.7	2.6	3.1	8	8.1	14
	3	10	1.5	2.0	2.2	6	7.2	11
	4	10	0.9	1.3	0.7	4	5.2	9
200	1	10	2.2	3.3	2.9	6	8.0	13
	2	10	3.1	4.0	4.2	6	9.2	11
	3	10	3.5	4.4	5.6	11	10.8	16
	4	10	2.7	3.4	3.0	8	7.8	13
300	1	10	3.5	4.0	3.1	9	8.0	14
	2	10	5.3	7.5	6.2	12	11.2	17
	3	10	6.4	7.6	10.1	15	15.5	20
	4	10	4.7	6.1	4.5	11	9.5	16
400	1	10	5.0	6.1	5.5	9	10.9	16
	2	10	7.4	8.6	7.8	12	13.4	20
	3	10	9.0	11.3	10.5	14	16.2	22
	4	10	7.5	9.4	7.8	23	12.9	29
500	1	10	7.2	9.5	6.9	14	12.2	21
	2	10	10.4	12.6	9.8	17	14.9	22
	3	10	11.9	13.4	12.3	17	17.4	22
	4	10	9.8	10.9	4.4	9	9.4	14

Table 2 BKP instances with $n = 100, 200, 300, 400, 500$ and $INS \leq 4$.

The results in the table are summarized in terms of average, maximum CPU time and number of optimal solutions obtained with a time limit of 60 seconds. Similarly to Table 1, we also report the average and maximum number of subproblems explored in Step 2, and the average and maximum number of times model $CRIT_2(c)$ is solved. The results illustrate the effectiveness of our approach. All instances are solved to optimality requiring 13.6 seconds at most for an instance with 500 items. The number of subproblems handled by Step 2 is in general lim-

ited, reaching a maximum value of 23 (in an instance with 400 items). Also, the number of models $CRIT_2(c)$ to be solved is generally limited and never larger than 29. We finally point out that the number of times constraints (20)/(35) are added to each subproblem is also limited: in the tested instances, the while–loop of Step 2 is executed 5 iterations at most.

To get a broader picture, we also tackled the instances generated in the recent work [19] where a heuristic approach is proposed for BKP and for other interdiction games. These instances are classified according to different correlations between profits and weights following the schemes in [25]. Nine classes were listed in [19] having the following distribution where u.r. stands for uniformly random.

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

All the instances were generated with $R = 100$, the leader's weights v_j u.r. in $[0, R]$ and follower's and leader's capacities generated as in [4]. We do not consider Class 9 here as it was mentioned in [19] that the instances of this class are trivial for BKP. Table 3 provides the relevant results in terms of average, maximum CPU time and number of optimal solutions obtained within a time limit of 300 seconds for the easier classes 1-2 where our algorithm kept the same parameters values indicated above.

			CPU Time		# Subproblems in Step 2		# $CRIT_2(\cdot)$ solved	
CORR	n	#Opt	Average	Max	Average	Max	Average	Max
	100	10	0.6	1.9	0.5	4	2.4	9
	200	10	1.4	3.3	1.2	6	3.2	11
1	300	10	2.6	6.9	2.3	12	4.3	17
	400	10	4.0	8.7	3.6	12	5.9	17
	500	10	4.9	10.5	2.5	8	4.5	13
	100	10	4.3	18.6	4.3	20	6.6	28
	200	10	12.2	39.1	6.8	29	10.7	41
2	300	10	12.5	52.2	8.6	37	11.5	42
	400	10	16.5	75.3	10.4	45	13.7	53
	500	10	33.7	180.9	11.9	48	22.5	113

Table 3 BKP instances from [19] (Classes 1-2).

Table 4 provides the relevant results on the harder correlations 3–8. For these instances, best performances were reached by slightly modifying the value of parameters Δ ($\Delta = 50$) and μ ($\mu = 2000$) while keeping unchanged the other parameters. For these instances, it is shown that the algorithm is capable of solving to optimality instances with size $n = 100$ with only four instances (two of class 6 and two of class 7, respectively) exceeding a CPU time limit of 3600 seconds, thus further highlighting the effectiveness of the proposed approach. In the table, the average CPU time is computed without considering the four instances exceeding the time limit.

			CPU Time		# Subproblems in Step 2		# $CRIT_2(\cdot)$ solved	
CORR	n	#Opt	Average	Max	Average	Max	Average	Max
3	100	10	148.6	1377.5	3.8	20	23.7	204
4	100	10	81.0	400.8	5.4	32	43.2	195
5	100	10	69.9	672.7	3.3	20	10.3	80
6	100	8	766.3	3600.0	1.6	13	18.1	145
7	100	8	737.1	3600.0	1.6	13	15.4	123
8	100	10	228.1	2234.4	4.1	20	17.4	138

Table 4 BKP instances from [19] (Classes 3-8, $n = 100$).

As mentioned at the end of Section 3, variables k_h could be substituted by a single integer variable. We tested this alternative formulation but the performances were consistently inferior on every non trivial instance. The total CPU time (summed up on all considered instances) ratio between the formulation with a single integer variable and the formulation with w_c binary variables is approximately equal to 4.

