Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2012, Article ID 504713, 23 pages
doi:10.1155/2012/504713

Research Article

An Exact Algorithm for Bilevel 0-1 Knapsack Problems

Raid Mansi,[1] Cláudio Alves,[1, 2] J. M. Valério de Carvalho,[1] and Saïd Hanafi[3]

[1] *Centro de Investigação Algoritmi da Universidade do Minho, Escola de Engenharia, Universidade do Minho, 4710-057 Braga, Portugal*
[2] *Departamento de Produção e Sistemas, Universidade do Minho, 4710-057 Braga, Portugal*
[3] *LAMIH-SIADE, UMR 8530, Université de Valenciennes et du Hainaut-Cambrésis, Le Mont Houy, 59313 Valenciennes Cedex 9, France*

Correspondence should be addressed to Cláudio Alves, claudio@dps.uminho.pt

Received 5 August 2011; Revised 26 October 2011; Accepted 31 October 2011

Academic Editor: Piermarco Cannarsa

We propose a new exact method for solving bilevel 0-1 knapsack problems. A bilevel problem models a hierarchical decision process that involves two decision makers called the leader and the follower. In these processes, the leader takes his decision by considering explicitly the reaction of the follower. From an optimization standpoint, these are problems in which a subset of the variables must be the optimal solution of another (parametric) optimization problem. These problems have various applications in the field of transportation and revenue management, for example. Our approach relies on different components. We describe a polynomial time procedure to solve the linear relaxation of the bilevel 0-1 knapsack problem. Using the information provided by the solutions generated by this procedure, we compute a feasible solution (and hence a lower bound) for the problem. This bound is used together with an upper bound to reduce the size of the original problem. The optimal integer solution of the original problem is computed using dynamic programming. We report on computational experiments which are compared with the results achieved with other state-of-the-art approaches. The results attest the performance of our approach.

1. Introduction

Bilevel optimization problems were introduced for the first time in [1] in connection with the well-known Stackelberg game [2]. These problems are related to the decision making process conducted by two agents, each with his own individual objective, under a given hierarchical structure. The agent that is at the top of this hierarchy is called the leader. His distinctive feature in the process is that he knows which decision is taken by the other agent (called the follower). As a consequence, he can optimize his own objective by taking into account the decision of the follower.

The bilevel 0-1 knapsack problem (BKP) that is addressed in this paper is defined in this context. It is a hierarchical optimization problem in which the set of feasible solutions depends on the set of optimal solutions of a parametric 0-1 knapsack problem. The BKP can be formulated as follows:

$$\begin{aligned} \text{(BKP)} \quad \max_{x,y} \quad & f^1(x,y) = d^1x + d^2y \\ \text{s.t.} \quad & x \in \{0,1\}^{n^1}, \\ & y \in \operatorname{Argmax}\left\{ f^2(y') = cy' : a^1x + a^2y' \le b, y' \in \{0,1\}^{n^2} \right\}. \end{aligned} \tag{1.1}$$

The n^1 decision variables related to the leader are denoted by x. The follower has n^2 decision variables which are denoted by y. The objective functions of the leader and the follower are denoted, respectively, by $f^1(x,y)$ and $f^2(y)$. The weights of the variables x and y in the objective function of the leader are denoted by d^1 and d^2, respectively, while the vector c represents the coefficients of the follower variables in his own objective function. The vectors a^1 and a^2 are the set of coefficients related to the decision variables of the leader and the follower in the knapsack constraint of the follower, respectively. The capacity of this knapsack constraint is denoted by b. All the coefficients of the problem are assumed to be positive. The standard 0-1 knapsack problem is a special case of BKP. It is obtained from (1.1) by setting $n^1 = 0$ and $d^2 = c$. As a consequence, the problem BKP is NP-hard.

Different methods have been proposed in the literature for bilevel programming problems with and without integer variables [3, 4]. A recent survey on the contributions for solving bilevel programming problems can be found in [5]. Many of these methods focus on problems with continuous variables.

The bilevel knapsack problem was addressed by Dempe and Richter [6], Plyasunov [7], and Brotcorne et al. [8], but only for the case where there is a single (continuous) variable for the leader and different binary variables for the follower. Note that, for these cases, there may be no optimal solution for the problem [4]. The branch-and-bound algorithm proposed by Moore and Bard in [9] and the method proposed by Brotcorne et al. in [10] can be adapted to solve the BKP addressed in this paper. At the end of the paper, we will compare our approach with the results obtained by these two algorithms.

Here, we consider the case where all the variables of the problem are binary variables. We propose a new exact approach for this problem based on several intermediate procedures. A lower bound for the problem is computed by applying first a polynomial time algorithm to solve the linear relaxation of BKP. The solutions generated by this algorithm are then used to compute feasible solutions to BKP and hence to obtain a valid lower bound for the problem. Using this lower bound and a given upper bound, the size of the problem is reduced by applying fixing rules. Dynamic programming rules are applied afterwards to obtain the optimal solution of BKP.

Bilevel programming problems have many applications on different fields including economics, engineering, the determination of pricing policies, production planning, transportation, and ecology. In [11], Dempe identified more than 80 references in the literature describing applications of bilevel problems. Other examples, namely, on the field of engineering are described in [12]. The BKP is a discrete bilevel problem that can be applied in many situations involving the interaction between two agents whose (binary) decisions are interrelated and with each one trying to optimize his own objective. Real applications of this problem can be found in revenue management, telecommunications, capacity allocation, and transportation, for example.

An application in revenue management may involve an individual searching for the best investment plan for his capital. The investor has the choice between placing his funds directly on financial applications with a guaranteed rate of return, letting an intermediary company (a broker, e.g.) decide how to invest these funds, or dividing the funds between these two possibilities. The intermediary company cannot invest more than the amount provided by the individual and it will do so in order to maximize its own profit. For this purpose, the intermediary will buy shares, bonds, or other financial assets that will provide it a revenue. Part of this revenue will be given back to the individual as a return on investment. In turn, the individual will decide on the amount to invest by itself and the amount to give to the intermediary with the objective of maximizing his own profit. In BKP, the individual will be the leader, while the intermediary will be considered as the follower. The value b in (1.1) represents the capital of the individual. The coefficients of a^1 represent the amounts that the individual can invest by itself and which will provide him a guaranteed rate of return given by the vector d^1. The alternative investment plans in which the intermediary company can invest, the revenue that these plans will provide to this company, and the revenue that will be paid back to the investor are represented in (1.1) by a^2, c, and d^2, respectively. In BKP, the decision of the leader has a direct impact on the knapsack constraint of the follower. In fact, the decision of the individual (the leader) will set the capacity of the knapsack and determine the total amount of money that the intermediary (the follower) will be allowed to invest.

An alternative application of BKP occurs in telecommunications, and in particular in the problem of allocating bandwidth to different clients. An application in this area was addressed in [13] using an approach based on a bilevel programming problem with knapsack constraints. The BKP can be used in this context to model the interaction between a service provider and its competitors. The service provider can use its installed capacity to serve directly its clients, or it can grant capacity to another company that may use this capacity to route the demand of its own clients through the network of the service provider. The latter charges the company for this service, while the company will choose or not to reroute the traffic of its clients through the network of the service provider according to the offers of other competitors and so as to maximize its own profit. In this case, the leader is the service provider and the follower is the other company. The total capacity of the service provider is the coefficient b in (1.1). The price that is charged by the service provider is represented by d^2, while the amount of traffic required by the clients is given by a^1 and a^2.

The remainder of the paper is organized as follows. In Section 2, we introduce different definitions and the notation that is used in the paper, and we describe the properties of BKP. In Section 3, we describe the details of our algorithm. We present our algorithm to solve the linear relaxation of BKP, the rules used to reduce the size of the problem, and the dynamic programming procedures developed to find the optimal solution of BKP. In Section 4, we report on computational results that illustrate the efficiency of our methods compared with the only available method from the literature [14]. Some final conclusions are drawn in Section 5.

2. The Bilevel 0-1 Knapsack Problem

2.1. Definitions

We introduce first the following standard definitions related to the bilevel 0-1 knapsack problem BKP described in the previous section:

(i) the relaxed feasible set:

$$S = \left\{ (x, y) \in \{0,1\}^{n^1+n^2} : a^1 x + a^2 y \le b \right\}; \tag{2.1}$$

(ii) the set of rational reactions of the follower for a fixed x:

$$P(x) = \operatorname{Argmax}\left\{ f^2(y) : a^2 y \le b - a^1 x, y \in \{0,1\}^{n^2} \right\}; \tag{2.2}$$

(iii) the Inducible Region (IR), that is, the space over which the leader optimizes:

$$\mathrm{IR} = \{ (x, y) \in S : y \in P(x) \}. \tag{2.3}$$

Using this notation, we can rewrite the BKP as follows.

$$\begin{aligned} \text{(BKP)} \quad \max_{x,y} \quad & f^1(x,y) = d^1 x + d^2 y \\ \text{s.t.} \quad & (x,y) \in \mathrm{IR}. \end{aligned} \tag{2.4}$$

When several optimal solutions exist for the follower problem, the previous model is not sufficient to define the optimal value of the problem because, for a leader decision, the follower can have several equivalent solutions. In this case, the solutions of the BKP can be defined either optimistically or pessimistically for each fixed leader variable [15]. These approaches can be described as follows.

