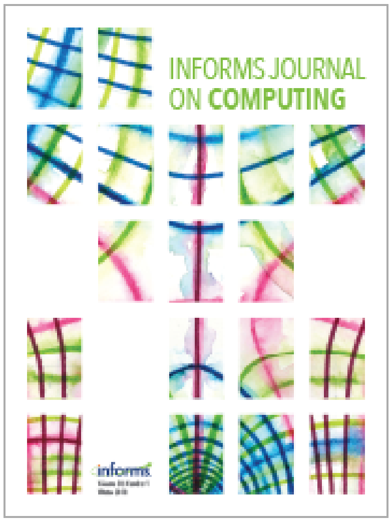

INFORMS Journal on Computing

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Bilevel Knapsack with Interdiction Constraints

Alberto Caprara†, Margarida Carvalho, Andrea Lodi, Gerhard J. Woeginger

INFORMS Journal on Computing
Vol. 28, No. 2, Spring 2016, pp. 319–333
ISSN 1091-9856 (print) | ISSN 1526-5528 (online)
http://dx.doi.org/10.1287/ijoc.2015.0676

Bilevel Knapsack with Interdiction Constraints

Alberto Caprara†
Late of University of Bologna, 40136 Bologna, Italy

Margarida Carvalho
INESC TEC and Faculdade de Ciências da Universidade do Porto, 4169-007 Porto, Portugal,
margarida.carvalho@dcc.fc.up.pt

Andrea Lodi
DEI, University of Bologna, 40136 Bologna, Italy, andrea.lodi@unibo.it

Gerhard J. Woeginger
Department of Mathematics and Computer Science, TU Eindhoven, 5600 MB Eindhoven, Netherlands,
gwoegi@win.tue.nl

We consider a bilevel integer programming model that extends the classic 0–1 knapsack problem in a very natural way. The model describes a Stackelberg game where the leader's decision interdicts a subset of the knapsack items for the follower. As this interdiction of items substantially increases the difficulty of the problem, it prevents the application of the classical methods for bilevel programming and of the specialized approaches that are tailored to other bilevel knapsack variants. Motivated by the simple description of the model, by its complexity, by its economic applications, and by the lack of algorithms to solve it, we design a novel viable way for computing optimal solutions. Finally, we present extensive computational results that show the effectiveness of the new algorithm on instances from the literature and on randomly generated instances.

Keywords: knapsack problem; bilevel programming; min-max problem
History: Accepted by Karen Aardal, Area Editor for Design and Analysis of Algorithms; received February 2015; revised August 2015; accepted September 2015. Published online April 11, 2016.

1. Introduction

In recent years, research on *mixed-integer bilevel programming* (MIBP) has grown substantially due to its wide applicability in the modeling of real-world problems. Bilevel programming and (more generally) multilevel programming are generalizations of standard single-level optimization. In the bilevel case, which is also referred to as a Stackelberg (1952) game, there are two noncooperating players who play two rounds. In the first round, the so-called leader takes action, and in the second round, the other player (called the follower) makes his decision while taking the decision of the leader into account. In the multilevel case, an entire hierarchy of n players make their decisions during a sequence of n rounds.

In this paper, we investigate a bilevel knapsack problem that was suggested in the Ph.D. thesis of DeNegre (2011), and hence will be called the *DeNegre bilevel knapsack* (DNeg). DNeg is an integer bilevel programming problem describing a situation where leader and follower hold their own private knapsacks and choose items from a common item set. First, the leader picks some of the items up to his own budget, and then the follower chooses some of the remaining items and packs them into his private knapsack. The objective of the follower is to maximize the profit of the items in his knapsack, whereas the objective of the (hostile) leader is to minimize the follower's profit by interdicting some items for the follower.

First, the leader picks some of the items up to his own budget, and then the follower chooses some of the remaining items and packs them into his private knapsack. The objective of the follower is to maximize the profit of the items in his knapsack, whereas the objective of the (hostile) leader is to minimize the follower's profit by interdicting some items for the follower.

One real-world application of DNeg is the so-called corporate strategy problem described in DeNegre (2011): a company B wishes to determine its marketing strategy

†Alberto Caprara died unexpectedly on April 2012 in a mountaineering accident. At that time the work that appears here was already in an advanced stage from the theoretical viewpoint and in progress on the computational side. Alberto's ideas, scientific touch, and vision contributed fundamentally to the way the authors decided to attack such a fascinating topic. Finishing this work and presenting to the scientific community on his behalf gives the authors a little relief for the deep loss felt on Alberto's premature death. The authors will always remember Alberto's very last afternoon at work that they had the privilege of spending together precisely on this topic.

for the upcoming fiscal year. Company B has to decide which demographic or geographic regions to target, subject to a specified marketing budget. There exists a cost to establish a marketing campaign for each target region and an associated benefit. The goal of company B is to maximize its marketing benefit. The larger company A has market dominance; whenever company A and company B target the same region, company B is unable to establish a worthwhile marketing campaign. In other words, company A can interdict regions for the marketing problem to be solved by company B.

The literature around MIBPs. The optimization literature only contains a handful of results on the solution of general MIBPs. Moore and Bard (1990) adapt the classical branch-and-bound scheme for *mixed-integer linear programming* (MIP) to MIBPs, and propose a number of simple heuristics. The approach in Moore and Bard (1990) is fairly basic and can only handle small instances with up to 20 integer variables. The first significant advances to the MIBP branch-and-bound scheme are due to the dissertation of DeNegre (2011), which added a number of interesting ingredients and, in particular, considered so-called *interdiction constraints*. For a comprehensive survey on solution methodologies for MIBPs, we refer the reader to Saharidis et al. (2013).

Hemmati et al. (2014) consider a more general bilevel min-max interdiction problem on networks. An effective cutting plane (CP) algorithm in the spirit of the one described in §3.3 is proposed and enhanced with valid inequalities that are specific to the considered problem on networks. Links to the general interdiction literature, especially from a homeland security perspective, are provided by Smith (2011) and Smith and Lim (2008).

Brotcorne et al. (2013) have studied a bilevel knapsack variant where the leader's decision interferes with the amount of budget available for the follower (but does not interdict items like DNeg). This allows the use of the traditional dynamic programming machinery for the 0–1 *knapsack problem* (KP) to compute the follower's best reactions for every possible capacity determined by the leader's strategy. Eventually, this leads to an equivalent single-level optimization formulation of pseudopolynomial size. For our problem DNeg, however, the number of possible scenarios generated by the leader grows exponentially with the number of items, so that there is no obvious way of enumerating the follower's best reactions within a reasonable amount of time. Another bilevel knapsack variant occurs in the work of Chen and Zhang (2011), where the leader's decision only interferes with the follower's objective function, but not with the follower's feasible region. This variant is computationally much easier, since the leader wants to maximize the social welfare (total profit) that leads to a coordination and alignment of the leader's and the follower's interests.

Caprara et al. (2013) show that the bilevel knapsack variant in Brotcorne et al. (2013) and the interdiction problem DNeg are Σ_2^p-complete, and hence are located at the second level of the polynomial hierarchy. This means that there is no way of formulating these problems like a single-level integer program of polynomial size *unless the polynomial hierarchy collapses* (a highly unlikely event, which would cause a revolution in complexity theory, quite comparable to the revolution that would be caused by a proof that $\mathrm{P} = \mathrm{NP}$). See also Jeroslow (1985) for more information on the polynomial hierarchy. The bilevel knapsack variant in Chen and Zhang (2011) is NP-complete, and hence can be formulated similar to a standard integer program.

Results and organization of the paper. Section 2 provides a clean mathematical programming formulation of problem DNeg, and reviews several well-known results on the 0–1 KP. In §3, we review the literature by discussing the applicability of existing algorithms to DNeg and describe a straightforward CP approach to solve the problem exactly. The ideas of this approach will be an ingredient of our algorithm (the central contribution of this paper) that we state in §4. Section 5 presents computational results on new randomly generated instances and on instances from the literature. Section 6 concludes the paper and suggests future research directions.

2. Definitions and Preliminaries

An instance of the DNeg bilevel KP looks as follows. There is a set $N = \{1, 2, \ldots, n\}$ of items, and for every item $i \in N$, there is a corresponding profit p_i, a leader's cost v_i, and a follower's cost w_i. Furthermore, there is a budget C_u for the leader and a budget C_l for the follower.

The corresponding Stackelberg game now works as follows. In the first round, the leader chooses a subset of items that fit into his own knapsack; his goal is to minimize the profit of the follower. In the second round, the follower chooses a subset of items that fit into his own knapsack and that have not been used by the leader; his goal is to maximize his own profit. This game can be modeled through the following bilevel formulation:

(DNeg)

$$\min_{(x,y)\in B^n\times B^n} \quad \sum_{i=1}^{n} p_i y_i \tag{1a}$$

$$\text{subject to} \quad \sum_{i=1}^{n} v_i x_i \le C_u, \tag{1b}$$

where $y_1, \ldots, y_n$ solves the follower's problem,

$$\max_{y\in B^n} \sum_{i=1}^{n} p_i y_i \quad \text{s.t.} \sum_{i=1}^{n} w_i y_i \le C_l \quad \text{and,} \tag{1c}$$

$$y_i \le 1 - x_i, \quad \text{for } 1 \le i \le n, \tag{1d}$$

where $B^n = \{0, 1\}^n$, and x and y are the binary decision vectors controlled by the leader and the follower, respectively. Without loss of generality, we will make the following three assumptions throughout:

$$p_i,\ v_i,\ w_i,\ C_u, \text{ and } C_l \text{ are positive integers}, \tag{2}$$

$$v_i < C_u \quad \text{and} \quad w_i < C_l, \quad \text{for all } i, \tag{3}$$

$$\sum_{i=1}^{n} v_i > C_u \quad \text{and} \quad \sum_{i=1}^{n} w_i > C_l. \tag{4}$$

As both agents work with the same objective function, the follower's reply yields the worst-possible result for the leader. Hence there is no need to distinguish between the (usual) optimistic and pessimistic cases; see, for instance, Colson et al. (2005).

