7^e *Conférence Francophone de MOdélisation et SIMulation - MOSIM'08 - du 31 mars au 2 avril* 2008 *- Paris - France*
Modélisation, Optimisation et Simulation des Systèmes : Communication, Coopération et Coordination

KNAPSACK PROBLEMS WITH SETUPS

S. Michel — ISEL, Université du Havre — michels@univ-lehavre.fr

N. Perrot — France-Télécom, division R&D — nancy.perrot@orange-ftgroup.com

F. Vanderbeck — IMB, Université Bordeaux 1 — fv@math.u-bordeaux1.fr

RÉSUMÉ : *We consider two variants of knapsack problems with setups arising as subproblems in a Dantzig-Wolfe decomposition approach to more complex combinatorial optimization problems. In the multiple-class binary knapsack problem with setups, items are partitioned into classes whose use implies a setup cost and associated capacity consumption. Item weights are assumed to be a multiple of their class weight. The total weight of selected items and setups is bounded. The objective is to maximize the difference between the profits of selected items and the fixed costs incurred for setting-up classes. In the continuous knapsack problems with setups, each class holds a single item and a fraction of an item can be selected while incurring a full setup. The paper shows the extent to which classical results for the knapsack problem can be generalized to these variants. In particular, an extension of the branch-and-bound algorithm of Horowitz and Sahni is developed for problems with positive setup costs. Our direct approach is compared experimentally with the approach proposed in the literature consisting in converting the problem into a multiple choice knapsack with pseudo-polynomial size.*

MOTS-CLÉS : *Knapsack problem, fixed cost, setup, variable upper bound, branch-and-bound.*

The Multiple-class Binary Knapsack problem with Setups (MBKS) is defined as follows. The knapsack has capacity W. There are n item classes, indexed by $i = 1, \ldots, n$, with associated setup cost, $f_i \in I\!R$, and setup capacity consumption, $s_i \in I\!R_+$. Each class is made of its own items (i, j) for $j = 1, \ldots, n_i \in I\!N$ with associated profit $p_{ij} \in I\!R$. The capacity consumption of item (i, j) is assumed to be a multiple of a class weight, $w_i \in I\!R_+$, i.e. $w_{ij} = m_{ij}\, w_i$ for some multiplicity $m_{ij} \in I\!N$ (assuming $w_{ij} \le W$). Moreover, there are lower and upper bounds, $a_i \le b_i \in I\!N$, on the total multiplicity of items that can be selected within each class. The objective is to maximize the sum of the profits associated with selected items minus the fixed costs incurred for setting-up classes.

Thus, model MBKS takes the form:

$$\max \sum_{i=1}^{n}\sum_{j=1}^{n_i} p_{i\,j}\, x_{i\,j} \quad - \quad \sum_{i=1}^{n} f_i y_i \tag{1}$$

$$\sum_{i=1}^{n}((\sum_{j=1}^{n_i} m_{i\,j}\, w_i\, x_{i\,j}) + s_i\, y_i) \quad \le \quad W \tag{2}$$

$$a_i\, y_i \le \sum_{j=1}^{n_i} m_{i\,j}\, x_{i\,j} \quad \le \quad b_i\, y_i \qquad \forall i \tag{3}$$

$$x_{i\,j} \quad \le \quad y_i \qquad \forall i, j \tag{4}$$

$$x_{i\,j} \quad \in \quad \{0, 1\} \qquad \forall i, j \tag{5}$$

$$y_i \quad \in \quad \{0, 1\} \qquad \forall i, \tag{6}$$

where x_{ij} indicates if item j is chosen within class i and $y_i = 1$ iff class i is setup.

When each class holds a single item, i.e. $n_i = 1\ \forall i$, and when a fraction of an item can be taken, we obtain the continuous knapsack problem with setups (CKS):

$$\max \sum_{i=1}^{n} p_i\, x_i \quad - \quad \sum_{i=1}^{n} f_i\, y_i \tag{7}$$

$$\sum_{i=1}^{n}(w_i\, x_i + s_i\, y_i) \quad \le \quad W \tag{8}$$

$$a_i\, y_i \le x_i \quad \le \quad b_i\, y_i \qquad \forall i \tag{9}$$

$$x_i \quad \ge \quad 0 \qquad \forall i \tag{10}$$

$$y_i \quad \in \quad \{0, 1\} \qquad \forall i. \tag{11}$$

Observe that, when $a_i = f_i = w_i = 0$ and $b_i = n_i = 1\ \forall i$, all the above models boil down to a standard binary knapsack problem. Hence, they are at least as hard as the standard binary knapsack problem.

These problems can get simpler if fixed costs are assumed to be non-negative.

Assumption 1 **(restrictive)** *$f_i \ge 0$ for all i,* or if there are no class lower bounds

Assumption 2 *(**restrictive**) $a_i = 0$ for all i.*

The Branch-and-Bound developed in the paper are made under these restrictive assumptions.

Model CKS arises as a sub-problem in capacitated multi-item lot sizing problem. (Goemans 1989) studied the structure of the CKS polyhedron, derived facet defining inequalities and proposed a heuristic separation procedure.

Model MBKS arises as a sub-problem in a branch-and-price approach to the cutting stock problem (Vanderbeck 1999). The special case of model MBKS with no setups is treated in (Vanderbeck 2002). Under the assumption $f_i = s_i = 0\ \forall i$, it is shown that the LP-relaxation can be solved by a greedy algorithm in linear time, a result that extends those of (Dantzig 1957) and (Balas & Zemel 1980) for the 0-1 knapsack problem (this result relies on the assumption that item weights are a multiple of their class weight); exact algorithms are derived (branch-and-bound or dynamic programs) by adapting existing algorithms for the 0-1 knapsack problem. Variant of model MBKS are considered in the literature (Chajakis & Guignard 1994, Jans & Degrave 2004).

The present paper proposes an analysis of models CKS and MBKS. The aim is to show the extent to which classical approach for the knapsack problem, such as the depth-first-search branch-and-bound algorithm of Horowitz and Sahni (see (Martello & Toth 1990) pages 30-31 or (Nemhauser & Wolsey 1988) pages 455-456) can be generalized to variants with setups. In particular, we show that under assumptions slightly less restrictive than Assumptions 1 and 2, the LP solution to these problems can be obtained in polynomial time by a greedy procedure. The key to these results are reformulation as continuous knapsack problems with multiple choice constraints (Johnson & Padberg 1981) or class bounds (Vanderbeck 2002). The formulation are polynomial in size while previously proposed reformulation as that of (Chajakis & Guignard 1994) are pseudo-polynomial. However, our reformulations are only valid for the LP-relaxation: their integer counterparts are not equivalent to our models. Therefore, the greedy LP solver does not immediately give rise to extension of standard branch-and-bound procedures. The other main contribution of the paper is a specific enumeration scheme for branch-and-bound for CKS and MBKS that exploit the property of optimal solutions and the greedy ordering of the LP bound. The resulting branch-and-bound algorithms are tested and compared to existing approaches.