For sake of completeness, we tested also on our machine the exact approaches of [4] and [17] (the related codes were kindly provided by the authors) with a time limit of 3600 seconds on the instances of Tables 3 and 4 with $n = 100$ for a total of 80 instances. Both approaches were typically able to solve in limited time the so-called easy instances with $INS = 5 - 10$, but ran out-of-time in most of the other instances. Specifically, the approach of [4] ran out of time on 28 instances while the approach of [17] ran out of time on 36 instances.
We tested then our approach on larger instances with $n = 200 - 500$ on the harder correlations 3–8. Table 5 provides the relevant results regrouped in three main categories ($INS = 1 - 3$, $INS = 4$ and $INS = 5 - 10$). For each category, we depict the number of problems solved to optimality along the six classes $3 - 8$ and the average CPU time. The results clearly indicate that instances with $INS = 4$ and $INS = 5 - 10$ remain consistently easy to solve for the proposed approach, while instances with $INS = 1 - 3$ are currently out of reach. For these instances involving strong correlations between the profits and the weights of the follower, we believe that dedicated approaches should be considered as it was done in the corresponding versions of the standard KP (see, e.g., the works [30] and [32] for handling strongly correlated and subset-sum instances, respectively.)

n	*INS*	#Opt	Aver. Time
200	1-3	2/18	3210.0
	4	6/6	23.4
	5-10	36/36	7.3
300	1-3	2/18	3225.6
	4	6/6	49.3
	5-10	36/36	16.1
400	1-3	0/18	3600.0
	4	6/6	71.5
	5-10	36/36	25.9
500	1-3	0/18	3600.0
	4	6/18	141.6
	5-10	36/36	36.1

Table 5 BKP instances from [19] (Classes 3-8, $n = 200, 300, 400, 500$).

Finally, we implemented a one-shot heuristic version of our procedure applying only step 1 and allowing a CPU time limit of 60 seconds. In the heuristic, the settings were the same used by our exact algorithm for the instances generated in [4] except for $\gamma = 5$ where for each model $CRIT_2(c)$ CPLEX 12.9 was allowed to run for 10 seconds. We compared our heuristic to the best heuristic algorithm denoted DR_+ in [19] on distributions [3-8] (as our exact algorithm solves all instances to optimality on distributions [1-2]). This heuristic procedure shows up to be just slightly inferior to the algorithm in [19] reaching the same objective function value on 281 over 300 instances and being outperformed on 19 instances only.

6 Extending the approach to the Interval Min-Max Regret Knapsack Problem

We now discuss the application of the derived algorithmic framework to the Interval Min-Max Regret Knapsack Problem (MRKP) first introduced in [24]. Some of the notation adopted for BKP recurs in this section with a different meaning. Consider the classical 0/1 Knapsack Problem (KP) and the related ILP formulation $\max \sum_{i=1}^{n} p_i x_i$ subject to $\sum_{i=1}^{n} w_i x_i \leq C, \quad x_i \in \{0,1\},\ i = 1, \ldots, n$. MRKP is a generalization of KP where the profit of each item i can assume any value between two values p_i^- and p_i^+, with $p_i^+ > p_i^-$. A set s of n profits $p_i^s \in [p_i^-, p_i^+]$ defines a scenario. The set of all possible scenarios is denoted by S_0, namely $S_0 := \{s : p_i^s \in [p_i^-, p_i^+],\ i = 1, \ldots, n\}$. We also denote by $z^s(opt)$ the KP optimal solution value under scenario s where each item has profit p_i^s and by $z^s(x)$ the solution value given by a feasible solution vector x, i.e. $z^s(x) = \sum_{i=1}^{n} p_i^s x_i,\ x_i \in x$. Correspondingly, the regret $r^s(x)$ associated with a solution x under scenario s is

$$r^s(x) = z^s(opt) - z^s(x). \tag{36}$$

MRKP consists in finding a feasible solution vector x such that the maximum regret obtainable over all scenarios is minimized. More formally the problem can

be stated as follows:

$$\min \quad \max_{s \in S_0} r^s(x) \tag{37}$$

$$\text{subject to} \quad \sum_{i=1}^{n} w_i x_i \leq C \tag{38}$$

$$x_i \in \{0,1\} \qquad i = 1, \ldots, n \tag{39}$$

We denote by x^* an optimal solution of model (37)-(39) with regret value z^*. The authors of [9] prove that the MRKP is Σ_2^p-complete and point out that the solution of even moderately sized instances of the problem is challenging and seems to require innovative approaches. An attempt in this direction has been provided in [20], where heuristic and exact approaches are proposed for the solution of the MRKP. The authors of [20] evaluate the performance of the proposed algorithms through extensive computational experiments on instances with up to 70 items (see Section 6.2). We refer the interested reader to [20] for a literature review about other min-max regrets optimization problems.