(i) Optimistic

We assume that the leader can influence the decision of the follower in his favor. In this case, the problem to solve becomes

$$\max_x \left\{ \max_y \left\{ f^1(x,y) : y \in P(x) \right\} (x,y) \in S \right\}, \tag{2.5}$$

and its optimal solution is called a *weak solution*.

(ii) Pessimistic

The follower takes his decision independently of the interests of the leader. In this case, the problem to solve becomes

$$\max_x \left\{ \min_y \left\{ f^1(x,y) : y \in P(x) \right\} (x,y) \in S \right\}, \tag{2.6}$$

and its optimal solution is called a *strong solution*.

A detailed discussion of each approach can be found in [15]. The algorithms described in this paper can find both the strong and the weak solution of the problem. However, and for the sake of clearness, we will focus our presentation on the optimistic approach.

2.2. *An Upper Bound for BKP*

The linear relaxation of BKP obtained by removing all the integrality constraints does not provide a valid upper bound for the problem. Hence, we resort to an upper bound for bilevel programming problems provided by a relaxation of (1.1) called the *high-point problem* [9, 16]. The high point problem is obtained by removing the objective function of the follower and the integrality constraints. It is defined formally as follows:

$$\begin{aligned} (\text{HBKP}) \quad \max_{x,y} \quad & f^1(x,y) = d^1x + d^2y \\ \text{s.t.} \quad & a^1x + a^2y \le b, \\ & x \in [0,1], \quad y \in [0,1]. \end{aligned} \tag{2.7}$$

The optimal solution of this relaxation can be computed using a classical procedure for the knapsack problem [17].

2.3. *Computing a Feasible Solution for BKP*

A feasible solution to BKP can be computed by solving a different optimization problem related to the follower problem as shown in the following proposition.

Proposition 2.1. *Let $z \in \{0,\dots,b\}$. An optimal solution to the following problem denoted by FBKP$_z$ is also feasible for BKP:*

$$(FBKP_z) \quad \max_{x,y} \quad f(x,y) = d^1x + cy \tag{2.8}$$

$$s.t. \quad a^1x = z, \tag{2.9}$$

$$\begin{aligned} & a^2y \le b - z, \\ & x \in \{0,1\}^{n^1}, \quad y \in \{0,1\}^{n^2}. \end{aligned} \tag{2.10}$$

Proof. As long as FBKP$_z$ admits a feasible solution, its optimal solution is feasible for the follower problem of BKP since the knapsack constraint of the follower is satisfied due to (2.9) and (2.10). This optimal solution is also optimal for the follower problem because it takes into account the follower objective function on the follower variables. □

An optimal solution (x^*, y^*) for BKP can then be defined using FBKP$_z$ as follows:

$$(x^*, y^*) \in \text{Argmax}\left\{d^1x_z^* + d^2y_z^* : (x_z^*, y_z^*) \text{ is an optimal solution of FBKP}_z, \text{ for } z = 0,\dots,b\right\}. \tag{2.11}$$

Clearly, finding an optimal solution for BKP by solving FBKP_z for each possible value of z is computationally expensive. To obtain a good feasible solution for BKP, in our algorithm, we solve the problem FBKP_z for a set of good candidate values for z, which are obtained by solving the linear relaxation of BKP with the polynomial time procedure described in Section 3.1.

3. An Exact Algorithm for BKP

Before we describe our exact algorithm for BKP, we define and discuss first its different components. Our algorithm relies on the computation of an upper and lower bound for BKP. The upper bound is computed by solving exactly the problem HBKP defined previously. The lower bound is obtained by solving first a linear relaxation of BKP using the polynomial time procedure described in Section 3.1, and then by solving the problem FBKP_z for different values of the parameter z. The values of z are associated to feasible solutions of the linear relaxation of BKP which are obtained by applying the polynomial procedure mentioned previously. The upper and lower bounds are used to fix variables of BKP to their optimal values (Section 3.2) and to further enhance the definition of the original problem so as to improve its resolution in the remaining steps (Section 3.3). The optimal value for the resulting problem is computed using dynamic programming. This value is then used to generate an optimal solution for BKP. The two phases of this dynamic programming procedure are described in Section 3.4. The outline of our exact algorithm is given in Section 3.5.

3.1. A Polynomial Time Solution Procedure for the Linear Relaxation of BKP

In this section, we show that the linear relaxation of BKP can be solved up to optimality in polynomial time, and we describe a procedure that computes this optimal solution. First, we recall the formal definition of this linear relaxation that will be denoted by CBKP (for continuous bilevel 0-1 knapsack problem):

$$\begin{aligned}
(\text{CBKP})\quad \max_{x,y}\quad & f^1(x,y) = d^1x + d^2y \\
\text{s.t.}\quad & 0 \le x \le 1, \\
& y \in \operatorname{Argmax}\left\{ f^2(y') = cy' : a^1x + a^2y' \le b, 0 \le y' \le 1 \right\}.
\end{aligned} \tag{3.1}$$

Now, we show that solving CBKP is equivalent to the resolution of the linear relaxation of a standard knapsack problem.

Proposition 3.1. *Assume that the follower variables $y_1, y_2, \dots, y_{n^2}$ are sorted in decreasing order of the relative value between their profit and their weight in the knapsack constraint, that is, such that $c_1/a_1^2 \ge c_2/a_2^2 \ge \cdots \ge c_{n^2}/a_{n^2}^2$. If $c_i/a_i^2 = c_j/a_j^2$, the order between the corresponding variables is determined according to the objective function of the leader, that is, $d_i^2/a_i^2 \ge d_j^2/a_j^2$ (in the pessimistic case, one will consider $d_i^2/a_i^2 \le d_j^2/a_j^2$). Let x^* be a decision of the leader. In this case, the total resource consumed by the leader in the knapsack constraint is given by a^1x^*. Furthermore, let k be defined such that $b - a^1x^* \in [\sum_{i=1}^k a_i^2 s, \sum_{i=1}^{k+1} a_i^2[$. The reaction of the follower related to the decision x^* will be as follows:*

$$y_1 = 1, y_2 = 1, \dots, y_k = 1, y_{k+1} = \frac{\left(b - a^1x^* - \sum_{i=1}^k a_i^2\right)}{a_{k+1}^2}, y_{k+2} = 0, \dots, y_{n^2} = 0. \tag{3.2}$$

```
Let $d_{\mathrm{opt}}$ denote the optimal value of the leader objective function;
Let $(x_{\mathrm{opt}}, y_{\mathrm{opt}})$ be an optimal solution of CBKP;
**Initialization**
  Sort the variables $x$ and $y$ in decreasing order of the ratio $d_j^1/a_j^1$ for $x$ and $c_j/a_j^2$
  and $d_j^2/a_j^2$ for $y$;
  Let
      $x_1^* = 1, x_2^* = 1, \ldots, x_k^* = 1, x_{k+1}^* = \frac{(b - \sum_{i=1}^{k} a_i^1)}{a_{k+1}^1}, x_{k+2}^* = 0, \ldots, x_{n^1}^* = 0$
  be the optimal solution for the following problem:
      $x^* \in \mathrm{Argmax}\{f^1(x) = d^1x : a^1x \le b, 0 \le x \le 1\}$;
  Let
      $y_1^* = 1, y_2^* = 1, \ldots, y_t^* = 1, y_{t+1}^* = \frac{(b - a^1x^* - \sum_{i=1}^{t} a_i^2)}{a_{t+1}^2}, y_{t+2}^* = 0, \ldots, y_{n^2}^* = 0$
  be the optimal solution for the following problem:
      $y^* \in \mathrm{Argmax}\{f^2(y) = cy : a^2y \le b - a^1x^*, 0 \le y \le 1\}$;
  /* The indexes $k+1$ and $t+1$ are respectively the indexes of the last
  leader and follower variables with a positive value according
  to the ordering */
  Let $d_{\mathrm{opt}} = d^1x^* + d^2y^*$;
$i = k+1$; $j = t+1$;
**while** $(i > 0)$ and $(j < n^2 + 1)$ **do**
  **if** $(a_i^1x_i^* > a_j^2(1 - y_j^*))$ **then**
    $x_i^* = x_i^* - a_j^2(1 - y_j^*)/a_i^1$; $y_j^* = 1$; $j = j + 1$;
  **end**
  **else if** $(a_i^1x_i^* < a_j^2(1 - y_j^*))$ **then**
      $x_i^* = 0$; $y_j^* = y_j^* - a_i^1x_i^*/a_j^2$; $i = i - 1$;
  **end**
  **else if** $(a_i^1x_i^* = a_j^2(1 - y_j^*))$ **then**
      $x_i^* = 0$; $y_j^* = 1$; $i = i - 1$; $j = j + 1$;
  **end**
  **if** $(d_{\mathrm{opt}} < d^1x^* + d^2y^*)$ **then**
    $d_{\mathrm{opt}} = d^1x^* + d^2y^*$; $(x_{\mathrm{opt}}, y_{\mathrm{opt}}) = (x^*, y^*)$;
  **end**
**end**
```

Algorithm 1: A polynomial time solution procedure for CBKP.