1. THE CONTINUOUS KNAPSACK PROBLEM WITH SETUPS

The formulation of the model CKS is given by (7-11). Here, bounds a_i and b_i are not necessarily integer, i.e. $a_i, b_i \in I\!R^+,\ \forall i$. Assumption 2 can be made without loss of generality (w.l.g.). Indeed, if $a_i > 0$ for some i, one can transform the problem as follows: $x'_i = x_i - a_i\, y_i$ and the lower bound is eliminated. Moreover, we can assume

Assumption 3 *(**w.l.g.**) $p_i \geq 0$ for all i.*

Otherwise, $x_i = 0$ in any optimal solution.

Assumption 4 *(**w.l.g.**) $f_i \leq 0$ for all i.*

Indeed, if $f_i \geq p_i\, b_i$ for some i, it is optimal to set $x_i = y_i = 0$ and consider the problem that remains on the other variables. While, if $0 < f_i < p_i\, b_i$ for some i, then, in any optimal solution, either $x_i = y_i = 0$ or $x_i \geq \frac{f_i}{p_i}$. Thus, $\frac{f_i}{p_i}$ can be interpreted as a lower bound a_i which can be eliminated as explained above.

Assumption 5 *(**w.l.g.**) $w_i = 1$ for all i.*

Otherwise, one can make a change of variables $x'_i = w_i\, x_i$.

In the rest of this section, we make Assumptions 2 to 4 without loss of generality, but we carry w_i in the notation for the sake of extending the results to model MBKS where Assumption 5 is not made. Similarly, when Assumption 1 is made, $f_i = 0\ \forall i$ (as implied by Assumption 4) but we keep f_i in the formulation.

1.1. Characterizations of optimal solutions

Some properties of optimal solutions are used to develop bounding procedure. First, note that under Assumption 5 if data are integer, i.e., if $a_i, b_i, s_i \in I\!N\ \forall i$ and $W \in I\!N$, the continuous variables x take integer value in any feasible extreme solutions. Then, the problem can be reformulated as a multiple choice knapsack problem:

$$\max\{\sum_i \sum_{x=a_i}^{b_i} (p_i\, x - f_i)\, \lambda^i_x : \sum_i \sum_{x=a_i}^{b_i} (x + s_i)\, \lambda^i_x \leq W, \sum_{x=a_i}^{b_i} \lambda^i_x \leq 1\ \forall i, \lambda^i_x \in \{0,1\}\ \forall i, x\}\ , \quad (12)$$

where $\lambda_x^i = 1$ iff $x_i = x$. This formulation has pseudo-polynomial size.

In the rest of this section we do not assume integer data. If one fixes y to $\tilde{y} \in \{0,1\}^n$, the problem reduces to a continuous knapsack problem, called $CKP(\tilde{y})$, that admits a greedy solution (Dantzig 1957). Let us explicitly state the characterization of extreme solutions to CKS for easy reference.

Observation 1 *An optimal solution exists to problem CKS where, for each i, one of the following case arises*

$$\begin{array}{ll} (i) & y_i = 1 \text{ and } x_i = b_i \\ (ii) & y_i = 1 \text{ and } 0 < x_i < b_i \\ (iii) & y_i = x_i = 0 \\ (iv) & y_i = 1 \text{ and } x_i = 0 \end{array}$$

Furthermore, case (ii) can only be assumed by one of the items which is called the critical item. Case (iv) can only arise if $f_i < 0$. Moreover, the level of the critical item, c, if any, is set so as to fill the remaining capacity of the knapsack. It is computed as $x_c = \frac{(W-\overline{W})}{w_c}$ where

$$\overline{W} = \sum_{k\in I}(w_k\, b_k + s_k) + s_c + \sum_{k\in J} s_k$$

for some sets $I \subseteq \{k : \frac{p_k}{w_k} \geq \frac{p_c}{w_c}\}$ and $J \subseteq \{k : f_k < 0$ and $\frac{p_c}{w_c} \geq \frac{p_k}{w_k}\}$.

The continuous relaxation of model CKS is given by (7-10) and $y_i \in [0,1]$ $\forall i$. Then, the contribution of the ith item to the LP solution can be anything in the convex hull of extreme solutions (i), (iii) and (iv) of Observation 1 (note that case (ii) is in this convex hull). For instance, one can generate any profit between 0 and $p_i\, b_i - f_i$ by setting $y_i = \frac{x_i}{b_i}$ and letting x_i vary between 0 and b_i. The associated capacity consumption is $(w_i + \frac{s_i}{b_i})\, x_i$. Similarly, if $f_i < 0$, any pair of $(p, w) = (-y_i\, f_i, y_i\, s_i)$ can be achieved by setting $x_i = 0$ and varying $y_i \in [0,1]$. Therefore, the continuous relaxation of CKS can be reformulated as follows:

$$\max \sum_{i=1}^{n}(p_i\, b_i - f_i) z_i^b \quad - \quad f_i z_i^f \qquad (13)$$

$$\sum_{i=1}^{n}(w_i\, b_i + s_i)\, z_i^b + s_i\, z_i^f \quad \leq \quad W \qquad (14)$$

$$z_i^b + z_i^f \quad \leq \quad 1 \qquad \forall i \qquad (15)$$

$$z_i^b \quad \in \quad [0,1] \qquad \forall i \qquad (16)$$

$$z_i^f \quad \in \quad [0,1] \qquad \forall i \qquad (17)$$

When $z_i^f = 0$, letting z_i^b vary from 0 to 1 allows to achieve any continuous solution in the convex hull of extreme solutions (i) and (iii) of Observation 1. Inversely, $z_i^b = 0$, letting z_i^f vary from 0 to 1 allows to achieve any continuous solution in the convex hull of extreme solutions (iii) and (iv). While, setting $z_i^f = 1 - z_i^b$, allows to achieve any continuous solutions of type (ii) in the convex hull of extreme solutions (i) and (iv). Any other solution for class i is LP dominated. The mapping from a solution z of (13-17) to a solution (x, y) for the LP relaxation of CKS is given by

$$x_i = b_i\, z_i^b \quad \text{and} \quad y_i = z_i^b + z_i^f \ .$$

Moreover, one can characterized the conditions under-which continuous solutions with $z_i^f > 0$ are dominated:

Observation 2 *If $\frac{p_i\, b_i - f_i}{w_i\, b_i + s_i} \geq \frac{-f_i}{s_i}$, there exists an optimal solution to (13-17) where $z_i^f = 0$.*

Note that Assumption 1 implies $\frac{p_i\, b_i - f_i}{w_i\, b_i + s_i} \geq \frac{-f_i}{s_i}$ $\forall i$.

