Hybrid approaches for the two-scenario max-min knapsack problem

Saïd Hanafi, Raïd Mansi, Christophe Wilbaut, Arnaud Fréville

INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

Intl. Trans. in Op. Res. 19 (2012) 353–378
DOI: 10.1111/j.1475-3995.2011.00836.x

Hybrid approaches for the two-scenario max–min knapsack problem

Saïd Hanafi[a], Raïd Mansi[b], Christophe Wilbaut[a] and Arnaud Fréville[a]

[a]*LAMIH, Université de Valenciennes, 59313 Valenciennes Cedex 9, France*
[b]*Escola de Engenharia, Universidade do Minho, 4710-057Braga, Portugal*
E-mail: said.hanafi@univ-valenciennes.fr [Hanafi]; raidmm@yahoo.fr [Mansi]; christophe.wilbaut@univ-valenciennes.fr [Wilbaut]

Received 18 February 2011; received in revised form 2 September 2011; accepted 3 September 2011

Abstract

In this paper, we deal with the two-scenario max–min knapsack (MNK) problem. First, we consider several formulations of MNK as a mixed integer programming problem. Then, we propose a hybrid method as an alternative to solve the MNK exactly. The approach combines relaxation technique and the temporary setting of variables to improve iteratively two sequences of upper and lower bounds. More precisely, pseudo-cuts are added to the problem to strengthen the bounds and reduce the gap between the best lower bound and the best upper bound. The algorithm stops when the proof of the optimality of the best solution is found. We also use a reduction technique to set some variables definitively at their optimal values. Numerical experiments demonstrate the robustness of the approach. In particular, our algorithm is efficient to solve large and correlated instances of MNK.

Keywords: Knapsack problem; max–min optimization; relaxation; mixed integer programming hybrid method

1. Introduction

The family of knapsack problems (KPs) is varied and includes numerous problems that are much studied in operations research and combinatorial optimization communities. KP is a well-known combinatorial optimization problem (Kellerer et al., 2004). From a set of N items, each characterized by an integer nonnegative weight and an integer nonnegative profit, the aim of the 0–1 KP is to select a subset of N such that the associated total profit is maximized and the associated total weight does not exceed a given knapsack capacity. KP can be formulated as the following 0–1 linear program:

$$(KP) \rightarrow \max \left\{ \sum_{j \in N} c_j x_j : \sum_{j \in N} a_j x_j \leq b, x_j \in \{0, 1\}, j \in N \right\},$$

where $n = |N|$ is the number of items, c_j and a_j refer to the profit and the weight of item $j \in N$, respectively, and b is the capacity of the knapsack. Although the KP has a natural and simple formulation, it remains a challenge because of its complexity as it is a nondeterministic polynomial time NP-hard problem (Garey and Johnson, 1979). KP has been studied extensively by the operational research community, in particular because of its applications such as cutting stock (Gilmore and Gomory, 1966) or capital budgeting (Lorie and Savage, 1955), among others. Based on the dynamic programming, branch-and-bound, or the core concept for instance, several efficient exact and heuristic approaches can be found in the literature (Balas and Zemel, 1980; Martello and Toth, 1990; Pisinger, 1995a; Toth, 1980). Many other problems can be derived from KP by integrating other constraints and/or changing the objective function. For instance, the multidimensional KP (Boussier et al., 2010; Fréville, 2004; Fréville and Hanafi, 2005; Wilbaut et al., 2008) is obtained from the KP when considering more than one resource constraint, whereas the multiple-choice (multidimensional) KP introduces a partition of the items and choice constraints associated to each group (or class) of items (Cherfi and Hifi, 2009; Kellerer et al., 2004; Pisinger, 1995b).

Max–min multiscenarios KP (MNK) is a particular case in which items' profit changes according to a set S of scenarios. As in KP, we have to choose a subset of items under a capacity constraint. However, in the case of MNK the choice of items is made so that the minimum value of the $s = |S|$ objective linear functions associated to items is maximized. In this paper, we address particularly the case where $s = 2$ that corresponds to the two-scenario max–min KP (Eben-Chaime, 1996; Yu, 1996). This problem can be formulated in the following form:

$$(MNK) \rightarrow \max \{ \min \{c^1 x, c^2 x\} : ax \leq b, x \in \{0, 1\}^n$$

where c^1 and c^2 represent the two profit vectors of the n items, a is the weight vector of the n items, and b represents the total capacity of the knapsack. Binary variable x_j is the decision variable that takes the value 1 if item j is selected and 0 otherwise for $j = 1, 2, \ldots, n$. We assume that all components c_j^1, c_j^2, a_j of associated input vector and the capacity b are nonnegative integer.

MNK is related to the particular "max–min optimization" approach of multiobjective optimization in which we look only for a solution maximizing the minimum profit value (Du and Pardalos, 1995; Aissi, 2005; Roy, 2010). Such approach is also referred to as the robust optimization in the literature (Yu, 1996). MNK can be reduced to KP when we consider only one objective function, implying MNK is NP-hard. In addition, MNK is strongly NP-hard when the number of scenarios is unbounded (Yu, 1996).

We found some papers in the literature dealing with the exact solution of MNK when $s > 2$. First, four branch-and-bound-based algorithms were proposed (Iida, 1999; Kouvelis and Yu, 1997; Yu, 1996; Amrani, 1997). In all the cases, authors use the surrogate relaxation of MNK to derive both upper and lower bounds. Iida also proposed the use of a Lagrangian relaxation to obtain other bounds. According to experiments reported in these papers, the algorithms were able to solve instances with $n \leq 90$ items, and $s \leq 30$ scenarios. Then, Taniguchi et al. (2008) proposed the use of a pegging test to reduce the size of the problem instance by setting variables at their optimal values in a first phase, and to optimally solve the remaining MNK using again a branch-and-bound approach in a second phase. According to their conclusions, the algorithm is able to solve optimally instances with up to 1000 items and 30 scenarios. Finally, another recent paper (Sbihi, 2010) presents a heuristic approach to tackle larger instances. The approach is a cooperative local-search-based

algorithm, which combines a greedy heuristic to build an initial solution, a restricted guided local search, and a very large local search. The cooperative local search controls the algorithm, using the previous subroutines to intensify and diversify the search. The algorithm is validated on instances with up to 10,000 items and 100 scenarios.

MNK with two scenarios has been studied by Taniguchi et al. (2009). It is an additional paper to Taniguchi et al. (2008), in which the authors introduce a specific technique to solve more efficiently the case with $s = 2$. The main difference lies in the use of a virtual pegging test allowing the fixing process to be improved and hence the size of the search tree of the branch-and-bound method to be limited. The method is based on the estimation of a lower bound of the problem as close as possible to the optimal value of the problem. If the algorithm does not find a solution corresponding to this bound in the current iteration, the value of the lower bound decreases, and the value used previously can serve as an upper bound. Experiments reported in the paper showed that this algorithm is very efficient for weakly correlated instances for which it can set a large number of variables in the first phase.

In this paper, we are interested in MNK with $s = 2$, and we propose an alternative approach to the one of Taniguchi et al. (2009) based on a similar philosophy: apply a heuristic to set as many variables of the initial problem as possible, and then solve the remaining problem with an exact method. The first phase also contributes to the solution of the MNK by reducing the gap between the lower and upper bounds. Contributions of our work can be summed up as following: (a) we propose several equivalent mixed integer linear program (MIP) formulations of MNK and compare these formulations when they are used in CPLEX MIP solver; (b) we propose another new hybrid approach allowing the setting of a large number of variables in the problem. This method can also be used as a convergent method, and so as an exact method to solve the initial problem. We strengthen it with the addition of cuts during the search. Our approach allows the reduction of large size instances (up to $n = 20{,}000$).

The rest of the paper is organized as follows. In Section 2, we present several models equivalent to MNK that will be evaluated and compared in our numerical experiments. Section 3 describes a surrogate heuristic to generate a feasible solution of MNK. Section 4 is devoted to the presentation of our method to reduce the initial problem by setting variables at their optimal values, and also to generate good lower and upper bound values. Section 5 presents a detailed analysis of the experiments performed to validate the approach on a set of 900 large and correlated instances.

2. Equivalent MIP formulations

In this section, we propose several alternative and equivalent formulations for MNK. In the rest of the paper, we consider the special case with $s = 2$. The most used equivalent formulation for the MNK in the literature is the reformulation as an MIP. This reformulation is obtained by introducing a new continuous variable z representing the minimum value between the two objective functions (i.e. $\min\{c^1x, c^2x\}$). The use of this continuous variable allows the writing of MNK as a mono-objective mixed integer linear program

$$(MIP)\begin{cases}\max z\\ s.t.\\ \quad z \le c^1 x & (1a)\\ \quad z \le c^2 x & (1b)\\ \quad ax \le b & (1c)\\ \quad x \in \{0,1\}^n \quad z \ge 0.\end{cases}$$

In the (MIP) problem, we maximize the lower bound z on the two objective functions c^1x and c^2x, which is equal to the minimum of c^1x and c^2x at the optimality. In the following, the resulting MIP formulation for the problem MNK will be referred to as the standard MIP formulation. The standard MIP formulation has n binary variables, one continuous variable, and three constraints. We will consider this formulation in our experiments.

The following proposition provides another equivalent formulation of MNK with n binary variables and one continuous variable, but only two constraints.

Proposition 1. *If $v(P)$ denotes the optimal value of problem P, then we have $v(MNK) = v(MIP1) = v(MIP2)$, where*

$$(MIP1)\begin{cases}\max c^1x - y\\ s.t.\\ \quad c^1x - y \le c^2x\\ \quad ax \le b\\ \quad x \in \{0,1\}^n \quad y \ge 0\end{cases} \quad and \rightarrow (MIP2) \rightarrow \begin{cases}\max c^2x - y\\ s.t.\\ \quad c^2x - y \le c^1x\\ \quad ax \le b\\ \quad x \in \{0,1\}^n \quad y \ge 0.\end{cases}$$

Proof. The MIP1 formulation is obtained from the MIP formulation as follows. First, we transform the inequality constraint (1a) into equality constraint $z = c^1x - y$, where variable $y \ge 0$ corresponds to the slack variable, and such that $c^1x - y$ is nonnegative. Next, we replace the value $z = c^1x - y$ into the objective function and into constraint (1b). Then, we can drop constraint (1a). Thus, we obtain the MIP1 formulation. Similarly, the MIP2 formulation is obtained from the MIP formulation by replacing inequality constraint (1b) by $z = c^2x - y$. □

Another formulation of MNK with only two constraints is possible. In that case, solving the MNK is equivalent to solving two bi-dimensional KPs (BKPs), as shown in the following proposition.

