Knapsack Problems with Setups

Sophie Michel, Nancy Perrot, François Vanderbeck

Knapsack Problems with Setups

S. Michel (1), N. Perrot (2) and F. Vanderbeck (3)

(1) ISEL-LMAH, Université du Havre, michels@univ-lehavre.fr

(2) France-Télécom, division R&D, nancy.perrot@orange-ftgroup.com

(1) Institut de Mathématiques de Bordeaux, Université Bordeaux 1, fv@math.u-bordeaux1.fr

November 27, 2008

Abstract

Knapsack problems with setups find their application in many concrete industrial and financial problems. Moreover, they also arise as subproblems in a Dantzig-Wolfe decomposition approach to more complex combinatorial optimization problems, where they need to be solved repeatedly and therefore efficiently. Here, we consider the multiple-class integer 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. A special case is the bounded integer knapsack problem with setups where each class holds a single item and its continuous version where 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 with setups. 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.

Keywords: Knapsack problem, fixed cost, setup, variable upper bound, branch-and-bound.

The Multiple-class Integer Knapsack problem with Setups (MIKS) is defined as follows. The knapsack has capacity W. There are n item classes, indexed by $i = 1, \ldots, n$, with associated setup cost, $f_i \in \mathbb{R}$, and setup capacity consumption, $s_i \in \mathbb{R}_+$. Each class is made of its own items (i, j) for $j = 1, \ldots, n_i \in \mathbb{N}$ with associated profit $p_{ij} \in \mathbb{R}$

and upper bound $u_{ij} \in I\!N$. 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 $w_{ij} \leq W$). Moreover, there are lower and upper bounds, $a_i \leq b_i \in I\!N$, on the total multiplicity of items that may 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 MIKS takes the form:

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

[MIKS] s.t.

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

$$a_i\, y_i \leq \sum_{j=1}^{n_i} m_{ij}\, x_{ij} \leq b_i\, y_i \qquad \text{for } i = 1, \ldots n \tag{3}$$

$$x_{ij} \leq u_{ij}\, y_i \qquad \text{for } i = 1, \ldots n \text{ and } j = 1, \ldots, n_i \tag{4}$$

$$x_{ij} \in I\!N \qquad \text{for } i = 1, \ldots, n \text{ and } j = 1, \ldots, n_i \tag{5}$$

$$y_i \in \{0,1\} \qquad \text{for } i = 1, \ldots, n\ , \tag{6}$$

where x_{ij} denotes the number of copies of item j that are chosen within class i and $y_i = 1$ iff class i is setup. The main developments of the paper are made under restrictive assumptions that simplify the characterization of extreme solutions of the continuous relaxation. In our branch-and-bound algorithms, we shall assume that fixed costs are non-negative:

Assumption 1 *(**restrictive**)* $f_i \geq 0$ *for all* i,

and that there are no class lower bounds:

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

Model MIKS has several interesting special cases. In a binary model, denoted MBKS, $x_{ij} \in \{0,1\}\ \forall ij$. When each class holds a single item, i.e. $n_i = 1\ \forall i$, MIKS gives rise to the integer knapsack problem with setups (IKS):

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

[IKS] s.t.

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

$$a_i\, y_i \leq x_i \leq b_i\, y_i \qquad \text{for } i = 1, \ldots n \tag{9}$$

$$x_i \in I\!N \qquad \text{for } i = 1, \ldots, n \tag{10}$$

$$y_i \in \{0,1\} \qquad \text{for } i = 1, \ldots, n\ . \tag{11}$$

Further relaxing the integrality constraint on x_i gives rise to the continuous knapsack problem with setups denoted CKS. Observe that, when $a_i = w_i = 0$ and $b_i = 1\ \forall i$, all the above models boil down to a standard binary knapsack problem (f.i., for IKS, as $w_i = 0$, it is optimal to set $x_i = b_i$ if $p_i \geq 0$). Hence, these models are at least as hard as the standard binary knapsack problem.