6.1 Bilevel reformulation of the MRKP

The following crucial result was proved in [1] (and recalled in [20]) for the MRKP within a general context of min-max regrets problems:

Lemma 1 *([1]) For any feasible solution x, the profits in its worst case scenario, denoted as $\sigma(x)$, are as follows:*

$$p_i^{\sigma(x)} = \begin{cases} p_i^- & \text{if } x_i = 1 \\ p_i^+ & \text{otherwise} \end{cases} \quad (i = 1, \ldots, n)$$

Therefore a given feasible solution x induces a unique worst case scenario ($\sigma(x)$) to be considered for the computation of the corresponding maximum regret $r^{\sigma(x)}$, namely

$$r^{\sigma(x)} = z^{\sigma(x)}(opt) - z^{\sigma(x)} = z^{\sigma(x)}(opt) - \sum_{i=1}^{n} p_i^- x_i \tag{40}$$

Notice that the last summation in (40) representing $z^{\sigma(x)}$ does not include terms p_i^+ as they disappear when $x_i = 0$. Given the result in Lemma 1, we can alternatively see the problem as a Stackelberg game and propose a related bilevel programming reformulation. To the authors' knowledge, this is the first time MRKP is formulated as a bilevel programming problem with interdiction constraints. Here the leader first chooses a feasible knapsack solution x with the goal of minimizing the regret $r^{\sigma(x)}$ associated with his decision. The follower aims to maximize the profits of the knapsack instance induced by the worst-case scenario $\sigma(x)$, thus computing $z^{\sigma(x)}(opt)$. To derive a bilevel linear program, we consider binary variables x_i ($i = 1, \ldots, n$) equal to one if the leader selects item i. We also introduce $2n$ items in the follower's knapsack problem: for each i we define a "low" item l_i with profit

p_i^- and weight w_i and a "high" item h_i with profit p_i^+ and weight w_i. Correspondingly, we introduce binary variables y_i^- and y_i^+ equal to one if the follower selects item l_i or h_i $(i = 1, \ldots, n)$ respectively. MRKP can be modeled as follows:

$$\min \quad \sum_{i=1}^{n}(p_i^- y_i^- + p_i^+ y_i^+) - \sum_{i=1}^{n} p_i^- x_i \tag{41}$$

$$\text{subject to} \quad \sum_{i=1}^{n} w_i x_i \leq C \tag{42}$$

$$x_i \in \{0,1\} \qquad i = 1, \ldots, n \tag{43}$$

where $y_1^-, \ldots, y_n^-, y_1^+, \ldots, y_n^+$ solve

$$\text{the follower's problem:} \quad \max \quad \sum_{i=1}^{n}(p_i^- y_i^- + p_i^+ y_i^+) \tag{44}$$

$$\text{subject to} \quad \sum_{i=1}^{n} w_i(y_i^- + y_i^+) \leq C \tag{45}$$

$$y_i^- \leq x_i \qquad i = 1, \ldots, n \tag{46}$$

$$y_i^+ \leq 1 - x_i \qquad i = 1, \ldots, n \tag{47}$$

$$y_i^-, y_i^+ \in \{0,1\} \qquad i = 1, \ldots, n \tag{48}$$

The leader's objective function (41) minimizes the regret (40) given by the difference between the follower's profits and $\sum_{i=1}^{n} p_i^- x_i$. The objective function (44) maximizes the follower's profits while constraints (42) and (45) respectively represent the leader's and follower's capacity constraint. Constraints (46) and (47) ensure that scenario $\sigma(x)$ is always induced in the follower's knapsack problem for any leader's solution x: the follower can in fact select only item l_i (with profit p_i^-) if $x_i = 1$ and only item h_i (with profit p_i^+) if $x_i = 0$ for each $i = 1, \ldots, n$. Constraints (43) and (48) define the domain of the variables.

Hence, we can generate a model which is structurally similar to the BKP bilevel formulation (1)-(7). This motivates us to employ the same algorithmic machinery setup for BKP with proper integrations. Model formulation and details of the approach are provided in Appendix.

6.2 Computational results on MRKP

The authors of [20] introduce several heuristic and exact approaches to tackle the MRKP. The solution approaches are based on classical optimization techniques such as Benders decomposition and branch and cut methods. The best performing exact algorithm given in [20] (running on a Pentium 4 @3.2 GHz), denoted as FIMY, is capable of solving to optimality instances with up to 70 items with a time limit of 3600 s, running out of time in 95 out of 486 instances. The instances (available at `http://www.or.deis.unibo.it/research_pages/ORinstances/MRKP_instances.zip`) were generated by considering different distributions of processing times and weights according to the same nine classes indicated in Section 5 with $R = 1000/10000$ and the following additional settings.

- 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 $\delta \in \{0.1, 0.2, 0.3\}$.

After preliminary experiments, we set the the parameters of our exact approach to the following entries: $\alpha = 2$, $\beta = 2$, $\Delta = 50$, $\mu = 1000$, $\gamma = 5$. $\omega = 5$ for Class 6, 7 and 8 which are the hardest classes to solve; $\alpha = 2$, $\beta = 2$, $\Delta = 20$, $\mu = 200$, $\gamma = 5$. $\omega = 5$ for the remaining classes. Table 6 compares FIMY (running on a different machine) to our approach on classes 1–3. Each row in the table considers six instances aggregated by the number of items n and value of δ. For both approaches, column "Sec" indicates the average CPU time, column "#f" (failures) indicates the total number of instances not solved to proven optimality. Similarly, Table 7 relates to classes 4–6 and Table 8 relates to classes 7–9. Entry "t.l." in the tables indicates that the approach reached an out of time status in *all* the instances of the category.