Proof. Indeed, for a given decision x^* of the leader, the problem CBKP becomes a standard 0-1 knapsack problem:

$$(\mathrm{CBKP}_{x^*}) \quad y \in \mathrm{Argmax}\left\{f^2(y) = cy' : a^2y' \le b - a^1x^*, 0 \le y' \le 1\right\}. \tag{3.3}$$

□

In Algorithm 1, we describe a polynomial time procedure that generates a weak solution for CBKP. The algorithm is based on the same idea that is used to solve the standard 0-1 knapsack problem. We start by solving the knapsack problem associated to the leader variables and objective function:

$$x^* \in \mathrm{Argmax}\left\{f^1(x) = d^1x : a^1x \le b, 0 \le x \le 1\right\}, \tag{3.4}$$

with x^* being the optimal solution of this problem. Then, we solve the follower knapsack problem that results from the leader decision x^*:

$$y^* \in \operatorname{Argmax}\left\{ f^2(y) = cy : a^2 y \leq b - a^1 x^*, 0 \leq y \leq 1 \right\}, \tag{3.5}$$

with y^* being the corresponding optimal solution. The algorithm enumerates all the nondominated feasible linear programming basic solutions starting by the solution (x^*, y^*). At each iteration, we move from a feasible basic solution to another by transferring the resources consumed by the leader to the follower. To clarify the procedure, we illustrate its execution in Example 3.2.

Example 3.2. Consider the following continuous bilevel 0-1 knapsack problem denoted by CBKP$_1$:

$$\begin{aligned}
(\mathrm{CBKP}_1) \quad \max_{x,y} \quad & f^1(x,y) = 3x_1 + 2x_2 + 7x_3 + 5y_1 + y_2 + 2y_3 + y_4 \\
\text{s.t.} \quad & 0 \leq x \leq 1, \\
& \max_{y} \quad f^2(y) = 2y_1 + 2y_2 + 3y_3 + 4y_4 \\
& \text{s.t.} \quad 3x_1 + x_2 + 2x_3 + y_1 + 2y_2 + y_3 + 4y_4 \leq 4 \\
& \qquad 0 \leq y \leq 1.
\end{aligned} \tag{3.6}$$

We start by sorting the variables of the leader in decreasing order of the ratio d_j^1/a_j^1, which results in the sequence x_3 (with $d_3^1/a_3^1 = 7/2$), x_2 (with $d_2^1/a_2^1 = 2/1$), and x_1 (with $d_1^1/a_1^1 = 3/3$). A similar ordering is applied to the variables of the follower resulting in the sequence y_3 (with $c_3/a_3^2 = 3/1$), y_1 (with $c_1/a_1^2 = 2/1$), y_2 (with $c_2/a_2^2 = 2/2$), and y_4 (with $c_4/a_4^2 = 4/4$). Note that y_2 and y_4 have the same ratio c_j/a_j^2, but y_2 with $d_2^2/a_2^2 = 1/2$ outperforms y_4 with $d_4^2/a_4^2 = 1/4$. In this example, we are considering the optimistic case. In the first phase of the procedure, we solve the following problem:

$$\begin{aligned}
\max_{x} \quad & f^1(x) = 3x_1 + 2x_2 + 7x_3 \\
\text{s.t.} \quad & 3x_1 + x_2 + 2x_3 \leq 4, \\
& 0 \leq x \leq 1.
\end{aligned} \tag{3.7}$$

The optimal solution of this problem is $x_3^* = 1$, $x_2^* = 1$, and $x_1^* = 1/3$. The optimal reaction of the follower for this decision of the leader is $y^* = (0,0,0,0)$ and $d_{\mathrm{opt}} = 10$. The solutions generated at each iteration of the Algorithm 1 are described as follows:

(1) $x_3^* = 1$, $x_2^* = 1$, $x_1^* = 1/3$ and $y_3^* = 0$, $y_1^* = 0$, $y_2^* = 0$, $y_4^* = 0$ with $d_{\mathrm{opt}} = 10$;

(2) $x_3^* = 1$, $x_2^* = 1$, $x_1^* = 0$ and $y_3^* = 1$, $y_1^* = 0$, $y_2^* = 0$, $y_4^* = 0$ with $d_{\mathrm{opt}} = 11$;

(3) $x_3^* = 1$, $x_2^* = 0$, $x_1^* = 0$ and $y_3^* = 1$, $y_1^* = 1$, $y_2^* = 0$, $y_4^* = 0$ with $d_{\mathrm{opt}} = 14$;

(4) $x_3^* = 0$, $x_2^* = 0$, $x_1^* = 0$ and $y_3^* = 1$, $y_1^* = 1$, $y_2^* = 1$, $y_4^* = 0$ with $d_{\mathrm{opt}} = 8$.

The value d_{opt} denotes the optimal value of the leader objective function as introduced in Algorithm 1. The optimal solution of CBKP_1 is obtained at the third iteration. This solution is achieved after a polynomial number of steps.

In Algorithm 1, all the nondominated feasible basic solutions of CBKP are visited. For each one of these solutions, we can associate a value for the parameter z in FBKP_z. In Example 3.2, the value of z is equal to 4, 3, 2, and 0 at the iterations 1 to 4, respectively. These values are equal to a^1x^0, with x^0 being the value of the leader variables of a given basic solution generated in Algorithm 1. As shown in Section 2.3, we can obtain a feasible solution for BKP by solving the problem FBKP_z using these values of z.

A basic solution of CBKP has at most two fractional variables (one for the leader, and another for the follower). If a basic solution of CBKP is integer for both the leader and the follower variables, then this solution is feasible for FBKP_z and for BKP too. If all the variables of the leader are integer, and only one variable of the follower is fractional, then we can fix the values of the leader variables in FBKP_z and solve the resulting problem which becomes a single knapsack problem. In these two cases, it is always possible to find a feasible solution for FBKP_z, and hence for BKP. However, when one of the leader variables is fractional, the problem FBKP_z may be infeasible. This is due to the fact that we are considering that $z \in \{\lfloor a^1x^0 \rfloor, \lfloor a^1x^0 \rfloor + 1\}$. Since there is no guarantee that the equation $a^1x = z$ in FBKP_z has a solution, the corresponding problem FBKP_z may be infeasible.

Solving the problem FBKP_z for a single value of the parameter z can be done efficiently using branch-and-bound, for example. Clearly, solving this problem for all the values of z in $\{0, \dots, b\}$ is much more expensive computationally. In our algorithm, our approach to generate a good feasible solution for BKP consists in inspecting a restricted set of good candidate values for z. For this purpose, we choose the values of z that are associated to the n best solutions generated by Algorithm 1. In Example 3.2, if we set $n = 2$, then the values of z associated to the two best solutions generated by Algorithm 1 (obtained at the iterations 2 and 3) will be used as a parameter in FBKP_z. The problems that will be solved in this case are FBKP_2 and FBKP_3.

The feasible solution (and corresponding lower bound) that is generated using this approach can be used together with the upper bound provided by HBKP to reduce the size of the original problem BKP. This can be done by fixing the values of some variables to their optimal values. The strategies used to reduce the size of the original BKP are described in the next section.

3.2. Reducing the Size of BKP Using Fixing Rules

A strategy to improve the resolution of 0-1 mixed integer programming problems which has been used extensively in the literature consists in fixing the values of some variables to their optimal value. Many authors [18–20] reported different procedures based on this idea to reduce the size of multidimensional knapsack problems. In this section, we show that it is also possible to apply fixing rules to BKP, and we describe the procedure that we used in our algorithm.

In many cases, fixing variables to their optimal value can be done via inexpensive operations. In the sequel, we show how variables can be fixed using information on the upper and lower bounds for the problem.

Proposition 3.3. *Let $\alpha \in \{0,1\}$ and LB be a lower bound for BKP. One will use the notation $v(\cdot)$ to indicate the optimal value of a given problem. The following fixing rules apply:*

(i) *for any* $j \in \{1,\ldots,n^1\}$, *if* $v(HBKP \mid x_j = \alpha) < LB$, *then* x_j *can be fixed to the value* $1-\alpha$*;*

(ii) *for any* $j \in \{1,\ldots,n^2\}$, *if* $v(HBKP \mid y_j = \alpha) < LB$, *then* y_j *can be fixed to the value* $1-\alpha$.

Proof. Let $v(\text{BKP})$ be the optimal value for BKP, and let q denote both the variables x and y of the leader and follower, respectively. Note that $v(\text{BKP}) = \max\{v(\text{BKP} \mid q_j = \alpha), v(\text{BKP} \mid q_j = 1-\alpha)\}$. Therefore, if $v(\text{HBKP} \mid q_j = \alpha) < \text{LB}$, then inevitably $v(\text{BKP}) = v(\text{BKP} \mid q_j = 1-\alpha)$, and the optimal value q_j^* can be fixed to $1-\alpha$. □

These fixing rules depend only on an upper and a lower bound for the problem. The stronger the upper and lower bounds are, the more effective will be the rules for fixing the variables.

