Accepted Manuscript

A dynamic programming algorithm for the bilevel knapsack problem

Luce Brotcorne, Saïd Hanafi, Raïd Mansi

PII: S0167-6377(09)00006-6
DOI: 10.1016/j.orl.2009.01.007
Reference: OPERES 5222

To appear in: *Operations Research Letters*

Received date: 5 February 2008
Accepted date: 9 January 2009

Please cite this article as: L. Brotcorne, S. Hanafi, R. Mansi, A dynamic programming algorithm for the bilevel knapsack problem, *Operations Research Letters* (2009), doi:10.1016/j.orl.2009.01.007

A Dynamic Programming algorithm for the Bilevel Knapsack Problem

Luce Brotcorne, Saïd Hanafi and Raïd Mansi[1]

LAMIH-ROI, ISTV2, UVHC-Le Mont Houy, 59313 Valenciennes cedex9

(luce.brotcorne, said.hanafi, raid.mansi) @ univ-valenciennes.fr

Abstract. We propose an efficient dynamic programming algorithm for solving a bilevel program where the leader controls the capacity of a knapsack, and the follower solves the resulting knapsack problem. We propose new recursive rules and show how to solve the problem as a sequence of two standard knapsack problems.

Keywords. Knapsack Problem; Bilevel Programming; Dynamic Programming.

[1]The list of authors on page one is the arbitrary alphabetical sequence.

2 Corresponding author: Luce Brotcorne, Luce.Brotcorne@univ-valenciennes.fr, LAMIH-ROI, Université de Valenciennes et du Hainaut Cambrésis, Le Mont Houy, F59313 Valenciennes, France

1. Introduction

Bilevel programs (see Colson et al. [2] for a survey) allow the modeling of situations in which a decision-maker, hereafter the leader, optimizes his objective by taking the followers' response to his decisions explicitly into account. In the case of the Bilevel Knapsack Problem (*BKP*), the leader determines the knapsack's capacity in order to maximize his profit, while the follower faces a 0-1 knapsack problem [9] involving the capacity set by the leader. *BKP* is well suited to model the "right financing problem", where an individual (the leader) shares its capital between a savings account with fixed rate of return and a more risky investment, through an intermediary, such as a bank or a broker. The intermediary (the follower) i) buys shares or bonds to maximize his revenue while respecting the leader's budget constraint (knapsack constraint) ii) obtains a return from its own investments. Similar applications occur in the field of revenue management [11], where a company may decide on the number of units to sell by itself or to hand over to an intermediary.

The *BKP* is a mixed integer bilevel program introduced by Dempe and Richter [5], who proposed for its solution a branch-and-bound framework. In this paper, we first extend Dempe and Richter's necessary and sufficient conditions for the existence of an optimal solution. Next, we propose a simple and efficient dynamic programming algorithm for its solution. In contrast with the approach of Dempe and Richter, where a list of undominated solutions is maintained, only the objective function values for both the leader and the follower are saved throughout dynamic programming process. Feasibility is implicitly controlled in the course of the algorithm.

Under some assumptions, the *BKP* can be formulated as a sequence of knapsack problems involving binary variables, one for each integer leader variable. This yields an algorithm that outperforms by orders of magnitude alternative approaches based on branch-and-bound.

2. The Bilevel Knapsack Problem: formulation and properties

In a knapsack of continuous capacity y, controlled by the leader, we assign to each item j a weight a_j, a revenue c_j for the follower, and a revenue d_j for the leader. The unit knapsack capacity cost is denoted by t. Given y, the follower selects a subset of items that respects the capacity constraint. This yields the bilevel program

$$(BKP)\begin{cases} \underset{y,x}{Max}\ f^1(y,x) = dx + ty \\ \quad s.t.\quad \underline{b} \le y \le \bar{b} \\ \qquad \underset{x}{Max}\ f^2(x) = cx \\ \qquad s.t.\quad ax \le y \ \ and \ \ x \in \{0,1\}^n \end{cases}$$

where a, c and d are integer-valued and all data a, c, d, $\underline{b}$ and $\bar{b}$ are non-negative. Throughout the paper, we denote by

$$S = \left\{(x,y) \in \{0,1\}^n \times [\underline{b}, \bar{b}] : ax \le y\right\}$$

the constraint region, by

$$P(y) = \left\{x \in Arg\max\left\{cx' : ax' \le y, x' \in \{0,1\}^n\right\}\right\}$$

the follower's rational reaction set (for fixed y), and by

$$IR = \left\{(x,y) \middle| (x,y) \in S, x \in P(y)\right\}$$

the inducible region over which the leader optimizes his objective function.