		Class 1				Class 2				Class 3			
		FIMY		Our Approach		FIMY		Our Approach		FIMY		Our Approach	
n	δ	Sec	#f	Sec	#f	Sec	#f	Sec	#f	Sec	#f	Sec	#f
50	0.1	< 0.1	0	0.2	0	< 0.1	0	3.0	0	3.2	0	13.0	0
	0.2	< 0.1	0	0.3	0	1.2	0	3.4	0	235.7	0	3.9	0
	0.3	< 0.1	0	0.3	0	2.9	0	1.4	0	195.2	0	5.3	0
Avg/Tot		< 0.1	0	0.3	0	1.4	0	2.6	0	144.7	0	7.4	0
60	0.1	< 0.1	0	0.3	0	0.2	0	5.4	0	31.8	0	17.0	0
	0.2	< 0.1	0	0.4	0	4.6	0	4.4	0	1835.7	3	3.5	0
	0.3	< 0.1	0	0.5	0	11.6	0	2.2	0	1906.2	2	4.7	0
Avg/Tot		< 0.1	0	0.4	0	5.5	0	4.0	0	1257.9	5	8.4	0
70	0.1	< 0.1	0	0.5	0	0.2	0	6.5	0	108.5	0	19.1	0
	0.2	< 0.1	0	0.5	0	10.9	0	3.8	0	2029.3	3	5.2	0
	0.3	< 0.1	0	0.5	0	53.7	0	2.7	0	3504.5	4	5.1	0
Avg/Tot		< 0.1	0	0.5	0	21.6	0	4.3	0	1880.8	7	9.8	0
Overall		< 0.1	0	0.4	0	9.5	0	3.6	0	1094.5	12	8.5	0

Table 6 MRKP instances from [20] (Classes 1-3).

		Class 4				Class 5				Class 6			
		FIMY		Our Approach		FIMY		Our Approach		FIMY		Our Approach	
n	δ	Sec	#f	Sec	#f	Sec	#f	Sec	#f	Sec	#f	Sec	#f
50	0.1	2.9	0	9.2	0	7.5	0	12.1	0	252.7	0	666.0	1
	0.2	82.0	0	4.3	0	55.5	0	4.3	0	1161.1	0	161.2	0
	0.3	30.4	0	2.0	0	72.1	0	2.1	0	315.1	0	18.1	0
Avg/Tot		38.4	0	5.1	0	45.1	0	6.1	0	576.3	0	281.8	1
60	0.1	21.5	0	16.3	0	46.6	0	29.2	0	3409.1	5	738.1	1
	0.2	1383.0	0	6.6	0	348.6	0	5.3	0	3235.3	4	352.3	0
	0.3	411.2	0	3.6	0	703.8	0	2.5	0	t.l.	6	59.8	0
Avg/Tot		605.2	0	8.8	0	366.4	0	12.3	0	3414.8	15	383.4	1
70	0.1	60.4	0	22.0	0	112.3	0	20.6	0	t.l.	6	944.3	1
	0.2	1849.8	3	6.2	0	1723.3	2	8.4	0	t.l.	6	553.4	0
	0.3	1662.9	0	3.6	0	2662.3	3	3.9	0	t.l.	6	137.0	0
Avg/Tot		1191.0	3	10.6	0	1499.3	5	11.0	0	t.l.	18	544.9	1
Overall		611.6	3	8.2	0	636.9	5	9.8	0	2530.3	33	403.4	3

Table 7 MRKP instances from [20] (Classes 4-6).

		Class 7				Class 8				Class 9			
		FIMY		Our Approach		FIMY		Our Approach		FIMY		Our Approach	
n	δ	Sec	#f	Sec	#f	Sec	#f	Sec	#f	Sec	#f	Sec	#f
50	0.1	234.4	0	987.0	1	2.1	0	47.4	0	< 0.1	0	0.4	0
	0.2	794.6	0	221.2	0	120.8	0	19.8	0	< 0.1	0	0.4	0
	0.3	594.6	0	75.8	0	103.8	0	14.9	0	< 0.1	0	0.4	0
Avg/Tot		541.2	0	428.0	1	75.6	0	27.3	0	< 0.1	0	0.4	0
60	0.1	3088.6	3	859.2	0	25.1	0	62.5	0	< 0.1	0	0.5	0
	0.2	t.l.	6	762.0	1	1834.7	3	24.4	0	< 0.1	0	0.5	0
	0.3	t.l.	6	312.9	0	900.3	0	20.3	0	< 0.1	0	0.5	0
Avg/Tot		3429.5	15	644.7	1	920.0	3	35.7	0	< 0.1	0	0.5	0
70	0.1	t.l.	6	1525.9	1	63.1	0	84.4	0	< 0.1	0	0.6	0
	0.2	t.l.	6	627.8	0	1997.7	3	29.7	0	< 0.1	0	0.6	0
	0.3	t.l.	6	180.6	0	2620.4	3	28.1	0	< 0.1	0	0.6	0
Avg/Tot		t.l.	18	778.1	1	1560.4	6	47.4	0	< 0.1	0	0.6	0
Overall		2523.5	33	616.9	3	852.0	9	36.8	0	< 0.1	0	0.5	0

Table 8 MRKP instances from [20] (Classes 7-9).