In the rest of this section, we will recall some standard concepts on the classical KP, like critical items; we refer the reader to Martello and Toth (1990) for more details.

Definition 1. Assume that the items are ordered by decreasing profit-to-weight ratios as $p_1/w_1 \geq p_2/w_2 \geq \cdots \geq p_n/w_n$. The item c defined by

$$c = \min\left\{ j\colon \sum_{i=1}^{j} w_i > C_l \right\}$$

is called the *critical item* of the knapsack instance.

The famous algorithm of Dantzig (1957) for the continuous relaxation of KP will play an important role in our algorithm.

Theorem 1. *Suppose that the items are ordered similar to Definition 1. The optimal solution y^* of the continuous relaxation of problem (1c) is given by*

$$y_i^* = 1, \quad \textit{for } i = 1, \ldots, c-1,$$

$$y_i^* = 0, \quad \textit{for } i = c+1, \ldots, n,$$

$$y_c^* = \left(C_l - \sum_{i=1}^{c-1} w_i \right) \cdot \frac{1}{w_c}.$$

The following result will be used to speed up our algorithm for the bilevel knapsack with interdiction constraints.

Corollary 1. *A trivial upper bound to the KP (1c) is given by*

$$U = \sum_{i=1}^{c-1} p_i + y_c^* p_c.$$

3. On the Exact Solution of DNeg

In this section, we first review some algorithmic approaches from the literature and then propose one straightforward scheme for problem DNeg. We start with algorithms for general bilevel problems (§3.1) and for bilevel knapsack variants (§3.2). The general linear integer bilevel approaches only solve instances of small size, and the known algorithms for bilevel knapsack variants are not applicable to our problem DNeg. A natural CP method to solve DNeg is presented in §3.3 by reformulating DNeg as a single-level optimization problem.

3.1. General Mixed-Integer Bilevel Algorithms

It is a well-known fact in mixed-integer bilevel optimization research that the techniques that successfully work for (classical, single-level) MIPs are not straightforward to generalize to the bilevel case; see, for instance, DeNegre (2011) or Moore and Bard (1990). Indeed, the *bilevel linear problem* (BLP) obtained by relaxing the integrality restrictions does not provide a lower bound on the original problem and even if the solution to the BLP relaxation is integral, it is not necessarily optimal for the original problem (see DeNegre 2011 for such examples). Therefore, computing lower bounds with good quality for MIBPs is a big challenge. The usual approach is to solve the so-called *high point problem* (see, for instance, Moore and Bard 1990), which consists of dropping the follower's optimality condition and integrality constraints. This may provide good lower bounds for problems in which the upper-level objective function takes (in some way) into account the follower's reaction.

The standard general procedure for MIBPs (see Moore and Bard 1990) is similar to the branch-and-bound approaches for single-level optimization problems. In the root, the *high point problem* is solved; through branch-and-bound fixed variables to satisfy the integrality requirement and solve in each promising node the corresponding bilevel optimization problem; whenever an integer solution is computed, verify its bilevel feasibility by solving the lower-level problem for the fixed leader's decision, to obtain an upper bound (in minimization problems). Unfortunately, this approach has a big drawback: the initial lower bound is, in general, considerably far from the optimum, so that the branch-and-bound tree is likely to be extremely big. This is, for instance, pointed out in the survey by Ben-Ayed (1993) on BLP problems.

The *high point problem* (H-DNeg) for (DNeg) is defined as follows:

(H-DNeg)

$$\min_{(x,y)\in[0,1]^n\times[0,1]^n} \quad \sum_{i=1}^{n} p_i y_i \tag{5a}$$

$$\text{subject to} \quad \sum_{i=1}^{n} v_i x_i \leq C_u, \tag{5b}$$

$$\sum_{i=1}^{n} w_i y_i \leq C_l, \tag{5c}$$

$$y_i \leq 1 - x_i, \quad \text{for } 1 \leq i \leq n. \tag{5d}$$

It can be seen that this high point problem has an optimal value (OPT) of zero, and hence does not provide an interesting lower bound for solving (DNeg). Under this general approach, we would continue by standard variable branching, and once a node has an integer solution verify its bilevel feasibility (which amounts to solving a KP for the follower). A bilevel feasible solution represents an upper bound, and therefore helps to prune some nodes. Unfortunately, for all possible leader's decisions, (H-DNeg) may have its optimum equal to zero if $y = 0$ (thus, these nodes are not pruned), meaning that the method would enumerate all the possible leader's decisions. Note that the number of feasible leader's solutions is $\Theta(2^n)$, so that this all boils down to a standard brute force approach.

A *mixed-integer interdiction problem* (MIPINT) is defined as a min-max problem where for each lower-level variable, there is a corresponding binary upper-level variable and a corresponding interdiction constraint, see, for instance, Israel (1999). DeNegre (2011) considers MIPINTs, and constructs a branch-and-cut scheme by adding some new ingredients to the basic method. (In DeNegre 2011, the disjunction is stated for the general interdiction problems, but for sake of clarity, we explicitly show it here for the DNeg problem.) Consider a node t, where the optimal solution (x^t, y^t) is integer but not bilevel feasible (that is, the best follower's reaction to x^t is $\hat{y}$ with $\sum_{i=1}^{n} p_i \hat{y}_i > \sum_{i=1}^{n} p_i y_i^t$). In such a node t, the method either adds valid inequalities (cuts) such that x^t becomes infeasible (the so-called nogood cuts), or exploits the interdiction structure of the problems by branching on the following disjunction: either the leader packs a set of items such that $\sum_{i:\, x_i^t = 0} x_i \geq 1$ or the leader packs a set of items such that $\sum_{i:\, x_i^t = 0} x_i \leq 0$ and the follower has a profit $\sum_{i=1}^{n} p_i y_i \geq \sum_{i=1}^{n} p_i \hat{y}_i$. Finally, some heuristics to improve the solutions obtained through the branch-and-cut method are presented in DeNegre (2011), but these are not successful in the context of DNeg because of the conflicting structure of leader and follower goals.

In §4, we will build a method that uses this disjunction idea to solve DNeg, but in a more sophisticated and efficient way.

3.2. Knapsack Bilevel Algorithms

Brotcorne et al. (2013) consider a bilevel KP in which the decision of the leader only modifies the budget available for the follower. The algorithm in Brotcorne et al. (2013) may be summarized as follows: compute an upper bound for the follower's budget, by ignoring the resources consumed by the leader; solve the follower's 0–1 KP considering this budget bound through the standard knapsack dynamic programming approach (see, for instance, Martello et al. 1999). More precisely, the best follower's reactions for all his possible budgets from zero to the bound are computed. (Note that in this case, different decisions of the leader may yield the same subproblem for the follower.) With this, the authors are able to define the follower's best reaction set for any fixed leader's decision through linear constraints, reducing the problem to single level.

If we mimic this procedure for problem (DNeg), we would have to consider all the leader's interdictions that imply different reactions of the follower. However, in this case, for every possible decision of the leader, the follower's KP is modified in terms of the (not interdicted) items available and not in terms of his budget. Since different decisions of the leader always yield different problems for the follower, the number of Lower-level subproblems for the follower grows with the number 2^n of item subsets and hence is exponential. In short, this is the reason why the methods developed in Brotcorne et al. (2013) cannot be applied to DNeg.

3.3. CP Approach

Problem (DNeg) is equivalent to the following single-level linear optimization problem:

$$\text{(BKP)} \quad \min_{(w,\, x) \in R \times B^n} \; w \tag{6a}$$

$$\text{subject to} \quad \sum_{i=1}^{n} v_i x_i \leq C_u, \tag{6b}$$

$$w \geq \sum_{i=1}^{n} y_i p_i (1 - x_i), \quad \forall\, y \in \mathbb{S}. \tag{6c}$$

Here, $\mathbb{S}$ is the collection of all feasible packings for the follower. As the size of $\mathbb{S}$ is $O(2^n)$, the use of the CP approach is the standard method to apply; see Algorithm 3.3.1.

Note that this type of single-level reformulation works for all MIPINT problems where the lower-level optimization problem can be replaced by a set of constraints explicitly taking into account all possible reactions to the leader's strategy. Note furthermore, that this reformulation is exponential in size.

Algorithm 3.3.1 (CP)

```
k = 1
Initialize 𝕊 (e.g., with the best follower's reaction
    when there is no interdiction)
Let (w^k, x^k) be an optimal solution to (BKP) with 𝕊
y(x^k) = BestReaction (x^k)
while w^k < Σ_{i=1}^{n} p_i y_i(x^k), do
    k = k + 1
    Add constraint w ≥ Σ_{i=1}^{n} y_i(x^k) p_i (1 − x_i) to BKP
                                              // update 𝕊
    Solve BKP and let (w^k, x^k) be the optimal
        solution
    y(x^k) = BestReaction (x^k)
end while
return w, (x^k, y(x^k)).
```

4. CCLW Algorithm: A Novel Scheme

Motivated by the previous section, we propose a new approach to tackle DNeg. The algorithm initialization is studied in §4.1 by computing an upper bound on DNeg. Section 4.2 constructs a naïve iterative method for solving DNeg exactly. Then, this basic scheme is enhanced through a sequence of improvements in the following sections. One such improvements takes into account the ideas of the CP approach presented in §3.3, thus mixing the advantages of this method with ours.