Proposition 2. *We have $v(MNK) = max\{v(BKP1), v(BKP2)\}$, where*

$$(BKP1)\begin{cases}\max c^1x\\ s.t.\\ \quad c^1x \le c^2x\\ \quad ax \le b\\ \quad x \in \{0,1\}^n\end{cases} \rightarrow \text{and} \rightarrow (BKP2)\begin{cases}\max c^2x\\ s.t.\\ \quad c^2x \le c^1x\\ \quad ax \le b\\ \quad x \in \{0,1\}^n.\end{cases}$$

Proof. Let x^* be an optimal solution of MNK. Then, by definition we have $v(\text{MNK}) = \min\{c^1x^*, c^2x^*\}$. If $c^1x^* \leq c^2x^*$ then we have $v(\text{MNK}) = c^1x^*$ and x^* is a feasible solution of BKP1, so we have $v(\text{MNK}) \leq v(\text{BKP1})$. Similarly, if $c^2x^* \leq c^1x^*$ then we have $v(\text{MNK}) = c^2x^*$ and x^* is a feasible solution of BKP2, thus we have $v(\text{MNK}) \leq v(\text{BKP2})$. Thus we have $v(\text{MNK}) \leq \min\{v(\text{BKP1}), v(\text{BKP2})\}$. By definition of the problems BKP1 and BKP2, we have $v(\text{BKP1}) \leq v(\text{MNK})$ and $v(\text{BKP2}) \leq v(\text{MNK})$. Therefore, we have $\max\{v(\text{BKP1}), v(\text{BKP2})\} \leq v(\text{MNK})$. Hence we showed that $v(\text{MNK}) = \max\{v(\text{BKP1}), v(\text{BKP2})\}$. □

To formulate MNK as a mono-objective problem, we can also use a quadratic formulation as shown in the next proposition.

Proposition 3. *We have $v(MNK) = v(Q)$, where*

$$(Q)\begin{cases}\max y(c^1 - c^2)x + c^2x \\ \quad s.t. \\ \quad\quad (2y-1)(c^1 - c^2)x \leq 0 \\ \quad\quad ax \leq b \\ \quad\quad x \in \{0,1\}^n \quad y \in \{0,1\}.\end{cases}$$

Proof. Let x^1 and x^2 the optimal solutions of BKP1 and BKP2, respectively. We introduce a binary variable y in order to put together the models BKP1 and BKP2 in one quadratic model (Q), where

$$y = \begin{cases}1 & \text{if an optimal solution is associated to BKP1 problem} \\ 0 & \text{if an optimal solution is associated to BKP2 problem.}\end{cases}$$

Then if $y = 1$, the objective function must be c^1x and the constraints are $(c^1 - c^2)\, x \leq 0$ and $ax \leq b$, and if $y = 0$ the objective function must be c^2x and the constraints are $(c^2 - c^1)\, x \leq 0$ and $ax \leq b$. So, unifying these two cases provides the quadratic objective function $yc^1x + (1 - y)c^2x$, and the constraints $y(c^1 - c^2)x + (1 - y)(c^2 - c^1)\, x \leq 0$ and $ax \leq b$, i.e.

$$(Q)\begin{cases}\max yc^1x + (1-y)\, c^2x \\ \quad s.t. \\ \quad\quad y(c^1 - c^2)x + (y-1)(c^1 - c^2)x \leq 0 \\ \quad\quad ax \leq b \quad x \in \{0,1\}^n \quad y \in \{0,1\}.\end{cases}$$

Note that if $y = 1$ then $(Q) \equiv$ (BKP1), whereas if $y = 0$ then $(Q) \equiv$ (BKP2). □

Recall that obtaining the optimal value of y allows us to identify the problem among BKP1 and BKP2 for which we find an optimal solution of MNK. Note that solving the MNK from BKP1 and BKP2 is easier than solving directly MNK because we solve a reduced model in this case. Solving a quadratic problem is generally very difficult in practice. In the following, we present a linearization of Q with the aim to simplify the search of an optimal value for y. We can use the linearization (LQ) of the previous quadratic model (Q), defined by:

$$(LQ)\begin{cases}\max z + c^2x \\ \quad s.t. \\ \quad (c^1 - c^2)x - M(1-y) \le \mathrm{z} \le (c^1 - c^2)x \\ \quad z \le M(1-y) \\ \quad -My \le z \le My \\ \quad ax \le b \\ \quad x \in \{0,1\}^n, \quad y \in \{0,1\}, (z)\end{cases} \Leftrightarrow \begin{cases}\max z + c^2x \\ \quad s.t. \\ \quad (c^1 - c^2)x \le M(1-y) \\ \quad (c^2 - c^1)x \le My \\ \quad (c^1 - c^2)x - M(1-y) \le \mathrm{z} \le (c^1 - c^2)x \\ \quad -My \le z \le My \\ \quad ax \le b \\ \quad x \in \{0,1\}^n, \quad y \in \{0,1\}, (z).\end{cases}$$

We also propose another linearization denoted as (L) for the previous quadratic model (Q) without a big parameter M.

Proposition 4. *We have $v(MNK) = v(L)$, where*

$$(L)\begin{cases} Max(c^1 - c^2)u + c^2x & (2a) \\ \quad s.t. & \\ \quad (c^1 - c^2)(2u - x) \le 0 & (2b) \\ \quad ax \le b & (2c) \\ \quad u \le ye & (2d) \\ \quad u \le x + (1-y)e & (2e) \\ \quad u \ge x + (y-1)e & (2f) \\ \quad (x,y) \in \{0,1\}^{n+1}, \quad u \in [0,1]^n & \end{cases}$$

where e denotes the vector of ones (1,1,. . . ,1) with an appropriate dimension.

Proof. It is obvious that $v(\mathrm{L}) = \max\{v(\mathrm{L}|y=0), v(\mathrm{L}|y=1)\}$. Now we will show that $v(\mathrm{L}|y=0) = v(\mathrm{BKP2})$ and $v(\mathrm{L}|y=1) = v(\mathrm{BKP1})$. If $y=0$, then constraint (2d) implies that $u=0$. Therefore the constraints (2e) and (2f) become redundant and we obtain the problem BKP2. So, we have $v(\mathrm{L}|y=0) = v(\mathrm{BKP2})$. If $y=1$, then constraint (2d) is redundant and the constraints (2e) and (2f) imply that $u=x$. So, we have $v(\mathrm{L}|y=1) = v(\mathrm{BKP1})$. Now, from the Proposition 2, we have $v(\mathrm{MNK}) = v(\mathrm{L})$. □

3. Upper and lower bounds of max–min KP

Relaxation and duality methods are very useful tools for mixed integer programming. They provide bounds for an optimization problem. Relaxing integrality constraints (LP-relaxation) provides directly an upper bound for the MNK problem. In the literature, some important properties related to the strength of the bounds are obtained with Lagrangian relaxation, surrogate relaxation, and composite relaxation. Lagrangian relaxation was introduced in the early 1970s through the pioneering work of Held and Karp (1970) on the traveling salesman problem. Subsequently, Geoffrion (1974) in the mid 1970s developed a general theory for applying the method by exploiting special problem structures. Many combinatorial optimization problems consist of an easy problem that is

complicated by the addition of extra constraints. Applying Lagrangian relaxation to these problems involves identifying these complicating constraints, then relaxing them by attaching penalties to the complicating constraints, and then absorbing them into the objective function. These penalties are known as the Lagrange multipliers. Due to the relaxation of the complicating constraints, the relaxed problem becomes much easier to solve. The next aim is to find tight upper and lower bounds to the problem by iteratively processing a sequence of modified subproblems. The surrogate relaxation, introduced by Glover (1975), replaces the original constraints with a single new constraint called a surrogate constraint.

As mentioned in the review of the literature, several authors used surrogate relaxation to derive upper and lower bounds for the MNK. This relaxation is then exploited in a branch-and-bound method to solve MNK exactly. Problem $S(\mu)$, for a given $\mu = (\mu_1, \mu_2) \geq 0$, defines a surrogate relaxation of MNK as follows:

$$S(\mu) \rightarrow \max\{\mu_1 c^1 x + \mu_2 c^2 x : ax \leq b, x \in \{0, 1\}^n\}.$$

Then the associated surrogate dual problem is defined as

$$(S) \rightarrow \min\{v(S(\mu)) : \mu \geq 0\}.$$

It is well known (see Glover, 1975) that, if for a multiplier μ, the optimal solution to $S(\mu)$ is feasible for MNK, then it is also optimal for MNK, because the two problems have the same objective function. Fréville and Plateau (1993, 1996) proposed a specific procedure able to find the solution of the surrogate dual within a finite number of iterations, practically independent of the number of variables. The property $v(S(\lambda\mu)) = v(S(\mu))$ for any $\lambda > 0$ and $\mu \geq 0$ allows the surrogate dual S to be solved in a one-dimensional search by normalizing the multiplier μ. The impact of the normalization on the convergence speed was studied by Hanafi (1993), and the author gives a unified proof of the finite convergence of the extended dichotomic procedure. Therefore, by exchanging the two constraints if necessary, a dichotomic-type search can be performed on a compact interval, using the information provided by the surrogate relaxation solution at each iteration. Extensive numerical experiments with randomly generated instances and up to 1,000 variables showed that this approach was superior to both Dyer's quasi-subgradient algorithm (Dyer, 1980) and simple dichotomic search, both in accuracy and computing time. In this paper, we consider the normalization L_1 for the multiplier μ, i.e. $\mu \in \{(\mu_1, \mu_2) : \mu_1 + \mu_2 = 1, \mu_1, \mu_2 \geq 0\}$. Using this normalization, the surrogate dual S can be solved in a one-dimensional search, i.e.

$$(S) \rightarrow \min\{v(S(\alpha)) : \alpha \in [0, 1]\},$$

where the surrogate relaxation is

$$S(\alpha) \rightarrow \max\{\alpha c^1 x + (1 - \alpha) c^2 x : ax \leq b, x \in \{0, 1\}^n\}.$$

In the following, we will consider $\mathrm{LP}(S) = \min\{v(\mathrm{LP}(S(\alpha))) : \alpha \in [0, 1]\}$, where $\mathrm{LP}(S(\alpha))$ is the LP-relaxation of $S(\alpha)$. The LP-relaxation $\mathrm{LP}(S(\alpha))$ can be easily solved (see Dantzig, 1957) by sorting the items so that $\frac{c_j^\alpha}{a_j} \geq \frac{c_{j+1}^\alpha}{a_{j+1}}$ for $j = 1, \ldots, n-1$, with $c_j^\alpha = \alpha c_j^1 + (1 - \alpha) c_j^2$, and determining

the critical item s^α defined by $s^\alpha = \min\{i : \sum_{j=1}^{i} a_j > b\}$. An optimal solution of LP($S(\alpha)$) denoted as $\overline{x}^\alpha$ is given by

$$\overline{x}_j^\alpha = \begin{cases} 1 & \text{for} \quad j = 1, \ldots, s^\alpha - 1 \\ \dfrac{b - \sum_{i=1}^{s^\alpha - 1} a_i}{a_{s^\alpha}} & \text{for} \quad j = s^\alpha \\ 0 & \text{for} \quad j = s^\alpha + 1, \ldots, n. \end{cases}$$

The solution $\overline{x}^\alpha$ can also be computed in $O(n)$ time, without sorting the items, by determining the critical item through median-finding techniques (see Balas and Zemel, 1980).