From the tables, even though tests were executed on different machines, we evince that the proposed approach strongly outperforms FIMY (except for Class 1 and Class 9 where FIMY is slightly faster but our approach requires 0.6 seconds at most) running out of time in six instances only (three of Class 6 and three of Class 7).

7 Concluding remarks

We proposed for the Bilevel Knapsack Problem with Interdiction Constraints a new exact approach which outperforms the state-of-the-art algorithms available in the literature. The algorithm relies on a new lower bound derived for the problem,

which is improved by exploiting the expected features of an optimal solution of the classical knapsack problem. It is shown that the proposed approach handles in few seconds uncorrelated instances with up to 500 items and is capable of solving all but *four* instances with $n = 100$ for various correlation classes between follower's weights and profits. The approach has been extended to deal with the Interval Min-Max Regret Knapsack Problem. Also in this case the proposed algorithm is shown up to be superior to the current state of the art literature. A very natural future research would be to study bilevel versions of several generalizations of KP, such as, for instance, Collapsing KP [12], Penalized KP [10], KP with Setups [11, 21, 29] and Product KP[8], for which very efficient exact algorithms have been recently proposed.

Acknowledgments

The very pertinent and useful comments of two anonymous referees are gratefully acknowledged. The authors wish to thank M. Carvalho for providing the benchmark instances of [4]. This work has been partially supported by "Ministero dell'Istruzione, dell'Università e della Ricerca" Award "TESUN-83486178370409 finanziamento dipartimenti di eccellenza CAP. 1694 TIT. 232 ART. 6".

References

1. Aissi, H., Bazgan, C., Vanderpooten, D.: Minmax and minmax regret versions of combinatorial optimization problems: A survey. European Journal of Operational Research. 197, 427–438 (2009).
2. Brotcorne, L., Hanafi, S., Mansi, R.: One-level reformulation of the bilevel Knapsack problem using dynamic programming. Discrete Optimization. 10, 1–10 (2013).
3. Caprara, A., Carvalho, M., Lodi, A., Woeginger, G.: A Complexity and Approximability Study of the Bilevel Knapsack Problem. Proceedings of IPCO 2013. Volume 7801 of LNCS, 98–109 (2013).
4. Caprara, A., Carvalho, M., Lodi, A., Woeginger, G.: Bilevel Knapsack with Interdiction Constraints. INFORMS Journal on Computing. 28, 319–333 (2016).
5. Caramia, M., Mari, R.: Enhanced exact algorithms for discrete bilevel linear problems. Optimization Letters, 9, 1447–1468 (2015).
6. Carvalho, M., Lodi, A., Marcotte, P.: A polynomial algorithm for a continuous bilevel knapsack problem. Operations Research Letters. 46, 185–188 (2018).
7. Chen, L., Zhang, G.: Approximation algorithms for a bi-level knapsack problem. Theoretical Computer Science. 497, 1–12 (2013).
8. D'Ambrosio, C., Furini, F., Monaci, M., Traversi, E.: On the Product Knapsack Problem. Optimization Letters. 12 (4), 691–712 (2018).
9. Deineko, V.G., Woeginger, G.: Pinpointing the complexity of the interval min–max regret knapsack problem. Discrete Optimization. 7, 191–196 (2010).
10. Della Croce, F., Pferschy, U., Scatamacchia, R.: New exact approaches and approximation results for the penalized knapsack problem. Discrete Applied Mathematics. 253, 122–135 (2019).
11. Della Croce, F., Salassa, F., Scatamacchia, R.: An exact approach for the 0-1 knapsack problem with setups. Computers and Operations Research 80, 61–67, (2019).
12. Della Croce, F., Salassa, F., Scatamacchia, R.: A new exact approach for the 0-1 collapsing knapsack problem. European Journal of Operational Research 260, 56–69, (2017).
13. Della Croce, F., Scatamacchia, R.: Lower Bounds and a New Exact Approach for the Bilevel Knapsack with Interdiction Constraints. In: Lodi A., Nagarajan V. (eds) Integer Programming and Combinatorial Optimization. IPCO 2019. Lecture Notes in Computer Science, vol 11480, 155–167. Springer International Publishing (2019).