4.1. An Upper Bound for DNeg

The unsuccessful search for *lower* bounds in bilevel optimization motivated us to try a completely different approach, which first computes an *upper* bound. In practice, this approach is very effective and enabled us to quickly find an optimal solution in almost all our experiments.

Theorem 2 formulates the first upper bound for DNeg that our algorithm computes. The underlying idea is simple: the set of follower's feasible strategies is extended (through the relaxation of his variables) and, consequently, the follower's profit is greater than or equal to the one obtained with the original set of strategies. This provides an upper bound to DNeg.

Theorem 2. *The optimal solution value of the following continuous-bilevel formulation provides an upper bound on the optimal solution value of problem* (DNeg).

(UB)

$$\min_{(x,y)\in B^n\times[0,1]^n} \sum_{i=1}^n p_i y_i \tag{7a}$$

$$\text{subject to } \sum_{i=1}^n v_i x_i \le C_u, \tag{7b}$$

where $y_1,\ldots,y_n$ solves the follower's problem,

$$\max_{y\in[0,1]^n} \sum_{i=1}^n p_i y_i, \quad \text{s.t.} \sum_{i=1}^n w_i y_i \le C_l \quad \textit{and} \tag{7c}$$

$$y_i \le 1 - x_i, \quad \textit{for } 1\le i\le n. \tag{7d}$$

Proof. The follower's problem (7c)–(7d) is a relaxation of problem (1c)–(1d) since the binary requirement on the y variables is removed. Therefore, given any fixed leader's interdiction x, the OPT of problem (7c)–(7d) is greater or equal than the OPT of problem (1c)–(1d), and thus provides an upper bound.

To complete the proof note that problems (DNeg) and (UB) are always bilevel feasible, which implies that (UB) always provides an upper bound to (DNeg). □

From the last proof, it is easy to see that an analogous result holds for any (general) MIPINT. Our motivation for introducing (UB) is that it can be written as a single-level MIP, thus leading to the possibility of applying effective solution methods as well as reliable software tools.

Theorem 3. *The bilevel problem* (UB) *is equivalent to the following*:

(MIP[1])

$$\min_{x\in B^n,\, z\in[0,\infty)^{n+1},\, u\in[0,\infty)^n} \left\{ z_0 C_l + \sum_{i=1}^n u_i \right\} \tag{8a}$$

$$\text{subject to } \sum_{i=1}^n v_i x_i \le C_u, \tag{8b}$$

$$u_i \ge 0, \quad \textit{for } 1\le i\le n, \tag{8c}$$

$$u_i \ge z_i - p_i x_i, \quad \textit{for } 1\le i\le n, \tag{8d}$$

$$w_i z_0 + z_i \ge p_i, \quad \textit{for } 1\le i\le n. \tag{8e}$$

Proof. The two main ingredients of our proof are the use of duality theory and the convex relaxation by McCormick (1976).

The follower's optimization problem (relaxed KP) is feasible and bounded for any x. Hence, it always has an optimal solution. In this way, according to the strong duality principle, we can write the single-level formulation equivalent to (UB) in the following way:

$$\min_{x\in B^n,\, z\in[0,\infty)^{n+1},\, y\in[0,1]^n} \sum_{i=1}^n p_i y_i \tag{9a}$$

$$\text{subject to } \sum_{i=1}^n v_i x_i \le C_u, \tag{9b}$$

$$z_0 C_l + \sum_{i=1}^n (1-x_i) z_i = \sum_{i=1}^n p_i y_i, \tag{9c}$$

$$\sum_{i=1}^n w_i y_i \le C_l, \tag{9d}$$

$$x_i + y_i \le 1, \quad \text{for } 1\le i\le n\ , \tag{9e}$$

$$w_i z_0 + z_i \ge p_i, \quad \text{for } 1\le i\le n, \tag{9f}$$

where the new variables z_i are the dual variables of the follower's relaxed KP.

Note that we can further simplify the above formulation by removing the decision vector y:

$$\min_{x\in B^n,\, z\in[0,\infty)^{n+1}} \left\{ z_0 C_l + \sum_{i=1}^n (1-x_i) z_i \right\} \tag{10a}$$

$$\text{subject to } \sum_{i=1}^n v_i x_i \le C_u \tag{10b}$$

$$w_i z_0 + z_i \ge p_i \quad \text{for } 1\le i\le n. \tag{10c}$$

Let us clarify this equivalence. Observe that any feasible solution (x^*, z^*, y^*) of (9) implies that (x^*, z^*) is feasible for (10), and thus (10) provides a lower bound to (9). In contrast, given any optimal solution (x^*, z^*) of (10), we may consider x^* fixed in the follower's relaxed KP and obtain an associated primal optimal solution y^*. This ensures that (x^*, z^*, y^*) is feasible to (9) and, in particular, optimal.

Finally, the bilinear terms $x_i z_i$ are linearized by adding the extra variables $u_i = (1 - x_i) z_i$ and the associated McCormick constraints (8c) and (8d). □

Before showing how the solution of (MIP^1) will be used to obtain an algorithm for problem (DNeg), it is worth noting that UB can be alternatively written as

$$\min_{(x,\,y)\in B^n\times[0,\,1]^n} \sum_{i=1}^{n} p_i y_i (1 - x_i)$$

$$\text{subject to } \sum_{i=1}^{n} v_i x_i \le C_u,$$

where $y_1, \ldots, y_n$ solves the follower's problem,

$$\max_{y\in[0,\,1]^n} \sum_{i=1}^{n} p_i y_i (1 - x_i) \quad \text{s.t.} \quad \sum_{i=1}^{n} w_i y_i \le C_l.$$

It is easy to verify that this is a reformulation of (UB) (same optimal solution value) and, that for any fixed vector x, we can use strong duality to obtain an equivalent single-level optimization problem. Indeed, for any fixed vector x, the interdiction constraints are embedded into the objective function, by setting to 0 the profit of all interdicted items. The advantage of this reformulation is that no variables of the leader do appear in the right-hand side of the follower's constraints, which implies that there are no bilinear terms in its dual. However, in practice, the reformulation does not have a significant impact on the computation times.

So far, we have built a mixed-integer linear problem (MIP^1) to compute an upper bound on DNeg. The first step of our algorithm is to solve (MIP^1) to optimality and to obtain the leader's decision vector x^1. This then is followed by solving the following KP, which we denote as follower's best reaction to x^1:

$$(\text{KP}^1) \qquad \max_{y\in B^n} \sum_{i=1}^{n} p_i y_i \tag{11a}$$

$$\text{subject to } \sum_{i=1}^{n} w_i y_i \le C_l, \tag{11b}$$

$$y_i \le 1 - x_i^1, \quad \text{for } 1 \le i \le n. \tag{11c}$$

Let $y(x^1)$ be an optimal solution of (KP^1). Then, $\sum_{i=1}^{n} p_i y_i(x^1)$ is our new upper bound. Figure 1 provides a pictorial illustration of the relationships between these solutions.

We will see in §5.1 that on our randomly generated test instances, $(x^1, y(x^1))$ provides a very tight approximation of the optimal solution value to DNeg. Before continuing, we note that if in the optimal solution of (UB) the follower's vector y is binary, then that solution is bilevel feasible but *not* necessarily optimal for DNeg.

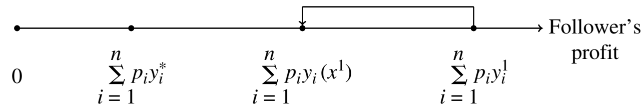

Figure 1 **Illustration of the Upper Bounds to** DNeg**, Where (x^*, y^*) Is an Optimal Solution to** DNeg**, (x^1, y^1) Is an Optimal Solution to** MIP^1**, and $(x^1, y(x^1))$ Is the Corresponding Bilevel Feasible Solution**

Example 1. Consider an instance with three items where

$$p = (4, 3, 3), \quad v = (2, 1, 1), \quad w = (4, 3, 2),$$
$$C_u = 2, \quad \text{and} \quad C_l = 4.$$

It is easy to check that the optimal solution for UB is binary with $x = (0, 1, 1)$ and $y = (1, 0, 0)$ with value 4. However, the optimal solution for DNeg has $x = (1, 0, 0)$ and $y = (0, 1, 0)$ (or $y = (0, 0, 1)$) with value 3. Indeed, when $x = (1, 0, 0)$ and the follower has the possibility of packing fractions of items, then the follower's reply is $y = (0, \frac{2}{3}, 1)$ with value 5.

4.2. Iterative Method

The basic scheme to solve problem (DNeg) is given by Algorithm 4.2.1. It consists of iteratively computing upper bounds by solving, at each iteration k, the MIP proposed in the previous section amended by a *nogood constraint* (NG$_0$) that forbids the leader to repeat his last strategy x^{k-1} (see, for instance, Balas and Jeroslow 1972 or D'Ambrosio et al. 2010):

$$\sum_{i:\,x_i^k=1} (1 - x_i) + \sum_{i:\,x_i^k=0} x_i \ge 1. \tag{12}$$

In this way, essentially the leader's strategies are enumerated until the last MIP is proven infeasible.