Thus, we have shown that

Proposition 1 *The LP relaxation of CKS is equivalent to the continuous relaxation of binary knapsack problem with multiple choice constraints or special ordered set (SOS) constraints (13-17) .*

The latter is known to admit a greedy solution (Johnson & Padberg 1981). Hence, this result extends to our model with setups. To be explicit, let us write the greedy LP solution. We do this under the assumption of Observation 2. Then, SOS constraints (15) are redundant and (13-17) reduces to a standard binary knapsack LP relaxation:

Observation 3 *If $\frac{p_i\, b_i - f_i}{w_i\, b_i + s_i} \geq \frac{-f_i}{s_i}$ $\forall i$, an optimal solution to the LP relaxation of problem CKS is given by indexing the items in order that*

$$\frac{(p_1\, b_1 - f_1)}{(w_1\, b_1 + s_1)} \geq \frac{(p_2\, b_2 - f_2)}{(w_2\, b_2 + s_2)} \geq \ldots \geq \frac{(p_n\, b_n - f_n)}{(w_n\, b_n + s_n)} \ .(18)$$

and setting

$$\begin{array}{rcl} x_i & = & b_i \text{ and } y_i = 1 \qquad \text{for } i < c, \\ x_c & = & \dfrac{W - \sum_{i<c}(w_i\, b_i + s_i)}{(w_c + \frac{s_c}{b_c})} \text{ and } y_c = \dfrac{x_c}{b_c}, \\ x_i & = & 0 \text{ and } y_i = 0 \qquad \text{for } i > c\ , \end{array}$$

where c is the index in the sorted item order such that

$$\sum_{i<c}(w_i\, b_i + s_i) < W \ \text{ but } \sum_{i\leq c}(w_i\, b_i + s_i) \geq W \ .$$

This observation merely translates the greedy LP solution of (13-17) in the (x, y) variables (when $z_i^f = 0\ \forall i$).

Now consider the integer version of formulation (13-17) where $z_i^b, z_i^f \in \{0,1\}\ \forall i$. It is important to note that, although model CKS and formulation (13-17) have the same LP solution, the integer version of (13-17) is not equivalent to model CKS. Indeed, a solution where case (ii) of Observation 1 is optimal for some item i cannot be represented as an integer solution in the z-formulation.

1.2. Branch-and-Bound

The standard Branch-and-Bound algorithm of Horowitz and Sahni for the 0-1 knapsack problem (Nemhauser & Wolsey 1988) is a specialized depth-first-search branch-and-bound where variables are fixed in an order that is greedy for both primal and dual bounding procedure. Hence, the first leaf node to be explored when plunging depth into the tree corresponds to a greedy primal solution while backtracking leads to exploring progressively more distant neighbors of this greedy solution. The dual bounds need not be recomputed after fixing a variable to one: their value differ from that of the parent node only for the branch where we part from the greedy ordering, i.e., when a variable is fixed to zero.

The extension to model CKS is not trivial. Indeed, the procedure requires that the same greedy approach solves both LP and IP problem. We have such greedy approach for the z-formulation but not for the (x, y)-formulation. As mentioned above, both formulation are not equivalent in IP term. This difficulty can be overcome by implicitly dealing with both the z-formulation (for dual decisions) and the (x, y)-formulation (for primal decisions). To illustrate how, we explicitly provide a branch-and-bound procedure under Assumption 1 (which simplifies the problem).

We use the greedy ordering (18) that was defined for the z variables. But, the fixing of z variables has a different interpretation for dual and primal bounds. The primal solution (x, y) associated with a dual solution z is obtained by rounding-up the fractional setup included in z (which yields a setup solution $\tilde{y}$) and by setting x is the optimal solution of CKP$(\tilde{y})$. This ensures that the solution we build obeys the characterization of Observation 1. The so called "forward moves" are sequences of branches where a z variable is fixed to 1. Forward moves are interspersed with "fixing-to-zero" branches. Primal and dual bounds are evaluated after each "fixing-to-zero" branching.

The branch-and-bound search is organized as follows: Items are considered in the order (18). For each item i, three cases are considered: (a) $z_i = 1$, (b) $y_i = 1$, and x is free, (c) $z_i = 0$. However, Observation 1 tells us that case (b) needs only be considered if the knapsack is filled at full capacity. Hence, we can manage with a binary tree where only two branches are defined for each item: the first branch to be explored corresponds to aggregated case (a) or (b), while the second branch corresponds to case (c). When case (a) is feasible given the residual capacity, the first branch is interpreted as case (a) as it dominates case (b). But when branching on case (a) is infeasible due to lack of capacity, all the previous left branches are interpreted as case (b). In the latter situation, we have reached a leaf node which we call "A type leaf node" (the knapsack is filled at full capacity) where $CKP(y)$ is solved. Another type of leaf node (called "B type") arises when there are no more items to consider. In the latter case, as the branch $z_n = 1$ has been explored, the branch $z_n = 0$ does not need to be explored as it is dominated. This algorithm is presented in Algorithm 1 (in a pseudo-language where we make use of some C++ notations).

```
**Initialization:** Sort items according to (18).
    Let $INC = Z = 0$; $C = W$; $x = y = 0$; $i = 1$.

**Compute UB:** Let $U = Z$; $K = C$; $l = i$;
    while ($l \leq n$) and ($K \geq w_l b_l + s_l$), do {
    $U$+= $(p_l b_l - f_l)$; $K$−= $(w_l\, b_l + s_l)$; $l = l + 1$. }
    If ($l \leq n$), $U$+= $(p_l b_l - f_l)\, \frac{K}{(w_l\, b_l + s_l)}$.

**Test Pruning:** if ($U \leq INC$), goto Backtracking.

**Forward Move:** While ($i \leq n$) and ($w_i b_i + s_i \leq C$),
    do { $Z$+= $(p_i b_i - f_i)$; $C$−= $(w_i\, b_i + s_i)$; $x_i = b_i$;
    $y_i = 1$; $i = i + 1$. }

**Type A Leaf Node:** If ($i \leq n$) /* and ($w_i b_i + s_i >$
    $C$) */, do {
    If ($s_i < C$), {
    let $y_i = 1$; $v = CKP(y)$; $\tilde{x} = \text{argmax}\{CKP(y)\}$;
    if ($v > INC$), $INC = v$, record $(\tilde{x}, y)$; }
    Let $x_i = y_i = 0$; $i = i + 1$; /* Zero Setting */
    and go to Compute UB. }

**Type B Leaf Node:** Else /* ($i = n + 1$) */, do {
    If ($Z > INC$), {$INC = Z$; record $(x, y)$;}
    Let $i = i - 1$;
    If ($y_i == 1$), { $Z$−= $(p_i b_i - f_i)$; $C$+= $(w_i b_i + s_i)$;
    $x_i = y_i = 0$.}}

**Backtracking:** Do ($i = i - 1$) while ($y_i == 0$) and
    ($i \geq 1$).
    If ($i == 0$), STOP.
    $Z$−= $(p_i b_i - f_i)$; $C$+= $(w_i\, b_i + s_i)$; $x_i = y_i = 0$;
    $i = i + 1$ /* Zero Setting */.
    Go to Compute UB.
```

Algorithm 1. BaB for CKS under Assumption 1.