14. DeNegre, S.: Interdiction and discrete bilevel linear programming. PhD thesis. Lehigh University (2011).
15. DeNegre, S., Ralphs, T.K.: A Branch-and-cut Algorithm for Integer Bilevel Linear Programs. Operations Research and Cyber-Infrastructure, volume 47 of Operations Research/Computer Science Interfaces, 65–78 (2009).
16. Fischetti, M., Ljubić, I., Monaci, M., Sinnl, M.: A New General-Purpose Algorithm for Mixed-Integer Bilevel Linear Programs. Operations Research. 65, 1615–1637 (2017).
17. Fischetti, M., Ljubić, I., Monaci, M., Sinnl, M.: Interdiction Games and Monotonicity, with Application to Knapsack Problems. INFORMS Journal on Computing. 31 (2), 390–410 (2019).
18. Fischetti, M., Ljubić, I., Monaci, M., Sinnl, M.: On the use of intersection cuts for bilevel optimization. Mathematical Programming. 172, 77–103 (2018).
19. Fischetti, M., Monaci, M., Sinnl, M.: A dynamic reformulation heuristic for Generalized Interdiction Problems. European Journal of Operational Research. 267, 40–51 (2018).
20. Furini, F., Iori, M., Martello, S., Yagiura, M.: Heuristic and Exact Algorithms for the Interval Min–Max Regret Knapsack Problem. INFORMS Journal on Computing. 27, 392–405 (2015).
21. Furini, F., Monaci, M., Traversi, E.: Exact approaches for the knapsack problem with setups. Computers and Operations Research, 90, 208–220 (2018).
22. Jeroslow, R.: The polynomial hierarchy and a simple model for competitive analysis. Mathematical Programming. 32, 146–164, (1985).
23. Kellerer, H., Pferschy. U., Pisinger, D.: Knapsack Problems. Springer (2004).
24. Kouvelis, P., Yu, G.: Robust Discrete Optimization and its Applications. Kluwer Academic Publishers, Boston (1997).
25. Martello, S., Pisinger, D., Toth, P.: Dynamic programming and strong bounds for the 0–1 knapsack problem. Management Science. 45, 414–424 (1999).
26. Martello, S., Toth, P.: Knapsack Problems: Algorithms and Computer Implementations. Wiley (1990).
27. Moore, J.T., Bard, J.F.: The mixed integer linear bilevel programming problem. Operations Research. 38, 911–921 (1990).
28. Pferschy, U., Nicosia, G., Pacifici A.: A Stackelberg knapsack game with weight control. Theoretical Computer Science 799, 149–159 (2019).
29. Pferschy, U., Scatamacchia, R.: Improved dynamic programming and approximation results for the knapsack problem with setups. International Transactions in Operational Research 25, 667–682 (2018).
30. Pisinger, D.: A fast algorithm for strongly correlated knapsack problems. Discrete Applied Mathematics 89, 197–212 (1998).
31. Pisinger, D.: A minimal algorithm for the 0–1 knapsack problem. Operations Research 45, 758–767 (1997).
32. Pisinger, D.: Linear time algorithms for knapsack problems with bounded weights. Journal of Algorithms 33, 1–14 (1999).
33. Qiu, X., Kern, W.: Improved approximation algorithms for a bilevel knapsack problem. Theoretical Computer Science. 595, 120–129 (2015).
34. Stackelberg, H.V.: The Theory of the Market Economy. Oxford University Press (1952).
35. Tang, Y., Richard, J.P.P., Smith, J.C.: A class of algorithms for mixed-integer bilevel min–max optimization. Journal of Global Optimization. 66 (2), 225–262 (2016).
36. Wang, L., Xu, P.: The Watermelon Algorithm for the Bilevel Integer Linear Programming Problem. SIAM Journal on Optimization. 27(3), 1403–1430 (2017).
37. Xu, P., Wang, L.: An exact algorithm for the bilevel mixed integer linear programming problem under three simplifying assumptions. Computers & Operations Research. 41, 309–318 (2014).

8 Appendix

8.1 Additional notation for MRKP

For further analysis, we define the set of items l_i by L and the set of items h_i by H. We consider the $2n$ items in the follower's knapsack ordered by nonincreasing

ratio profit over weight. For each item $j = 1, \ldots, 2n$ in the ordering we denote by $i(j)$ the corresponding index i. Similarly as in Section 2, we denote by $KP(x)$ the follower's knapsack problem induced by a leader's solution x, here with item set

$$J := \left\{ \begin{array}{l} j : x_{i(j)} = 1 \text{ and } j \in L, \\ j : x_{i(j)} = 0 \text{ and } j \in H, \\ x_i \in x \end{array} \right\}$$

and denote by $KP^{LP}(x)$ the corresponding LP relaxation. We again denote as *critical* the last item with a strictly positive value in the optimal solution of $KP^{LP}(x)$. Notice that for each x_i one item between l_i and h_i will be always available for insertion in the follower's knapsack. Hence, under the standard assumption that $\sum_{i=1}^{n} w_i > C$ in the MRKP, we also have $\sum_{j=1}^{2n} w_{i(j)} > C$. This implies that there alway exists a critical item c in $KP^{LP}(x)$ and a split solution with value

$$\sum_{\substack{j \in J: \\ j<c, j \in L}} p^{-}_{i(j)} + \sum_{\substack{j \in J: \\ j<c, j \in H}} p^{+}_{i(j)} = \sum_{j<c, j \in L} p^{-}_{i(j)} x_{i(j)} + \sum_{j<c, j \in H} p^{+}_{i(j)} (1 - x_{i(j)}) \qquad (49)$$

8.2 Computing a lower bound for MRKP

As for BKP, we aim to guess the critical item of $KP^{LP}(x^*)$. To this purpose we modify model $CRIT_2(c)$ as follows. We replace term $\sum_{h=1}^{w_c} hk_h$ with an integer variable θ associated with the weight contribution of the critical item c in the follower's capacity constraint, with $1 \leq \theta \leq w_{i(c)}$. The objective function has term $\sum_{i=1}^{n} p_i^{-} x_i$ as negative contribution plus the split solution value (49) and the additional profit contribution that can be gained by a feasible solution derived by the split solution. Such a profit is captured by variable π through a set of constraints $\mathcal{F}(\pi, x, \theta, v, \tau)$ involving additional binary variables v_p ($p = 1, \ldots, \frac{|\mathcal{F}(\pi,x,\theta,v,\tau)|}{2}$). We explain the construction of set $\mathcal{F}(\pi, x, \theta, v, \tau)$ next. We obtain the following model, denoted as $CRIT_3(c)$.