Algorithm 4.2.1 (Basic iterative method)

$k = 1$; $BEST = +\infty$;
Build MIP^k
while MIP^k is feasible, **do**
  $x^k = \arg\min\{\text{MIP}^k\}$
  $y(x^k) = BestReaction(x^k)$
    // solves the follower's KP by fixing x^k
  if $\sum_{i=1}^{n} p_i y_i(x^k) < BEST$, **then**
    $BEST = \sum_{i=1}^{n} p_i y_i(x^k)$;
    $(x^{BEST}, y^{BEST}) = (x^k, y(x^k))$
  end if
  MIP^{k+1} ← add (NG$_0$) in x^k to MIP^k

$$\sum_{i:\,x_i^k=1} (1 - x_i) + \sum_{i:\,x_i^k=0} x_i \ge 1$$

  $k = k + 1$
end while
$OPT = BEST$; $(x^{OPT}, y^{OPT}) = (x^{BEST}, y^{BEST})$;
return OPT, (x^{OPT}, y^{OPT}).

In Algorithm 4.2.1, function *BestReaction* receives as input the leader's decision x^k from the optimal solution of a MIP^k, and computes a rational reaction $y(x^k)$ for the follower, that is, the KP optimum to interdiction x^k. It is clear that Algorithm 4.2.1 finds an optimal solution to DNeg. However, it is a very inefficient process and a number of improvements can be applied to make it more effective in theory and in practice. More precisely, we will propose several improvements that lead to an enhanced and substantially faster version of Algorithm 4.2.1; this final version is discussed in §4.

Throughout the paper, we use the notation of Algorithm 4.2.1. The leader interdiction computed in iteration k is denoted by x^k, the follower's optimal solution to x^k is denoted by $y(x^k)$, $BEST$, and (x^{BEST}, y^{BEST}) are the minimum value and associated solution among all bilevel feasible values computed up to iteration k and OPT and (x^{OPT}, y^{OPT}) are DNeg OPT and associated solution. Denote by y^k, the follower's optimal relaxed solution to x^k, which, although not used from the algorithmic point of view, theoretically, it will play an important role.

4.3. Strengthening the Nogood Constraints

Let us first concentrate on strengthening the nogood constraints.

Definition 2. A feasible strategy x^k for the leader is maximal, if $\nexists j \in \{i\colon x_i^k = 0\}$ such that $\sum_{i=1}^n v_i x_i^k + v_j \le C_u$.

A strategy for the leader is maximal, if he does not have enough budget left to pick more items. A maximal strategy dominates an associated nonmaximal strategy, since it leaves the follower with a smaller set of options, at least one further item cannot be taken by the follower due to the interdiction constraints. Algorithm 4.3.1 takes a not necessarily maximal strategy and turns it into a maximal one.

Algorithm 4.3.1 (MakeMaximal—Complete x^k by adding the available items)

1: $Residual = C_u - \sum_{i=1}^n v_i x_i^k$
2: $i = 1$
3: **while** $i \le n$ and $Residual > 0$, **do**
4: **if** $x_i^k == 0$ and $Residual - v_i \ge 0$, **then**
5: $Residual = Residual - v_i$
6: $x_i^k = 1$
7: **end if**
8: $i = i + 1$
9: **end while**
10: **return** x^k.

Once a strategy x^k for the leader and its corresponding bilevel solution $(x^k, y(x^k))$ have been evaluated, there is no need to keep x^k feasible, because we want to concentrate in new bilevel feasible solutions potentially decreasing the follower's profit.

Definition 3. If x^k is a maximal strategy for the leader, then $\sum_{i\colon x_i^k=0} x_i \ge 1$ is called a strong maximal constraint (NG_1).

It is easy to see that a NG_1 constraint dominates a NG_0 one when both are associated with the same leader interdiction.

The strong maximal constraints can be strengthened further in the following way. Let $(x^k, y(x^k))$ denote a bilevel feasible solution for DNeg. There is no point in generating new solutions for the leader where the set of items picked by the follower in $y(x^k)$ is available, because the follower would have a profit at least as high as the previous one.

Definition 4. If x^k is a maximal strategy for the leader, then $\sum_{i\colon y_i(x^k)=1} x_i \ge 1$ is called a nogood constraint for the follower (NG_2).

It is easy to see that given a maximal strategy for the leader, the corresponding strong maximal constraint is dominated by the associated nogood constraint for the follower, as $y_i(x^k) = 1$ implies $x_i^k = 0$. If $(x^k, y(x^k))$ is not the optimal solution of DNeg, then under the strategy $y(x^k)$, the follower is packing an item interdicted in any optimal solution. This establishes the validity of the nogood constraints for the follower.

Thus, at each iteration k of the algorithm in which the (standard) nogood cuts are replaced by the follower's nogood cuts, either an optimal solution has already been obtained or any optimal strategy for the leader satisfies all the follower's nogood constraints already added. This shows the correctness of the substitution of (standard) nogood with follower's nogood constraints.

A further strengthening of the follower's nogood constraints can be achieved by paying close attention to the CP approach described in §3.3.

Theorem 4. *Consider an iteration k of Algorithm 4.2.1. If $BEST$ is not the* OPT *of problem* (DNeg), *then there is an optimal admissible interdiction x^* for the leader such that*

$$\sum_{i=1}^n p_i y_i (1 - x_i^*) \le BEST - 1, \quad \forall\, y \in B^n$$

$$\text{such that } \sum_{i=1}^n w_i y_i \le C_l. \tag{13}$$

Proof. Let (x^*, y^*) be an optimal solution of DNeg. Then

$$\sum_{i=1}^n y_i p_i (1 - x_i^*) \le \sum_{i=1}^n p_i y_i^* \quad \forall\, y\colon \sum_{i=1}^n w_i y_i \le C_l.$$

Moreover, if $BEST$ at iteration k is not an OPT of DNeg, then $\sum_{i=1}^n p_i y_i^* \le BEST - 1$. □

With the help of Theorem 4, it is easy to derive the following new type of valid constraints, to be introduced in each iteration k to strengthen MIP^k:

(NG$_3$) cutting plane constraint

$$\sum_{i=1}^{n} y_i(x^k)p_i(1-x_i) \leq BEST - 1. \tag{14}$$

In this way, whenever $BEST$ is updated in the iterative procedure, also the right-hand sides of the previous CP constraints are updated.

It is easy to show that a CP constraint dominates a follower's nogood constraint when associated with the same leader interdiction. Indeed, after solving MIP^k in an arbitrary iteration k, a best reaction of the follower to x^k is computed and then it is checked whether this leads to a better solution for DNeg. At that point, the following inequality holds:

$$\sum_{i=1}^{n} p_i y_i(x^k) \geq BEST.$$

Hence, to satisfy the associated CP constraint

$$\sum_{i=1}^{n} y_i(x^k)p_i(1-x_i) \leq BEST - 1,$$

the leader must interdict at least one item packed with the strategy $y(x^k)$.

Next, the general dominance of the CP constraints over the remaining presented ones is established.

Proposition 1. *Consider Algorithm 4.2.1 amended by making the leader's strategy maximal* (*after step* 4) (*call it Algorithm*$_0$) *and replacing the nogood constraint* (*step* 10) *by*

—*Algorithm*$_1$: *the strong maximal constraint;*
—*Algorithm*$_2$: *the follower's nogood constraint;*
—*Algorithm*$_3$: *the CP constraint.*

Assume that if in an iteration k, Algorithm$_2$ *and Algorithm*$_3$ *have a common optimal interdiction* x^k, *then both select* x^k *and the same associated best reaction* $y(x^k)$. *Then, for* $i = 1, 2, 3$, *Algorithm*$_i$ *returns the optimal solution after a number of iterations less or equal than Algorithm*$_{i-1}$.

Proof. For Algorithm$_i$ denote as MIP^{k,i} and $F^{k,i}$ the optimization problem MIP^k and the associated feasible region for the leader maximal interdictions at iteration k. Define $F^{k,i}$ as equal to the empty set if Algorithm$_i$ had returned the optimal solution in a number of iterations less or equal to k. Denote $x^{k,i}$ as the leader optimal solution to MIP^{k,i}.

For each Algorithm$_i$, note that the purpose of each iteration k is to cut off nonoptimal leader's maximal interdictions, therefore it is enough to concentrate on the set $F^{k,i}$. In other words, it is sufficient to show that $F^{k,i} \subseteq F^{k,i-1}$ holds for any iteration k since it directly implies that Algorithm$_i$ enumerates a less or equal number of bilevel feasible solutions in comparison with Algorithm$_i$. We will prove that this result holds for $i = 1, 2$ through induction in k.

In the first iteration, $k = 1$, all algorithms solve the same MIP^1, and thus $F^{1,2} = F^{1,1} = F^{1,0}$.

Next, assume that $F^{m,i} \subseteq F^{m,i-1}$ holds for $m = k$. The induction hypothesis implies that the optimal solution value of MIP$^{m,i-1}$ is a lower bound to MIP^{m,i}.

Recall that we have argued before that for the same leader interdiction: the nogood constraint is dominated by the strong maximal constraint; the strong maximal constraint is dominated by the follower's nogood constraint.