2. THE MULTIPLE-CLASS BINARY KNAPSACK WITH SETUPS

The formulation of the model MBKS is given by (1-6). Here, Assumption 2 is restrictive and so is Assumption 3: indeed, if $a_i > 0$ there is a knapsack sub-problem to be solved to decide which items (i,j) within class i should be selected to satisfy this lower bound, given that they have profits $p_{ij} \neq m_{ij}\, p_i$. An item (i,j) with negative profit $p_{ij} < 0$ might even be worth selecting to satisfy the lower bound a_i. However, if $a_i = 0$, one can make non restrictive assumptions:

Assumption 6 *(***w.l.g.***) Under Assumption 2, $p_{ij} \geq 0$ for all j.*

Otherwise, $x_{ij} = 0$ in any optimal solution.

Assumption 7 *(***w.l.g.***) $f_i < \max\{\sum_j p_{ij}\, x_{ij} : \sum_j m_{ij}\, x_{ij} \leq b_i,\ x_{ij} \in \{0,1\}\ \forall j\}$.*

Otherwise, it is optimal to set $x_{ij} = 0\ \forall j$ and $y_i = 0$.

2.1. Characterizations of optimal solutions

Integer solutions to model MBKS have some structure although not as simple as for CKS. In the line of the work of (Chajakis & Guignard 1994), MBKS can be decomposed into knapsack subproblem associated with each class. Since we assumed that all items have a weight that is a multiple of the class weight, the capacity consumption of a class i, i.e., $w_i\,(\sum_j m_{ij}\, x_{ij})$, is the same for all solutions $x_{i.}$ yielding the same total multiplicity $\sum_j m_{ij}\, x_{ij}$. As a result, the optimization within each class can be done independently of the global optimization of the use of the knapsack capacity W.

A combination of the x_{ij} variables that yields a dominated pair $(M_i = \sum_j m_{ij} x_{ij}, P_i = \sum_j p_{ij}\, x_{ij})$ for class i need not be considered: a pair (M_i, P_i) is dominated if another class i solution achieves a profit at least as large with a smaller multiplicity or a greater profit for the same multiplicity. This leads to a reformulation as a multiple choice knapsack:

$$\begin{aligned}\max \quad \{ \quad & \sum_i \sum_{M=a_i}^{b_i} (P_i(M) - f_i)\lambda_M^i : \\ & \sum_i \sum_{M=a_i}^{b_i} (M\, w_i + s_i)\, \lambda_M^i \leq W, \\ & \sum_{M=a_i}^{b_i} \lambda_M^i \leq 1\ \forall i, \\ & \lambda_M^i \in \{0,1\} \forall i, M \quad \} \end{aligned} \tag{19}$$

where $P_i(M)$ is the best profit that can be achieved within class i using a multiplicity M:

$$P_i(M) = \max\{\sum_j p_{ij}\, x_{ij} : \sum_j m_{ij}\, x_{ij} = M, \quad x_{ij} \in \{0,1\}\ \forall j\} \tag{20}$$

Thus, computing $P_i(M)$'s requires solving a "all-capacities" knapsack problem over each class i (a dynamic program is well suited for this). Moreover, this reformulation involves a pseudo-polynomial number of variables.

Note that an optimal solution may have $y_i = 1$ while the x_{ij}'s are set to the minimum value that allows to satisfy the class lower bound a_i. However, this is not the case when $a_i = 0$ and $f_i \geq 0$.

Observation 4 *Under Assumptions 1 and 2, there exists an optimal solution where $y_i = 0$ when $\sum_j x_{ij} = 0$.*

The LP solution to MBKS can also be characterized in view of the above decomposition:

Observation 5 *Consider solutions to the LP relaxation of MBKS. Their projection in the subspace $(x_i = \sum_j m_{ij}\, x_{ij},\ y_i)$ associated with class i are convex combinations of the following extreme points:*

$$\begin{array}{ll} (i) & x_i = 0\ (i.e.\ x_{ij} = 0\ \forall j) \text{ and } y_i = 0 \\ (ii) & x_i = \sum_j m_{ij}\, x_{ij} = a_i \text{ and } y_i = 1 \\ (iii) & x_i = \sum_j m_{ij}\, x_{ij} = b_i \text{ and } y_i = 1 \end{array}$$

If the profit per unit of knapsack capacity of extreme solution (ii) is less than that of (iii), i.e., if

$$\frac{P_i^{LP}(a_i) - f_i}{w_i\, a_i + s_i} \leq \frac{P_i^{LP}(b_i) - f_i}{w_i\, b_i + s_i}$$

where $P_i^{LP}(M)$ is the solution to the LP relaxation of (20), then one only needs to consider solutions that are convex combination of cases (i) and (iii). The reverse case, i.e. $\frac{P_i^{LP}(a_i)-f_i}{w_i\, a_i+s_i} > \frac{P_i^{LP}(b_i)-f_i}{w_i\, b_i+s_i}$, can only arise if $a_i > 0$ or $f_i < 0$.

Similarly to what we did for model CKS, we can derive from Observation 5 a z-reformulation of the LP relaxation of MBKS. In the continuous relaxation of MBKS, item (i,j) can yield a profit per unit equal to

$$\text{either} \quad \frac{p_{ij}\, \frac{a_i}{m_{ij}} - f_i}{w_i\, a_i + s_i} \quad \text{or} \quad \frac{p_{ij}\, \frac{b_i}{m_{ij}} - f_i}{w_i\, b_i + s_i}$$

or a convex combination of these two, depending of whether it is contributing to the class effort of targeting extreme solution (*ii*) or (*iii*) of Observation 5 or their combination. Case (*ii*) can be split in two sub-cases, either $a_i > 0$ or $f_i < 0$. Let $I^a = \{i : a_i > 0\}$ and $I^f = \{i : a_i = 0 \text{ and } f_i < 0\}$. Thus, $I^a \cap I^f = \emptyset$. For $i \in I^f$, the extreme solution (*ii*) takes the form $(x_i = 0, y_i = 1)$. Hence, we introduce variable $z_i^f \in [0,1]$ such that $z_i^f = 1$ represents solution $(x_i = 0, y_i = 1)$. Similarly, for $i \in I^a$, we define a variable z_{ij}^a for each item (i,j) of class i, such as $z_{ij}^a = 1$ if item (i,j) contributes in full to targeting extreme solution (*ii*). Symmetrically, variables z_{ij}^b are defined for $i \in I = \{1, \ldots, n\}$ and associated with extreme solutions (*iii*). With these notations, the LP relaxation of MBKS can be reformulated as

$$\max \quad \sum_{i \in I^a} \sum_{j=1}^{n_i} (p_{ij} - \frac{f_i}{a_i}\ m_{ij})\ z_{i\,j}^a + \sum_{i \in I} \sum_{j=1}^{n_i} (p_{ij} - \frac{f_i}{b_i}\ m_{ij})\ z_{i\,j}^b - \sum_{i \in I^f} f_i z_i^f \quad (21)$$

s.t.