The binary search method to solve problem LP(S) is described in Algorithm 1. We use this procedure to compute an initial feasible solution for MNK. In steps 2 and 3 of Algorithm 1, an integer feasible solution $\tilde{x}^\alpha$ is generated from the optimal solution $\overline{x}^\alpha$ of the associated LP-relaxation LP($S(\alpha)$ by setting the variable corresponding to the critical item s^α to 0, i.e. $\tilde{x}^\alpha_{s^\alpha} = 0$.

Algorithm 1. Dichotomic Surrogate Heuristic

Input: An instance P of the problem
Step 1: Let $\alpha_{\mathrm{L}} = 0$ and $\alpha_{\mathrm{R}} = 1$, solve LP(S(α_{R})) and LP(S(α_{L})) to obtain $\overline{x}^{\alpha_L}$ and $\overline{x}^{\alpha_R}$, respectively.
Step 2: Set $\overline{v} = \min\{v(\mathrm{LP}(\mathrm{S}(\alpha_{\mathrm{R}}))), v(\mathrm{LP}(\mathrm{S}(\alpha_{\mathrm{L}})))\}$. Set $\tilde{x}^{\alpha_L} = \overline{x}^{\alpha_L}$, $\tilde{x}^{\alpha_L}_{s^{\alpha_L}} = 0$ and $\tilde{x}^{\alpha_R} = \overline{x}^{\alpha_R}$, $\tilde{x}^{\alpha_R}_{s^{\alpha_R}} = 0$.
Let $\underline{v} = \max\{\min\{c^1\tilde{x}^{\alpha_L}, c^2\tilde{x}^{\alpha_L}\}, \min\{c^1\tilde{x}^{\alpha_R}, c^2\tilde{x}^{\alpha_R}\}\}$.
Step 3: Let $\alpha = (\alpha_{\mathrm{L}} + \alpha_{\mathrm{R}})/2$ and solve LP($S(\alpha)$) to obtain $\overline{x}^\alpha$. Set $\tilde{x}^\alpha = \overline{x}^\alpha$, $\tilde{x}^\alpha_{s^\alpha} = 0$ and $\underline{v} = \max\{\min\{c^1\tilde{x}^\alpha, c^2\tilde{x}^\alpha\}, \underline{v}\}$.
Step 4: If $a\overline{x}^\alpha \leq b$, or $\alpha_{\mathrm{R}} - \alpha_{\mathrm{L}} \leq \varepsilon$ then stop.
Step 5: If $c^1\overline{x}^\alpha > c^2\overline{x}^\alpha$ then set $\alpha_{\mathrm{R}} = \alpha$, else set $\alpha_{\mathrm{L}} = \alpha$. Go to Step 2.
Output : $\underline{v}$ and $\overline{v}$.

We can also reinforce the lower bound found by applying a local-search process at each iteration of Algorithm 1 around the current feasible solution $\tilde{x}^\alpha$. Note that in our implementation we use the value $\varepsilon = 0.01$ in Algorithm 1 as a stopping condition.

4. Iterative linear programming based heuristic

Solving hard combinatorial optimization problems remains a challenge, in particular when the size of the instances grows. Several recent publications are devoted to hybrid methods, in which projection techniques are used to set some variables at particular values, to generate reduced problems that are easier to solve at optimality. Different fixation processes can be identified, such as the use of adaptive memory as in Glover (2005) or the use of external branching framework as in Fischetti and Lodi (2003). Generally speaking, these methods at each iteration build a neighborhood, which is explored optimally or heuristically. Additional information is also introduced to guide the search process. In this paper, we propose the use of another hybrid method, based also on a temporary fixation of variables, and the construction of reduced problems that are easier to solve at optimality.

This approach can be considered as a general method to solve 0–1 mixed integer programming problems.

Recently, Hanafi and Wilbaut (2011) proposed a general method that can be used for solving 0–1 mixed integer programming problems, and they call it the iterative relaxation-based heuristic (IRH). This approach can be considered as an extension and a generalization of the previous work for solving pure 0–1 integer programming problems (Soyster et al., 1978). The main idea of the method is to generate a sequence of nonincreasing upper bounds (in the case of a maximization problem), and a sequence of nondecreasing lower bounds, to converge to an optimal solution of the problem. This can be done with the addition of *pseudo-cuts* to the problem, to strengthen the search and avoid the blocking of the search process. Generally speaking, an instantiation of the IRH follows three main steps: (a) solve one or more relaxations of the current problem P to generate one or more pseudo-cuts, and update the best upper bound of the problem; (b) solve one or more reduced problems induced by the optimal solutions of the previous relaxations to obtain one or more feasible solutions of the initial problem, and update the best lower bound of the problem; (c) if the stopping criterion is satisfied then return the best lower bound and the best upper bound. Otherwise, add the pseudo-cuts generated in the first step to P and repeat the process.

From this general description, we can easily observe that several versions can be considered, according to the relaxation(s) we use, the way we solve the reduced problem(s) (heuristically or at optimality), the stopping condition we define, etc.

In this paper, we use the Linear Programming (LP) relaxation, and denote as the iterative LP based heuristic (ILPH) the method we propose. To ensure an easier comprehension of the paper, we briefly recall the main principles of the ILPH. Let P be a 0–1 integer programming problem formulation as proposed in Section 2. From now, we assume that the MNK is formulated as the following 0–1 MIP:

$$(P) \rightarrow \begin{cases} \max cx \\ \quad s.t. \\ \quad\quad Ax \leq b \\ \quad\quad x_j \in \{0,1\}, \quad j \in N. \end{cases}$$

First, we introduce the notion of a reduced problem. Given an arbitrary binary solution x° and a subset of variables $J \subseteq N$, the problem reduced from the original problem P and associated with x° and J is derived from P by setting variables with indices in J at values of x°. This problem is referred to as $P(x^\circ, J)$. A major element in the ILPH is the use of pseudo-cuts. A pseudo-cut is a linear inequality that excludes certain points from being feasible as solutions to the input problem. More precisely, let x and y be two binary solutions of problem P. The distance between x and y is defined as $\delta(x, y) = \sum_{j \in N} |x_j - y_j|$. From this definition, it is easy to define the partial distance between x and y relative to a given subset J of N as $\delta(J, x, y) = \sum_{j \in J} |x_j - y_j|$. Let $\overline{x}$ be an optimal solution of the LP relaxation of P, and let $B(\overline{x}) = \{j \in N | \overline{x}_j \in \{0, 1\}\}$ (i.e. the set of indices of variables in $\overline{x}$ with integer values). Then, for $J \subseteq B(\overline{x}) : \delta(J, x, \overline{x}) = \sum_{j \in J} x_j(1 - \overline{x}_j) + \overline{x}_j(1 - x_j)$. With these different ingredients, we are able to give an algorithmic description of a standard implementation of the ILPH, as in Algorithm 2.

Algorithm 2. An instantiation of the ILPH.

Input: An instance P of the problem.

Output: An optimal solution x^* of P

$Q = P$; $stop =$ false;
Find an initial solution x^* of P;
$\overline{v} = +\infty$; $\underline{v} = cx^*$;
while $stop =$ false do
  Let $\overline{x}$ be an optimal solution of LP(Q);
  Update the best upper bound value $\overline{v}$ if $c\,\overline{x} < \overline{v}$;
  if $\overline{x} \in \{0, 1\}^n$ then update $\underline{v}$ and x^* and return x^*;
  Let x° be an optimal solution of problem $P(x^\circ, B(\overline{x}))$;
  Update the best lower bound value _and vx^* if $cx^\circ > cx^*$;
  Apply a fixation technique to set some variables at their optimal values;
  Add a pseudo-cut to $Q : Q = (Q|\delta(B(\overline{x}), x, \overline{x}) \geq 1)$;
  if$\lfloor \overline{v} - \underline{v} \rfloor < 1$ then $stop =$ true;
end while
return x*;

According to Algorithm 2, the ILPH repeats the solution of the LP-relaxation of the current problem Q, then the exact solving of the reduced problem associated with an optimal solution of this relaxation. After that, the pseudo-cut $\delta(B(\overline{x}), x, \overline{x}) \geq 1$ is added to the current problem to discard solution $\overline{x}$ from the search space. It is easy to prove that the addition of this pseudo-cut guarantees that reduced problems already explored are not revisited. In addition, the use of this pseudo-cut strengthens the problem, and helps the method to construct a nonincreasing sequence of upper bounds. Then, it can be proved that the ILPH converges to an optimal solution of the input problem (under some conditions) in a finite number of iterations (for more details, see Hanafi and Wilbaut, 2011). That is why condition $\lfloor \overline{v} - \underline{v} \rfloor < 1$ is used in Algorithm 2. The convergence of the algorithm can be very time consuming, in particular if the gap between the LP-value of the initial problem and an optimal solution is large. In that case, it is possible to use the ILPH as a heuristic, by fixing the number of iterations to be performed in the loop, or by fixing a maximum running time for example. Recall that for MNK we use the search heuristic described in Section 3 to obtain an initial feasible solution x^*.

In this paper, we propose to enhance the behavior of the ILPH for the MNK, by using a fixation technique. Fixation techniques are frequently used to solve hard combinatorial optimization problems (Savelsberg, 1994). This is particularly the case when solving KPs (Fayard and Plateau, 1977). In this paper, we propose the use of the well-known reduced cost constraint, in particular to exploit the improvement of the upper bound and the lower bound during the search process.