By contradiction, suppose that $F^{m+1,i} \not\subseteq F^{m+1,i-1}$. This implies the existence of $x \in F^{m+1,i}$ such that $x \notin F^{m+1,i-1}$. Since $F^{m+1,i} \subset F^{m,i} \subseteq F^{m,i-1}$, then $x \in F^{m,i-1}$. Therefore x only violates the additional constraint of $F^{m+1,i-1}$ associated with $F^{m,i-1}$. This is only possible if x is the optimal solution of MIP$^{m,i-1}$. Because MIP$^{m,i-1}$ provides a lower bound to MIP^{m,i} and $x \in F^{m,i}$, x is the optimal solution of MIP^{m,i}. However, this means that x will be cut off from $F^{m,i}$, and thus $x \notin F^{m+1,i}$, leading to a contradiction.

It remains to prove that Algorithm$_3$ finishes in a number of iterations less or equal than Algorithm$_2$. To this end, the following assumption is necessary.

As mentioned before, in the first iteration MIP^{1,2} = MIP^{1,3}, and thus, by the proposition assumption, $y(x^{1,2}) = y(x^{1,3})$. This fact implies that MIP^{2,2} = MIP^{2,3} since $BEST = \sum_{i=1}^{n} p_i y_i(x^{1,2})$ means that the NG$_3$ constraint is equivalent to NG$_2$ regarding $y(x^{1,2})$. Moreover, $y(x^{2,2}) = y(x^{3,2})$ and, consequently, the associated NG$_3$ constraint dominates NG$_2$. We conclude that $F^{3,3} \subseteq F^{3,2}$. At this point, Algorithm$_3$ has advantage over Algorithm$_2$ because the set of interdictions $F^{3,3}$ is at most as large as $F^{2,3}$. Note that if there is an iteration $k \geq 3$ such that $y(x^{k,3}) \neq y(x^{k,2})$, then Algorithm$_3$ is reducing the set of feasible interdictions through NG$_3$ associated with $y(x^{k,3})$ and Algorithm$_3$ might end up computing $y(x^{k,2})$ latter on in an iteration $m > k$, which shows that Algorithm$_3$ progresses more or as fast as Algorithm$_2$. □

We conclude this section with two observations. First, the improvements described above are purely based on the fact that we are dealing with an interdiction problem. Hence, any type of interdiction problem for which we can prove an adaptation of Theorem 3 can be attacked by the basic iterative method with CP constraints. Secondly, all constraints described so far depend solely on the decision variables of the leader. Therefore the statement of Theorem 3 also applies to all improvements, and each MIP^k is equivalent to a bilevel optimization problem in which the follower solves a relaxed knapsack problem.

4.4. Stopping Criteria

Our next goal is to add a condition for the whole algorithm to stop. Let $p_{\max} = \max_{i=1,\dots,n} p_i$.

Proposition 2. *At an iteration k of the basic iterative method BEST cannot be decreased in the current and forthcoming iterations if*

$$\sum_{i=1}^{n} p_i y_i^{BEST} + p_{\max} \le \sum_{i=1}^{n} p_i y_i^k.$$

Proof. Let OPT be the OPT to DNeg and assume that the condition in the proposition holds. For any leader's optimal solution x^*, Corollary 1 implies that the OPT of the follower's continuous knapsack with interdiction x^* lies within the interval

$$[OPT, OPT + p_{\max}]. \tag{15}$$

Because y^{BEST} is the follower's strategy corresponding to the best solution computed up to iteration k, obviously,

$$\sum_{i=1}^{n} p_i y_i^{BEST} + p_{\max} \ge OPT + p_{\max}.$$

Then, $\sum_{i=1}^{n} p_i y_i^k$ is not in the range (15), which implies that x^k is not an optimal interdiction. Furthermore, since the OPT of the MIPs is monotonically increasing with the algorithm iterations, none of the upcoming iterations returns a leader's optimal solution. □

In other words, the quantity $p_{\max}$ is an upper bound on the amount by which the continuous solution value of any follower's reaction can decrease. If $\sum_{i=1}^{n} p_i y_i^k - p_{\max}$ is already bigger than the current incumbent solution value, then no further improvement is possible since (of course) $\sum_{i=1}^{n} p_i y_i^{k+1} \ge \sum_{i=1}^{n} p_i y_i^k$.

4.5. Saving Some Knapsack Computations

In an iteration k of our algorithm, the leader's interdiction just built may lead to an improvement if the following necessary condition holds. The following observation follows from Corollary 1.

Proposition 3. *At an iteration k, the pair $(x^k, y(x^k))$ does not decrease BEST if*

$$\sum_{i=1}^{n} p_i y_i^k - p_{c^k} y_{c^k}^k \ge \sum_{i=1}^{n} p_i y_i^{BEST},$$

where c^k is the critical item for the follower's continuous knapsack with interdiction x^k.

Thus, whenever the above condition is violated, we do not need to compute the best reaction by solving the associated 0–1 knapsack. Our next goal is to embed the condition of Proposition 3 as a constraint inside MIP^k. For that purpose, the following lemma and theorem will be crucial. Lemma 1 follows from Corollary 1.

Lemma 1. *Let x^k be a leader's interdiction. Then,*

$$\sum_{i=1}^{n} p_i y^k - \sum_{i=1}^{n} p_i y_i(x^k) \le p_{c^k}.$$

Note that p_{c^k} provides yet another upper bound on the value of the improvement due to *BestReaction*. The following theorem makes the upper bound independent of the critical item computation. Let $w_{\max} = \max_{i=1,\dots,n} w_i$.

Theorem 5. *Let x^k be a leader's interdiction. Then, for the corresponding follower's relaxed rational reaction to x^k, there exists a dual solution that satisfies*

$$z_0^k w_{\max} \ge \sum_{i=1}^{n} p_i y_i^k - \sum_{i=1}^{n} p_i y_i(x^k).$$

Proof. By Theorem 1, there exists a solution in which *at most one* entry of y^k is not binary in the relaxed rational reaction to x^k; furthermore, if such an entry does exist, then its value equals $y_{c^k}^k$. By strong duality, there is a corresponding optimal dual solution with $z_{c^k}^k = 0$. The c^k dual constraint (9f) implies

$$z_0^k w_{c^k} \ge p_{c^k} \;\Rightarrow\; z_0^k w_{\max} \ge z_0^k w_{c^k} \ge p_{c^k}.$$

By using Lemma 1, we get

$$z_0^k w_{\max} \ge z_0^k w_{c^k} \ge p_{c^k} \ge \sum_{i=1}^{n} p_i y^k - \sum_{i=1}^{n} p_i y_i(x^k).$$

Otherwise, if all follower's variables are binary

$$\sum_{i=1}^{n} p_i y^k - \sum_{i=1}^{n} p_i y_i(x^k) = 0 \le z_0^k w_{\max}$$

because $z_0^k \ge 0$. □

To use the upper bound derived above, the following proposition establishes yet another necessary condition, which is similar in spirit to Proposition 3.

Proposition 4. *At an iteration k, BEST will not decrease if*

$$\sum_{i=1}^{n} p_i \lfloor y_i^k \rfloor > BEST - 1.$$

In other words, if we round down the relaxed rational reaction of the follower to the leader strategy x^k, then the resulting feasible solution for the follower has a profit strictly smaller than the best bilevel feasible bound known. Because of Theorem 5,

$$\sum_{i=1}^{n} p_i y^k - \sum_{i=1}^{n} p_i y_i(x^k) \le \sum_{i=1}^{n} p_i y^k - \sum_{i=1}^{n} p_i \lfloor y_i^k \rfloor \le p_{c^k} \le z_0^k w_{\max},$$

and it is easy to see that also the following holds:

$$z_0^k C_l + \sum_{i=1}^{n} \overbrace{(1 - x_i^k) z_i^k}^{u_i^k} - z_0^k w_{\max} \le BEST - 1. \tag{16}$$

The following theorem turns condition (16) into an inequality that can be added to MIP^k.

Theorem 6. *In the end of iteration k, the strong cut*

$$z_0 C_l + \sum_{i=1}^{n} u_i - z_0 w_{\max} \le BEST - 1$$

is valid for MIP^{k+1}.

Proof. The dual of the follower's relaxed problem (D_FKP) with the introduction of the strong cut (and replacing u_i) is

$$\text{(D\_FKP)} \quad \min_{z \ge 0} \left\{ z_0 C_l + \sum_{i=1}^{n} (1 - x_i^k) z_i \right\} \tag{17a}$$

$$\text{subject to} \quad w_i z_0 + z_i \ge p_i, \quad \text{for } 1 \le i \le n. \tag{17b}$$

$$z_0 C_l + \sum_{i=1}^{n} (1 - x_i^k) z_i - z_0 w_{\max} \le BEST - 1, \tag{17c}$$

and the follower's relaxed problem is

$$\text{(P\_FKP)} \quad \max_{y \ge 0} \left\{ \sum_{i=1}^{n} y_i p_i - (BEST - 1) y_{n+1} \right\} \tag{18a}$$

$$\text{subject to} \quad \sum_{i=1}^{n} y_i w_i - (C_l - w_{\max}) y_{n+1} \le C_l, \tag{18b}$$

$$y_i - (1 - x_i^k) y_{n+1} \le 1 - x_i^k, \quad \text{for } 1 \le i \le n. \tag{18c}$$

Essentially, we are dealing with a new item $n+1$ whose profit $-(BEST - 1)$ and weight $-(C_l - w_{\max})$ are negative. We will show that no optimal solution will use this new item: then $y_{n+1}^k = 0$ holds, and the prior primal problem collapses to the previous continuous KP for which the critical item exists. Hence, let us first ignore the new item and solve the continuous knapsack as before. Let c be the critical item, and let S be the set of (indices of) items that are fully taken. Then, clearly,

$$\frac{\sum_{i \in S} p_i}{\sum_{i \in S} w_i} \ge \frac{p_c}{w_c}.$$

Moreover, by Proposition 4, we may assume $\sum_{i \in S} p_i \le BEST - 1$. Finally, $\sum_{i \in S} w_i \ge C_l - w_{\max}$ because otherwise c would not be the critical item. Altogether, this yields

$$\frac{BEST - 1}{C_l - w_{\max}} \ge \frac{\sum_{i \in S} p_i}{\sum_{i \in S} w_i} \ge \frac{p_c}{w_c}.$$

As the profit-to-weight ratio of the new item is at least as large as the profit-to-weight ratio of the critical item and because profit and weight of the new item are negative, the new item will not be used in an optimal solution. □

In §5, we will show that this cut is crucial in practice, because it significantly reduces the number of leader interdictions in the enumeration. This is the reason why the iterative approach is currently superior to the CP approach. It is relatively easy to embed additional conditions to reduce the search space of the iterative approach, whereas additional cutting planes to enhance CP seem difficult to be developed.