$$\sum_{i \in I^a} \sum_{j=1}^{n_i} (w_i + \frac{s_i}{a_i})\ m_{i\,j}\ z_{i\,j}^a \ + \ \sum_{i \in I} \sum_{j=1}^{n_i} (w_i + \frac{s_i}{b_i})\ m_{i\,j}\ z_{i\,j}^b + \sum_{i \in I^f} s_i\ z_i^f \ \leq \ W \quad (22)$$

$$\sum_{j=1}^{n_i} [\frac{m_{i\,j}}{a_i}\ z_{i\,j}^a + \frac{m_{i\,j}}{b_i}\ z_{i\,j}^b] \ \leq \ 1 \qquad \forall i \in I^a \quad (23)$$

$$\sum_{j=1}^{n_i} \frac{m_{i\,j}}{b_i}\ z_{i\,j}^b + z_i^f \ \leq \ 1 \qquad \forall i \in I^f \quad (24)$$

$$\begin{aligned} z_{i\,j}^a + z_{i\,j}^b &\leq 1 && \forall i \in I^a, j \\ z_{i\,j}^a &\in [0,1] && \forall i \in I^a, j \\ z_{i\,j}^b &\in [0,1] && \forall i \in I, j \\ z_i^f &\in [0,1] && \forall i \in I^f \qquad (25) \end{aligned}$$

A solution z translates into a solution for the LP relaxation of MBKS as follows:

$$x_{ij} = z_{ij}^a + z_{ij}^b \quad \text{and} \quad y_i = \sum_j (\frac{m_{i\,j}}{a_i} z_{i\,j}^a + \frac{m_{i\,j}}{b_i} z_{i\,j}^b) + z_i^f \quad (26)$$

Constraints (23-24) are required to enforce $y_i \in [0,1]$. Observe that constraints (3) are built in the definition of the change of variables. Indeed, if we replace x and y by their expression in z in (3), in the case $a_i > 0$, we obtain:

$$\begin{aligned} a_i \sum_j (z_{ij}^a\ \frac{m_{ij}}{a_i} + z_{ij}^b\ \frac{m_{ij}}{b_i}) &\leq \sum_j m_{ij}\ (z_{ij}^a + z_{ij}^b) \\ \sum_j m_{ij}\ (z_{ij}^a + z_{ij}^b) &\leq b_i \sum_j (z_{ij}^a\ \frac{m_{ij}}{a_i} + z_{ij}^b\ \frac{m_{ij}}{b_i}) \end{aligned}$$

which is always satisfied because $\frac{a_i}{b_i} \leq 1$ in the left-hand-side and $\frac{b_i}{a_i} \geq 1$ in the right-hand-side. In the case $a_i = 0$, (3) is trivially verified. Hence, we have shown that

Proposition 2 *The LP relaxation of MBKS is equivalent to the continuous relaxation of binary knapsack problem with class bounds and SOS constraints (21-25).*

On one hand, it is known that the LP relaxation of binary knapsack problem with SOS constraints admits a greedy solution (Johnson & Padberg 1981). On the other hand, (Vanderbeck 2002) shows that the LP relaxation of a binary knapsack with class bounds can also be solved using a greedy procedure. But, solving problem (21-25) requires dealing with both SOS constraints and class bounds. We have not found a greedy procedure to solve this case involving both complexities. Instead, we develop a greedy LP solution for the special case where SOS constraints are redundant, a result that extends that of reference (Vanderbeck 2002) to the case with setups.

We make the simplifying assumption that all class i items target a filling up to b_i because this corresponds to a better ratio:

Assumption 8 *(***restrictive)**

$$\frac{p_{ij}\ \frac{a_i}{m_{ij}} - f_i}{w_i\ a_i + s_i} \ \leq \ \frac{p_{ij}\ \frac{b_i}{m_{ij}} - f_i}{w_i\ b_i + s_i} \quad \forall (i,j)\,.$$

Note that Assumption 1 implies the above since, when $f_i \geq 0$, either $a_i > 0$ and $(p_{ij} - \frac{f_i}{a_i} m_{ij}) \leq (p_{ij} - \frac{f_i}{b_i} m_{ij})$ while $(w_i + \frac{s_i}{a_i}) \geq (w_i + \frac{s_i}{b_i})$, or $a_i = 0$ and $p_{ij} \geq 0$ as stated in Assumption 6. Hence, Assumption 8 can be understood as less restrictive than Assumption 1.

Observation 6 *Under Assumption 8, problem (21-25) admits a solution where all variables z_{ij}^a and z_i^f have value zero.*

Indeed, if Assumption 8 holds and $z_{ij}^a > 0$, one can modify the solution by setting ${z_{ij}^a}' = 0$ and ${z_{ij}^b}' = z_{ij}^b + \frac{(w_i + \frac{s_i}{a_i})\, m_{ij}}{(w_i + \frac{s_i}{b_i})\, m_{ij}}\ z_{ij}^a$. This solution modification is feasible with regard to knapsack constraint (22) by construction but also with regard to constraint (23) as it can be easily checked. Moreover, the profit value of the modified solution is not less than the original. Similarly, if $z_i^f > 0$, decreasing its value allows to increase some z_{ij}^b value of better profit ratio.

Then, we can give a greedy LP solution to MBKS:

Proposition 3 *If Assumption 8 holds, an optimal solution to the LP relaxation of MBKS is given by the following procedure. Sort the items (i, j) in non-increasing order of their ratio:*

$$\frac{(p_{ij} - \frac{f_i}{b_i}\, m_{ij})}{(w_i + \frac{s_i}{b_i})\, m_{ij}} \tag{27}$$

Let $m = \sum_i n_i$ and $k = 1, \ldots, m$ be the item indices in that ordering. K^i is the set of items k that belong to class i:

$$K^i = \{k : \exists j \in \{1, \ldots, n_i\} \text{ with } k = (i, j)\} \,.$$

For $i \in \{1, \ldots, n\}$, let the critical item for class i be $c_i \in K^i$, be such that

$$\sum_{k \in K^i,\, k < c_i} m_k \le b_i \quad \text{but} \quad \sum_{k \in K^i,\, k \le c_i} m_k > b_i \,. \tag{28}$$

Let $K^i(l) = \{k \in K^i : k < c_i \text{ and } k < l\}$, $I^b(l) = \{i : c_i < l\}$, and

$$W(l) = \sum_{i, k \in K^i(l)} (w_i + \frac{s_i}{b_i}) m_k + \sum_{i \in I^b(l)} (w_i + \frac{s_i}{b_i})(b_i - \sum_{k \in K^i(l)} m_k) \,. \tag{29}$$