Bilevel programs exist in two variants [7]. In the *optimistic case*, whenever the rational reaction set is not a singleton, the follower implements the solution that maximizes the objective of the leader. The corresponding solutions is called a *strong solution*. In the *pessimistic* case, the leader assumes that, whenever he is facing ties, the follower selects the solution that minimizes the leader's objective, yielding a *weak solution*.

Proposition 1 (Dempe and Richer [5]) If the unit investment cost t is non-positive, then an optimal solution exists for *BKP*.

We note that, if *t* is positive, the *BKP* may fail to have an optimal solution, as illustrated in the following example:

$$\begin{cases} \underset{y,x}{Max}\ f^1(y,x) = 5x_1 + x_2 + x_3 + x_4 + y \\ \quad s.t.\ \ 1 \le y \le 4 \\ \qquad \underset{x}{Max}\ f^2(x) = 4x_1 + 5x_2 + 10x_3 + 15x_4 \\ \qquad\quad s.t.\quad x_1 + 2x_2 + 3x_3 + 4x_4 \le y \\ \qquad\qquad x \in \{0,1\}^4 \end{cases}$$

where $f^1(y,x)$ is a linear piecewise discontinuous function in the leader variable *y* (*see Figure 1*). For *y* belonging to [1, 2[, the follower always selects the first object with unit capacity, and the leader's objective function becomes $f^1(y, x) = 5 + y$. In the same way, when $y \in [2,4[$, $f^1(y, x) = 1 + y$. Although the leader's objective function $f^1(x, y)$ is bounded from above by 7 ($\lim_{y \xrightarrow{<} 2} f^1(y,x) = 7$),this bound is not achieved.

****** [[Insert figure 1 about here]] *****

The next proposition links the capacity cost with the leader's "revenue/weight" ratio.

Proposition 2: Let $\underline{b} = 0$ and $t < 0$. If $|t| > \max_{1 \le j \le n}\left(d_j / a_j\right)$, then $(y^*, x^*) = (0, 0_n)$ is an optimal solution.

Proof:

Let (*y*, *x*) be a feasible solution for the *BKP*. First, multiplying the constraint $ax \le y$ by *t* ($t < 0$) yields $(ta+d)\ x \ge ty + dx = f^1(y, x)$. Second, since $ta_j + d_j < 0$ for $j=1,..,n$ and $x \in \{0,1\}^n$, it follows that $(ta+d)\ x \le 0$, and thus $f^1(y,x) \le 0$. Finally, since $(y^*, x^*) = (0, 0_n)$ is a feasible solution for the *BKP* in which $f^1(y, x) = 0$. This solution is not only feasible, but also optimal. □

If $\infty > \bar{b} \ge \sum_{i=1}^{n} a_i$ and $t>0$, then the optimal solution x^* is trivially equal to (1,..,1) and $y^* = \bar{b}$. If *d* and *c* are collinear ($d = \alpha c$, where $\alpha > 0$) and $t \ge 0$, then solving the *BKP* is equivalent to solving the follower's knapsack problem with capacity $\bar{b}$.

Definition: A discrete bilevel knapsack problem (*BKPd*) is a bilevel knapsack problem in which the leader variable are discrete.

Proposition 3

If $t \le 0$, then any solution (y^*, x^*) that is optimal for *BKPd* is also optimal for *BKP*.

- If $t > 0$, and if an optimal solution exists for *BKP*, then it is also optimal for *BKPd*.

Proof:

Case 1: According to Theorem 1, an optimal solution, (y^*, x^*) exists. In addition, $IR(BKPd) \subset IR(BKP)$ and, according to *Theorem* 4 by [5], y^* is integer.

