A POLYNOMIAL ALGORITHM FOR A CONTINUOUS BILEVEL KNAPSACK PROBLEM

Margarida Carvalho
Andrea Lodi
Patrice Marcotte

May 2017

DS4DM-2017-006

A polynomial algorithm for a continuous bilevel knapsack problem

Margarida Carvalho [*] Andrea Lodi [†] Patrice Marcotte [‡]

submitted: 11 May, 2017

Abstract

In this note, we analyze a bilevel interdiction problem, where the follower's program is a parametrized continuous knapsack. Based on the structure of the problem and an inverse optimization strategy, we propose for its solution an algorithm with worst-case complexity $O(n^2)$.

Keywords— Bilevel programming, Continuous knapsack problem, Polynomial time.

1 Model

Recently, a number of papers have been devoted to bilevel programs involving integer-valued knapsacks at the lower level [2]–[6],[9], all of them NP-hard. A question that arises naturally is the complexity of the continuous variants of these problems. More precisely, we consider a *continuous bilevel knapsack problem with interdiction constraints* that consists in a relaxation of the problem analyzed by Caprara et al.[4], where the goal of the leader is to minimize the value of the follower's knapsack, through the interdiction of a subset of items. Its mathematical formulation is

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

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

$$\text{where } y_1,\dots,y_n \text{ solves the follower's problem}$$

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

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

First, with the aim of minimizing the follower's profit, the leader decides for each item i the fraction x_i that is interdict to the follower subject to the budget constrain (1b). Afterwards the follower reacts with the aim of maximizing profit by selecting for each item i a fraction y_i to be packed that respects constraint (1d), the leader's interdiction, and subject to a budget constraint (1c).

[*]IVADO Fellow, Canada Excellence Research Chair, École Polytechnique de Montréal, C.P. 6128, Succursale Centre-Ville, Montréal, QC H3C 3A7, Canada.

[†]Canada Excellence Research Chair, École Polytechnique de Montréal, C.P. 6128, Succursale Centre-Ville, Montréal, QC H3C 3A7, Canada.

[‡]DIRO, Université de Montréal, C.P. 6128, Succursale Centre-Ville, Montréal, QC H3C 3J7, Canada.

Replacing the lower level problem by its necessary and sufficient KKT conditions, one obtains a single level reformulation as the bilinear program (the reader is referred to [4] for details):

$$
\begin{aligned}
(CBK1) \qquad & \min_{x\in[0,1]^n,\ z\in[0,\infty)^{n+1}} && z_0 C_l + \sum_{i=1}^{n} z_i - \sum_{i=1}^{n} x_i z_i && \text{(2a)}\\
& \text{subject to} && \sum_{i=1}^{n} v_i x_i \le C_u && \text{(2b)}\\
& && w_i z_0 + z_i \ge p_i \qquad \text{for } 1 \le i \le n, && \text{(2c)}
\end{aligned}
$$

whose worst case complexity is not obvious, given that its objective is bilinear. This question will be settled by proposing an $O(n^2)$ algorithm, whose validity actually extends to problems involving distinct linear objectives at both levels of decision making.

2 A polynomial algorithm

Let us focus on the follower's knapsack, and assume that items are sorted in decreasing order of the profit-weight ratios p_i/w_i. An optimal solution is then trivially obtained by selecting items in the natural order of indices, until the budget C_l is exhausted, and declare item c *critical* if it corresponds to the smallest one (or rather, its index) that does not fit entirely (see, e.g., [7]). The value of x_c is then fractional or equal to 0.

For a given upper level decision x, the optimal objective of the follower's continuous knapsack is

$$
\sum_{i=1}^{c-1} p_i(1-x_i^*) + p_c \frac{C_l - \sum_{i=1}^{c-1} w_i(1-x_i^*)}{w_c},
$$

where c is the critical item corresponding to x^*.