To introduce a new fixing rule, we rewrite the problem HBKP (used to derive an upper bound for BKP) in its standard form as follows:

$$\max\left\{d^1x + d^2y : a^1x + a^2y + s = b, e \geq x \geq 0, e \geq y \geq 0, s \geq 0\right\}, \tag{3.8}$$

where s corresponds to the vector of slack variables and e is the vector with all elements equal to 1. Let $N^1 = \{1,\ldots,n^1\}$ and $N^2 = \{1,\ldots,n^2\}$ be the indices of the leader and follower variables x and y, respectively.

By reference to the LP basis that produces $(\overline{x},\overline{y})$, we define $\mathcal{B} = \{j \in N^1 : x_j \text{ is basic}\} \cup \{j \in N^2 : y_j \text{ is basic}\}$ and $\overline{\mathcal{B}} = \{j \in N^1 : x_j \text{ is nonbasic}\} \cup \{j \in N^2 : y_j \text{ is non-basic}\}$. We subdivide $\overline{\mathcal{B}}$ to identify the four subsets $\overline{\mathcal{B}}_x^0 = \{j \in \overline{\mathcal{B}} : \overline{x}_j = 0\}$, $\overline{\mathcal{B}}_y^0 = \{j \in \overline{\mathcal{B}} : \overline{y}_j = 0\}$, $\overline{\mathcal{B}}_x^1 = \{j \in \overline{\mathcal{B}} : \overline{x}_j = 1\}$, and $\overline{\mathcal{B}}_y^1 = \{j \in \overline{\mathcal{B}} : \overline{y}_j = 1\}$.

Assume that $(\overline{x},\overline{y})$ is an optimal basic solution of HBKP. The problem HBKP can be written in the optimal basis related to $(\overline{x},\overline{y})$ in the following way:

$$\begin{aligned}\max\quad & v(\text{HBKP}) + \sum_{j\in\overline{\mathcal{B}}_x^0}\widehat{d}_j^1x_j + \sum_{j\in\overline{\mathcal{B}}_y^0}\widehat{d}_j^2y_j - \sum_{j\in\overline{\mathcal{B}}_x^1}\widehat{d}_j^1(1-x_j) - \sum_{j\in\overline{\mathcal{B}}_y^1}\widehat{d}_j^2(1-y_j) + \widehat{l}s\\ \text{s.t.}\quad & \widehat{a}^1x + \widehat{a}^2y + \widehat{t}s = \widehat{b},\\ & x \in [0,1]^{n^1}, \quad y \in [0,1]^{n^2}, \quad s \geq 0,\end{aligned} \tag{3.9}$$

with $v(\text{HBKP})$ being the optimal value of HBKP, and $(\widehat{d}^1,\widehat{d}^2,\widehat{l})$ the vector of reduced costs corresponding to the variables (x,y,s) of the optimal basis. For a given lower bound LB for BKP, we have

$$v(\text{HBKP}) + \sum_{j\in\overline{\mathcal{B}}_x^0}\widehat{d}_j^1x_j + \sum_{j\in\overline{\mathcal{B}}_y^0}\widehat{d}_j^2y_j - \sum_{j\in\overline{\mathcal{B}}_x^1}\widehat{d}_j^1(1-x_j) - \sum_{j\in\overline{\mathcal{B}}_y^1}\widehat{d}_j^2(1-y_j) + \widehat{l}s \geq \text{LB}. \tag{3.10}$$

The quantity $\widehat{l}s$ is negative because of the negative reduced cost vector $\widehat{l}$ associated to the optimal basic solution, and the positive slack variables s. Moreover, since $\widehat{d}_j^1 \leq 0$ for $j \in \overline{\mathcal{B}}_x^0$

(resp., $\widehat{d}_j^2 \leq 0$ for $j \in \overline{\mathcal{B}}_y^0$), and $\widehat{d}_j^1 \geq 0$ for $j \in \overline{\mathcal{B}}_x^1$ (resp., $\widehat{d}_j^2 \geq 0$ for $j \in \overline{\mathcal{B}}_y^1$), we can consider the following cut based on the reduced costs:

$$\sum_{j\in\overline{\mathcal{B}}_x^0} \widehat{d}_j^1 x_j + \sum_{j\in\overline{\mathcal{B}}_y^0} \widehat{d}_j^2 y_j - \sum_{j\in\overline{\mathcal{B}}_x^1} \widehat{d}_j^1 (1-x_j) - \sum_{j\in\overline{\mathcal{B}}_y^1} \widehat{d}_j^2 (1-y_j) \leq v(\mathrm{HBKP}) - \mathrm{LB}. \tag{3.11}$$

This inequality can be used to derive the fixing rule introduced in the next proposition.

Proposition 3.4. *If $\overline{x}_j$ (resp., $\overline{y}_j$) is a nonbasic variable, and $|\widehat{d}_j^1| > v(HBKP) - LB$ (resp., $|\widehat{d}_j^2| > v(HBKP) - LB$), then at optimality one has $x_j^* = \overline{x}_j$ (resp., $y_j^* = \overline{y}_j$).*

Proof. The proof comes directly from the previous inequality (3.11). □

Applying these fixing rules is useful for reducing the size of the original problem BKP, and hence to improve its resolution. However, because they do not take into account the objective function of the follower, these rules may result in problems whose solutions are infeasible for the original BKP. The problem occurs when the leader has solutions with the same value than the optimal solution, but which are infeasible for the original BKP because they are not optimal for the follower. To clarify this issue, we apply these rules on the case described in Example 3.2. The results are given in the following example.

Example 3.5. Consider the instance of BKP whose linear relaxation is given by CBKP$_1$ in Example 3.2. We will denote this instance of BKP by BKP$_1$. In Table 1, we describe an optimal solution $(\overline{x}, \overline{y})$ for the corresponding problem HBKP, and we report on the values of the associated vectors of reduced costs $(\widehat{d}^1, \widehat{d}^2)$. Furthermore, we specify whether a given variable is a basic variable or not by reference to the solution $(\overline{x}, \overline{y})$, and we identify the variables that can be fixed according to the fixing rules described previously.

Let UB and LB denote, respectively, the value of an upper and lower bound for this instance of BKP. The value of the solution given in Table 1 is 14, and hence we have UB = 14. By applying Algorithm 1, we obtain a lower bound of value LB = 14, as shown in Example 3.2. According to Proposition 3.4, since UB − LB = 0, all the nonbasic variables with an absolute reduced cost greater than 0 can be fixed, and hence we have $x_1^* = 0$, $x_3^* = 1$, $y_1^* = 1$, $y_2^* = 0$, and $y_4^* = 0$. The variable x_2 cannot be fixed because the absolute value of its reduced cost is not greater than UB − LB. Similarly, the variable y_3 cannot be fixed because it is a basic variable. Applying the fixing rules leads to the following problem:

$$\begin{aligned}
(\mathrm{BKP}_1) \quad \max_{x,y} \quad & f^1(x,y) = 2x_2 + 2y_3 + 12 \\
\text{s.t.} \quad & x_2 \in \{0,1\}, \\
& \max_{y} \quad f^2(y) = 3y_3 \\
& \text{s.t.} \quad x_2 + y_3 \leq 1, \\
& \qquad\quad y_3 \in \{0,1\}.
\end{aligned} \tag{3.12}$$

Table 1: An optimal solution for HBKP (Example 3.5).

	x_1	x_2	x_3	y_1	y_2	y_3	y_4
$(\overline{x}, \overline{y})$	0	0	1	1	0	1	0
$(\widehat{d}^1, \widehat{d}^2)$	−3	0	3	3	−3	0	−7
Basic variable	no	no	no	no	no	yes	no
Variable can be fixed	yes	no	yes	yes	yes	no	yes

The resulting problem has two equivalent solutions. The first one consists in the leader action $x_2 = 1$ and the follower reaction $y_3 = 0$. In this case, the complete solution (denoted by sol_1) for the original problem is $x_1 = 0$, and $x_2 = 1$, $x_3 = 1$ and $y_1 = 1$, $y_2 = 0$, $y_3 = 0$, and $y_4 = 0$. The second solution consists in the leader action $x_2 = 0$ and the follower reaction $y_3 = 1$. The complete solution for the original problem in this case (denoted by sol_2) is $x_1 = 0$, $x_2 = 0$, and $x_3 = 1$ and $y_1 = 1$, $y_2 = 0$, $y_3 = 1$, and $y_4 = 0$. The value of both sol_1 and sol_2 is equal to 14. However, the optimal solution of original problem BKP $_1$ is given by sol_2, since for the leader action $x_1 = 0$, $x_2 = 0$, and $x_3 = 1$ the reaction of the follower $y_1 = 1$, $y_2 = 0$, $y_3 = 1$, and $y_4 = 0$ is optimal for the follower problem. On the contrary, sol_1 is not feasible for the problem because for the leader action $x_1 = 0$, $x_2 = 1$, and $x_3 = 1$, the follower reaction should not be $y_1 = 1$, $y_2 = 0$, $y_3 = 0$, and $y_4 = 0$ with the value 2 for the follower objective function. In this case, the follower reaction should be $y_1 = 0$, $y_2 = 0$, $y_3 = 1$, and $y_4 = 0$ with a corresponding value for the follower objective function that is equal to 3.