$CRIT_3(c)$:

$$\min \quad \sum_{j<c, j\in L} p^-_{i(j)} x_{i(j)} + \sum_{j<c, j\in H} p^+_{i(j)}(1 - x_{i(j)}) + \pi - \sum_{i=1}^{n} p^-_i x_i \tag{50}$$

$$\text{subject to} \quad \sum_{i=1}^{n} w_i x_i \leq C \tag{51}$$

$$\sum_{j<c, j\in L} w_{i(j)} x_{i(j)} + \sum_{j<c, j\in H} w_{i(j)}(1 - x_{i(j)}) + \theta = C \tag{52}$$

$$1 \leq \theta \leq w_{i(c)} \tag{53}$$

$$\mathcal{F}(\pi, x, \theta, v, \tau) \tag{54}$$

$$x_{i(c)} = 1 \text{ if } c \in L \,, x_{i(c)} = 0 \text{ if } c \in H \tag{55}$$

$$x_i \in \{0, 1\} \quad i = 1, \ldots, n \tag{56}$$

$$v_p \in \{0, 1\} \quad p = 1, \ldots, \frac{|\mathcal{F}(\pi, x, \theta, v, \tau)|}{2} \tag{57}$$

$$\theta \in N \tag{58}$$

$$\pi \geq 0 \tag{59}$$

The objective function (50) minimizes the difference between the value of a follower's knapsack solution and sum $\sum_{i=1}^{n} p^-_i x_i$. Constraint (51) represents the leader's capacity constraint. Constraints (52) and (53) guarantee that item c is critical as it is the last item packed, with a weight in the interval $[1, w_{i(c)}]$. Constraint (55) enforces the availability of item c in the follower's knapsack: if $c \in L$ we must have $x_{i(c)} = 1$ or else $x_{i(c)} = 0$. Constraints (56)-(59) define the domain of the variables.

We build a *proper* set $\mathcal{F}(\pi, x, \theta, v, \tau)$, namely a set of constraints that considers only feasible solutions in the follower's knapsack, as follows. For a given subset τ of items around the critical item c, the corresponding profit p^τ and weight w^τ now are:

$$p^\tau = - \sum_{\substack{j\in\tau: \\ j<c, j\in L}} p^-_{i(j)} - \sum_{\substack{j\in\tau: \\ j<c, j\in H}} p^+_{i(j)} + \sum_{\substack{j\in\tau: \\ j\geq c, j\in L}} p^-_{i(j)} + \sum_{\substack{j\in\tau: \\ j\geq c, j\in H}} p^+_{i(j)} \tag{60}$$

$$w^\tau = - \sum_{j\in\tau: i<c} w_{i(j)} + \sum_{j\in\tau: j\geq c} w_{i(j)}. \tag{61}$$

We consider in set $\mathcal{F}(\pi, x, \theta, v, \tau)$ only subsets of items improving upon the split solution, i.e. with $p^\tau > 0$, that could be feasible, i.e. with $w^\tau \leq w_{i(c)}$. Also we do not consider subsets with two items j and j' corresponding to the same x_i, namely with $i(j) = i(j')$, as these items cannot be both available for packing. For the $p - th$ subset τ considered for selection in $\mathcal{F}(\pi, x, \theta, v, \tau)$, we will have binary variable v_p equal to one only if the follower's capacity constraint is not violated, i.e. $w^\tau \leq \theta$. This condition will be enforced by adding to $\mathcal{F}(\pi, x, \theta, v, \tau)$ the constraint $(w_{i(c)} - w^\tau)v_p \geq \theta - w^\tau$, that we translate as follows to be added as a linear constraint

$$(w_{i(c)} - w^\tau + \frac{1}{2})v_p \geq \theta - w^\tau + \frac{1}{2}, \tag{62}$$

where constant term $\frac{1}{2}$ is introduced to have $v_p = 1$ when $w^\tau = \theta$. Then an improvement π can be determined by adding to $\mathcal{F}(\pi, x, \theta, v, \tau)$ the following constraint

$$\pi \geq p^\tau (v_p - \sum_{j \in \tau, j \in L} (1 - x_{i(j)}) - \sum_{j \in \tau, j \in H} x_{i(j)}). \tag{63}$$

Notice that constraints (62) and (63) ensure that only items available for insertion in the follower's knapsack are considered and the follower's capacity constraint is not violated, validating tuple τ. The following proposition holds.