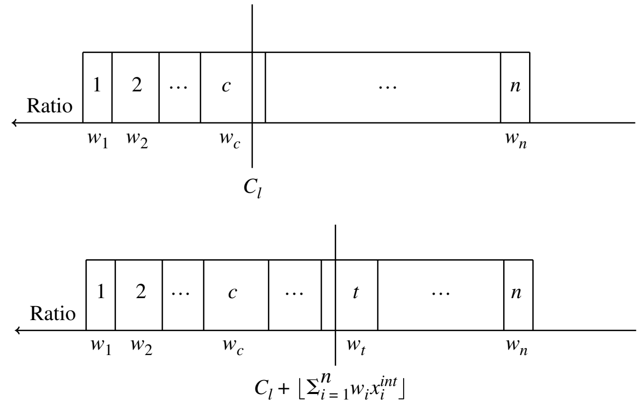

Figure 2 **Illustration of the Follower's Preferences When His Knapsack Is Relaxed: Items from 1 to $c-1$ and from $t+1$ to n Are Never Critical**

4.6. Preprocessing

For the approach developed so far, it is crucial to compute good upper bounds to the profit and weight of the items that may act as a critical item. We describe a preprocessing routine that tightens these bounds, and hence leads to a stronger approach.

Recall that in this context, we are dealing with the relaxed KP for the follower. Suppose that the follower could pack all the items from 1 to $c-1$ as illustrated in Figure 2. Since the follower has incentive to fully pack the available items from 1 to $c-1$, these items can never be critical. Another interesting observation is that some of the less valuable items for the follower are never packed by him, and hence are not critical: this occurs because the follower uses his entire budget on the most valuable available items. All in all, we are interested in computing a bound on the maximum follower's weight interdicted by the leader. This trivially can be achieved by solving the following relaxed KP:

$$x^{\text{int}} = \arg\max_{x \in [0,1]^n} \sum_{i=1}^{n} w_i x_i \tag{19a}$$

$$\text{subject to} \quad \sum_{i=1}^{n} v_i x_i \le C_u. \tag{19b}$$

Therefore the leader interdicts at most $\lfloor \sum_{i=1}^{n} w_i x_i^{\text{int}} \rfloor$ of the total available weight of the follower. It is easy to see from Figure 2 that the items from $t+1$ to n are never critical. In conclusion, with $t = \min\{j\colon C_l + \lfloor \sum_{i=1}^{n} w_i x_i^{\text{int}} \rfloor \le \sum_{i=1}^{j} w_i\}$, we have

$$p_{\max} = \max_{i=c,\ldots,t} p_i \quad \text{and} \quad w_{\max} = \max_{i=c,\ldots,t} w_i.$$

The running time of this preprocessing is $O(n \log n)$, and hence slightly more expensive than the simple $O(n)$ procedure by computing $p_{\max}$ and $w_{\max}$ by taking all n items.

We could improve these bounds even further by adding so-called sensitive intervals for identifying the critical item candidates; see Brotcorne et al. (2013). However, this comes at the cost of adding more constraints to our MIPs. For that reason, we will apply this improvement only to the very hard instances as explained in the last paragraphs of §5.1.

4.7. Caprara-Carvalho-Lodi-Woeginger Algorithm (CCLW) Algorithm

Our main algorithm is summarized in Algorithm 4.7.1. For ease of reference, we call it the Caprara-Carvalho-Lodi-Woeginger Algorithm (CCLW).

Algorithm 4.7.1 CCLW

```
Compute p_max, w_max according to the
    Preprocessing
k = 1; BEST = +∞;
Build MIP^k
while MIP^k is feasible, do
  x^k = arg min{MIP^k}
  if BEST + p_max ≤ OPT of
      MIP^k (= Σ_{i=1}^n p_i y_i^k), then
    STOP;
  else
    x^k = MakeMaximal(x^k)
    y(x^k) = BestReaction(x^k) // solves the
                follower's KP by fixing x^k
    if Σ_{i=1}^n p_i y_i(x^k) < BEST, then
      BEST = Σ_{i=1}^n p_i y_i(x^k);
      (x^BEST, y^BEST) = (x^k, y(x^k))
      MIP^{k+1} ← if k = 1 add strong cut
      z_0 C_l + Σ_{i=1}^n u_i − z_0 w_max ≤ BEST − 1,
      otherwise, update the right-hand side of
        the strong cut and NG_3s with BEST − 1.
    end if
    MIP^{k+1} ← add NG_3 in y(x^k) to the MIP^k:
    Σ_{i: y_i(x^k)=1} p_i (1 − x_i) ≤ BEST − 1
  end if
  k = k + 1
end while
OPT = BEST; (x^OPT, y^OPT) = (x^BEST, y^BEST);
return OPT, (x^OPT, y^OPT).
```

5. Computational Results

In this section, we computationally evaluate the algorithms from the preceding section in two phases. First, in §5.1, we compare CCLW with CP. There we also discuss the importance of the main ingredients of algorithm CCLW, as well as the structural difficulty of bilevel knapsack instances regarding our algorithms. Secondly, in §5.2, we compare CCLW with the results of DeNegre (2011) and DeNegre and Ralphs (2009).

All algorithms have been coded in Python 2.7.2, and each MIP has been solved with Gurobi 5.5.0. The experiments were conducted on a Quad-Core Intel Xeon processor at 2.66 GHz and running under Mac OS X 10.8.4.

5.1. Method Comparisons

In this section, CP and CCLW are compared against each other. Moreover, we discuss the structural difficulty of bilevel knapsack instances regarding the performance of CCLW.

Generation of instances. For building the follower's data, we have used the knapsack generator described in Martello et al. (1999); the profits p_i and weights w_i are taken with uncorrelated coefficients from the interval $[0, 100]$. For each value n, 10 instances were generated; these instances are available upon request from the second author. According to Martello et al. (1999), the budget C_l is set to $\lceil (\text{INS}/11) \sum_{i=1}^{n} w_i \rceil$ for the instance number instance identifier (INS). The leader's data v_i and C_u all were generated by using Python's random module; see Python Software Foundation (2012). In particular, v_i and C_u were chosen uniformly at random from $[0, 100]$ and $[C_l - 10, C_l + 10]$, respectively.

Note that if the leader's budget is significantly smaller than the follower's budget, then there are fewer feasible solutions for the leader and the instance would be easier. In contrast, if the leader's budget is significantly bigger than the follower's budget, then all the items may be packed by leader and follower together and again the instance would be easier. We will see below that CCLW is very efficient for these cases.

CP vs. CCLW. In an attempt of asserting the importance of each ingredient of algorithm CCLW, we performed some tests with its basic scheme (Algorithm 4.2.1). It turned out that within one hour of CPU time, the Basic Scheme can only solve instances with up to 15 items. Although this is comparable to the size of problems reported in DeNegre (2011), DeNegre and Ralphs (2009) (discussed in detail in §5.2), CP and CCLW can go much higher in terms of number of items. For this reason, no detailed results for Algorithm 4.2.1 are reported here.