As shown in Example 3.5, the optimal solution of the problem can be found even when the fixing rules described in this section are applied. However, an additional treatment on the optimal solutions of the resulting problem is necessary to identify the solutions that are optimal for the follower problem (and hence feasible for the original problem BKP). To overcome this issue, in our algorithm, we fixed only the leader variables that are not directly influenced by the objective function of the follower.

3.3. *Reducing the Interval of Values for the Parameter Z in FBKP$_Z$*

In this section, we show how to decrease the knapsack capacity b of the follower problem, and hence the size of the interval of possible values for z in FBKP_z. Let lb_z and ub_z be the values of a lower and upper bound for z in the problem FBKP_z. Initially, we have $\text{lb}_z = 0$ and $\text{ub}_z = b$, and hence $z \in [0, b]$. The smaller the size of the interval $[\text{lb}_z, \text{ub}_z]$ is, the easier the problem BKP will be to solve.

To improve the values of lb_z and ub_z, we solve the following two linear programming problems (denoted by LB_z and UB_z) which relies on a lower bound LB for BKP. The optimal value of LB_z leads to a feasible value for lb_z, while UB_z leads to a feasible value for ub_z:

$$\begin{aligned}
(\text{LB}_z)\quad \min\quad & z \\
\text{s.t.}\quad & a^1 x = z, \\
& a^2 y \le b - z, \\
& d^1 x + d^2 y \ge \text{LB}, \\
& x \in [0,1]^{n^1},\ y \in [0,1]^{n^2},\ z \in [0,b],
\end{aligned}$$

$$\begin{aligned}(\mathrm{UB}_z)\quad \max\quad & z\\ \text{s.t.}\quad & a^1x = z,\\ & a^2y \le b - z,\\ & d^1x + d^2y \ge \mathrm{LB},\\ & x \in [0,1]^{n^1},\ y \in [0,1]^{n^2},\ z \in [0,b].\end{aligned} \tag{3.13}$$

Optimizing over the variable z with the additional constraint $d^1x + d^2y \geq \mathrm{LB}$ in these two linear programs ensures that the resulting lower and upper bound for z will not cut the optimal solution of the original BKP. In the next section, we show how this new interval helps in improving the performance of the dynamic programming component of our algorithm for BKP.

3.4. Computing an Optimal Solution of BKP Using Dynamic Programming

In this section, we describe an approach based on dynamic programming to compute the optimal solution of BKP. The approach is divided into two phases. The first phase is a forward procedure whose objective is to find the value of an optimal solution for BKP. This forward phase divides in turn into two steps which are applied, respectively, to the leader variables and to the follower variables. The dynamic programming rules used for the follower variables are an extension of those used in [8]. In the second phase, a backtracking procedure is applied to generate a solution for the BKP with the value found in the forward phase. This dynamic programming algorithm has a pseudo-polynomial complexity, and it is able to solve both the optimistic and pessimistic cases mentioned previously. For the sake of brevity, we will focus our presentation on the optimistic case.

3.4.1. Computing the Optimal Value of BKP: The Forward Phase

As alluded previously, the objective of the forward phase is to find the optimal value of BKP. This phase consists in two steps. The first step applies to the variables of the leader in BKP, and it considers only the objective function of the leader. The definition of this step relies on the interaction between the leader and the follower. For a given decision x of the leader, the follower has to maximize his total profit cy using the corresponding residual capacity $b - a^1x$. For each value of $\vartheta \in [0,b]$, the best action for the leader has to be determined: $x_{\vartheta\in[0,b]} \in \mathrm{Argmax}\{d^1x : a^1x = \vartheta, x \in \{0,1\}^{n^1}\}$. Hence, the dynamic programming subproblem for the leader states as follows:

$$f_k^1(\vartheta) = \max\left\{\sum_{j=1}^{k} d_j^1 x_j : \sum_{j=1}^{k} a_j^1 x_j = \vartheta, x \in \{0,1\}^k\right\}, \tag{3.14}$$

with $k \in N^1$.

```
$f_1^1(\vartheta) = 0$ if $\vartheta = 0$, $d_1^1$ if $\vartheta = a_1^1$, $-\infty$ otherwise;
  **for** $k = 2$ to $n^1$ **do**
    **for** $\vartheta = 0$ to $b$ **do**
      **if** $\vartheta < a_k^1$ **then**
        $f_k^1(\vartheta) = f_{k-1}^1(\vartheta)$;
      **end**
      **else**
        $f_k^1(\vartheta) = \max(f_{k-1}^1(\vartheta), f_{k-1}^1(\vartheta - a_k^1) + d_k^1)$;
      **end**
    **end**
  **end**
```

Algorithm 2: Forward procedure for the leader.

Table 2: First step of the forward phase for the leader of BKP$_1$.

		k	
	1	2	3
ϑ	$f_1^1(\vartheta)$	$f_2^1(\vartheta)$	$f_3^1(\vartheta)$
0	0	0	0
1	$-\infty$	2	2
2	$-\infty$	$-\infty$	7
3	3	3	9
4	$-\infty$	5	5

The dynamic programming procedure for the leader in this first step of the forward phase is described in Algorithm 2. To illustrate the execution of this algorithm, we show in the following example how it applies to the instance of the BKP described in Example 3.2.

Example 3.6. Let us recall first the definition of the instance BKP$_1$:

$$\begin{aligned}
(\mathrm{BKP}_1) \quad & \max_{x,y} \quad f^1(x,y) = 3x_1 + 2x_2 + 7x_3 + 5y_1 + y_2 + 2y_3 + y_4 \\
& \text{s.t.} \quad x \in \{0,1\}^3, \\
& \qquad \max_{y} \quad f^2(y) = 2y_1 + 2y_2 + 3y_3 + 4y_4 \\
& \qquad \text{s.t.} \quad 3x_1 + x_2 + 2x_3 + y_1 + 2y_2 + y_3 + 4y_4 \le 4, \\
& \qquad\qquad y \in \{0,1\}^4.
\end{aligned} \tag{3.15}$$

The results of the first step of the forward phase applied to the leader variables of BKP$_1$ are given in Table 2. In this table, we report on the optimal values of the associated subproblems at this step.

Note that the value of $f_3^1(4)$ is smaller than $f_3^1(3)$ because there is no solution x with a better value which consumes exactly 4 units of capacity.

Table 3: Second step of the forward phase for the follower of BKP$_1$.

	k							
	1		2		3		4	
β	$f_1^2(\beta)$	$\widehat{f}_1^1(\beta)$	$f_2^2(\beta)$	$\widehat{f}_2^1(\beta)$	$f_3^2(\beta)$	$\widehat{f}_3^1(\beta)$	$f_4^2(\beta)$	$\widehat{f}_4^1(\beta)$
0	0	0	0	0	0	0	0	0
1	2	5	2	$\underline{\underline{5}}$	3	$\underline{\underline{2}}$	3	2
2	2	5	$\underline{2}$	$\underline{5}$	5	7	5	7
3	2	5	4	6	5	7	5	7
4	2	5	4	6	7	8	7	8

In the second step of the forward phase, we focus on the variables of the follower. The problem that is solved at this stage is the following:

$$\begin{aligned} &\max_{y} \quad d^2 y \\ &\text{s.t.} \quad y \in \operatorname{Argmax}\left\{ f^2(y') = cy' : a^2 y' \le b, y' \in \{0,1\}^{n^2} \right\}. \end{aligned} \tag{3.16}$$

Let $\beta = b - \vartheta \in [0, b]$ denote the residual capacity associated with the leader action x_ϑ. In this second step, we consider both the leader and the follower objective functions, and we apply the forward procedure based on dynamic programming described in [8]. The objective is to determine all the reactions of the follower for a given action of the leader.

Two tables are generated in the second step of the forward phase: one that stores the optimal values of the follower ($f_k^2(\beta)$), and a second one that stores the optimal values of the leader values ($\widehat{f}_k^1(\beta)$) with

$$\begin{aligned} f_k^2(\beta) &= \max\left\{ \sum_{j=1}^{k} c_j y_j : \sum_{j=1}^{k} a_j^2 y_j \le \beta, y \in \{0,1\}^k \right\}, \\ \widehat{f}_k^1(\beta) &= \max\left\{ \sum_{j=1}^{k} d_j^2 y_j : y \in \operatorname{Argmax}\left\{ f_k^2(\beta) \right\} \right\}, \end{aligned} \tag{3.17}$$

and $k \in N^2$. To illustrate the execution of the forward procedure for the follower, we applied it to the instance BKP$_1$ used in the previous examples. The results are reported in Example 3.7.