Case 2: Straightforward from (ii) in Proposition 1.□

It follows from Proposition 3 that solving the *BKP* is equivalent to solving *BKPd* whenever *t* is negative. If *t* is positive, an optimal solution (see Proposition 1), whenever it exists, is achieved at a point $(\bar{b}, x^*)$ where $x^* \in P(\bar{b})$. Note that an optimal solution always exists for *BKPd*. Thus, if a *BKP* has an optimal solution, it can be obtained by solving a sequence of knapsack problems involving binary variables, one for each feasible value of *y*. The algorithm that we propose, described in the next section, exploits this property.

3. A dynamic programming algorithm for *BKP*

Our dynamic algorithm *DPBK* is a two-phase procedure that can address both the optimistic and pessimistic cases, and has worst-case complexity $\theta(n\bar{b})$. For simplicity, we only discuss the optimistic case.

According to Proposition 3, solving the *BKP* is equivalent to solving a follower knapsack problem for each integer capacity value *y* belonging to $[\underline{b}, \bar{b}]$. In the course of the algorithm, the recursions concurrently take

into account both objective functions. This is accomplished in the two phases described below.

3. 1. Forward Phase

The forward phase is composed of two loops: an outer loop related to steps $k \in [1..n]$ and an inner loop related to the integer capacities $y \in [\underline{b}, \bar{b}]$. During this phase, two tables are generated. The first one stores the optimal follower values

$$f_k^2(y) = \max\left\{\sum_{j=1}^{k} c_j x_j : \sum_{j=1}^{k} a_j x_j \le y, x \in \{0,1\}^k\right\}$$

and the second one the optimal leader values

$$\tilde{f}_k^1(y) = \max\left\{\sum_{j=1}^{k} d_j x_j : x \in P(y)\right\},$$

at each step k and for each capacity y. To construct these dynamic programming tables, the recursion is performed for all values of y between 0 and $\bar{b}$, for each item. Note that the leader's function $\tilde{f}_k^1(y)$ does not take the knapsack capacity cost into account, and thus $f_k^1(y) = \tilde{f}_k^1(y) + ty$.

Forward procedure

For $k = 2$ ***to*** n ***do***
For $y = 0$ ***to*** $\bar{b}$ ***do***
If $y < a_k$ ***then***
$f_k^2(y) = f_{k-1}^2(y)$ ***and*** $\tilde{f}_k^1(y) = \tilde{f}_{k-1}^1(y)$
Otherwise
$f_k^2(y) = \max\left(f_{k-1}^2(y), f_{k-1}^2(y-a_k) + c_k\right)$ (1)
If $f_{k-1}^2(y) \neq f_{k-1}^2(y-a_k) + c_k$ ***then***

$$\tilde{f}_k^1(y) = \begin{cases} \tilde{f}_{k-1}^1(y) & \text{if } f_k^2(y) = f_{k-1}^2(y) \\ \tilde{f}_{k-1}^1(y-a_k) + d_k & \text{if } f_k^2(y) = f_{k-1}^2(y-a_k) + c_k \end{cases} \quad (2)$$

End if
If $f_{k-1}^2(y) = f_{k-1}^2(y-a_k) + c_k$ ***then***
$\tilde{f}_k^1(y) = \max\left(\tilde{f}_{k-1}^1(y), \tilde{f}_{k-1}^1(y-a_k) + d_k\right)$ (***Opt.***) (3)
$\tilde{f}_k^1(y) = \min\left(\tilde{f}_{k-1}^1(y), \tilde{f}_{k-1}^1(y-a_k) + d_k\right)$ (***Pes.***) (4)
End if
End if
End for

In the first step (*i.e.*, k =1), a single item x_1 is considered, and the optimal leader and follower values for each capacity y is computed as follows:

$$f_1^2(y) = \begin{cases} 0 & \text{for} \quad y = 0,..,a_1 - 1 \\ c_1 & \text{for} \quad y = a_1,.....,\bar{b} \end{cases}$$

$$\tilde{f}_1^1(y) = \begin{cases} 0 & \text{for} \quad y = 0,..,a_1 - 1 \\ d_1 & \text{for} \quad y = a_1,.....,\bar{b} \end{cases}$$

i.e., the follower selects the first item only if the corresponding capacity y is sufficient.