Table 1 reports the results of algorithms CP and CCLW. For each instance, the table shows the number of items ($n \in \{35, 40, 45, 50\}$), the INS, and the OPT. For algorithm CP, we further report the number of CP iterations (#It.s), and the CPU time in seconds (time), but for algorithm CCLW, we report the value of MIP[1] (ObjF), the number of iterations (#MIPs), the iteration in which the optimal solution has been found (OPT_{iter}), and the CPU time in seconds (time). Finally, for algorithm CCLW we also report some data on the mostexpensive MIP solved, namely, the CPU time in

Table 1 **Comparison between CP and CCLW**

			CP		CCLW					
n	INS	OPT	#It.s	Time	ObjF	#MIPs	OPT_{iter}	Time	WMIP time	WMIP nodes
35	1	279	16	0.34	288.07	14	2	0.79	0.05	14
	2	469	40	1.59	474.00	33	1	2.57	0.09	171
	3	448	253	55.61	455.88	203	1	40.39	0.50	1,635
	4	370	397	495.50	374.56	11	1	1.48	0.14	363
	5	467	918	[451]	472.00	5	2	0.72	0.19	660
	6	268	155	71.43	268.00	2	1	0.06	0.03	0
	7	207	298	144.46	207.00	2	1	0.06	0.03	0
	8	41	11	0.25	41.00	2	1	0.04	0.01	0
	9	80	25	0.97	80.00	2	1	0.03	0.00	0
	10	31	8	0.12	31.00	2	1	0.03	0.00	0
40	1	314	24	0.66	326.12	21	1	1.06	0.05	60
	2	472	77	6.67	483.78	67	2	7.50	0.19	805
	3	637	338	324.61	644.78	244	1	162.80	2.52	4,521
	4	388	530	1,900.03	396.56	3	1	0.34	0.13	165
	5	461	653	[457]	466.18	2	1	0.22	0.15	66
	6	399	534	2,111.85	399.00	2	1	0.09	0.04	0
	7	150	254	83.59	150.00	2	1	0.05	0.02	0
	8	71	33	1.73	71.00	2	1	0.04	0.01	0
	9	179	404	137.16	179.00	2	1	0.08	0.03	4
	10	0	2	0.03	0.00	2	1	0.03	0.00	0
45	1	427	45	1.81	434.60	33	1	2.37	0.08	74
	2	633	97	13.03	642.36	74	1	11.64	0.25	903
	3	548	845	[547]	558.69	387	1	344.01	2.86	10,638
	4	611	461	[566]	624.84	108	1	38.90	1.01	8,611
	5	629	462	[568]	630.00	15	7	3.42	0.30	1,179
	6	398	639	3,300.76	398.00	2	1	0.07	0.03	0
	7	225	141	60.43	225.00	2	1	0.04	0.01	0
	8	157	221	60.88	157.00	2	1	0.05	0.01	0
	9	53	23	0.83	53.00	2	1	0.05	0.01	0
	10	110	11	0.40	110.00	2	1	0.05	0.01	0
50	1	502	58	2.86	514.12	39	1	4.55	0.12	114
	2	788	733	1,529.16	798.00	695	2	1,520.56	7.29	6,352
	3	631	467	[612]	638.47	212	1	105.59	2.03	7,909
	4	612	310	[586]	621.04	17	1	3.64	0.32	954
	5	764	287	[657]	768.88	3	1	0.60	0.27	369
	6	303	385	1,046.85	303.00	2	1	0.05	0.01	0
	7	310	617	2,037.01	310.00	2	1	0.09	0.04	0
	8	63	49	2.79	63.00	2	1	0.05	0.01	0
	9	234	717	564.97	234.00	2	1	0.10	0.05	3
	10	15	5	0.09	15.00	2	1	0.04	0.01	0

seconds (WMIP time) and the number of nodes (WMIP nodes). The algorithms had a limit of one hour to solve each instance. The entries in square brackets mark the cases where algorithm CP reached the time limit, and in such cases, we report the lower bound value instead of the computing time.

The results in Table 1 clearly illustrate that algorithm CCLW is superior to algorithm CP. In particular, CCLW usually finds an optimal solution within two iterations, which shows that, in practice, we will find the optimum very early and the only challenge is to prove optimality. Looking at the number of MIPs solved and at the computing times, we observe that for any number of items, algorithm CCLW is extremely powerful, for instances, with INS ≥ 5. An optimal solution is computed by MIP^1 and optimality is proved by MIP^2, except in three cases with INS $= 5$. Considering the way in which the instances are generated, the next theorem shows that this behavior is structural.

Theorem 7. *If for any leader's maximal interdiction, the follower can pack the remaining items, then* CCLW *solves* DNeg *in two iterations.*

Proof. Given that the follower is able to pack all the items left by any maximal interdiction of the leader, we get that the follower's budget constraint is not binding. In particular, the solution of the follower's relaxed problem to any leader's maximal interdiction is binary. Hence the MIPs' OPTs are bilevel feasible and the DNeg optimum is consequently found in the first iteration of CCLW.

In the second iteration, MIP^2 uses the additional strong cut

$$z_0 C_l + \sum_{i=1}^{n} u_i - z_0 w_{\max} \leq BEST - 1.$$

The dual variable z_0 corresponds to the follower's budget constraint (7c). Initially noted, constraint (7c) is not binding, which together with strong duality, implies that the associated optimal dual solution has $z_0 = 0$. However, with $z_0^2 = 0$, the strong cut imposes

$$\sum_{i=1}^{n} u_i^2 \leq BEST - 1.$$

This means that the OPT of MIP^2 is strictly better then the value obtained in MIP^1. But this is absurd, because MIP^2 equals MIP^1 plus an additional constraint (the strong cut). Consequently, MIP^2 is infeasible, and CCLW stops in the second iteration. □

As INS increases its value, larger budget capacities are associated with the leader and the follower. Therefore, it is likely that these instances fall into the condition of Theorem 7.

Strength of the CCLW Ingredients. To evaluate the effectiveness of CCLW main algorithmic ingredients, we have performed two additional sets of experiments. First, we considered what happens to the basic enumerative scheme (Algorithm 4.2.1) if it is strengthened by the nogood cuts described in §4.3. The results are reported in Table 2 for instances with $n \in \{30, 35\}$.

Table 2 **Algorithm 4.2.1 with Strengthened Nogood Constraints**

n	INS	OPT	ObjF	#MIPs	OPT_{iter}	Time	WMIP time	WMIP nodes
30	1	272	282.80	13	2	0.27	0.02	9
	2	410	423.29	34	1	0.95	0.04	223
	3	502	513.63	110	1	10.56	0.28	1,036
	4	383	385.00	151	2	36.65	1.06	7,094
	5	308	308.00	301	1	121.27	1.85	7,730
	6	223	223.00	239	1	44.22	0.81	5,580
	7	146	146.00	121	1	8.32	0.15	1,072
	8	88	88.00	70	1	2.03	0.05	281
	9	113	113.00	83	1	2.71	0.07	674
	10	82	82.00	73	1	1.99	0.04	276
35	1	279	288.07	19	2	0.72	0.04	16
	2	469	474.00	53	1	3.20	0.08	524
	3	448	455.88	303	1	102.23	1.31	2,673
	4	370	374.56	474	1	1,203.90	19.49	74,265
	5	467	472.00	1,152	2	tl	9.30	26,586
	6	268	268.00	234	1	222.66	5.78	35,510
	7	207	207.00	471	1	321.08	3.97	28,962
	8	41	41.00	42	1	1.24	0.04	49
	9	80	80.00	98	1	5.28	0.09	285
	10	31	31.00	33	1	0.85	0.03	9

The results in Table 2 show that this (simple) strengthening already allows us to double the size of the instances that the basic scheme can settle (recall the discussion at the beginning of the previous section). More precisely, all instances with 30 items can be solved to optimality in rather short computing times, whereas size 35 becomes troublesome.

If we compare these results to the corresponding results in Table 1, we note that the number of MIPs needed to prove optimality is much bigger, in particular, for the cases INS ≥ 3. This behavior becomes dramatic for INS ≥ 5, where CCLW generally proves optimality in two iterations (as suggested by Theorem 7), whereas the improved version of the basic scheme still needs a large number of iterations. The difference in behavior seems to be mainly caused by the strong cut described in §4.5.

This observation is also confirmed by our second set of experiments, in which we removed the strong cut from algorithm CCLW. The corresponding results are reported in Table 3. Indeed, the results in Table 3 illustrate that without the strong cut, the number of MIPs required by CCLW blows up significantly. The algorithm is only slightly better (because of the stopping criteria, see §4.4) than the basic scheme with strengthened nogood cuts (see Table 2).

Solving Large(r) Instances. What are the computational limits of Algorithm CCLW? How does it scale to larger values of n? Table 4 provides some partial answers to these questions by displaying the results for CCLW on instances with 55 items. Again, we see that MIP^1 is very effective in computing the leader's strategy, like in most of the cases, we obtain the optimal DNeg solution already at iteration 1. In general, the machinery discussed in the previous sections seems to be able to keep the enumeration of leader strategies under control: CCLW succeeds in solving all but two instances. The two exceptions are the instances with INS $\in \{3, 4\}$, on which CCLW exceeded its time limit of one hour of CPU time (the "tl" entries in the table).

For the most challenging instances, we implemented a preprocessing step based on the idea of computing

Table 3 **CCLW Without the Strong Cut**

n	INS	OPT	ObjF	#MIPs	OPT_{iter}	Time	WMIP time	WMIP nodes
35	1	279	288.07	14	2	0.89	0.04	16
	2	469	474.00	33	1	1.76	0.05	207
	3	448	455.88	218	1	43.27	0.50	1,443
	4	370	374.56	277	1	216.96	2.40	14,651
	5	467	472.00	1,152	2	tl	9.26	26,586
	6	268	268.00	59	1	3.76	0.10	756
	7	207	207.00	202	1	25.86	0.27	1,667
	8	41	41.00	21	1	0.62	0.03	49
	9	80	80.00	30	1	1.06	0.04	207
	10	31	31.00	2	1	0.03	0.00	0