Example 3.7. The results after the second step of the forward phase are reported in Table 3. This example shows that the values of the leader subproblems do not increase always because of the choice of the follower. For y_3 ($k = 3$) and $\beta = 1$, the value for the leader decreases from 5 (for y_2 and $\beta = 1$) to 2. This new value is associated with $d_3^2 = 2$ in order to satisfy the objective of the follower. Note that we applied the dynamic recurrence rules on the leader objective function for y_2 ($k = 2$) and $\beta = 2$. The two values of the follower are equivalent: $f_{y_1}^2(2) = f_{y_1}^2(0) + c_2 = 2$. In this case, the value for the leader is 5 because $(\widehat{f}_{y_1}^1(2) = 5) > (\widehat{f}_{y_1}^1(0) + d_2^2 = 1)$.

```
$\vartheta \leftarrow \vartheta^*$;
for $k = n^1$ to 2 do
    if $f_k^1(\vartheta) = f_{k-1}^1(\vartheta - a_k^1) + d_k^1$ then
        $x_k^* = 1$;
        $\vartheta \leftarrow \vartheta - a_k^1$;
    end
    else
        $x_k^* = 0$;
    end
end
if $f_1^1(\vartheta) = 0$ then $x_1^* = 0$; else $x_1^* = 1$; end
```

Algorithm 3: Backtracking procedure for the leader.

This dynamic programming approach can be improved by fixing some variables of the problem BKP to their optimal value, and by reducing the size of the interval $[\mathrm{lb}_z, \mathrm{ub}_z]$ as discussed in the previous sections. Once this new interval has been computed, the first step of the forward phase can be applied with $\vartheta \in [0, \mathrm{ub}_z]$ instead of $\vartheta \in [0, b]$. Since $b \geq \mathrm{ub}_z$, this may reduce the number of steps of Algorithm 2. We do not apply this dynamic programming procedure up to the value of b, because there is no solution with a value better than LB for $\vartheta > \mathrm{ub}_z$. Similarly, in the second step of the forward phase, we use the following interval for β: $\beta \in [0, b - \mathrm{lb}_z]$.

3.4.2. Generating an Optimal Solution for BKP: The Backtracking Phase

Let (x^*, y^*) be an optimal solution for BKP. The objective of the backtracking phase is to generate a solution (x^*, y^*) with a value that is equal to the value computed in the forward phase. Before we introduce the backtracking procedure, we define first the optimal value that is determined in the forward phase. The optimal solution can be defined using the following rule:

$$d^1 x^* + d^2 y^* = f_{n^1}^1(\vartheta^*) + \widehat{f}_{n^2}^1(b - \vartheta^*) = \max_{0 \leq \vartheta \leq b} \left\{ f_{n^1}^1(\vartheta) + \widehat{f}_{n^2}^1(b - \vartheta) \right\}. \tag{3.18}$$

The main idea is based on the fact that for each leader decision with ϑ resources consumed, the follower reaction has to be optimum for the remaining $b - \vartheta$ resources.

From the value ϑ^*, we apply the backtracking procedure on the leader variables described in Algorithm 3. For the follower variables, we apply the backtracking procedure described in [8] by taking into account both the leader and the follower objective functions, and starting with the value $b - \vartheta^*$.

For a given k, if the follower has different equivalent choices, the value of y_k^* is determined according to the profit of the leader. Note that the variable y_k^* can take the value 0 or 1, if the two choices are equivalent for the leader and the follower. In Example 3.8, we illustrate the execution of the backtracking procedure on the instance BKP_1 used in the previous examples.

Table 4: Backtracking procedure for the follower and leader of BKP_1.

	k										
	1		2		3		4		1	2	3
ϑ/β	$f_1^2(\beta)$	$\widehat{f}_1^1(\beta)$	$f_2^2(\beta)$	$\widehat{f}_2^1(\beta)$	$f_3^2(\beta)$	$\widehat{f}_3^1(\beta)$	$f_4^2(\beta)$	$\widehat{f}_4^1(\beta)$	$f_1^1(\vartheta)$	$f_2^1(\vartheta)$	$f_3^1(\vartheta)$
0	0	0	0	0	0	0	0	0	(0)	(0)	0
1	2	(5)	2	(5)	3	2	3	2	$-\infty$	2	2
2	2	5	2	5	5	(7)	5	(7)	$-\infty$	$-\infty$	(7)
3	2	5	4	6	5	7	5	7	3	3	9
4	2	5	4	6	7	8	7	8	$-\infty$	5	5

Example 3.8. The optimal value for the problem BKP_1 described in Example 3.2 is determined from Tables 2 and 3 as follows:

$$\max_{0\le\vartheta\le4}\left\{f_{n^1}^1(0)+\widehat{f}_{n^2}^1(4), f_{n^1}^1(1)+\widehat{f}_{n^2}^1(3), f_{n^1}^1(2)+\widehat{f}_{n^2}^1(2), f_{n^1}^1(3)+\widehat{f}_{n^2}^1(1), f_{n^1}^1(4)+\widehat{f}_{n^2}^1(0)\right\}=14, \tag{3.19}$$

with $\vartheta^*=2$.

The results of the backtracking procedure for the follower and leader of BKP_1 are given in Table 4. To determine the optimal action of the leader, we apply Algorithm 3 starting with $\vartheta^*=2$. The optimal action for the leader is $x_1^*=0$, $x_2^*=0$, and $x_3^*=1$. For the follower, we apply the backtracking procedure described in [8], starting from $4-\vartheta^*=2$. The optimal reaction of the follower is $y_1^*=1$, $y_2^*=0$, $y_3^*=1$, and $y_4^*=0$.

3.5. Outline of the Algorithm

The outline of our exact algorithm for BKP is given in Figure 1. Each box in this figure corresponds to a step of our algorithm. The numbers identify the sequence by which the operations are performed.

The algorithm starts by computing an upper bound for BKP through the exact resolution of HBKP. The next step consists in finding a good lower bound for BKP by computing a feasible solution for the problem. For this purpose, we solve first the problem CBKP using Algorithm 1. As referred to in Section 3.1, each solution generated by Algorithm 1 can be associated to a value of the parameter z in $FBKP_z$. From the set of solutions found by Algorithm 1, we select the n best solutions, and we solve the n problems $FBKP_z$ for the corresponding values of the parameter z.

The upper and lower bounds (denoted, resp., by UB and LB in Figure 1) obtained in the previous steps are used to fix the variables of the leader to their optimal values. This is done by applying the fixing rules discussed in Section 3.2. The resulting problem is called the reduced problem in Figure 1. The lower bound LB is then used to reduce the size of the interval of possible values for z. The new computed interval may help in reducing the number of steps of the dynamic programming procedures that are applied next. Similarly, the size of the reduced problem (i.e., solved with dynamic programming in the next step of the algorithm) is smaller than the original BKP, and hence, it is easier to solve using dynamic programming.

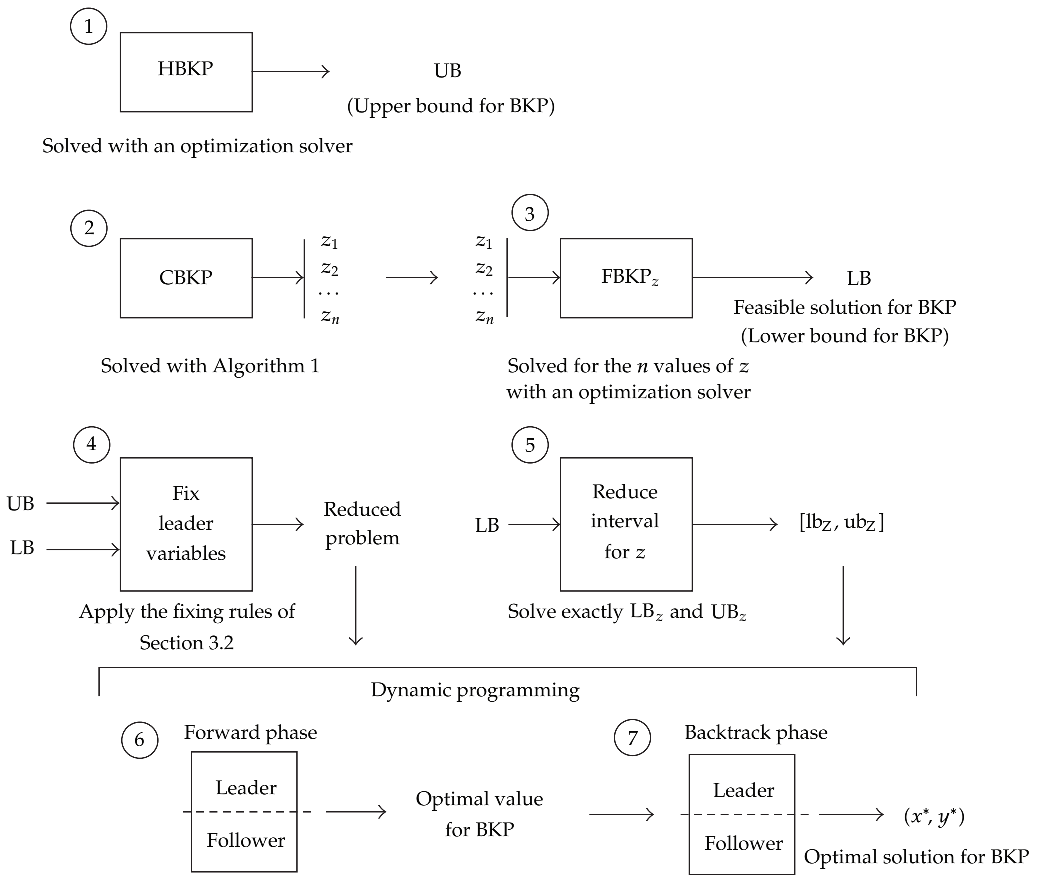

Figure 1: Outline of the algorithm.