Model CKS arises as a sub-problem in *capacitated multi-item lot sizing problem* when setting up the machine for the production of an item requires setup time and cost: once the demand covering constraints are dualized the problem decomposes into a CKS problem for each period. Then, w_i, f_i and s_i are respectively the processing time, the setup cost and the setup time for item i, p_i is the difference between the dual value for covering item i demand and its production cost, W is the machine capacity of that period, and $[a_i, b_i]$ defines an interval of valid production levels for item i. A lower bound a_i on production may arise due to a business rule to amortize setup or due to technical constraints that translate into a minimum batch size. Note that, for this application, fixed cost naturally satisfy Assumption 1. Goemans [4] studied the structure of the CKS polyhedron, derived facet defining inequalities and proposed a heuristic separation procedure.

Model IKS is encountered as a sub-problem in solving the cutting stock problem by branch-and-price [10]. When using branching constraints that enforce integrality of the number of cutting patterns that involve a given item i, the knapsack subproblem must be modified to include a fixed cost. Nonzero lower bounds, a_i, may arise in the course of a rounding heuristic when, in order to achieve a feasible solution to the residual problem, one must impose a minimum production level in remaining cutting patterns. A special case of model IKS was studied by Sural et al. [13]: they assume $f_i = 0$ and $w_i = 1$ for all i. Then, they show how to generalize the Dantzig's upper bound and they propose a primal heuristic. Both are used for setting up a depth-first search branch-and-bound algorithm. Their motivations for studying this model were applications in finance and in machine scheduling. The assumption $w_i = 1$ is not restrictive for IKS but assuming $f_i = 0$ is restrictive. However, the Dantzig's upper bound can be generalized to the case $f_i \neq 0$ as shown in this paper.

Model MBKS also arises as a sub-problem in a branch-and-price approach to the cutting stock problem. Most fractional solutions can be cut-off by bounding the number of cutting patterns that involves a specific binary variable of the knapsack subproblem in its 0-1 form [14] [1]. When a branching constraint is added that restricts the number of

[1] The pricing problem is normally an integer knapsack problem: $\max\{\sum_i p_i x_i : \sum_i w_i\, x_i \leq W, x_i \leq b_i, x_i \in I\!N\ \forall i\}$. A standard 0-1 transformation consists in introducing $n_i = \lfloor \log_2 b_i \rfloor + 1$ binary items (i, j) for each integer item i with multiplicity $m_{ij} = 2^{j-1}$ for $j = 1, \ldots, n_i - 1$ and $m_{i\,n_i} = b_i - \sum_{j=1}^{n_i - 1} m_{ij}$.

columns involving a specific binary item (i, j), the item dual price is modified in the pricing problem, leading to objective coefficients $p_{ij} \neq m_{ij}\, p_i$. If one combines such branching with that on the number of cutting pattern involving a specific item i, then fixed costs are also nonzero, giving rise to model MBKS. Observe that if the binary decomposition of the pricing problem is not done a priori but dynamically as branching constraints are introduced on specific binary items (see [16] for details), then model MIKS must be used.

The special case of model MBKS with no setups is treated in [15]. 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 [3] and Balas and Zemel [1] 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.

Variants of model MBKS are considered in the literature. Chajakis and Guignard [2] consider a model where $m_{ij} = 1\ \forall ij$, class bound constraints are replaced by $\sum_{j=1}^{n_i} x_{ij} \geq y_i\ \forall i$, and the item weights are not restricted to be a multiple of a class weight (hence, the result of [15] concerning a polynomial greedy solution of the LP relaxation does not extend to the model of [2]). The application that motivated their study is the scheduling of parallel unrelated machines with setups where this knapsack problem arises as a subproblem. They propose and test two approaches: either a dynamic programming solver or a two-stage approach. In the latter, the problem is transformed into a standard multiple choice 0-1 knapsack problem and solved either by dynamic programming or branch-and-bound. The transformation consists in defining a "pseudo-item" for each dominant feasible solutions within a class. These dominant solutions are the states of a dynamic program for solving the binary knapsack problem defined on a single class. There is a pseudo-polynomial number of them. They found that, for correlated instances with small knapsack capacity (they assume integer data and consider $W \leq 500$), the direct dynamic programming approach is the most efficient. When the number of families or the knapsack capacity increases, the two-stage approach using branch-and-bound for the second stage is the most efficient.