Table 4 CCLW Computational Results on Instances with $n = 55$

			CCLW					
n	INS	OPT	ObjF	#MIPs	OPT_{iter}	Time	WMIP time	WMIP nodes
55	1	480	489.21	103	2	18.57	0.37	1,090
	2	702	706.15	419	1	443.53	4.33	11,097
	3	778	783.67	926	1	tl	8.85	21,491
	4	889	899.34	787	1	tl	14.67	41,813
	5	726	726.00	2	1	0.24	0.13	158
	6	462	462.00	2	1	0.09	0.04	0
	7	370	370.00	2	1	0.08	0.03	0
	8	387	387.00	2	1	0.10	0.04	0
	9	104	104.00	2	1	0.06	0.01	0
	10	178	178.00	2	1	0.06	0.02	0

sensitive intervals (as done in Brotcorne et al. 2013). Ideally, in each iteration k of CCLW, we would like to know the profit p_{c^k} of the critical item in the optimal solution for the follower's continuous knapsack. (Recall Theorem 5, which shows that $z_0^k w_{\max}$ is an upper bound on p_{c^k} in each iteration k.) To reach this goal, we compute sensitive intervals with the function

$$\phi(Z_0^+ \to Z_0^+)\colon\ \sum_{i=1}^{c} w_i x_i \to \max_{i=c',\ldots,t} p_i, \tag{20}$$

where $c' = \min\{j\colon \sum_{i=1}^{c} w_i x_i + C_l \le \sum_{i=1}^{j} w_i\}$. In this way, instance INS = 4 in Table 4 was solved within the time limit. The computation took 2,796.20 CPU seconds (roughly half an hour), and the speedup was mainly due to a strong reduction in the number of MIPs (693 versus *at least* 787). In principle, sensitivity interval preprocessing could achieve the same kind of reduction in all considered instances. Note, however, that this preprocessing adds five constraints and up to n binary variables to every MIP solved by CCLW. Hence there is a trade-off between performing fewer iterations and working with larger MIPs, and this is also the reason why we decided not to include sensitivity interval preprocessing in the standard version of CCLW: it slightly slows down the computing time, whereas only few additional hard instances can be solved with it. (Note that it does not manage to solve the instance $n = 55$ and INS = 3 to optimality.)

All in all, we conclude that new algorithmic ideas will be needed to attack the hard instances with INS ≤ 4 for larger values of n. For instance, for $n = 100$, computation times of one hour CPU time (as we reached for the smaller instances in this section) seem currently out of reach.

5.2. Literature Comparison

DeNegre (2011) and DeNegre and Ralphs (2009) solved knapsack interdiction instances by using the branch-and-cut procedure described in §3. These authors present two branching strategies: maximum infeasibility and strong branching. We compare our method CCLW against these two procedures in Table 5 (the instances have kindly been provided by the authors of DeNegre 2011, DeNegre and Ralphs 2009). The data in the table averages more than 20 instances, and the computing times for DeNegre and Ralphs (2009) refer to an Intel Xeon 2.4 GHz processor with 4 GB memory. A "—" indicates that due to memory requirements, no instance of the corresponding size was solved.

Table 5 Summary of Results for Instances in DeNegre (2011), DeNegre and Ralphs (2009)

	Branch-and-cut DeNegre and Ralphs (2009)		
Variable	Maximum infeasibility	Strong branching	CCLW
n	Avg CPU time	Avg CPU time	Avg CPU time
10	3.17	4.69	0.009
11	6.63	9.13	0.009
12	13.27	17.50	0.009
13	27.54	35.84	0.010
14	60.08	71.90	0.011
15	124.84	145.99	0.011
16	249.19	296.16	0.014
17	516.65	—	0.013

Although it is always difficult to compare different computing codes running on different computers, we believe that from the results in Table 5, it is safe to conclude that, for these instances, CCWL outperforms the branch-and-cut method. In particular, the highest average number of branch-and-bound nodes explored by Gurobi for solving the MIPs is 4.55 for the instances with $n = 16$, thus the impact of the parallelism associated with our computing platform to be Quad-Core is negligible. We noticed that in all the instances introduced in DeNegre (2011), DeNegre and Ralphs (2009), CCLW executes only two iterations and the optimum is always found in the first iteration. The second iterations are only needed to prove optimality, because leader and follower have enough capacity to pack all the items. Theorem 7 shows that in these cases, the strong cut makes MIP^2 infeasible.

6. Conclusions

We have analyzed a special class of interdiction problems and proposed an exact algorithm for solving it. Our method uses a new way of generating (enumerating) solutions, which seems to hit the optimal solution at a very early stage, and thus allows us to concentrate on techniques for proving optimality. This behavior is quite different from classical branch-and-bound methods, which usually starts from infeasible (superoptimal) solutions and apply extensive enumerations. Of course, the classical branch-and-bound scheme has proven very effective for classical MIPs, whereas our results might indicate that this is not the case for MIBPs. Furthermore, we introduce a new cut for the leader's variables, which seems to be much stronger than the ones used in the literature and which significantly decreased the number of enumerated bilevel feasible solutions. Also, cuts limiting the objective function range had a big impact in speeding up the method.

We were able to solve instances with up to 100 binary variables, which is significantly larger than the size of instances solved in the literature. Our method is very efficient on instances where leader and follower have a large budget. Consequently, the challenging and hard instances are those in which the budget of leader and follower forces them to evaluate a large number of strategies.

The comparison of our algorithm CCLW with the best ones from the literature demonstrates its advantage, and stresses the importance that problem-specific algorithms currently have in solving bilevel programming. A promising line for future research on general interdiction problems is to exploit the follower's integrality relaxation; this is in harsh contrast to the classical high point relaxation where the follower is forgotten as a decision maker.

Acknowledgments

The authors thank Scott DeNegre and Ted Ralphs for inspiring discussions on the topic and for sharing the instances of the problem. The authors also thank the three anonymous referees for a very careful reading and useful remarks that led to a significant improvement of the paper. This work was supported by a STSM Grant from COST Action [TD1207]. The second author acknowledges the support of the Portuguese Foundation for Science and Technology through a Ph.D. program [Grants SFRH/BD/79201/2011, POPH/FSE]. The third author acknowledges the support of MIUR, Italy [Grant PRIN2009].

References

Balas E, Jeroslow R (1972) Canonical cuts on the unit hypercube. *SIAM J. Appl. Math.* 23:61–69.

Ben-Ayed O (1993) Bilevel linear programming. *Comput. Oper. Res.* 20:485–501.

Brotcorne L, Hanafi S, Mansi R (2013) One-level reformulation of the bilevel knapsack problem using dynamic programming. *Discrete Optim.* 10:1–10.

Caprara A, Carvalho M, Lodi A, Woeginger GJ (2013) A complexity and approximability study of the bilevel knapsack problem. Goemans M, Correa J, eds. *Integer Programming and Combinatorial Optimization*, Lecture Notes in Computer Science, Vol. 7801 (Springer, Berlin), 98–109.

Chen L, Zhang G (2011) Approximation algorithms for a bi-level knapsack problem. Wang W, Zhu X, Du D-Z, eds. *Combinatorial Optimization and Applications*, Lecture Notes in Computer Science, Vol. 6831 (Springer, Berlin), 399–410.

Colson B, Marcotte P, Savard G (2005) Bilevel programming: A survey. *4OR* 3:87–107.

D'Ambrosio C, Frangioni A, Liberti L, Lodi A (2010) On interval-subgradient and no-good cuts. *Oper. Res. Lett.* 38:341–345.

Dantzig GB (1957) Discrete-variable extremum problems. *Oper. Res.* 5:266–288.

DeNegre S (2011) Interdiction and discrete bilevel linear programming. Ph.D. thesis, Lehigh University, Bethlehem, PA. http://gradworks.umi.com/34/56/3456385.html.

DeNegre S, Ralphs TK (2009) A branch-and-cut algorithm for integer bilevel linear programs. Chinneck JW, Kristjansson B, Saltzman MJ, eds., *Operations Research and Cyber-Infrastructure*, Operations Research/Computer Science Interfaces, Vol. 47 (Springer, New York), 65–78.

Hemmati M, Smith JC, Thai MT (2014) A cutting-plane algorithm for solving a weighted influence interdiction problem. *Comput. Optim. Appl.* 57:71–104.

Israel E (1999) System interdiction and defense. Ph.D. thesis, Naval Postgraduate School, Monterey, CA.

Jeroslow R (1985) The polynomial hierarchy and a simple model for competitive analysis. *Math. Programming* 32:146–164.

Martello S, Toth P (1990) *Knapsack Problems: Algorithms and Computer Implementations* (John Wiley & Sons, Inc., New York).

Martello S, Pisinger D, Toth P (1999) Dynamic programming and strong bounds for the 0–1 knapsack problem. *Management Sci.* 45:414–424.

McCormick GP (1976) Computability of global solutions to factorable nonconvex programs: Part I—Convex underestimating problems. *Math. Programming* 10:147–175.

Moore JT, Bard JF (1990) The mixed integer linear bilevel programming problem. *Oper. Res.* 38:911–921.

Python Software Foundation (2012) Python Language Reference, version 2.7. Accessed March 7, 2016, http://docs.python.org/library/random.html.

Saharidis GKD, Conejo AJ, Kozanidis G (2013) Exact solution methodologies for linear and (mixed) integer bilevel programming. Talbi E-G, Brotcorne L, eds. *Metaheuristics for Bi-Level Optimization*, Studies in Computational Intelligence, Vol. 482 (Springer, Berlin), 221–245.

Smith JC (2011) Basic interdiction models. Cochran J, ed. *Wiley Encyclopedia of Operations Research and Management Science* (John Wiley & Sons, Inc., Hoboken, NJ), 323–330.

Smith JC, Lim C (2008) Algorithms for network interdiction and fortification games. Chinchuluun A, Pardalos P, Migdalas A, Pitsoulis L, eds. *Pareto Optimality, Game Theory and Equilibria*, Springer Optimization and Its Applications, Vol. 17 (Springer, New York), 609–644.

Stackelberg HV (1952) *The Theory of the Market Economy* (Oxford University Press, Oxford, UK).