In the family of KPs, the reduced cost constraints have been used intensively with success for solving the 0–1 multidimensional KP (Boussier et al., 2010; Vimont et al., 2008). To introduce this technique, we consider the problem formulation P introduced previously in which we suppose we have n binary variables and m constraints. The associated LP relaxation in the standard form, obtained by introducing s—a vector of m slack variables, is defined by: $\max\{cx|Ax + Es = b, x_j \leq 1, j \in N, x, s \geq 0\}$. Let $\overline{v}$ be the optimal value of the LP-relaxation,

$B^+(x) = \{j \in N | x_j = 1\}$, $B^-(x) = \{j \in N | x_j = 0\}$, $M = \{1, 2, \ldots, m\}$ and E the identity matrix. Then, the LP relaxation can be rewritten in an optimal basis as

$$\begin{cases} \max \bar{v} + \sum_{j \in B^-(\bar{x})} \bar{c}_j x_j - \sum_{j \in B^+(\bar{x})} \bar{c}_j (1 - x_j) + \sum_{j \notin N} \bar{c}_j x_j + \sum_{i \in M} \bar{d} s \\ \quad s.t. \\ \quad \bar{A}x + \bar{E}s = \bar{b} \\ \quad x_j \leq 1, \quad j \in N \\ \quad x, s \geq 0 \end{cases}$$

with $\bar{x}$ an optimal solution, $(\bar{c}, \bar{d})$ the vector of the reduced costs corresponding to variables (x, s) for the basic solution $(\bar{x}, \bar{s})$, and $\bar{A}$, $\bar{E}$, and $\bar{b}$ the values for the basis associated with $(\bar{x}, \bar{s})$. Then, if $\underline{v}$ is the lower bound value associated with the current best feasible solution found during the search, each better solution x must satisfy the following so-called reduced cost constraint: $\sum_{j \in B^-(\bar{x})} |\bar{c}_j| x_j + \sum_{j \in B^+(\bar{x})} |\bar{c}_j| (1 - x_j) \leq \bar{v} - \underline{v}$. It follows from this constraint that the simplest rule that attempts to set a variable $x_j (j \in N)$ to the complement of the LP-relaxation value of $\bar{x}_j$, is: if $|\bar{c}_j| > \bar{v} - \underline{v}$ then $x_j = \bar{x}_j$.

It is easy to integrate the use of the reduced cost constraints in the ILPH for the MNK, taking into account a given MIP formulation. At each iteration, after solving the LP-relaxation of the current problem, we try to set some new variables by applying the previous test. In addition, the associated reduced cost constraints are added, if necessary. In the next section, we present the results obtained with the ILPH, when using different models for the MNK presented in the previous section.

5. Computational experiments

5.1. Environment of experiments

In this section, we provide a summary of experiments performed to evaluate both our approaches and the different MIP formulations presented in Section 2 for MNK. We used a set of randomly generated instances, obtained by applying the same method as in Taniguchi et al. (2009), which can be described as follows. For a given number of variables n, weight a_j for item j is randomly and uniformly distributed over [1, 1,000]. Knapsack capacity b is set to $500 \times n \times \rho$, where ρ is either 0.25, 0.5, or 0.75. As for other KPs, three kinds of correlation degree are used.

(1) *Uncorrelated*: In this case c_j^1 and c_j^2 associated with item j are distributed independently and uniformly over [1, 1,000].
(2) *Weakly correlated*: In this case, weight a_j of item j is considered to generate c_j^1 and c_j^2, which are distributed independently and uniformly over $[a_j, a_j + 200]$.
(3) *Strongly correlated*: In this case, c_j^1 is set to $a_j + 100$, and c_j^2 is distributed independently and uniformly over $[a_j, a_j + 200]$.

We note in what follows σ is used to refer to the correlation degree of the instance. All our algorithms are coded in C++ language, compiled with g++ compiler and option –O2. We use a

Table 1
Parameters used in randomly generated instances.

σ	n	ρ	σ	n	ρ	σ	n	ρ
Uncorrelated	5,000	0.25	Weakly correlated	5,000	0.25	Strongly correlated	500	0.25
	7,000	0.5		7,000	0.5		1,000	0.5
	10,000	0.75		10,000	0.75		4,000	0.75
	13,000			13,000			5,000	
	15,000			15,000			7,000	
	18,000			18,000			10,000	
	20,000			20,000				

Pentium IV computer with 3.4 GHz processor and 4 GB RAM to run the algorithms. To compare models of MNK and compute upper bounds, we use the general purpose MIP solver CPLEX 11.2. Preliminary experiments demonstrated that CPLEX is able to optimally solve instances with a high number of items in a reasonable CPU time (less than 10 minutes). According to this observation, we set the values of parameters (σ, n, ρ) as in Table 1, and we generate 15 instances for each value.

We have a total of 315 uncorrelated instances, 315 weakly correlated instances, and 270 strongly correlated instances. All the instances are available upon request. When solving MNK instances, we use the total limit time of 10 minutes for all the algorithms. In addition, we set parameter CPX_PARAM_EPGAP of CPLEX to 0.0 due to numerical sensitivity when the size of instances grows and to avoid an anticipated stop of the search by the solver without an optimal solution for the instance. Indeed, some experiments with value 1×10^{-06} (default value is 1×10^{-04}) provide possible termination for CPLEX with status 102, corresponding to a solution considered as an optimal solution with a tolerance, although this solution was not an optimal solution.

5.2. Comparison of models for MNK

In this section, we compare the potential of models presented in Section 2 for MNK. Thus, we apply directly the CPLEX MIP solver to solve all the instances using the different models. The aim of this phase is to identify a *better* model for MNK when using an exact approach. The first data we can consider to compare the models are the associated LP relaxation. In fact, we observe that models MIP, MIP1, and MIP2 show the same LP values for all the instances. When considering the models BKP1 and BKP2, the conclusions are the same in most of the cases. For a few instances, one of the two models has an associated LP-value less than the other models. However, it happens when the associated optimal value does not correspond to an optimal value of the initial MNK instance (due to the max in the formulation between the two models, $v(\text{MNK}) = \max\{v(\text{BKP1}), v(\text{BKP2})\}$).

The situation is different when we consider the linearization (L) of the quadratic model. In that case, the optimal LP values are larger than the optimal LP values for other models (with the values of parameters reported in Table 1). In addition, the CPU time needed to solve the relaxation is also largely more important. In particular, it is not possible for CPLEX MIP solver to solve the LP relaxation in 600 seconds for some large instances. In that case, the status returned by CPLEX is 11, which means that a feasible or an infeasible solution is obtained before the time limit is reached.

That can be explained by the size and the structure of the problem. We have in model (L) $2n + 1$ variables, whereas other models imply n or $n + 1$ variables, and the number of constraints is also clearly more important. In the case of linearization (LQ), we have, in addition, to deal with an additional parameter M. According to these observations, we do not consider the linearization LQ in the following, in particular due to the size of our instances (CPLEX is not able to solve optimally any of the instances).

In Tables 2–4, we provide elements to compare the efficiency of the different models over uncorrelated, weakly correlated, and strongly correlated instances, respectively. In these tables, columns n and ρ correspond, respectively, to the number of variables and the correlation of the instance. Then, column *gap* provides the average gap values obtained by CPLEX when solving the instances (i.e. the difference between the initial upper bound and the final lower bound). We observe that the average gap obtained with models MIP1, MIP2, MIP, or BKP is almost the same. Thus, to avoid an overload of the presentation, we only report one value for all these models. We also use the notation BKP to refer to the use of both models BKP1 and BKP2. Then, we report for each model the number of instances solved exactly (*#opt*) over the 15 instances we considered, and the average CPU time (in seconds) needed to solve one instance. From these three tables, we can observe the following:

Table 2
Average results obtained over uncorrelated instances.

			MIP1		MIP2		MIP		BKP	
n	ρ	gap	#opt	CPU	#opt	CPU	#opt	CPU	#opt	CPU
5,000	0.25	7.74	15	20.46	15	17.75	15	26.74	15	22.21
7,000		6.07	15	43.03	15	41.37	15	54.75	15	43.76
10,000		4.38	15	103.51	15	78.28	15	135.83	15	59.60
13,000		4.00	14	186.60	15	151.71	15	193.59	15	70.74
15,000		3.08	15	141.06	15	125.92	15	192.09	15	67.88
18,000		2.58	15	273.75	15	245.76	11	405.60	15	68.02
20,000		2.58	12	415.82	10	384.37	7	485.33	15	72.77
5,000	0.5	6.71	15	25.77	15	22.97	15	37.61	15	29.30
7,000		5.07	15	86.20	15	83.57	15	114.12	15	51.97
10,000		3.85	14	262.19	13	247.78	13	334.75	15	105.38
13,000		3.32	13	310.39	13	311.64	10	413.14	15	85.86
15,000		3.04	13	354.22	10	351.59	10	441.57	15	92.72
18,000		2.69	1	594.44	4	536.70	1	594.25	15	116.82
20,000		2.17	7	564.81	11	401.35	7	555.63	15	114.69
5,000	0.75	7.84	15	19.24	15	15.61	15	22.02	15	16.35
7,000		5.65	15	31.66	15	30.61	15	42.93	15	26.66
10,000		3.96	15	87.73	15	55.86	15	122.21	15	34.23
13,000		3.58	15	135.88	15	143.80	14	194.72	15	52.03
15,000		2.73	14	203.10	15	141.38	14	314.15	15	61.55
18,000		2.70	14	221.97	14	248.95	9	505.75	15	46.45
20,000		2.60	11	346.70	12	364.26	3	585.08	15	67.50

Table 3
Average results obtained over weakly correlated instances.