The next step of our algorithm consists in applying the forward phase of the dynamic programming procedure to the reduced problem in order to compute the value of an optimal solution for BKP. Finally, from this optimal value, an optimal solution for BKP is generated using the backtracking procedure described in Section 3.4.2.

4. Computational Results

In this section, we report on the results of the computational study that was performed to evaluate the performance of our algorithm. Our approach is compared with other methods proposed in the literature. The limits of our algorithm are discussed, and a strategy to overcome these limits is presented and tested. All the algorithms analyzed in this section were coded in C++. We used the version 12.0 of the commercial optimization solver CPLEX. The experiments were conducted on a PC with 2.4 GHz with 4 GB of RAM.

Three sets of computational experiments were performed. In the first one, we compare our algorithm with the branch-and-bound algorithm proposed by Moore and Bard in [9]. In the second one, we perform a comparative study between our algorithm and the method of Brotcorne et al. described in [10]. The third set of experiments was conducted to analyze the impact of each component of our algorithm in the performance of our global approach. The

Table 5: Computing time for BM and MACH1 for uncorrelated instances with $L = 100$ and $\alpha = 0.25$.

n^1	n^2	t_{BB}	t_{MACH1}
10	10	0	0
10	50	0	0
10	100	0	1
50	10	2	0
50	50	4	0
50	100	22	1
100	10	6	0
100	50	28	0
100	100	130	1

limits of our algorithm are illustrated from a computational standpoint, and a variant that handles these issues is described and tested.

We used the generator proposed by Martello et al. [21] to generate instances of the knapsack problem. This generator gives us the data for the coefficients a^1, a^2, d^1, and c of BKP. The value of b is computed as follows: $b = \alpha(\sum_{i=1}^{n^1} a_i^1 + \sum_{j=1}^{n^2} a_j^2)$, with $\alpha \in \{0.50, 0.75\}$. The input data for the leader (d^2) is generated randomly, such that all the coefficients are in the interval $[1, L]$, with $L \in \{100, 1000\}$. We generated instances with uncorrelated coefficients (UC), and with correlated coefficients (C) [17].

In Table 5, we compare the performance of our algorithm (denoted by MACH1) with the branch-and-bound algorithm proposed in [9] (denoted by BM). For these experiments, we used a set of small instances with uncorrelated coefficients with $\alpha = 0.25$ and $L = 100$. We generated 5 instances for each set of instances characterized by the parameters n^1 and n^2. In Table 5, we report on the average computing time (in seconds) required by BM and MACH1 to find an optimal solution for these instances. The computing times for BM and MACH1 are given, respectively, in the columns t_{BM} and t_{MACH1}.

Table 5 shows the difficulty of the branch-and-bound algorithm of Moore and Bard in solving these instances, while our approach remains very fast. Note that the branch-and-bound algorithm is not able to find the optimal solution of medium instances with $\alpha = 0.5$, $n^1 = 200$, and $n^2 = 200$ and correlated coefficients in less than one hour. As we will see in the next experiments, our approach can solve these (and larger) instances very efficiently.

The results of our second set of experiments are reported in Table 6. We compare the performance of our algorithm with the method described in [10]. In the sequel, the latter will be denoted by BHM. The algorithm BHM is composed by two phases: the first phase is a dynamic programming procedure applied to the follower problem to determine all the possible reactions of the follower; in the second phase, a reformulated integer problem is solved by making the link between the actions of the leader and the reactions of the follower.

For these experiments, we used harder instances. The sets of instances are characterized by the parameters n^1, n^2, and α. Again, we generated randomly 5 instances for each set. The parameters were chosen as follows: $n^1 \in \{50, 100\}$, $n^2 \in \{50, 100\}$, and $\alpha \in \{0.50, 0.75\}$. The coefficients were generated in the interval $[1, 1000]$, and we considered both uncorrelated and correlated instances. For these experiments, we used a maximum time limit of 600 seconds.

In column $\mathrm{opt}_{\mathrm{BHM}}$, we give the number of times the algorithm BHM finds a proven optimal solution within the maximum time limit. Note that our algorithm always finds

Table 6: Comparison between BHM and MACH1.

n^1	n^2	α	UC			C		
			opt$_{\text{BHM}}$	t_{BHM}	t_{MACH1}	opt$_{\text{BHM}}$	t_{BHM}	t_{MACH1}
50	50	0.5	5	0.4	0.6	2	485.8	1.2
50	100	0.5	5	5	0.4	0	>600	1.0
100	50	0.5	5	3.8	0.6	0	>600	1.2
100	100	0.5	4	421.6	1.4	0	>600	2.0
50	50	0.75	5	0.4	0.4	4	177.8	0.8
50	100	0.75	5	2.8	0.8	0	>600	1.2
100	50	0.75	5	0.8	0.8	0	>600	1.2
100	100	0.75	5	12.0	0.8	0	>600	1.4

an optimal solution within this time limit. The average computing time required by BHM and MACH1 to find a proven optimal solution is given in the columns t_{BHM} and t_{MACH1}, respectively. For BHM, the average time reported in Table 6 corresponds only to the cases where this algorithm finds a proven optimal solution within the time limit of 600 seconds.

From the results of Table 6, it is clear that our algorithm outperforms the approach of Brotcorne et al. [10]. Our approach remains very fast both for the uncorrelated and the correlated instances, while BHM is not able to find the optimum solution for most of the correlated instances. The performance of our algorithm is due in a large part to the strategies used for fixing the value of some variables, to our procedures for computing lower and upper bounds, and in particular to the strategy for reducing the interval of values for the parameter z in FBKP$_z$. Note that our algorithm does not have any difficulty in proving the optimality of the solution found in the backtracking phase, since the optimal value is known at the end of the forward phase. The algorithm BHM spends more time precisely in its second phase when it solves the reformulated problem. At this stage, this algorithm has no information for the value of the optimal solution. Its computing time increases quickly with the correlation of the instances because in this case the size of the reformulated integer problem becomes larger.

In our final set of experiments, we focus on our algorithm. Despite its efficiency compared with other approaches, our algorithm may experience some difficulties with memory space for larger instances. These difficulties are illustrated in Table 7. The instances used in this case were generated as in the previous experiments, and with the parameters n^1, n^2, and α given in Table 7. For each case, we generated 5 instances. Since these difficulties are due to the dynamic programming part of our algorithm, we used in these experiments a version of MACH1 in which the procedures described in Sections 3.1, 3.2, and 3.3 are disabled. We will denote this version by MACH1′. Table 7 reports the average computing time for the MACH1′ and for a variant that will be described hereinafter. The entry mem in Table 7 means that MACH1′ did not complete because of the memory space required for its execution. This problem arises for the largest instances with $n^1 = 500$, $n^2 = 500$, and $\alpha = 0.75$. Recall that, for the coefficient b, we have $b = \alpha(\sum_{i=1}^{n^1} a_i^1 + \sum_{j=1}^{n^2} a_j^2)$. In this case, the value of b can be very large, and that is the main cause for this memory problem.

To overcome this issue, we propose a variant of the algorithm MACH1 (denoted by MACH2) that consists in replacing the backtracking phase based on dynamic programming in MACH1 by the exact resolution of the problem FBKP$_z$ right after the forward phase. The forward phase gives us the optimal value for BKP. This optimal value is used for solving FBKP$_z$. Since we know this optimal value in advance, the resolution of FBKP$_z$ becomes easier.