Conversely, if the lower level solution y is fixed, inverse (bilevel) optimization consists in optimizing the upper level problem under the constraint that y is lower-level optimal. In our context, rather than fixing the y vector, we only assume knowledge of the critical index c. For given c this yields the leader's tri-knapsack problem

$$
\begin{aligned}
(CBK_c) \qquad & \min_{x} && \sum_{i=1}^{c-1} p_i(1-x_i) + p_c \frac{C_l - \sum_{i=1}^{c-1} w_i(1-x_i)}{w_c} && (3)\\
& \text{subject to} && \sum_{i=1}^{c-1} v_i x_i \le C_u && (4)\\
& && \sum_{i=1}^{c-1} w_i(1-x_i) \le C_l && (5)\\
& && w_c + \sum_{i=1}^{c-1} w_i(1-x_i) \ge C_l && (6)\\
& && 0 \le x_i \le 1 \qquad \text{for } 1 \le i \le c-1 && (7)
\end{aligned}
$$

where constraints (5) and (6) guarantee that item c is indeed critical. Algorithm CBK (Figure 1) consists in solving CBK_c for $c \in [1,\ldots,n]$ and retaining the best solution.

Theorem 1 *Algorithm CBK computes the optimal solution of the continuous bilevel knapsack problem with interdiction constraints in polynomial time.*

Proof. Algorithm CBK enumerates over all possible critical indices. If an optimal critical index exists, the algorithm determines the corresponding best upper level solution by solving CBK_c, which is the solution of the original problem. At Step 13, the algorithm checks the case when no index is critical, i.e., the budget of the follower allows herself to pack all remaining items. In this situation, it is optimal for the leader to pack the items according to her own utility[1], i.e., by decreasing order of the ratio $\frac{p_i}{v_i}$. Since exactly $n+1$ linear programs CBK must be solved, it follows that the continuous bilevel knapsack problem with interdiction constraints is polynomially solvable. □

Algorithm CBK

```
1: $best = +\infty$; $k = -1$
2: for $c = 1, \ldots, n$ do
3:   if $CBK_c$ feasible then
4:     // $c$ is the critical item
5:     $val := CBK_c$ and let $x^*$ be the optimal solution
6:     if $val < best$ then
7:       $best := val$
8:       $x_{best} := x^*$
9:       $k := c$
10:    end if
11:  end if
12: end for
13: $val := \min_{x \in [0,1]^n, \sum_{i=1}^n v_i x_i \le C_u, \sum_{i=1}^n w_i(1-x_i) \le C_l} \sum_{i=1}^n p_i(1-x_i)$ and let $x^*$ be the optimal solution
14: if feasible then
15:   if $val < best$ then
16:     $best := val$
17:     $x_{best} := x^*$
18:   end if
19: end if
20: return $(best, x_{best})$
```

Figure 1: Algorithm to solve CBK.

We next show that constraints (5) and (6) are automatically satisfied if x is upper-level optimal, hence they can be ignored throughout Algorithm CBK. The proof relies on the following two lemmate.

Lemma 1 *If for a given $1 \le c \le n$ and a leader's interdiction x, it holds $w_c + \sum_{i=1}^{c-1} w_i(1-x_i) < C_l$, then*

$$\sum_{i=c}^{k-1} p_i + \frac{p_k}{w_k}\left(C_l - \sum_{i=1}^{c-1} w_i(1-x_i) - \sum_{i=c}^{k-1} w_i\right) < \frac{p_c}{w_c}\left(C_l - \sum_{i=1}^{c-1} w_i(1-x_i)\right),$$

where $c < k \le n+1$ is the critical item associated with the follower's optimal solution to x.

Proof. Since items are sorted in nonincreasing order of their prize-weight ratios, there holds

$$\sum_{i=c}^{k-1}\left(\frac{p_i}{w_i} - \frac{p_k}{w_k}\right) w_i < \sum_{i=c}^{k-1}\left(\frac{p_c}{w_c} - \frac{p_k}{w_k}\right) w_i.$$

Since k is the critical item, we have $C_l - \sum_{i=1}^{c-1} w_i(1-x_i) > \sum_{i=c}^{k-1} w_i$. Those two inequalities yield

$$\sum_{i=c}^{k-1}\left(\frac{p_i}{w_i} - \frac{p_k}{w_k}\right) w_i < \left(\frac{p_c}{w_c} - \frac{p_k}{w_k}\right)\left(C_l - \sum_{i=1}^{c-1} w_i(1-x_i)\right).$$

Upon rearrangement of the terms, the result follows. □

[1]This observation is clarified by the following example: $n = 3, C_u = C_l = 2$, $p = \{100, 60, 50\}$, $v = \{2, 1, 1\}$, $w = \{1, 1, 1\}$, in which the leader's optimal solution is $x = (\frac{1}{2}, 1, 0)$ (or $x = (0, 1, 1)$).