			MIP1		MIP2		MIP		BKP	
n	ρ	gap	#opt	CPU	#opt	CPU	#opt	CPU	#opt	CPU
5,000	0.25	2.07	15	4.26	15	3.81	15	174.39	15	10.99
7,000		1.63	15	5.91	15	4.83	11	424.40	15	11.94
10,000		1.26	15	4.95	15	3.47	0	600.08	15	9.64
13,000		1.19	15	5.64	15	4.97	0	600.10	15	12.25
15,000		0.97	15	13.38	15	6.28	6	432.22	15	15.30
18,000		0.97	0	600.17	0	600.11	0	600.11	15	9.97
20,000		0.82	0	600.07	0	600.11	0	600.08	15	13.05
5,000	0.5	1.50	15	8.98	15	4.89	14	171.09	15	12.31
7,000		1.17	15	7.61	15	5.28	13	254.56	15	15.76
10,000		1.05	15	11.03	15	7.38	6	424.48	15	20.39
13,000		0.83	15	12.37	15	6.71	7	444.98	15	13.22
15,000		0.73	15	14.11	15	8.09	6	425.98	15	15.48
18,000		0.94	0	600.12	0	600.12	0	600.07	15	18.61
20,000		0.82	0	600.07	0	600.09	0	600.08	15	19.10
5,000	0.75	1.71	15	6.65	15	5.59	14	102.81	15	11.98
7,000		1.45	15	11.42	15	5.96	12	258.44	15	14.06
10,000		1.02	15	8.82	15	4.01	11	266.98	15	14.91
13,000		0.82	15	11.06	15	6.38	10	388.95	15	15.20
15,000		0.94	15	11.58	15	5.64	0	600.05	15	16.88
18,000		0.89	2	558.85	2	541.42	0	600.07	15	18.04
20,000		0.76	0	600.07	1	587.55	0	600.07	15	20.87

Table 4
Average results obtained over strongly correlated instances.

			MIP1		MIP2		MIP		BKP	
n	ρ	gap	#opt	CPU	#opt	CPU	#opt	CPU	#opt	CPU
500	0.25	58.26	0	600.15	0	600.14	0	600.23	0	1200.25
1,000		49.21	0	600.09	0	600.06	0	600.09	0	1200.21
4,000		51.20	0	600.02	1	560.06	1	560.05	0	1200.13
5,000		34.77	0	600.09	0	600.04	1	560.06	0	1200.15
7,000		42.77	0	600.05	1	560.04	0	600.05	0	1200.10
10,000		41.34	0	600.03	1	560.06	1	560.06	0	1200.11
500	0.5	53.43	0	600.09	1	560.14	1	560.09	2	1000.35
1,000		50.19	0	600.14	0	600.12	0	600.04	0	1160.21
4,000		45.65	0	600.04	1	560.03	0	600.04	0	1200.06
5,000		37.05	0	600.09	1	560.09	1	560.03	0	1200.08
7,000		54.61	5	400.33	5	400.17	5	400.06	1	965.43
10,000		54.75	2	520.75	2	520.22	2	520.05	0	1120.77
500	0.75	65.16	1	560.15	2	520.07	2	520.25	1	1080.22
1,000		60.53	3	509.09	4	440.12	6	360.04	2	1047.62
4,000		47.59	0	600.03	2	520.09	1	560.06	0	1160.08
5,000		46.87	2	520.06	2	520.10	2	520.03	1	1080.15
7,000		43.49	0	600.05	0	600.03	0	600.06	0	1200.12
10,000		47.39	0	600.05	0	600.05	0	600.04	0	1200.10

(1) When comparing models MIP1, MIP2, MIP, and BKP, we can observe a worse performance for model MIP for uncorrelated instances (in particular for $n \geq 18{,}000$, and for weakly correlated instances when $n \geq 10{,}000$).
(2) The number of instances solved exactly decreases in general with the size of the instances and the correlation degree for all the models. However, we can observe that using models BKP shows very impressive results for uncorrelated and weakly correlated instances, since CPLEX MIP solver is able to prove the optimality of the solution obtained in all the cases. In general, instances with $\rho = 0.5$ are the most difficult to solve with CPLEX MIP solver.
(3) The average gap is very small for uncorrelated and weakly correlated instances. When considering strongly correlated instances, a more important average value is observed.

According to previous observations 1 and 2, we can conclude that CPLEX MIP solver is able to find an optimal solution for almost all the uncorrelated and weakly correlated instances with models MIP1, MIP2, MIP, and BKP. The main difference between these models is the number of times the solution can be proved to be optimal within the imposed running time.

Columns #*opt* demonstrate the following:

- Globally, model MIP is dominated by models with two constraints. In particular, we can observe that CPLEX is able to obtain an optimal solution with this model for 249 uncorrelated instances, whereas it obtains 278, 282, and 315 optimal solutions with model MIP1, MIP2, and BKP, respectively. The difference is more important for weakly correlated instances for which the number of optimal solutions found with model MIP decreases to 125, against 227, 228, and 315 with model MIP1, MIP2, and BKP, respectively. However, we can also observe a better behavior for strongly correlated instances with 23 optimal solutions, against 13, 23, and 7 for MIP1, MIP2, and BKP, respectively.
- Solving model MIP1 or model MIP2 shows on average equivalent results. In fact, model MIP1 obtains better results in a few cases (for instance when considering uncorrelated instances with $\rho = 0.25$), but model MIP2 obtains on average slightly better results (in particular for strongly correlated instances with 23 optimal solutions whereas model MIP1 obtains 13 optimal solutions). Globally, model MIP2 obtains 533 optimal solutions against 518 for model MIP1.
- Finally, using model BKP seems to be an interesting way for solving MNK instances, in particular for uncorrelated and weakly correlated instances. However, strongly correlated instances are clearly difficult to solve with this model. A simple explanation comes from the fact that we have to solve exactly two models to find the optimal value of the problem. In the case of strongly correlated instances, CPLEX is not able, in almost all the cases, to solve the two models at optimality (even if we use the time limit of 10 minutes to solve each of the two models. That explains why the average running time for strongly correlated instances with model BKP can be more than 600 seconds).

Experiments presented in this section seem to demonstrate that no model totally dominates the others. In fact, model BKP show better average results for uncorrelated and weakly correlated instances. For strongly correlated instances, CPLEX solves more instances when using models MIP2 and MIP. If we consider only the number of instances solved at optimality, model BKP is ranked first with 637 instances out of 900, followed by model MIP2 with 533, model MIP1 with 518, and model MIP with 397.

5.3. Using ILPH to improve models' efficiency

In this section, we propose the use of the ILPH as a preprocessing phase to obtain good lower and upper bounds, and to set variables at their optimal values, and then to solve the remaining problem with CPLEX MIP solver, taking into account information generated by the ILPH. In that case, we apply the algorithm with a limited number of iterations:

- To find an initial feasible solution and improve it in a short number of iterations. The lower bound associated with this solution can be used to cut-off the tree of the search later by CPLEX.
- To improve the upper bound on the optimal value because of the addition of the pseudo-cuts into the problem. With this process we wish to reduce the gap between the upper bound and the lower bound quickly.
- To set a large part of the variables to their optimal value by using reduced cost constraints. If the fixation process is sufficiently efficient, we can expect the final reduced problem easier to solve with an exact method.

According to preliminary experiments, we set to 20 the number of iterations performed with the ILPH. As we see in the following, it is sufficient to obtain an interesting performance.

In Tables 5–7, we give average results to evaluate the efficiency of this approach. We apply the ILPH based on models MIP1, MIP2, MIP, and BKP, and denoted as ILPH-MIP1, ILPH-MIP2, ILPH-MIP, and ILPH-BKP, respectively. In all the cases, we provide the relative error associated with the initial solution provided by the heuristic. If LP(P) denotes the objective value of the LP-relaxation of problem P and if v denotes the objective value of the initial solution, the relative error is computed as $((\mathrm{LP(P)} - v)/v) \times 100$. These values are reported in column $rerror^0$. Note that this value is the same for all the ILPH-based algorithms, since the initial surrogate heuristic used is the same regardless of the model used. Then, for each variant, we report the average relative error obtained at the end of the ILPH in column *rerror*. This value depends on the final lower bound provided by the ILPH (after 20 iterations) and the final upper bound refined with pseudo-cuts added iteratively. Columns *fix* provide the average percentage of variables set at their optimal value at the end of the ILPH. Then, columns Δ(opt) give the difference between the number of optimal solutions found by the method "ILPH + CPLEX" and the number of optimal solutions found by CPLEX alone. Thus, a positive value in column Δ(opt) means that using ILPH as a preprocessing phase increases the number of instances solved exactly. In the same way, values reported in columns Δ(cpu) correspond to the difference between the average running time of the method "ILPH + CPLEX" and the average running time needed by CPLEX alone. A negative value in column Δ(cpu) means that using the ILPH as a preprocessing phase decreases the average running time to solve MNK. Note that we do not use columns *rerror* and *fix* when using BKP. Indeed, in that case we have to solve two different problems, so the average values do not have exactly the same meaning.

According to values reported in Tables 5–7, we can observe the following:

- The use of ILPH with models MIP1 and MIP2 is an efficient way to improve the behavior of CPLEX MIP solver with these two models. In particular, we can see that the number of optimal solutions found increases from 278, 227, and 13 when using CPLEX alone with model MIP1

Table 5
Average results with ILPH over uncorrelated instances.