The variant of model MBKS that is considered by Jans and Degraeve [5] is simpler. They also assume $m_{ij} = 1\ \forall ij$ and $w_{ij} \neq m_{ij}\, w_i$, but their model has $b_i = 1\ \forall i$. This

However, this transformation introduces multiple 0-1 representation of a given integer solution. The alternative 0-1 decomposition proposed in [15] is to set $m_{i\,n_i} = 2^{n_i}$. Then, one needs to introduce explicit class upper bounds: $\sum_{j=1}^{n_i} m_{ij}\, x_{ij} \leq b_i\ \forall i$. It guarantees a unique representation of each integer solution. This is essential to avoid the enumeration of symmetric solutions. A numerical comparison of branch-and-bound approaches based on the standard 0-1 transformation versus the multiple class model is presented in [15]; it shows the increase branch-and-bound tree size that may result from ignoring this symmetry.

subproblem arises in a decomposition of a multi-item lot-sizing problem per time period. It takes the form

$$\max\{\sum_i(\sum_j p_{ij}\, x_{ij} - f_i\, y_i) : \sum_i(\sum_j w_{ij}\, x_{ij} + s_i\, y_i) \leq W, \sum_j x_{ij} \leq y_i\ \forall i,\ x_{ij}, y_i \in \{0,1\}\}$$

where setting $x_{ij} = 1$ amounts to producing item i so as to cover demands from the current period t up to $t + j - 1$. Moreover, their application assumes positive fixed cost $f_i \geq 0$. In this special case, feasible solutions verify $x_{ij} = x_{ij}\, y_i$. Therefore, their model reduces to a standard multiple choice knapsack problem

$$\max\{\sum_{ij} \tilde{p}_{ij} z_{ij} : \sum_{ij} \tilde{w}_{ij} z_{ij} \leq W, \sum_j z_{ij} \leq 1\ \forall i,\ z_{ij} \in \{0,1\}\}\ ;$$

where $z_{ij} = x_{ij}\, y_i$, $\tilde{p}_{ij} = p_{ij} - f_i$, and $\tilde{w}_{ij} = w_{ij} + s_i$. Jans and Degraeve [5] developed their own branch-and-bound algorithm for it.

The present paper proposes an analysis of models CKS and MBKS. Their extensions to models IKS and MIKS are also discussed. 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 or dynamic programs (see [8] pages 30-31 or [9] 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 [6] or class bounds [15]. The formulation are polynomial in size while previously proposed reformulations such as that of [2] 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 extensions 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.

Dynamic programming recursion can also be derived for these knapsack models with setups. They are straightforward extension of results for the standard knapsack problem. We present them for the sake of establishing the complexity of the various models. Of course, the improvements of the basic techniques for knapsack problems: linear time computation of upper bound [1], improved variants of Dantzig's bounds, improved dynamic recursion (f.i. using bounds to eliminate intermediate states, exploiting the core, or so-called balanced enumeration), more sophisticated branch-and-bound (f.i. making use of dominance rules) and hybrid methods [12] could also be reviewed for the case of problems

with setups, but this is beyond our scope. We purposely keep the presentation to basic techniques to get a point across: to demonstrate that knapsack problems with setups are not much harder than standard knapsack problems.