Lemma 2 *If for a given $1 \le c \le n$ and a leader's interdiction x, it holds $\sum_{i=1}^{c-1} w_i(1-x_i) > C_l$, then*

$$\frac{p_k}{w_k}\left(C_l - \sum_{i=1}^{k-1} w_i(1-x_i)\right) < \sum_{i=k}^{c-1} p_i(1-x_i) + \frac{p_c}{w_c}\left(C_l - \sum_{i=1}^{c-1} w_i(1-x_i)\right),$$

where $1 \le k < c$ is the critical item associated with the follower's optimal solution to x.

Proof. The proof is similar to that of Lemma 1. Since items are sorted in non-increasing order of their prize-weight ratio, there holds

$$\sum_{i=k}^{c-1}\left(\frac{p_k}{w_k} - \frac{p_c}{w_c}\right) w_i(1-x_i) < \sum_{i=k}^{c-1}\left(\frac{p_i}{w_i} - \frac{p_k}{w_k}\right) w_i(1-x_i).$$

Since k is the critical item, we have $C_l - \sum_{i=1}^{k-1} w_i(1-x_i) < \sum_{i=k}^{c-1} w_i(1-x_i)$. Those two inequalities yield

$$\left(\frac{p_k}{w_k} - \frac{p_c}{w_c}\right)\left(C_l - \sum_{i=1}^{k-1} w_i(1-x_i)\right) < \sum_{i=k}^{c-1}\left(\frac{p_i}{w_i} - \frac{p_k}{w_k}\right) w_i(1-x_i).$$

Upon rearrangement of the terms, the result follows. □

Theorem 2 *The optimal solution of the continuous bilevel knapsack problem with interdiction constraints can be solved in $O(n^2)$ time.*

Proof. From Lemmate 1 and 2, it follows that constraints (5) and (6) can be ignored. Since the resulting relaxed program is a continuous knapsack that can be solved in linear time, the overall algorithm reduces to the solution of $n+1$ continuous knapsacks. Since each knapsack can be solved in linear time [1], the results follows. □

Example 1 *Consider the following instance: $n = 3$, $C_u = 2$, $C_l = 3$, $p = \{10, 8, 6\}$, $v = \{3, 1, 1\}$ and $w = \{4, 4, 4\}$.*

- *Items are sorted in non-increasing order of the ratios p_i/w_i.*
- *In accordance with Algorithm CBK, Step 2, one computes the leader's optimal objective value consistent with critical item $c \in \{1, 2, 3\}$.*
- *For $c = 1$, $val = 10 \times \frac{3}{4} = 7.5$ and the 'if' at Step 3 is true. Thus, $best = 7.5$ and $x_{best} = (0, 0, 0)$.*
- *If $c = 2$, $val = \frac{20}{3}$ and the 'if' in step 3 is true. Step 6 is also true, and thus, $best = \frac{20}{3}$ and $x_{best} = (\frac{2}{3}, 0, 0)$.*
- *If $c = 3$, $val = \frac{43}{6}$ and the 'if' in step 3 is true. In step 6, $val > best$.*
- *At Step 13, no item is critical, leading to $val = 10$ and $x^* = (0, 1, 1)$. However, the follower is not able to fully pack item 1, 'if' at Step 14.*
- *The optimal solution value is $best = \frac{20}{3}$ with $x_{best} = (\frac{2}{3}, 0, 0)$.*

If in this example we had $v = \{3, 1, 3\}$ and $w = \{2, 2, 2\}$, Algorithm CBK would evolve as in Table 1 and the optimal solution value would be $best = \frac{35}{3}$ with $x_{best} = (\frac{1}{3}, 1, 0)$.

	Critical c			
	1	2	3	step 13
val	15	$\frac{38}{3}$	$\frac{35}{3}$	$\frac{38}{3}$
x^*	(0,0,0)	($\frac{2}{3}$,0,0)	($\frac{1}{3}$,1,0)	($\frac{1}{3}$,1,0)
best	15	$\frac{38}{3}$	$\frac{35}{3}$	$\frac{35}{3}$

Table 1: Algorithm CBK computations.