For k>1, the recursion (1) is applied to the follower's knapsack problem to generate the follower's table. The leader's table can be generated according to the cardinality of the follower's optimal solution set. If the follower's solution is unique, the leader' s objective function is given by (2). Otherwise, the leader's objective function is updated according to (3) (in the optimistic case) or (4) (in the pessimistic case).

3. 2. Backtracking Phase

The backtracking phase is used to identify an optimal solution (y^*, x^*) that corresponds to the optimal value determined in the forward phase. The optimal capacity value y^* is generated by the n^{th} column of the leader table, as described in the next proposition.

Proposition 4:
Let (y^*, x^*) be an optimal solution for *BKPd*.

- If $t \le 0$, then $\tilde{f}_n^1(y^*) + ty^* = Max\left\{\tilde{f}_n^1(y) + ty : y \in \{\underline{b}, \underline{b}+1,.., \bar{b}\}\right\}$.
- If $t > 0$, then two possibilities exist:
  i. If $\underset{\underline{b} \le y < \bar{b}}{Max}\left\{\tilde{f}_n^1(y) + t(y+1)\right\} \le \tilde{f}_n^1(\bar{b}) + t\bar{b}$, then $(\bar{b}, x^*)$ is an optimal solution for the *BKP*, where $x^* \in P(y^*)$ and $y^* = \bar{b}$; or
  ii. *BKP* has no optimal solution.

Proof:
Case 1: $t \le 0$. Follows from Proposition 2.
Case 2: $t > 0$. $\forall\ \varepsilon \in [0,1[$ and $y \in \{\underline{b},..., \bar{b}-1\}$, we have that $f^1((y+\varepsilon), x) = f^1(y, x) + t\varepsilon < f^1(y, x) + t$, $(x \in P(y))$. Thus, if $f^1(\bar{b}, x) \ge f^1(y, x) + t$, $\forall\ y \in \{\underline{b}, \underline{b}+1,..., \bar{b}-1\}$,

then $y^* = \bar{b}$ is an optimal solution. Otherwise, the set of optimal solutions is empty. □

Starting from the optimal leader solution y^*, the backtracking phase applies the dynamic programming recursion associated with both the leader and follower problems. The procedure for the backtracking phase is presented below.

Backtracking procedure

$y \leftarrow y^*$
For $k = n$ ***to*** 2 ***do***
If $f_{k-1}^2(y) \neq f_{k-1}^2(y-a_k)+c_k$ ***then***
 if $f_k^2(y) = f_{k-1}^2(y)$ ***then*** $x^*_k = 0$
 if $f_k^2(y) = f_{k-1}^2(y-a_k)+c_k$
  then $x^*_k = 1$ ***and*** $y \leftarrow y-a_k$
Else
 If $\tilde{f}_k^1(y) = \tilde{f}_{k-1}^1(y)$ ***then*** $x^*_k = 0$,
 If $\tilde{f}_k^1(y) = \tilde{f}_{k-1}^1(y-a_k)+d_k$
  then $x^*_k = 1$ ***and*** $y \leftarrow y-a_k$
End if
End for
If $f_1^2(y) = 0$ ***then*** $x^*_1 = 0$ ***else*** $x^*_1 = 1$

If the leader faces equivalent choices, then the variable x_k^* can take either value 0 or 1. However, the value of x_1^*, which does not depend on the recursion, is set to 0 if $f_1^2(y) = 0$, and to 1 otherwise. If y^* is not unique, then the backtracking procedure is applied to each value of y^* to find all the optimal solutions. The procedure is illustrated on the following example.

Example 2:

$$\begin{cases} \underset{y,x}{Max}\ f^1(y,x) = 3x_1 + 5x_2 + x_3 + 9x_4 - 2y \\ \quad s.t.\quad 0 \le y \le 6 \\ \qquad \underset{x}{Max}\ f^2(x) = x_1 + x_2 + x_3 + x_4 \\ \qquad s.t.\quad 5x_1 + 3x_2 + 2x_3 + x_4 \le y \ \text{ and } \ x \in \{0,1\}^4 \end{cases}$$

Forward phase

Table 1 contains the optimal values of the leader $\tilde{f}_k^1(y)$ and follower $f_k^2(y)$ sub-problems at each step of the algorithm.