1 The continuous knapsack problem with setups

In model CKS, the item selection variables, x, are allowed to take continuous values. Hence, the formulation is:

$$\max \quad \{ \quad \sum_{i=1}^{n}(p_i\, x_i - \sum_{i=1}^{n} f_i\, y_i) \;\; : \sum_{i=1}^{n}(w_i\, x_i + s_i\, y_i) \leq W \;, \\ a_i\, y_i \leq x_i \leq b_i\, y_i \;\; \forall i \;, \quad x_i \geq 0 \;\; \forall i \;, \quad y_i \in \{0,1\} \;\; \forall i \;.\} \tag{12}$$

Here, bounds a_i and b_i are not necessarily integer, i.e. a_i and $b_i \in I\!R^+$, $\forall i$. Assumption 2 can be made without loss of generality. Indeed, if $a_i > 0$ for some i, one can transform the problem as follows: let $a'_i = 0$, $b'_i = b_i - a_i$, $s'_i = s_i + w_i\, a_i$, $f'_i = f_i - a_i p_i$; its solution (x'_i, y'_i) translates into a solution for the original problem as follows: $x_i = (a_i + x'_i)\, y'_i$ and $y_i = y'_i$. Moreover, we can assume

Assumption 3 *(**without loss of generality**)* $p_i \geq 0$ *for all* i.

Indeed, if $p_i < 0$ for some i, $x_i = 0$ in any optimal solution. Also, we have

Assumption 4 *(**without loss of generality**)* $f_i \leq 0$ *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 $x_i \geq \frac{f_i}{p_i}$, because a solution where $0 < x_i < \frac{f_i}{p_i}$ can be improved by setting $x_i = y_i = 0$. Thus, $\frac{f_i}{p_i}$ can be interpreted as a lower bound, a_i, which can be eliminated as explained above by re-setting $b'_i = b_i - \frac{f_i}{p_i}$, $s'_i = s_i + w_i\, \frac{f_i}{p_i}$, $f'_i = f_i - \frac{f_i}{p_i} p_i = 0$. Finally, one can also assume

Assumption 5 *(**without loss of generality**)* $w_i = 1$ *for all* i.

Otherwise, one can make a change of variables $x'_i = w_i\, x_i$ and redefine the associated coefficients: $p'_i = \frac{p_i}{w_i}$, $b'_i = w_i\, b_i$.

Under Assumption 5, problem CKS can be reformulated as a multiple choice knapsack problem. When data are integer, i.e., when $a_i, b_i, s_i \in I\!N$ $\forall i$ and $W \in I\!N$, observe that the continuous variables x take integer value in any feasible extreme solutions. Therefore, within each class, one can optimize the use level x by enumeration. Hence, the problem can be reformulated as a multiple choice knapsack problem:

$$\max\{\sum_i \sum_{x=a_i}^{b_i} (p_i\, x - f_i)\, \lambda_x^i : \; \sum_i \sum_{x=a_i}^{b_i} (w_i x + s_i)\, \lambda_x^i \leq W, \; \sum_{x=a_i}^{b_i} \lambda_x^i \leq 1 \; \forall i, \; \lambda_x^i \in \{0,1\} \; \forall i, x\} \;, \tag{13}$$

where $\lambda_x^i = 1$ iff $x_i = x$. This formulation has pseudo-polynomial size but it leads to a possible solution approach using a solver for the multiple choice knapsack problem, which we shall use in numerical comparison to our algorithm.

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. Thus, our model is given by (12) where $a_i = 0$, $p_i \geq 0$ and $f_i \leq 0$.

1.1 Characterizations of optimal solutions

Some properties of optimal solutions are used to develop bounding procedure or dynamic programs. To analyse the structure of extreme solutions, note that if one fixes y to $\tilde{y} \in \{0,1\}^n$, the problem reduces to a continuous knapsack problem that admits a greedy solution.