We close this section by noting that, as Example 1 illustrates, val is not monotone either increasing or decreasing, with respect to the critical index. This suggests that no further improvement is possible, although we could not prove this conjecture. In the same vein, we note that the leader's preference with respect to item i depends on the value $-\frac{p_i}{v_i} + \frac{p_c w_i}{w_c v_i}$, which in turn depends on the current iteration, so that reoptimization with respect to previously computed preference is not trivial.

3 Generalization

In this section we extend the analysis to the case where the leader's objective is arbitrary, say

$$\sum_{i=1}^{n} c_i^x x_i + \sum_{i=1}^{n} c_i^y y_i.$$

Since the follower's optimization problem is still a continuous knapsack, our line of proof still applies if one replaces the objective function in CBK_c by

$$\sum_{i=1}^{n} c_i^x x_i + \sum_{i=1}^{c-1} c_i^y (1 - x_i) + c_c^y \frac{C_l - \sum_{i=1}^{c-1} w_i(1-x_i)}{w_c},$$

which we designate by CB_c. However, we note that in this more general case, constraints (5) and (6) cannot be ignored, as the following example illustrates.

Example 2 *Consider the following data:* $n = 7$, $C_u = 97$, $C_l = 98$, $c^x = \{0,0,0,0,0,0,0\}$, $c^y = \{1,1,1,1,1,1,1\}$, $v = \{14, 37, 26, 10, 30, 49, 95\}$ *and* $w = \{1, 2, 82, 86, 52, 57, 63\}$. *Essentially, the leader aims at minimizing the number of items selected by the follower, not their total value.*

If constraints (5) and (6) are ignored, one of them is violated at every iteration of the algorithm, which implies that no critical item can be found. On the other hand, if these constraints are included (see Table 2), Algorithm CBK finds the optimal value best $= 1.15$ *and the corresponding optimal solution* $x_{best} = (1, 1, 0.85, 0, 0, 0, 0)$.

	Critical c							
	1	2	3	4	5	6	7	step 13
val	inf	inf	inf	1.15	1.15	1.16	1.69	2.85
$best$	$+\infty$	$+\infty$	$+\infty$	1.15	1.15	1.15	1.15	1.15

Table 2: Algorithm CBK computations. Entries with 'inf' correspond to CBK_c infeasible.

In summary, Theorem 1 still holds. A linear program with n variables and a fixed number of constraints can be solved in $O(n)$ time through multidimensional search [8, 5]. Therefore, CB_c can be solved in $O(n)$ time for each $c = 1, \dots, n+1$, since it has 3 constraints. Thus, this problem generalization can be solved in $O(n^2)$ time.

References

[1] E. Balas, E. Zemel. An algorithm for large zero-one knapsack. *Operations Research* **28**: 1130-1154, 1980.

[2] L. Brotcorne, S. Hanafi, R. Mansi. Dynamic programming for the bilevel knapsack problem. *Operations Research Letters* **37**: 215-218, 2009.

[3] L. Brotcorne, S. Hanafi, R. Mansi. One-level reformulation of the bilevel Knapsack problem using dynamic programming. *Discrete Optimization* **10**: 1–10, 2013.

[4] A. Caprara, M. Carvalho, A. Lodi, G.J. Woeginger. Bilevel knapsack with interdiction constraints. *INFORMS Journal on Computing* **28**: 319–333, 2016.

[5] Clarkson, K. L. Linear programming in $O(n \times 3^{d^2})$ time. *Journal Information Processing Letters* **22**: 21–24, 1986.

[6] S. Dempe, K. Richter. Bilevel programming with knapsack constraints. *Central European Journal of Operations Research* **8**: 93-107, 2000.

[7] Martello, S., P. Toth. Knapsack problems: algorithms and computer implementations. John Wiley & Sons, Inc., New York, NY, USA. 1990.

[8] Megiddo, N. Linear programming in linear time when the dimension is fixed. *Journal of the ACM* **31**: 114–127, 1984.

[9] Qiu, X. and W. Kern. Improved approximation algorithms for a bilevel knapsack problem. *Theoretical Computer Science* **595**: 120–129, 2015.