For *k = 1*, the follower does not select the first item until the knapsack capacity is sufficient (i.e., the values of f_1^2 and $\tilde{f}_1^1$ are equal to zero for $y < a_1 = 5$: $\left(f_1^2, \tilde{f}_1^1\right) = (0,0)$). When $y \ge 5$, the follower selects item 1: $\left(f_1^2, \tilde{f}_1^1\right) = (c_1, d_1) = (1,3)$.

The dynamic programming rules are applied for *k* = 2, 3 and 4. More precisely, for *k* = 2, the second item cannot be selected unless the knapsack capacity is sufficient. Therefore, when $y<a_2=3$, $\left(f_2^2, \tilde{f}_2^1\right) = \left(f_1^2, \tilde{f}_1^1\right)$. When $y \ge 3$, the dynamic programming RR (1) and (2) are applied. Thus $f_2^2(3) = \max\left(f_1^2(3), f_1^2(0)+1\right) = \max(0,1) = 1$. Since $f_2^2(3)$ is obtained from $f_1^2(3-a_2)+c_2$, $\tilde{f}_2^1(3) = \tilde{f}_1^1(3-a_2)+d_2 = \tilde{f}_1^1(0)+5 = 5$.

When *k* = 4 and *y* = 5, there are several optimal solutions for the follower problem. Indeed, $f_4^2(5) = \max\left(f_3^2(5), f_1^2(4)+1\right) = 2$. In the optimistic case, the follower chooses the solution that supports the leader's objective. Recursion (3) is applied, leading to $\tilde{f}_4^1(5) = \max\left(\tilde{f}_4^1(5), \tilde{f}_4^1(5-a_4)+d_4\right) = 14$.

****** [[Insert table 1 about here]] *****

Backtracking phase:

According to Proposition 3, an optimal solution y^* for the leader is determined using: $f_{opt}^1 = \tilde{f}_n^1(y^*) + ty^* = \underset{\underline{b} \le y \le \bar{b}}{Max}\left\{\tilde{f}_n^1(y) + ty\right\} =$ *max {15-12, 14-10, 14-8, 10-6, 9-4,* ***9-2****, 0-0} =7*. Thus, $y^* = 1$ and $\left(f_{opt}^1, f_{opt}^2\right) = (7,1)$.

In this example, we only consider the line associated with $y \le 1$ (Table 1). First, $x_4^* = 1$, since the values in the cell $(1, x_4^*)$ of the follower table are defined by the sum of the value in the cell $(0, x_3^*)$ and $c_3 = 1$, $\left(f_4^2(1) = f_3^2(0)+1\right)$. In the same way, $x_3^* = 0$, $x_2^* = 0$ and $x_1^* = 0$. At the end of this phase, the strong optimal solution is defined by $x^* = (0,0,0,1)$ and $y^* = 1$.

****** [[Insert table 2 about here]] *****

4. Computational Experiments

It should be clear that an algorithm whose time requirement is twice that of a

knapsack problem should outperform procedures based on branch-and-bound, most likely by an order of magnitude. In order to quantify that statement, we compared *DPBK* to the algorithm proposed by Dempe and Richter (*D&R*) [5], which was designed to solve *BKP*, as well as the generic branch-and-bound scheme of Bard and Moore (*B&M*) [1]. The mixed-integer programs involved in the latter two algorithms were solved by the commercial Cplex 9.1 software [3].

The knapsack problem instances were randomly produced using the generator proposed by Martello et al. [8], with varying degrees of correlation between the coefficients of the problem. Input data for the leader (*d* and *t*) were randomly generated according to a uniform distribution over the interval [1, *Max*], with *Max* fixed at either 100 or 1000. The results, displayed in Tables 3 and 4, support our previous claims concerning the efficiency of the method.

5. Conclusion

We close this section by mentioning a potential drawback of the *DPKB* algorithm, namely the memory requirement for instances involving large *Max* values. We believe that this difficulty could be sidestepped by resorting to sparse dynamic programming techniques [9].

Acknowledgments : This research was supported by the International Campus on Safety and Intermodality in Transportation, the Nord-Pas-de-Calais Region, the European Community, the Regional Delegation for Research and Technology, the Ministry of Higher Education and Research, and the National Center for Scientific Research. The authors are grateful to an anonymous referee for his constructive comments.