Then, let the global critical item, $c \in \{1, \ldots, m\}$, be the highest index item such that

$$W(c) \le W \quad \text{but} \quad W(c) + (w_{i_c} + \frac{s_{i_c}}{b_{i_c}})\, m_c > W \tag{30}$$

where i_c refers to the class containing the global critical item (i.e. $c \in K^{i_c}$) and set

$$\begin{aligned}
x_k &= 1 && \text{for } k \in K^i(c) \text{ and } i = 1, \ldots, n && (31)\\
x_{c_i} &= \frac{1}{m_{c_i}} (b_i - \sum_{k \in K^i(c)} m_k) && \text{for } i \in I^b(c) && (32)\\
x_c &= \frac{1}{(w_{i_c} + \frac{s_{i_c}}{b_{i_c}})\, m_c} (W - W(c)) && \text{if } c \in K^{i_c} && (33)\\
x_k &= 0 && \text{otherwise} && (34)\\
y_i &= 1 && \text{for } i \in I^b(c) && (35)\\
y_{i_c} &= \frac{\sum_{k \in K^{i_c}(c)} m_k + m_c\, x_c}{b_{i_c}} && \text{for } i : c \in K^i && (36)\\
y_i &= 0 && \text{otherwise.} && (37)
\end{aligned}$$

Proof: Observation 6 implies that in the LP formulation (21-25) we only keep the z^b_{ij} variables. The simplified formulation is that of a continuous multiple class knapsack problem that admits a greedy solution as proved in (Vanderbeck 2002). Converting the greedy solution z into the original variables x and y provides the desired result. ■

2.2. Branch-and-Bound

The greedy procedure that allows to solve (21-25) under Assumption 8 can be the basis for a specialized branch-and-bound algorithm for MBKS that generalizes the depth-first-search algorithm of Horowitz and Sahni. However, we are confronted with the same difficulty as for model CKS: the greedy procedure is defined for the z-formulation whose IP counterpart is not equivalent to model MBKS. To overcome it, we branch on z variables but make corrections to the primal solutions to make them feasible for MBKS.

To simplify the presentation, we make Assumptions 1 and 2 that imply Assumption 8. Then, the integer version of the z-formulation takes the form

$$\begin{aligned}
\max \sum_{i=1}^{n} \sum_{j=1}^{n_i} (p_{ij} - \frac{f_i}{b_i}\, m_{ij})\, z_{ij} && && (38)\\
\text{s.t.} \sum_{i=1}^{n} \sum_{j=1}^{n_i} (w_i + \frac{s_i}{b_i})\, m_{ij}\, z_{ij} &\le W && (39)\\
\sum_j m_{ij}\, z_{ij} &\le b_i \quad \forall i && (40)\\
z_{ij} &\in \{0, 1\} \quad \forall i, j \,. && (41)
\end{aligned}$$

Problem MBKS and problem (38-41) admit the same LP solution but not the same integer solution. The Branch-and-Bound strategy is to apply the Horowitz and Sahni scheme to the above z-formulation, to translate z solution into a feasible solution for MBKS, by applying the mapping (26) and rounding-up the y variables, and to adapt the residual problem dynamically at each branch-and-bound node.

Fixing z_{ij} to 1, translate into fixing $x_{ij} = 1$ and also $y_i = 1$ (if the class set-up was not already set to 1). Hence, for the residual problem, the attractivity of the items (i, j) from class i with $y_i = 1$ is proportional to their ratio

$$\frac{p_{ij}}{w_i m_{ij}}. \tag{42}$$

Thus, at a given branch-and-bound node, the z-formulation of the residual problem takes the form

$$\max \sum_{i \in I^0} \sum_{j \in J_i} (p_{ij} - \frac{f_i}{b_i}\, m_{ij})\, z_{ij} + \sum_{i \in I^1} \sum_{j \in J_i} p_{ij}\, z_{ij} \tag{43}$$

s.t.

$$\begin{aligned}
\sum_{i \in I^0, j} (w_i + \frac{s_i}{b_i}) m_{ij} z_{ij} + \sum_{i \in I^1, j} w_i m_{ij} z_{ij} &\le C && (44)\\
\sum_{j \in J_i} m_{ij}\, z_{ij} &\le C_i \quad \forall i && (45)\\
z_{ij} &\in \{0, 1\} \quad \forall i, j && (46)
\end{aligned}$$

where I^0 (resp. I^1) is the set of classes for which y_i is still at value zero (resp. is already set to 1), J_i denotes the set of class i items that have not yet been fixed to 0 or 1, C denotes the remaining knapsack capacity, and C_i the residual upper bound on class i items. By extension of the above arguments, one can easily be convinced that, at a given branch-and-bound node, problem (43-46) and the residual MBKS problem in its (x, y)-formulation have the same LP value and therefore computing the LP-solution of (43-46) provides a valid dual bound for that node. Finally, observe that a preprocessing can be applied to (43-46). Any item (i, j) with $i \in I^0$ that is such that $(w_i\, m_{ij} + s_i) > C$ cannot be in the solution of the (x, y)-formulation of the residual problem. Therefore, it can be removed from the z-formulation (43-46) before computing its LP value.

The LP-relaxation of (43-46) admits a greedy solution. Items whose class is in I^0 are sorted by decreasing ratio (27), while those whose class is in I^1 are sorted by decreasing ratio (42) and the two sorted lists are merged to define the greedy ordering. Then, items are considered in that order and taken into the LP solution while there remains some knapsack capacity and some class multiplicity. Thus, the "Compute UB" step of our branch-and-bound is implemented so as to compute the upper bound of Proposition 3 adapted to the residual problem: for items whose class is in I^1 we use ratio (42) instead of (27), capacity consumption $w_i\, m_{ij}$ and profit p_{ij}. This upper bound will have to be recomputed after each fixing to zero as in the Horowitz and Sahni algorithm, but also after each fixing to 1 of a y variable, as it modifies the partition I^0, I^1, and hence the partition of the items.

Our branch-and-bound algorithm is presented below in Algorithm 2. Items k denotes the next item in the greedy ordering for the current residual problem (43-46). It is obtained dynamically from two static lists that are sorted a priori. Each item (i, j) is represented twice, once in list L^0 as an item whose class is not setup and once in list L^1 as an item whose class is setup (defining specific p_k, w_k, and m_k value as described in Algorithm 2). Each list is appropriately sorted by decreasing ratio $\frac{p_k}{w_k}$. One moves forward in list L^O (resp. L^1) ignoring items whose class is setup (resp. not setup) and a function named *next* compares the next candidate from the two lists and returns that with the largest ratio $\frac{p_k}{w_k}$. However the cursor in each list must be *reset* after each modification of a class setup. The order in which items have been considered is memorized in a data structure to enable backtracking through a call to function *prev*.

Initialization: Let $N = \sum_i n_i$. Each item (i, j) is duplicated. The first copy is defined by $m_k = m_{ij}$, $w_k = (w_i + \frac{s_i}{b_i})\, m_{ij}$, and $p_k = p_{ij} - \frac{f_i}{b_i} m_{ij}$, $k = 1, \ldots, N$; and L^0 is the list of the first copies sorted in decreasing order of their ratio (27). The second copies are defined by $m_k = m_{ij}$, $w_k = w_i\, m_{ij}$, and $p_k = p_{ij}$, $\forall k$; and L^1 is the list of the second copies sorted in decreasing order of their ratio (42). Let $w_{\min} = \min_k\{w_{i_k}\, m_k\}$; $C = W$; $C_i = b_i$, $\forall i$; $Z = 0$; $x = y = 0$; $INC = 0$; k is the first item of L^0. Let $next(k)$ be a subroutine that returns the best item next to k in greedy order for the current residual problem (43-46) by proper extraction from either L^0 or L^1, *reset* reactualizes it after each class setup modification, and *prev* returns the item fixed at the previous BaB node.