			ILPH-MIP1				ILPH-MIP2				ILPH-MIP				ILPH-BKP	
n	ρ	rerror^0	Rerror	fix	Δ(opt)	Δ (cpu)	rerror	fix	Δ(opt)	Δ (cpu)	rerror	fix	Δ(opt)	Δ (cpu)	Δ (opt)	Δ (cpu)
5,000	0.25	0.0073	0.0020	95.01	0	−12.89	0.0020	95.01	0	−10.13	0.0020	94.85	0	−13.37	0	−3.08
7,000		0.0072	0.0021	92.46	0	−29.87	0.0021	92.46	0	−29.24	0.0021	92.46	0	−30.45	0	−13.54
10,000		0.0058	0.0012	93.90	0	−80.95	0.0012	93.90	0	−63.71	0.0012	93.81	0	−84.36	0	−18.29
13,000		0.0044	0.0009	94.60	1	−158.95	0.0009	94.60	0	−126.73	0.0009	94.01	0	−108.86	0	−5.81
15,000		0.0052	0.0009	93.24	0	−117.75	0.0009	93.24	0	−94.13	0.0008	93.65	0	−91.57	0	−9.54
18,000		0.0043	0.0006	94.79	0	−230.42	0.0006	94.79	0	−199.56	0.0006	94.45	4	−259.59	0	−16.96
20,000		0.0034	0.0006	94.26	2	−256.19	0.0006	94.26	5	−256.83	0.0006	94.26	6	−218.84	0	−1.42
5,000	0.5	0.0049	0.0015	93.37	0	−15.33	0.0015	93.37	0	−14.85	0.0014	93.60	0	−21.94	0	−7.81
7,000		0.0090	0.0012	92.24	0	−68.42	0.0012	92.24	0	−63.64	0.0013	91.92	0	−71.09	0	−10.00
10,000		0.0148	0.0007	93.60	1	−234.27	0.0007	93.60	2	−216.98	0.0007	94.00	2	−212.86	0	−47.74
13,000		0.0019	0.0005	94.07	2	−240.52	0.0005	94.07	2	−256.60	0.0005	94.00	4	−206.39	0	−21.80
15,000		0.0044	0.0006	92.68	1	−251.48	0.0006	92.68	4	−261.26	0.0006	92.76	4	−186.89	0	−9.87
18,000		0.0068	0.0005	92.03	11	−432.10	0.0005	92.03	8	−340.50	0.0005	92.10	5	−90.73	0	5.33
20,000		0.0041	0.0003	94.05	8	−502.74	0.0003	94.05	4	−359.61	0.0004	93.70	4	−253.08	0	−41.35
5,000	0.75	0.0102	0.0012	94.06	0	−11.27	0.0012	94.06	0	−9.80	0.0011	94.56	0	−11.12	0	−2.33
7,000		0.0035	0.0010	93.69	0	−19.92	0.0010	93.69	0	−19.29	0.0010	93.37	0	−23.34	0	−1.49
10,000		0.0082	0.0006	94.47	0	−62.75	0.0006	94.47	0	−33.66	0.0006	94.63	0	−87.05	0	−2.40
13,000		0.0047	0.0004	95.53	0	−89.64	0.0004	95.53	0	−122.60	0.0004	95.33	1	−99.52	0	−5.54
15,000		0.0054	0.0004	94.46	1	−170.27	0.0004	94.46	0	−92.65	0.0004	94.57	1	−216.93	0	−23.93
18,000		0.0043	0.0003	95.68	1	−146.85	0.0003	95.68	1	−212.90	0.0003	95.53	5	−320.33	0	7.20
20,000		0.0048	0.0003	94.91	3	−262.44	0.0003	94.91	2	−273.29	0.0003	95.10	9	−247.55	0	−17.34

Table 6
Average results with ILPH over weakly correlated instances.

			ILPH-MIP1				ILPH-MIP2				ILPH-MIP				ILPH-BKP	
n	ρ	rerror^0	rerror	fix	Δ(opt)	Δ (cpu)	rerror	fix	Δ(opt)	Δ (cpu)	rerror	fix	Δ(opt)	Δ (cpu)	Δ(opt)	Δ (cpu)
5,000	0.25	0.0041	0.0019	84.63	0	0.17	0.0019	84.63	0	21.37	0.0019	84.66	0	−71.73	0	−1.11
7,000		0.0036	0.0010	88.95	0	20.07	0.0010	88.95	0	38.92	0.0009	90.18	2	−201.01	0	−0.61
10,000		0.0033	0.0008	86.81	−4	189.41	0.0008	86.81	0	17.43	0.0008	87.75	4	−125.06	0	1.62
13,000		0.0025	0.0004	91.21	−3	153.61	0.0004	91.21	−5	198.85	0.0004	91.56	1	−36.68	0	−1.80
15,000		0.0025	0.0005	89.11	−2	73.74	0.0005	89.11	−1	45.32	0.0005	88.42	1	−75.28	0	3.88
18,000		0.0058	0.0004	87.70	11	−435.11	0.0004	87.70	13	−515.07	0.0004	89.28	0	2.95	0	4.12
20,000		0.0018	0.0004	86.21	12	−438.06	0.0004	86.21	12	−473.37	0.0004	86.21	1	−15.18	0	7.75
5,000	0.5	0.0022	0.0011	80.10	0	27.13	0.0011	80.10	0	9.37	0.0011	80.33	0	−67.00	0	−0.94
7,000		0.0021	0.0008	79.04	−1	39.85	0.0008	79.04	−1	55.83	0.0008	78.94	0	−117.93	0	−5.04
10,000		0.0017	0.0004	86.80	−3	103.71	0.0004	86.80	−1	−0.82	0.0004	84.98	1	−53.85	0	−1.52
13,000		0.0016	0.0005	77.99	−2	94.45	0.0005	77.99	−3	143.12	0.0004	79.37	1	−73.83	0	4.60
15,000		0.0014	0.0003	83.83	−1	31.70	0.0003	83.83	−1	40.69	0.0003	82.72	1	−68.01	0	3.38
18,000		0.0017	0.0004	75.38	12	−474.49	0.0004	75.38	13	−514.06	0.0004	73.18	0	1.41	0	5.71
20,000		0.0029	0.0002	85.98	13	−511.42	0.0002	85.98	12	−473.29	0.0002	84.99	0	2.27	0	2.69
5,000	0.75	0.0022	0.0009	81.03	0	−2.89	0.0009	81.03	0	0.74	0.0008	81.60	1	−59.04	0	1.25
7,000		0.0020	0.0004	86.66	−1	38.79	0.0004	86.66	−1	64.38	0.0004	86.66	1	−107.18	0	−1.65
10,000		0.0012	0.0004	83.65	−1	10.07	0.0004	83.65	−1	6.81	0.0004	83.99	1	−68.20	0	1.29
13,000		0.0015	0.0002	87.38	−1	63.87	0.0002	87.38	0	49.40	0.0003	86.41	1	−120.72	0	5.29
15,000		0.0016	0.0003	80.11	−2	74.94	0.0003	80.11	−1	58.99	0.0003	80.97	8	−185.00	0	4.42
18,000		0.0021	0.0002	83.97	12	−510.10	0.0002	83.97	13	−534.45	0.0002	82.45	2	−50.87	0	4.71
20,000		0.0011	0.0002	86.29	11	−394.02	0.0002	86.29	11	−461.05	0.0002	85.47	2	−24.16	0	−0.91

Table 7
Average results with ILPH over strongly correlated instances.

			ILPH-MIP1				ILPH-MIP2				ILPH-MIP				ILPH-BKP	
n	ρ	rerror^0	rerror	fix	Δ(opt)	Δ (cpu)	rerror	fix	Δ(opt)	Δ (cpu)	rerror	fix	Δ(opt)	Δ (cpu)	Δ(opt)	Δ (cpu)
500	0.25	0.0724	0.0643	41.17	0	4.88	0.0643	41.17	0	5.33	0.0643	41.21	0	5.74	0	10.27
1,000		0.0357	0.0300	48.19	0	4.54	0.0300	48.19	0	5.12	0.0300	48.19	1	−35.27	0	8.74
4,000		0.0084	0.0084	42.12	0	2.06	0.0084	42.12	0	2.34	0.0084	42.12	−1	42.24	0	4.02
5,000		0.0104	0.0082	53.81	0	2.73	0.0082	53.81	1	−37.42	0.0082	53.81	−1	42.95	0	−34.75
7,000		0.0068	0.0053	37.84	1	−38.60	0.0053	37.84	−1	41.68	0.0054	37.78	0	1.66	0	−37.18
10,000		0.0049	0.0047	38.12	0	0.91	0.0047	38.12	0	1.24	0.0046	38.12	0	0.94	0	2.00
500	0.5	0.0354	0.0328	36.23	2	−75.70	0.0328	36.23	2	−74.96	0.0329	36.07	1	−35.14	1	−31.59
1,000		0.0164	0.0157	39.21	1	−36.54	0.0184	31.59	0	4.45	0.0157	39.33	1	−35.48	1	−32.01
4,000		0.0053	0.0046	34.10	0	1.81	0.0046	34.10	−1	42.04	0.0046	34.10	0	1.76	0	4.17
5,000		0.0063	0.0051	41.38	0	2.27	0.0051	41.38	0	2.21	0.0051	41.35	−1	42.04	0	−36.16
7,000		0.0037	0.0037	25.51	0	0.91	0.0037	25.51	−3	120.72	0.0037	25.52	0	1.46	1	−43.23
10,000		0.0029	0.0028	27.83	0	0.22	0.0028	27.83	1	−39.62	0.0028	27.83	0	0.55	2	−119.33
500	0.75	0.0286	0.0282	31.00	0	4.26	0.0282	31.00	0	4.79	0.0282	31.00	−1	44.63	0	11.85
1,000		0.0139	0.0133	33.28	3	−146.79	0.0132	33.41	−2	83.91	0.0132	33.41	0	2.79	0	−121.01
4,000		0.0027	0.0026	47.30	1	−38.02	0.0026	47.30	−1	42.39	0.0026	47.28	1	−37.65	0	−35.74
5,000		0.0059	0.0034	40.30	0	1.43	0.0034	40.30	0	1.60	0.0034	40.30	−1	42.02	1	−77.12
7,000		0.0052	0.0038	30.88	0	1.28	0.0038	30.88	0	1.44	0.0038	30.88	0	1.37	0	2.55
10,000		0.0029	0.0021	37.02	0	0.69	0.0021	37.02	0	0.72	0.0021	37.02	0	0.68	0	1.36

to 309, 277, and 21 with the ILPH, for uncorrelated, weakly correlated, and strongly correlated instances, respectively. In the same way, the number of optimal solutions found increases from 282 and 228 when using CPLEX alone with model MIP2 to 310 and 287 with the ILPH, for uncorrelated and weakly correlated instances, respectively. However, ILPH-MIP2 obtains average worse results than CPLEX with model MIP2 for strongly correlated instances, with 19 optimal solutions found against 23. A consequence of these results is the decrease of the average running time needed to solve instances at optimality when using the ILPH, in particular for uncorrelated and large weakly correlated instances. This efficiency is clearly in correlation with the fixation process. Indeed, we can observe the very important values in columns *fix* for uncorrelated and weakly correlated instances. As for other KPs, the fixation process is less efficient for strongly correlated instances, even if it is in general greater than 30%. For strongly correlated instances, values in columns Δ(cpu) clearly depend on the number of instances solved at optimality. Finally, the average final gap values and the average fixations observed are very similar for both ILPH-MIP1 and ILPH-MIP2. Some differences occur for strongly correlated instances, for example when $n = 1{,}000$ and $\rho = 0.5$ or $n = 1{,}000$ and $\rho = 0.75$.