Observation 1 *(Dantzig, [3])*
Let $I = \{i : \tilde{y}_i = 1\}$ and $\tilde{W} = W - \sum_{i \in I} s_i$. The resulting problem in the x variables is:

$$[CKP(\tilde{y})] \equiv \max\{\sum_{i \in I} p_i\, x_i :\ \sum_{i \in I} w_i\, x_i \leq \tilde{W}, 0 \leq x_i \leq b_i\ \forall i \in I\} \tag{14}$$

An optimal solution is obtained as follows. Assume an indexing of the items in I such that

$$\frac{p_1}{w_1} \geq \frac{p_2}{w_2} \geq \ldots \geq \frac{p_{|I|}}{w_{|I|}}\ .$$

Let c be the index for which $\sum_{i<c} b_i\, w_i < \tilde{W}$ but $\sum_{i \leq c} b_i\, w_i \geq \tilde{W}$. Then, set

$$x_i = b_i \qquad \text{for } i < c, \tag{15}$$

$$x_c = \frac{\tilde{W} - \sum_{i<c} w_i\, b_i}{w_c}, \tag{16}$$

$$x_i = 0 \qquad \text{otherwise.} \tag{17}$$

This standard observation yields to the conclusion that $x_i \in \{0, b_i\}$ for all i but one. The same observation was made in [13] for the case $f_i = 0\ \forall i$. Moreover, in that case, [13] adds that the item with $0 < x_i < b_i$, if any, has the smallest ratio $\frac{p_i}{w_i}$ of non zero items. This property generalizes trivially to our case. Let us explicitly state this characterization of extreme solutions to CKS for easy reference.

Observation 2 *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 $J \subseteq \{k :\ f_k < 0 \text{ and } \frac{p_c}{w_c} \geq \frac{p_k}{w_k}\}$.

Indeed, if $y_i = 1$ and $x_i = 0$ but $f_i = 0$, y_i can feasibly be set to zero without decreasing the profit. The rest of the observation results from the following procedure. Let (x^*, y^*) be an optimal solution to problem CKS. Let us fix the y variables to their optimal 0-1 values y^*. From observation 1, we derive an associated solution x' defined by (15-17). Solution (x', y^*) is optimum and has the above form.

The continuous relaxation of model CKS is given by (12) where $y_i \in \{0, 1\}$ is replaced by $y_i \in [0, 1]\ \forall i$. Then, the value of the class i solution in the LP solution can be anything in the convex hull of extreme solutions (i), (iii) and (iv) of Observation 2 (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:

$$\max \quad \sum_{i=1}^{n} ((p_i\, b_i - f_i) z_i^b - f_i z_i^f) \tag{18}$$

$$\sum_{i=1}^{n} ((w_i\, b_i + s_i)\, z_i^b + s_i\, z_i^f) \leq W \tag{19}$$

$$z_i^b + z_i^f \leq 1 \qquad \text{for } i = 1, \ldots, n \tag{20}$$

$$z_i^b \in [0, 1] \qquad \text{for } i = 1, \ldots, n \tag{21}$$

$$z_i^f \in [0, 1] \qquad \text{for } i = 1, \ldots, n \tag{22}$$

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 2. Inversely, setting $z_i^b = 0$ and 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 has a profit to capacity consumption ratio that is worse than those that we considered. The mapping from a solution z of (18-22) to a solution (x, y) for the LP relaxation of CKS is given by $x_i = b_i\, z_i^b$ and $y_i = z_i^b + z_i^f$.

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 (18-22).*

The latter is known to admit a greedy solution [6]. Hence, this result extends to our model with setups. Moreover, one can characterized the conditions under-which continuous solutions with $z_i^f > 0$ are dominated:

Observation 3 *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 (18-22) where* $z_i^f = 0$.

Note that Assumption 1 implies $\frac{p_i\, b_i - f_i}{w_i\, b_i + s_i} \geq \frac{-f_i}{s_i}\ \forall i$. Let us write the greedy LP solution. Under the assumption of Observation 3, the special ordered set (SOS) constraints (20) are redundant and (18-22) reduces to a standard binary knapsack LP relaxation:

Observation 4 *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*

$$\frac{(p_1\, b_1 - f_1)}{(w_1\, b_1 + s_1)} \geq \frac{(p_2\, b_2 - f_2)}{(w_2\, b_2 + s_2)} \geq \ldots \geq \frac{(p_n\, b_n - f_n)}{(w_n\, b_n + s_n)}\ . \tag{23}$$

and setting

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

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 (18-22) in the (x, y) variables (when $z_i^f = 0\ \forall i$).