Compute UB: Let $U = Z$; $K = C$; $K_i = C_i$, $\forall i$; and let $l = k$.

Step A: While $(l \le N)$ and $(w_l \le K)$ and $(m_l \le K_{i_l})$ {
If $(w_{i_l}\, m_{i_l, j_l} + s_{i_l}(1 - y_{i_l}) > C)$, {
$l = next(l)$, goto Step A. }
$U {+}{=}\, p_l$; $K {-}{=}\, w_l$; $K_{i_l} {-}{=}\, m_l$; $l = next(l)$.}

Step B: If $(l > N)$, goto Test Pruning.

Step C: If $((m_l > K_{i_l})$ and $(w_l\, \frac{K_{i_l}}{m_l} \le K))$, {
$U {+}{=}\, p_l\, \frac{K_{i_l}}{m_l}$; $K {-}{=}\, w_l\, \frac{K_{i_l}}{m_l}$; $K_{i_l} = 0$;
$l = next(l)$; go to Step A }.

Step D: /*$(w_l > K)$ or even $(w_l\, \frac{K_{i_l}}{m_l} > K)$*/
$U {+}{=}\, p_l\, \frac{K}{w_l}$.

Test Pruning: If $(U \le INC)$, goto Backtracking.

Forward Move: While $(k \le N)$ and $(w_{i_k}\, m_k + s_{i_k}\,(1 - y_{i_k}) \le C)$ and $(m_k \le C_{i_k})$, do {
$x_{(i_k, j_k)} = 1$; $C {-}{=}\, w_k$; $C_{i_k} {-}{=}\, m_k$; $Z {+}{=}\, p_k$;
if $(y_{i_k} == 1)$, then $k = next(k)$;
else $\{y_{i_k} = 1$; $C {-}{=}\, s_{i_k}$; $Z {-}{=}\, f_{i_k}$; *reset*; goto Compute UB; } }
If $(k > N)$ or $(C < w_{\min})$, /* leaf node */ goto Record Incumb.

Set item to 0: /* $((w_{i_k}\, m_k + s_i\,(1 - y_{i_k}) > C)$ or $(m_k > C_{i_k})$ */
$k = next(k)$. If $(k > N)$, goto Record Incumb. Else, goto Compute UB.

Record Incumb: If $(Z > INC)$, then $INC = Z$ and record (x, y).

Pre-backtrack: If $(k > N)$, $\{k = prev$; if $(x_{(i_k, j_k)} == 1)$, **WithdrawItem**(k); }

Backtracking: Do $\{k = prev;\}$ while $((x_{(i_k, j_k)} == 0)$ and $(k \ge 1))$.
If $(k == 0)$, STOP.
/*$(x_{(i_k, j_k)} == 1)$ */ **WithdrawItem**(k);
$k = next(k)$. Go to Compute UB.

Algorithm 2. BaB for MBKS when $f_i \geq 0$ and $a_i = 0\ \forall i$.

In Algorithm 2, i_k denotes the class index of item k, j_k its index within the class, and Z denotes the current profit. In "Forward Moves", we set z_k's to one in formulation (43-46), which amount to fixing $x_{(i_k,j_k)}$ to 1 in MBKS. If $i_k \in I^0$, we set $y_{i_k} = 1$ and we update C, Z, and the resulting residual problem formulation (placing $i_k \in I^1$), and we reset the greedy ordering of the remaining items. Then, we return to the "Compute UB" step. Otherwise, we continue our sequence of fixing z_k's to one. This is repeated while there remains some class capacity and some knapsack capacity to insert further items, i.e., while $C \geq w_{\min}$, where $w_{\min}$ is the smallest item weight. Otherwise, the next item is set to zero and the dual bound must be computed. A leaf node is reached when the knapsack is filled or there are no more items to consider. In the latter case, as the branch $z_n = 1$ has been explored, the branch $z_n = 0$ does not need to be explored as it is dominated. "Backtracking" must insure that the class setup is set to zero when the last positive item of that class is set to zero. This is done in the WithdrawItem(k) subroutine of Algorithm 3. Within this branch-and-bound procedure lies a primal greedy heuristic that could be used independently. Note that alternative primal heuristics can be developed that make use of decomposition of the problem into knapsack subproblems for each class.

```
Let $x_{(i_k,j_k)} = 0$; $C += w_k$; $C_{i_k} += m_k$; $Z -= p_k$.
If ($C_{i_k} == b_i$), do {
$y_{i_k} = 0$; $C += s_{i_k}$; $Z += f_{i_k}$; $reset$; }.
```

Algorithm 3. **WithdrawItem(k)** of the BaB for MBKS.

3. NUMERICAL TESTS

Our purpose in this paper was not to develop the most effective algorithm for the knapsack problem with setups, but instead, to show that the properties on which the knapsack algorithms rely can be extended to the case with setups. Thus, the focus of our numerical section is on testing our extensions of Horowitz and Sahni branch-and-bound algorithm. We compare our branch-and-bound algorithms to a standard MIP solver (Xpress-MP in our tests) and to the best approach of the previous literature for knapsack problems with setups, namely the conversion into a multiple choice knapsack with pseudo-polynomial size followed by the application of a specialized algorithm for the multiple choice knapsack. We must insist on the fact that the algorithms of this paper have not been developed so as to incorporate the latest techniques for improved efficiency outlined in the introduction (Pisinger & Toth 1998). This makes comparison to specialized solver for multiple choice knapsack somewhat bias.

For the tests on model CKS, we generate 10 random instances with $n \in \{1000, 5000, 10000\}$ and $W = 100 * n$. We want approximately 20% of the items in the solution, hence on average $(s_i + b_i w_i) = \frac{W}{0.2n}$. Thus for each i, $s_i + w_i$ are generated from an uniform distribution in interval $[\frac{0.35W}{0.2n}, \frac{0.65W}{0.2n}]$, hence $s_i + w_i$ is on average equal to $\frac{0.5W}{0.2n}$ and b_i must be approximately equal to $\frac{\frac{W}{0.2n} - s_i}{w_i}$. We set $s_i = \alpha\, w_i$, where α is a parameter in $[0, 4]$ and draw b_i uniformly in $[1, \lfloor \frac{\frac{1.5W}{0.2n} - s_i}{w_i} \rfloor]$. The profit p_i is uniformly distributed in $[(1-\beta)w_i, (1+\beta)w_i]$ where β is a parameter in $[0, 1]$ measuring the correlation between item weight and profit, and f_i is uniformly distributed in $[(1-\beta)s_i, (1+\beta)s_i]$.