- Results obtained by ILPH-MIP are similar. In particular, the number of optimal solutions found increases significantly from 249 and 125 to 294 and 153 for uncorrelated and weakly correlated instances, respectively. In the same way, the average running time decreases significantly from 275 and 436 seconds to 139 and 365 seconds, respectively. However, as for ILPH-MIP2, results are similar for CPLEX and ILPH-MIP for strongly correlated instances, with 23 and 22 optimal solutions found, and 549 and 553 seconds on average, respectively. Finally, we can observe some differences in columns *fix* and *rerror* between ILPH-MIP and ILPH-MIP1 or 2. However, the differences are never significant, and occur more often for uncorrelated and weakly correlated instances (sometimes to the advantage of model MIP and sometimes to the advantage of model MIP1 or MIP2).
- It is interesting to observe the behavior of the ILPH when using model BKP. Indeed, previous results show that CPLEX MIP solver was able to solve at optimality all the uncorrelated and all the weakly correlated instances with this model (in 10 minutes). From the optimality point of view, a first conclusion is that using ILPH as a preprocessing allows also CPLEX to solve all these instances. However, from the running time point of view, the gain is not in the same order as for other models. Model BKP used directly in CPLEX needs 62 and 15 seconds on average for uncorrelated and weakly correlated instances, respectively, to be solved at optimality. With the ILPH, the average running time measured is 50 and 16 seconds, respectively. So, we can conclude equivalence for these instances. However, another positive conclusion can be extracted from results for strongly correlated instances. Indeed, in that case ILPH-BKP shows better results than CPLEX (whereas it was not the case for models MIP1 and MIP2). With model BKP, the algorithm is able to solve 13 strongly correlated instances against 7 when using CPLEX alone, and the average running time decreases from 1,145 seconds to 1,116 seconds.

To conclude this section, we can say that using the ILPH before solving the problem with CPLEX is an efficient approach for MNK. Indeed, average performances are increased for all the models considered in our experiments. Considering the number of instances solved at optimality, we have the following overall progressions:

- Model BKP: Increase from 637 to 643.
- Model MIP2: Increase from 533 to 616.
- Model MIP1: Increase from 518 to 607.
- Model MIP: Increase from 397 to 469.

Thus, globally the previous hierarchy is still valid, but more than 50% of the instances can be solved at optimality with the four models.

5.4. Using the ILPH as an exact method

As presented in Section 4, the ILPH can be used as an exact method under some conditions. In this section, we provide the results obtained when CPLEX MIP solver is used to solve all the reduced problems and all the LP-relaxations at optimality during the search, inside the ILPH. As MNK is a pure 0–1 integer programming, when condition $\lfloor \bar{v} - \underline{v} \rfloor < 1$ is satisfied, the solution associated to lower bound $\underline{v}$ is optimal for the input problem. To evaluate the use of the ILPH *alone*, we limit the total running time to 600 seconds, as in previous experiments. Due to space limitations, we only present the results obtained by the ILPH with model MIP2. We report the synthesis of the results in Tables 8–10. First, we consider uncorrelated and weakly correlated instances in Tables 8 and 9, respectively. In these tables, we provide the average percentage of variables set during the process in column *fix*, the average iteration associated with the best (or optimal) solution found by the ILPH in column *iter**, the associated average CPU time needed to reach this solution in column CPU*. Column #*opt* and CPU report the number of solutions proved to be optimal during the allowed CPU time, and the average running time of the algorithm, respectively. Finally, we provide results obtained by our implementation of the virtual pegging test algorithm in columns VPT: the average final relative error (column *rerror*, and computed as previously), the average percentage of variables set at their optimal value (column *fix*), the number of optimal solutions found (column #*opt*), and the average running time of the algorithm (column CPU).

Results obtained by the ILPH as an exact method are really encouraging, in particular for uncorrelated and weakly correlated instances. Indeed, the average CPU time needed to solve exactly the instances decreases as far as when CPLEX is used alone (see Tables 2–3). More precisely, using CPLEX directly with model MIP2 requires 190 and 172 seconds for uncorrelated and weakly correlated instances, respectively. The use of ILPH alone requires only 166 and 10 seconds for these instances. In addition, the number of optimal solutions found increases from 282 and 228 to 307 and 315 for uncorrelated and weakly correlated instances, respectively. The difference with the method combining the ILPH and CPLEX is less important (see Tables 5–6). However, we can observe that the ILPH is able to find all the optimal solutions for weakly correlated instances (whereas the method using ILPH and CPLEX solves 287 instances at optimality in that case). These results can be explained by two main dependent reasons: the initial gap between the LP-value and the optimal value is very small, in particular for weakly correlated instances. Then, the use in the ILPH of pseudo-cuts clearly helps to improve the value of the upper bound during the process, and to apply the optimality test. In addition, the fixation process during the search is really efficient for these instances, with more than 98% of variables set at their optimal values at the end of the algorithm in almost all the cases. Results obtained by the VPT presented in Tables 8 and 9 are clearly less

Table 8
Average results with ILPH and VPT over uncorrelated instances.

		ILPH-alone					VPT			
n	ρ	fix	iter*	CPU*	#opt	CPU	rerror	fix	#opt	CPU
5,000	0.25	99.21	127.53	23.30	15	84.62	0.0760	5.63	8	345.49
7,000		99.39	113.07	25.26	15	134.45	0.0791	1.81	4	476.53
10,000		99.55	106.33	32.47	15	163.37	0.0812	3.15	1	564.07
13,000		99.58	159.07	93.40	13	276.65	0.0630	1.43	1	571.13
15,000		99.72	133.53	43.48	15	130.00	0.0732	0.28	1	574.92
18,000		99.81	100.67	27.22	15	105.25	0.0600	2.92	2	575.69
20,000		99.78	109.87	29.51	14	200.07	0.0834	1.03	1	594.08
5,000	0.5	99.17	115.73	29.89	15	103.82	0.0906	1.19	3	514.71
7,000		99.38	129.40	54.70	15	172.85	0.1060	0.00	0	600.01
10,000		99.53	113.07	46.72	14	256.40	0.0710	1.16	2	545.45
13,000		99.64	116.67	35.76	14	237.52	0.0524	1.08	4	526.71
15,000		99.66	117.33	38.25	14	252.43	0.0788	0.05	0	600.01
18,000		99.71	119.53	55.07	13	281.16	0.0701	0.00	0	600.01
20,000		99.79	116.53	53.31	15	139.61	0.0595	1.52	0	600.01
5,000	0.75	99.08	117.53	15.67	15	84.44	0.0482	1.83	4	480.59
7,000		99.41	114.80	21.23	15	95.75	0.0430	7.24	1	560.75
10,000		99.58	119.40	23.31	15	105.54	0.0479	2.13	1	562.32
13,000		99.68	102.53	23.77	15	179.72	0.0315	2.28	2	540.12
15,000		99.75	111.80	32.98	15	115.43	0.0465	0.00	0	600.01
18,000		99.75	108.47	33.48	15	159.99	0.0403	0.54	1	595.19
20,000		99.79	122.20	39.63	15	205.50	0.0457	0.38	0	600.01

impressive. We can conclude from these values that the size of the instances we used is too important for this algorithm, since it is not able to solve at optimality almost all the instances, even in the uncorrelated case of MNK.

In Table 10, we report the results obtained for strongly correlated instances. We provide as additional information the final relative error for the solution returned by the ILPH in column *rerror*, computed as previously: ((ub – lb) / lb) × 100, with the final upper bound value (ub) and the final lower bound value (lb), obtained at the end of the ILPH. Values in Table 10 show that the ILPH is clearly less efficient as an exact method when strongly correlated instances are considered. The method is able to obtain five optimal solutions within the time limit we set. The more important initial gap with the LP-value and the less important efficiency of the fixation process are two main reasons to explain these results. However, we can observe that the final relative error is not very important, and that the best solution found by the ILPH is produced in general in a short time. These observations are encouraging results for using the ILPH also as a heuristic approach when considering more difficult and large instances of MNK.

Experiments presented in this section demonstrate that the ILPH is an efficient approach to solve at optimality large instances of MNK, in particular when the data are not strongly correlated. Indeed, the efficiency of the pseudo-cuts added during the search to reduce the gap between the

Table 9
Average results with ILPH and VPT over weakly correlated instances.

		ILPH-alone					VPT			
n	ρ	fix	iter*	CPU*	#opt	CPU	rerror	fix	#opt	CPU
5,000	0.25	99.25	88.60	7.11	15	27.09	0.0546	4.26	2	561.43
7,000		99.62	77.47	8.84	15	18.88	0.0491	5.01	1	561.02
10,000		99.16	57.60	2.61	15	10.34	0.0388	0.00	0	600.01
13,000		99.43	59.73	4.56	15	8.26	0.0299	0.00	0	600.01
15,000		99.32	57.60	4.33	15	5.10	0.0264	1.66	1	580.09
18,000		99.03	58.20	5.06	15	5.44	0.0302	0.00	0	600.01
20,000		98.30	54.40	5.21	15	5.23	0.0250	0.00	0	600.01
5,000	0.5	98.65	63.80	2.67	15	9.74	0.0416	0.23	1	570.52
7,000		98.90	69.00	4.08	15	5.65	0.0354	0.00	2	591.32
10,000		97.72	60.87	4.39	15	8.94	0.0244	0.00	1	570.58
13,000		98.83	59.67	3.58	15	4.94	0.0275	0.00	0	600.01
15,000		99.14	71.40	5.59	15	7.23	0.0229	0.00	0	600.01
18,000		98.75	65.47	6.71	15	8.53	0.0244	0.00	0	600.01
20,000		98.73	58.67	7.18	15	9.41	0.0199	0.00	0	600.01
5,000	0.75	98.82	62.20	3.41	15	13.70	0.0230	2.62	3	543.89
7,000		98.61	70.00	4.92	15	19.51	0.0223	0.00	1	562.33
10,000		98.96	54.47	2.78	15	8.04	0.0145	1.13	1	576.65
13,000		98.35	56.13	3.42	15	7.79	0.0166	0.00	0	600.01
15,000		98.75	59.73	4.25	15	7.81	0.0134	0.00	0	600.01
18,000		98.78	56.20	4.41	15	4.89	0.0117	0.00	0	600.01
20,000		99.22	65.73	6.28	15	7.19	0.0128	0.00	0	600.01

Table 10
Average results with ILPH and VPT over strongly correlated instances.