Table 7: Comparison between MACH1′ and MACH2′.

n^1	n^2	α	UC $t_{\text{MACH1}'}$	UC $t_{\text{MACH2}'}$	C $t_{\text{MACH1}'}$	C $t_{\text{MACH2}'}$
250	250	0.5	5.0	5.0	5.1	5.4
250	500	0.5	13.8	14.0	14.0	14.6
500	250	0.5	11.6	12	11.8	12.8
500	500	0.5	22.4	23.2	22.4	23.2
250	250	0.75	8.2	8.4	8.4	8.4
250	500	0.75	20.2	20.4	21.0	21.4
500	250	0.75	17.0	17.0	17.6	17.6
500	500	0.75	mem	33.8	mem	35.6

Table 8: Comparison between MACH2′ and MACH2.

n^1	n^2	α	UC qual$_h$ %	UC t_h	UC fix %	UC int$_z$ %	UC $t_{\text{MACH2}'}$	UC t_{MACH2}	C qual$_h$ %	C t_h	C fix %	C int$_z$ %	C $t_{\text{MACH2}'}$	C t_{MACH2}
500	500	0.5	99.0	3.0	9.8	8.51	21.0	12.6	99.2	0.8	0	45.5	21.4	15.8
500	1000	0.5	99.0	2.6	0.0	9.98	50.2	35.8	99.0	1.4	0	48.9	51.0	44.8
1000	500	0.5	99.0	3.8	8.0	6.85	43.0	25.6	99.0	1.0	0	34.47	44.2	31.8
1000	1000	0.5	99.0	2.4	0.0	9.39	82.8	49.6	99.0	1.4	0	44.5	85.2	65.2
500	500	0.75	99.2	2.2	12.6	6.22	30.8	15.8	99.0	1.0	0	25.5	31.2	21.4
500	1000	0.75	99.0	1.6	0.0	5.78	74.6	49.8	99.4	1.2	0	24.3	75.0	56.6
1000	500	0.75	99.0	2.4	19.2	4.36	63.0	26.6	99.0	1.2	0	18.7	64.0	41.2
1000	1000	0.75	99.0	2.8	0.0	5.63	122.0	69.8	99.0	1.0	0	25.7	124.0	81.4

In MACH2, we keep only two columns for dynamic programming at each iteration of the forward phase, and hence, the memory space that is necessary decreases.

In Table 7, we report on the average computing time required by a version of this variant without the procedures described in Sections 3.1, 3.2, and 3.3 (as in MACH1′). This version will be denoted by MACH2′. All the instances are solved up to optimality with a very small increase in the computing time compared to MACH1′. With this new variant, the problem with memory space does not occur anymore.

In Table 8, we compare the complete version of the algorithm MACH2 with the version MACH2′. In our implementation of MACH2, we solved the problem FBKP$_z$ with the 10 best solutions generated by Algorithm 1 to find a valid lower bound. The objective of these experiments was to evaluate the impact of the additional components of our approach, namely, the polynomial procedure for solving CBKP described in Section 3.1 (Algorithm 1) and the reduction procedures described in Sections 3.2 and 3.3. We generated randomly 5 instances for each set of instances as in the previous experiments with the parameters n^1, n^2, and α given in Table 8. The meaning of the entries in Table 8 is the following:

(i) qual$_h$: quality of the solution obtained with Algorithm 1 described in Section 3.1 (value of the best solution given by Algorithm 1 divided by the value of the optimal solution of the BKP);

(ii) t_h: computing time (in seconds) required by Algorithm 1;

(iii) fix: percentage of variables that were fixed;

(iv) int_z: measure of the reduction achieved with the procedure described in Section 3.3; the values in this column are computed as follows: $(\text{ub}_z - \text{lb}_z)/b$;

(v) $t_{\text{MACH2}'}$: computing time (in seconds) required by MACH2′;

(vi) t_{MACH2}: computing time (in seconds) required by MACH2.

From the results of Table 8, we can observe that the additional components of the algorithm have a positive impact on the performance of our global approach. The average computing times for all the sets of instances decreased with MACH2. For the set of instances with $n^1 = 1000$, $n^2 = 500$, and $\alpha = 0.75$, the reduction is greater than 50%.

The lower bound given by CBKP is strong. Furthermore, it is computed very efficiently with Algorithm 1. The average computing time required by this algorithm is always smaller than 4 seconds. The fixing rules presented in Section 3.2 have a limited impact on the correlated instances. This can be explained by the quality of the upper bound that was considered (given by HBKP), and by the correlation between the coefficients of the instances. These rules perform better on the uncorrelated instances. While the lower bound on the optimal value of BKP does not seem to be very useful for fixing the values of the variables, it is for reducing the interval of feasible values for z. Although the size of this interval decreases for all the instances, it is more significant for the uncorrelated instances. The reduction of the size of this interval has a strong influence on the resolution of the reduced problem with dynamic programming. Indeed, it implies reducing the capacity of the knapsack constraint at each step of the dynamic programming procedures. That explains in part the better performance of MACH2 compared with MACH2′.

5. Conclusions

In this paper, we described a new exact algorithm for bilevel 0-1 knapsack problems (BKPs). We developed an original method for solving the linear relaxation of BKP, and we proposed a method for computing good feasible solutions for this problem using the information provided by the solutions of this linear relaxation. We described different strategies to enhance the resolution of BKP based on a valid upper and lower bound for the problem. Finally, we presented a dynamic programming procedure to find the integer optimal solution of the problem. To evaluate the performance of our approach, we conducted a set of computational experiments. Our results were compared with other algorithms proposed in the literature. The results that we obtained show that our algorithm clearly outperforms other state-of-the-art methods presented so far.

Acknowledgments

This work was partially supported by the Portuguese Science and Technology Foundation through the postdoctoral Gsssrant SFRH/BPD/64766/2009 for Raid Mansi, by the Algoritmi Research Center of the University of Minho for Cláudio Alves and José Valério de Carvalho, and 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 for Saïd Hanafi. The authors would like to thank the anonymous referees for their constructive comments, which helped improving the quality of the paper.

References

[1] J. Bracken and J. T. McGill, “Mathematical programs with optimization problems in the constraints,” *Operations Research*, vol. 21, pp. 37–44, 1973.

[2] H. Stackelberg, *The Theory of the Market Economy*, Oxford University Press, Oxford, UK, 1952.

[3] K. Shimizu, Y. Ishizuka, and J. F. Bard, *Nondifferentiable and Two-Level Mathematical Programming*, Kluwer Academic, 1996.

[4] S. Dempe, *Foundations of Bilevel Programming*, vol. 61, Kluwer Academic, Dordrecht, The Netherlands, 2002.

[5] B. Colson, P. Marcotte, and G. Savard, “Bilevel programming: a survey,” *Quarterly Journal of the Belgian*, vol. 3, no. 2, pp. 87–107, 2005.

[6] S. Dempe and K. Richter, “Bilevel programming with knapsack constraints,” *Central European Journal of Operations Research*, vol. 8, no. 2, pp. 93–107, 2000.

[7] A. Plyasunov, “The bilevel optimization problem with multiple-choice knapsack problem in lower level,” *Discrete Analysis and Research Operations*, vol. 10, pp. 44–52, 2003.

[8] L. Brotcorne, S. Hanafi, and R. Mansi, “A dynamic programming algorithm for the bilevel knapsack problem,” *Operations Research Letters*, vol. 37, no. 3, pp. 215–218, 2009.

[9] J. T. Moore and J. F. Bard, “The mixed integer linear bilevel programming problem,” *Operations Research*, vol. 38, no. 5, pp. 911–921, 1990.

[10] L. Brotcorne, S. Hanafi, and R. Mansi, “One-level reformulation of the bilevel knapsack problem using dynamic programming,” Tech. Rep., Université de Valenciennes et du Hainaut-Cambrésis, France, 2011.

[11] S. Dempe, “Annotated bibliography on bilevel programming and mathematical programs with equilibrium constraints,” *Optimization*, vol. 52, no. 3, pp. 333–359, 2003.

[12] O. Marcotte and G. Savard, *Bilevel Programming: Applications*, Kluwer Academic, Dordrecht, The Netherlands, 2001.

[13] S. Kosuch, P. Le Bodic, J. Leung, and A. Lisser, “On a stochastic bilevel programming problem with knapsack constraints,” in *Proceedings of the International Network Optimization Conference*, 2009.

[14] J. F. Bard and J. T. Moore, “An algorithm for the discrete bilevel programming problem,” *Naval Research Logistics*, vol. 39, no. 3, pp. 419–435, 1992.

[15] P. Loridan and J. Morgan, “Weak via strong Stackelberg problem: new results,” *Journal of Global Optimization*, vol. 8, no. 3, pp. 263–287, 1996.

[16] T. A. Edmunds and J. F. Bard, “Algorithms for nonlinear bilevel mathematical programs,” *IEEE Transactions on Systems, Man and Cybernetics*, vol. 21, no. 1, pp. 83–89, 1991.

[17] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*, John Wiley and Sons, New York, NY, USA, 1990.

[18] E. Balas and C. H. Martin, “Pivot and complement-a heuristic for 0-1 programming,” *Management Science*, vol. 26, no. 1, pp. 86–96, 1980.

[19] D. Fayard and G. Plateau, “An algorithm for the solution of the 0-1 knapsack problem,” *Computing*, vol. 33, no. 5, pp. 1259–1273, 2006.

[20] B. Gavish and H. Pirkul, “Efficient algorithms for solving multiconstraint zero-one knapsack problems to optimality,” *Mathematical Programming*, vol. 31, no. 1, pp. 78–105, 1985.

[21] S. Martello, D. Pisinger, and P. Toth, “Dynamic programming and strong bounds for the 0-1 knapsack problem,” *Management Science*, vol. 45, no. 3, pp. 414–424, 1999.