The results are presented in the Table 1. The first column indicates the chosen parameters. The following columns give the average computation time obtained using the specialized multiple-choice knapsack dynamic program solver of (Pisinger 1995) ("MCKP") after application of transformation (12) and the average time of our Branch-and-Bound algorithm ("BB"). The average computation time for the standard MIP solver (XPress-MP 2001) is 6 times greater than the average time "MCKP". Note that if one uses a basic dynamic program (which would be more in line with our basic Branch-and-Bound solver), the computation time for $n1000 - \alpha 2 - \beta 0.5$ is 11m35s16t and for $n5000 - \alpha 2 - \beta 0.5$ is 3h40m26s49t.

param	MCKP	BB
$n1000 - \alpha 2 - \beta 0.5$	2s17t	5t
$n5000 - \alpha 2 - \beta 0.5$	48s77t	55t
$n5000 - \alpha 2 - \beta 0.2$	50s68t	1s1t
$n5000 - \alpha 2 - \beta 0.8$	47s70t	38t
$n5000 - \alpha 4 - \beta 0.5$	36s63t	44t
$n10000 - \alpha 2 - \beta 0.5$	3m12s85t	1s28t

Table 1. Computation time for CKS

For the tests on model MBKS, we generate 10 random instances with a number of item classes $n \in \{10, 50, 100\}$, $W = 1000 * n$ and we impose that 50% of classes have a positive setup ($f_i > 0$, $s_i > 0$). For each class i, $s_i + w_i$ are generated from an uniform distribution in interval $[\frac{W}{20n}, \frac{W}{2n}]$ (so $s_i + w_i \in [50, 500]$), with $s_i = \alpha\, w_i$ where α is a parameter in $[0, 4]$ (α is set to 0 if the class has no positive setup); $a_i = 0$, b_i is uniformly distributed in $[\lfloor \frac{W - s_i}{2w_i} \rfloor, \lfloor \frac{W - s_i}{w_i} \rfloor]$, n_i is uniformly distributed in $[b_i, 5b_i]$ (which can result in large values of $n_i \in [200, 15800]$), f_i is uniformly distributed in $[(1-\beta)s_i, (1+\beta)s_i]$ where β is a parameter in $[0, 1]$ measuring the correlation between weight and profit. For each item j in class i, m_{ij} is generated in interval $[1, \frac{b_i}{2}]$ and $w_{ij} = m_{ij} w_i$. We try to have ratios $r_{ij} = \frac{p_{ij}}{m_{ij} w_i}$ that take different values in $[1-\beta, 1+\beta]$ for items of the same class and also between items of different classes (we generate g_{ij} in $[0, M]$ where M is a big parameter and set $r_{ij} = 1 - \beta + \beta * g_{ij} * \frac{2}{M}$). Then, we compute $p_{ij} = \lfloor m_{ij} w_i r_{ij} \rfloor$.

The results for MBKS are presented in the Table 2. The last two columns give the average time to apply transformation (19) and then the specialized multiple-choice knapsack dynamic program solver of (Pisinger 1995) ("MCKP") and the average time taken by our Branch-and-Bound algorithm ("BB"). For (XPress-MP 2001), the gap after 3h was 0.05% for $n = 50$. This approach was not applied on instances with $n = 100$.

In average BB is faster than MCKP, but we observe that for 10% of instances, MCKP is faster. For the correlated problem instances (with $\beta = 0.2$), BB computing times exhibits large variation around the average value (BB takes more than 5h for one instance and less than 20s for two others). While, the computation time for MCKP is similar for every instance in a group. In the MCKP approach, the bottleneck is the time spent at transforming the problem into a multiple choice knapsack (taking more than 95% of the computing time). One has to solve a "all-capacities" knapsack problem for each class, computing the best profit (20) for each multiplicity $M = 0, \cdots, b_i$. (The "all-capacities" knapsack problem is to the knapsack problem what the "all-pairs" shortest path problem is to the shortest path problem, see (Kellerer et al. 2004), section 1.3). To do this, there is no much better algorithm than a a basic dynamic program (which is what we use).

parameters	MCKP	BB
$n10 - \alpha 2 - \beta 0.5$	3s34t	14t
$n50 - \alpha 2 - \beta 0.5$	6m35s64t	2m35s48t
$n100 - \alpha 2 - \beta 0.5$	1h10m1s33t	20m4s17t
$n100 - \alpha 2 - \beta 0.2$	1h8m40s22	1h6m52s15t
$n100 - \alpha 2 - \beta 0.8$	1h9m23s15t	22m18s6t
$n100 - \alpha 4 - \beta 0.5$	3h6m1s46t	1h29m47s78t

Table 2. Computation time for MBKS

4. CONCLUSION

The paper discusses various reformulations of knapsack problems with setups and presents specialized branch-and-bound procedures extending the standard algorithm for the knapsack problem. Numerical experimentation shows that the latter are competitive approaches to knapsack problems with setups. The analysis of model CKS and MBKS can be extended to their integer counterpart. The greedy enumeration scheme and greedy dual bounds of our branch-and-bound procedures could be exploited to developed hybrid dynamic programming approaches in future work in the line of the best performing approaches for the standard knapsack problem.

References

Balas, E. & Zemel, E. (1980). An algorithm for large zero-one knapsack problems, *Operations research* **28**: 1130–1154.

Chajakis, E. & Guignard, M. (1994). Exact algorithms for the setup knapsack problem, *INFOR* **32**: 124–142.

Dantzig, G. (1957). Discrete variable extremum problems, *Operations research* **5**: 266–277.

Goemans, M. X. (1989). Valid inequalities and separation for mixed 0-1 constraints with variable upper bounds, *Oper. Res. Lett.* **8**: 315–322.

Jans, R. & Degrave, Z. (2004). Improved lower bounds for the capacitated lot sizing problem with setup times, *Oper. Res. Letters* **32**: 185–195.

Johnson, E. & Padberg, M. W. (1981). A note on the knapsack problem with special ordered sets., *Oper. Res. Lett.* **1**: 18–22.

Kellerer et al., H. (2004). *Knapsack Problems*, Springer, Berlin, Germany.

Martello, S. & Toth, P. (1990). *Knapsack Problems: Algorithms and computer Implementations*, Wiley-Interscience Series in Discrete Mathematics and Optimization.

Nemhauser, G. & Wolsey, L. (1988). *Integer and Combinatorial Optimization*, John Wiley & Sons, Inc.

Pisinger, D. (1995). A minimal algorithm for the multiple-choice knapsack problem, *European Journal of Operational Research* **83**: 394–410.

Pisinger, D. & Toth, P. (1998). Knapsack problems, *Handbook of Combinatorial Optimization* **1**: 299–428.

Vanderbeck, F. (1999). Computational study of a column generation algorithm for bin packing and cutting stock problems, *Mathematical Programming* **86**: 565–594.

Vanderbeck, F. (2002). Extending dantzig's bound to the bounded multi-class binary knapsack problem, *Mathematical Programming* **94**: 125–136.

XPress-MP (2001). *User guide and Reference Manual*, Release 12, Dash Optimization.