		ILPH-alone						VPT			
n	ρ	rerror	fix	iter*	CPU*	#opt	CPU	rerror	fix	#opt	CPU
500	0.25	0.0495	41.92	209.33	55.06	0	601.97	0.2454	2.52	1	573.83
1,000		0.0251	52.19	81.53	11.84	0	604.26	0.0909	4.61	0	600.01
4,000		0.0071	49.18	40.40	16.63	0	604.35	0.0295	8.32	0	600.01
5,000		0.0061	64.77	229.13	13.38	0	605.38	0.0187	18.17	0	600.01
7,000		0.0043	47.62	140.27	40.59	0	604.87	0.0181	4.11	0	600.01
10,000		0.0033	51.99	64.27	30.49	0	605.86	0.0094	14.20	0	600.01
500	0.5	0.0272	37.80	147.27	7.68	1	563.64	0.1383	4.91	1	599.63
1,000		0.0135	40.23	223.67	32.69	0	604.44	0.0851	0.36	0	600.01
4,000		0.0036	43.08	163.60	31.25	0	606.24	0.0188	5.25	0	600.01
5,000		0.0037	52.36	224.07	41.22	1	570.32	0.0155	3.41	0	600.01
7,000		0.0029	33.71	121.00	17.45	0	604.00	0.0095	6.04	0	600.01
10,000		0.0027	29.46	41.93	2.75	1	605.39	0.0106	1.88	0	600.01
500	0.75	0.0261	31.79	51.13	13.46	1	566.41	0.1097	4.96	1	560.11
1,000		0.0117	34.40	472.00	81.12	0	603.56	0.0606	3.15	0	600.01
4,000		0.0025	47.86	151.53	41.85	0	606.25	0.0100	13.91	0	600.01
5,000		0.0031	45.94	122.80	19.54	1	566.51	0.0116	8.48	0	600.01
7,000		0.0029	41.65	76.07	36.13	0	602.99	0.0098	4.86	0	600.01
10,000		0.0018	40.95	55.07	4.70	0	605.30	0.0063	7.67	0	600.01

upper bound and the lower bound, and the efficiency of the fixation process, are clearly correlated to the initial problem.

6. Conclusion

In this paper, we consider the two scenarios max–min Knapsack Problem (KP), a variant of the classical zero-one KP, where the values of the items differ under two possible scenarios. This problem is a particular case of the max–min multiscenarios KP (MNK) in which items' profit changes according to a set of objective functions. The aim is to choose a subset of items so that the minimal objective value associated to the selected items under the different scenarios is maximal, and so that the total weight of the selected items does not exceed a given capacity. In a first part, we introduce several formulations of the MNK as a mixed-integer programming problem (MIP). These formulations can be used to provide an upper bound of the problem, and to solve exactly the MNK. We also consider a quadratic formulation, and two possible linearization models. Then, we present a hybrid method combining heuristics and mathematical programming techniques to provide strong upper and lower bounds of the MNK. Lower bounds are obtained by solving subproblems in which variables are fixed temporarily. The algorithm converges to an optimal solution of the problem, by iteratively adding pseudo-cuts into the problem to strengthen the upper bound, and by reducing the gap between the bounds. An initial solution is derived from a surrogate-based heuristic, and we integrate a fixation technique to reduce the size of the initial problem by setting definitively variables at their optimal values. We performed several experiments on a set of 900 large and correlated instances with up to 20,000 items to evaluate and compare: (a) the different MIP models when using a black-box solver to solve the MNK; (b) the impact of using our algorithm as a preprocessing phase, before solving the problem with the solver; (c) the use of our approach as an exact or a heuristic approach to solve large instances. The results demonstrate that MIP models can be used efficiently by the solver, more particularly when the number of constraints is small, even if strongly correlated instances are more difficult to solve at optimality in reasonable running time. In addition, using directly our approach as an exact method is an efficient technique to solve MNK if the initial gap of the instance is small. Finally, the preprocessing phase performed by our algorithm provides a significant improvement in the average performance of the solver when considering large and correlated instances. This improvement is due in particular to the setting of an important number of variables in the initial problem, and to the reduction of the initial gap because of the addition of pseudo-cuts.

Future work may consider the solution of MNK when the number of scenarios increases. Exact methods available in the literature have trouble in exactly solving instances with more than 30 scenarios, in particular for strongly correlated instances. According to the results presented in this paper, MIP formulations need to be improved (in particular for strongly correlated instances), even when considering only two scenarios. We think that the use of our approach could be an interesting prospect, in particular by integrating additional features such as efficient reduction techniques and valid inequalities to strengthen the problem. The use of different relaxations to derive strong upper bounds and the use of temporarily setting and adaptive memory to obtain good lower bounds are two possible ways to solve efficiently MNK.

Acknowledgements

This work was partially supported by the Nord/Pas-de-Calais Region and the French Environment and Energy Management Agency (ADEME) under project ENTRe. This support is gratefully acknowledged. We also would like to thank the referees for their valuable suggestions in improving this paper.

References

Aissi, H., 2005. Approximation et résolution des versions min-max et min-max regret de problèmes d'optimisation combinatoire. PhD thesis, Université Paris-Dauphine, France.

Amrani, F., 1997. Problèms de Min-Max (Max-Min) en variables 0–1: Algorithmes de résolution exacte et approchée. PhD thesis, University of Valenciennes, France.

Balas, E., Zemel, E., 1980. E. An algorithm for large zero-one knapsack problems. *Operations Research* 28, 1130–1154.

Boussier, S., Vasquez, M., Vimont, Y., Hanafi, S., Michelon, P., 2010. A multi-level search strategy for the 0–1 multidimensional knapsack problem. *Discrete Applied Mathematics* 158, 97–109.

Cherfi, N., Hifi, M., 2009. Hybrid algorithms for the multiple-choice multidimensional knapsack problem. *International Journal of Operational Research* 5, 89–109.

Dantzig, G. B., 1957. Discrete–variable extremum problems. *Operations Research* 5, 266–277.

Du, D.Z., Pardalos, P.M. (eds.), 1995. *Minimax and Applications*. Kluwer Academic Publishers, Dordrecht.

Dyer, H.E., 1980. Calculating surrogate constraints. *Mathematical Programming* 19, 255–278.

Eben-Chaime, M., 1996. Parametric solution for linear bicriteria knapsack models. *Management Science* 42, 1565–1575.

Fayard, D., Plateau, G., 1977. Reduction algorithm for single and multiple constraints 0–1 linear programming problems. Presented at Conference on Mathematical Programming, Zakopane, Poland.

Fischetti, M., Lodi, A., 2003. Local branching. *Mathematical Programming* 98, 23–47.

Fréville, A.., 2004. The multidimensional 0–1 knapsack problem: An overview. *European Journal of Operational Research* 155, 1–21.

Fréville, A, Hanafi, S., 2005. The multidimensional 0–1 knapsack problem—bounds and computational aspects. *Annals of Operational Research* 139, 195–227.

Fréville, A., Plateau, G., 1993. An exact search for the solution of the surrogate dual of the 0–1 bidimensional knapsack problem. *European Journal of Operational Research* 68, 413–421.

Fréville, A., Plateau, G., 1996. The 0–1 bidimensional knapsack problem: Towards an efficient high-level primitive tool. *Journal of Heuristics* 2, 147–167.

Garey, M.R., Johnson, D.S., 1979. Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman, New York.

Geoffrion, A.M., 1974. The Lagrangian relaxation for integer programming. *Mathematical Programming Study* 2, 82–114.

Gilmore, P.C., Gomory, R.E., 1966. The theory and computation of knapsack functions. *Operations Research* 14, 1045–1075.

Glover, F., 1975. Surrogate constraints duality in mathematical programming. *Operations Research* 23, 434–451.

Glover, F., 2005. Adaptive memory projection methods for integer programming. In: Rego, C., Alidaee, B. (eds) *Metaheuristic Optimization Via Memory and Evolution*. Academic Publishers, Kluwer, pp. 425–440.

Hanafi, S., 1993. Contribution à la résolution de problèmes duaux de grande taille en optimisation combinatoire. PhD thesis, University of Valenciennes, France.

Hanafi, S. and Wilbaut, C., 2011. Improved convergent heuristics for the 0–1 multidimensional knapsack problem. *Annals of Operations Research* 183, 125–142.

Held, M., Karp, R.M., 1970. The traveling salesman problem and minimum spanning trees. *Operations Research* 18, 1138–1162.

Iida, H., 1999. A note on the max–min 0–1 knapsack problem. *Journal of Combinatorial Optimization* 3, 89–94.

Kellerer, H., Pferschy, U., Pisinger, D., 2004. *Knapsack Problems*. Springer, Berlin.

Kouvelis, P., Yu, G., 1997. *Robust Discrete Optimization and its Applications*. Kluwer Academic Publishers, Dordrecht.
Lorie, J.H., Savage, L. J., 1955. Three problems in capital rationing. *Journal of Business* 28, 229–239.
Martello, S., Toth, P., 1990. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley, New York.
Pisinger, D., 1995a. An expanding-core algorithm for the exact 0–1 knapsack problem. *European Journal of Operational Research* 87, 175–187.
Pisinger, D., 1995b. A minimal algorithm for the multiple-choice knapsack problem. *European Journal of Operational Research* 83, 394–410.
Roy, B., 2010. Robustness in operational research and decision aiding: A multi-faceted issue. *European Journal of Operational Research* 200(3), 629–638.
Savelsberg, M.W.P., 1994. Preprocessing and probing techniques for mixed integer programming problems. *ORSA Journal of Computing* 6, 445–454.
Sbihi, A., 2010. A cooperative local search-based algorithm for the multiple-scenario max–min knapsack problem. *European Journal of Operational Research* 202, 339–346.
Soyster, A. L., Lev, B., Slivka, W., 1978. Zero-one programming with many variables and few constraints. *European Journal of Operational Research* 2, 195–201.
Taniguchi, F., Yamada, T., Kataoka, S., 2008. Heuristic and exact algorithms for the max-min optimization of the multi-scenario knapsack problem. *Computers and Operations Research* 35, 2034–2048.
Taniguchi, F., Yamada, T., Kataoka, S., 2009. A virtual pegging approach to the max–min optimization of the bi-criteria knapsack problem. *International Journal of Computer Mathematics* 86, 779–793.
Toth, P., 1980. Dynamic programming algorithms for the zero-one knapsack problem. *Computing* 25, 29–45.
Vimont, Y., Boussier, S., Vasquez, M., 2008. Reduced costs propagation in an efficient implicit enumeration for the 01 multidimensional knapsack problem. *Journal of Combinatorial Optimization* 15, 165–178.
Wilbaut, C., Hanafi, S., Salhi, S., 2008. A survey of effective heuristics and their application to a variety of knapsack problems. *IMA Journal of Management Mathematics* 19, 227–244.
Yu, G., 1996. On the maximin knapsack problem with robust optimization applications. *Operations Research* 44, 407–415.