Bibliography

[1] J.F. Bard, J.T. Moore, An Algorithm for the Discrete Bilevel Programming Problem, Naval Research Logistics 39 (1992) 419-435.

[2] B. Colson, P. Marcotte, G. Savard, Bilevel Programming, A survey, 4OR 3 (2005) 87-107.

[3] CPLEX, Using the cplex callable library and cplex mixed integer programming (2006).

[4] S. Dempe, Foundation of Bilevel Programming, Kluwer Academic Publishers, 2002.

[5] S. Dempe, K. Richter, Bilevel Programming with Knapsack Constraint, Central European Newspaper of Operations Research 8 (2000) 93-107.

[7] P. Loridan, J. Morgan, Weak Via Strong Stackelberg Problem: New Results, Journal of Global Optimization 8 (1996), 263-287.

[8] S. Martello, D. Pisinger, P. Toth, Dynamic Programming and Strong Bounds for the 0–1Knapsack Problem, Management Science 45 (1999) 414–424.

[9] S. Martello, P. Toth, Knapsack Problems Algorithms and Computer Implementations, John Wiley and Sons, 1990.

[11] K. T. Talluri, G. J. Van Ryzin, Kluwer Academic Publishers, 2005.

Figure 1: No optimal solution for BKP_1

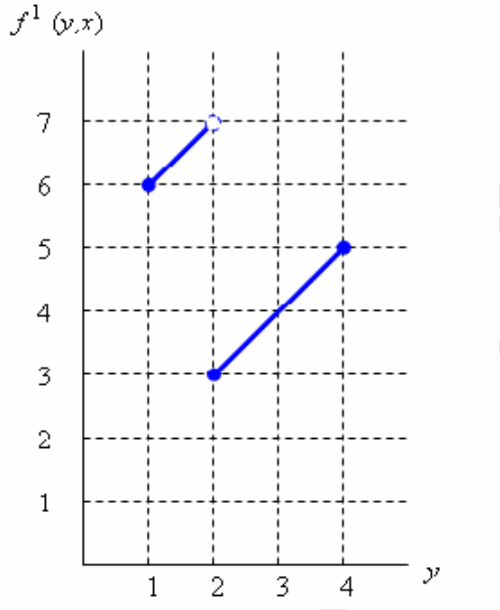

Table 1: Values of $\tilde{f}_n^1(y)$ and $f_n^2(y)$ for BKP_2

y \ k	x_1		x_2		x_3		x_4	
	f_1^2	$\tilde{f}_1^1$	f_2^2	$\tilde{f}_2^1$	f_3^2	$\tilde{f}_3^1$	f_4^2	$\tilde{f}_4^1$
0	***0***	***0***	0	0	0	0	0	0
1	0	0	0	0	0	0	1	9
2	0	0	0	0	1	1	1	9
3	***0***	***0***	***1***	***5***	1	5	2	10
4	0	0	1	5	***1***	***5***	2	14
5	***1***	***3***	1	5	2	6	***2***	***14***
6	1	3	1	5	2	6	3	15

Table 2: the backtracking phase for example 2

y \ *k*	x_1		x_2		x_3		x_4	
0	***0***	***0***	***0***	***0***	***0***	***0***	0	0
1	0	0	0	0	0	0	***1***	***9***

Table 3 : Algorithm computation times for *Max* = 1000

	Size of the coefficients: 1000 * time limit (3600 sec.) exceeded								
	Uncorrelated			Correlated			Strongly correlated		
n	B&M	D&R	DPBK	B&M	D&R	DPBK	B&M	D&R	DPBK
10	1.949	0.000	0.002	8.164	0.002	0.000	1.336	0.001	0.001
25	41.672	0.003	0.011	93.314	0.078	0.008	38.589	1.607	0.009
50	220.61	0.087	0.039	669.72	1.976	0.037	1135.46 (7)	90.812	0.045
75	1006.95 (6)	0.906	0.098	1487.63 (5)	9.693	0.092	* (0)	828.891	0.090
100	1429.19 (3)	2.440	0.167	1361.65 (2)	75.689	0.156	* (0)	3537.700	0.164