Proposition 3 *If $KP^{LP}(x^*)$ has critical item c and model $CRIT_3(c)$ has a proper set $\mathcal{F}(\pi, x, \theta, v, \tau)$, then $z(CRIT_3(c)) \leq z^*$.*

Proof We apply the same argument of Propositions 1 and 2 and observe that model $CRIT_3(c)$ considers feasible solutions for $KP(x^*)$. Let z_1 denote the solution value of the best of these KP solutions. We have

$$z(CRIT_3(c)) \leq z_1 - \sum_{i=1}^{n} p_i^- x_i^* \leq z(KP(x^*)) - \sum_{i=1}^{n} p_i^- x_i^* = z^*,$$

where the first inequality is implied by noticing that x^* is feasible but not necessarily optimal for model $CRIT_3(c)$. □

We finally notice that the introduction of variable θ is motivated by the presence of large follower's weights (with values up to 10000) in the benchmark MRKP literature instances we considered in our computational tests. As remarked in Section 3, large weights compromise the use of variables k_h and the solution of the corresponding ILP models.

8.3 A new exact approach for MRKP

We employ the same algorithmic framework of BKP with the following few modifications. We construct set $\mathcal{F}(\pi, x, \theta, v, \tau)$ in model $CRIT_3(c)$ by a procedure denoted as *MRKPTuples*. The procedure is very similar to *DefineTuples*, so we do not report the corresponding pseudo code. The only differences are the presence of $2n$ items in the follower's knapsack and the insertion of μ pairs of constraints (62)-(63) to set $\mathcal{F}(\pi, x, \theta, v, \tau)$ and constraints related to the adding of the critical item (Lines 6-7 in the pseudo code of *DefineTuples*).
Similarly to (28), we then define the first possible critical item q as

$$q := \min\{j : \sum_{k=1}^{j} w_{i(k)} \geq C\}. \tag{64}$$

We also test variables x_i and v_p for fixing by reduced costs and consider the following constraint instead of constraint (35) during the exploration of a subproblem $CRIT_3(c)$

$$\sum_{i:\bar{x}_i=0}^{n} x_i + \sum_{i:\bar{x}_i=1}^{n} (1 - x_i) \geq 1. \tag{65}$$

Finally notice that for MRKP we do not have to handle the absence of the critical item in the follower's knapsack problem (see section 8.1). Below is reported the pseudo code of the derived approach.

MRKP exact solution approach

Input: MRKP instance, parameters α, β, Δ, μ, γ, ω.
$z^* \leftarrow +\infty$;
Identify the candidate critical items and build models $CRIT_3(c)$:
Compute the interval of critical items $[q, 2n]$: $q \leftarrow$ apply (64);
for all c in $[q, 2n]$ **do**
    Build model $CRIT_3(c)$ by procedure *MRKPTuples(c, α, β, Δ, μ)*;
    Solve model $CRIT_3^{LP}(c)$;
end for
Sort models $CRIT_3(c)$ by increasing $z(CRIT_3^{LP}(c))$.
$\Longrightarrow$ Create a list of ordered critical items $L = \{c_1, c_2, \dots\}$;
Compute feasible MRKP solutions:
for $i = 1, \dots, \gamma$ **do**
    if $z(CRIT_3^{LP}(c_i)) < z^*$ **then** $\hat{x} \leftarrow$ solve $CRIT_3(c_i)$;
        if $z(CRIT_3(c_i)) < z^*$ **then** solve $KP(\hat{x})$;
            if $(z(KP(\hat{x})) - \sum_{i=1}^{n} p_i^- \hat{x}_i) < z^*$ **then** $x^* = \hat{x}$, $z^* = z(KP(\hat{x})) - \sum_{i=1}^{n} p_i^- \hat{x}_i$;
            end if
        end if
    end if
end for
▷ Step 2
Solve subproblems:
for all c in list L **do**
    if $z(CRIT_3^{LP}(c)) \geq z^*$ **then return** x^*; **end if**
    Fix variables x_i and v_p by reduced costs in $CRIT_3(c)$;
    $\bar{x} \leftarrow$ solve $CRIT_3(c)$;
    while $z(CRIT_3(c)) < z^*$ **do**
        solve $KP(\bar{x})$;
        if $(z(KP(\bar{x})) - \sum_{i=1}^{n} p_i^- \bar{x}_i) < z^*$ **then** $x^* = \bar{x}$, $z^* = z(KP(\bar{x})) - \sum_{i=1}^{n} p_i^- \bar{x}_i$;
        end if
        Compute subset $\bar{\tau}$;
        if $\bar{\tau}$ *is not empty* **then**
        add constraints (62)-(63), as indicated in section 8.2,
        to $CRIT_3(c)$; **end if**
        if $\bar{\tau}$ *is empty* **then**
        add constraint (65) to $CRIT_3(c)$; **end if**
        if $z(CRIT_3(c))$ *does not increase for ω iterations* **then**
        Consider the remaining tuples computed by *MRKPTuples(c, α, β, Δ, μ)*
        with a weight $\leq \theta$;
        Add alltogether at most μ pair of constraints (62)-(63) to $CRIT_3(c)$,
        see section 8.2; **end if**
        $\bar{x} \leftarrow$ solve $CRIT_3(c)$;
    end while
end for
return x^*.