Now consider the integer version of formulation (18-22) where $z_i^b, z_i^f \in \{0, 1\}\ \forall i$. It is important to note that, although model CKS and formulation (18-22) have the same LP solution, the integer version of (18-22) is not equivalent to model CKS. Indeed, a solution where case (ii) of Observation 2 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 [9] 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 problems. 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 (23) 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 to the value of the optimal solution of CKP($\tilde{y}$) defined in Observation 1. This ensures that the solution we build obeys the characterization of Observation 2. The so called "forward moves" are sequences of branches where a z_i variable is fixed to 1. Forward moves are interspersed with "fixing-to-zero" branches where a z_i variable is set to zero which we interpret as fixing the associated x_i to zero for the primal bound. 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 (23). For each item i, three cases are considered: (a) $z_i = 1$ (which, for the primal bound, corresponds to setting $x_i = b_i$ and $y_i = 1$), (b) $y_i = 1$, and x is free, (c) $z_i = 0$. However, Observation 2 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 according to Observation 1. 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 Table 1 (in a pseudo-language where we make use of some C++ notations). In our notations, INC is the value of the incumbent solution, Z is the value of the current partial solution, U is the current dual bound, C is the current residual capacity and v is the return value of the routine solving $CKP(y)$. The greedy heuristic for the primal problem used in this branch-and-bound procedure can be used independently. It is given in Table 2.

Table 1: Branch-and-Bound algorithm for CKS under Assumption 1

Initialization:	Sort items according to (23).
	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.

Table 2: Primal heuristic for CKS under Assumption 1

Step A: Sort items according to (23). Let $C = W$; $x = y = 0$; incumbent $= 0$; $i = 1$.
Step B: While $(C > w_i b_i + s_i)$ and $(i \leq n)$, do { $x_i = b_i$; $y_i = 1$; $C{-}{=} (w_i\, b_i + s_i)$; $i{+}{+}$.}
Step C: If $(i > n)$, { record solution (x, y); update the incumbent; and STOP.}
Step D: Let $y_i = 1$; solve $CKP(y)$ according to Observation 1 and update the incumbent.
Step E: Let $y_i = 0$; $i = i + 1$; and go to Step B.

1.3 Dynamic programming recursion for the CKS problem

Here, we assume integer data: s_i, w_i, and $W \in I\!N$. To solve problem CKS, one can implement a dynamic programming recursion that computes the best solution of the form described in Observation 2. Let the item indexing be such that

$$\frac{p_1}{w_1} \geq \frac{p_2}{w_2} \geq \ldots \geq \frac{p_n}{w_n}\,.$$

Let $V^k(C)$ be the best value that can be accumulated using items 1 up to k at level $x_k \in \{0, b_k\}$ and using a capacity C. Setting $V^0(C) = 0$ for all $0 \leq C \leq W$, the values $V^k(C)$ can be computed recursively:

$$V^k(C) = \max\{\underbrace{V^{k-1}(C)}_{x_k=y_k=0}, \underbrace{V^{k-1}(C - w_k\, b_k - s_k) + p_k\, b_k - f_k}_{x_k=b_k,\, y_k=1}\}$$

for all $0 \leq C \leq W$ and $k = 1, \ldots, n$. This requires $O(nW)$ operations. Let $U^k(C)$ be the best value that can be accumulated using items n down to k at level $x_k = 0$ but with $y_k \in \{0, 1\}$ and using a capacity C. Setting $U^{n+1}(C) = 0$ for all $0 \leq C \leq W$, the values $U^k(C)$ can be computed recursively:

$$U^k(C) = \max\{\underbrace{U^{k+1}(C)}_{y_k=0}, \underbrace{U^{k+1}(C - s_k) - f_k}_{x_k=0,\, y_k=1}\}$$

for all $0 \leq C \leq W$ and $k = n, \ldots, 1$. This requires another $O(nW)$ operations. Then, the optimal solution value for CKS can be obtained as

$$\max\{V^n(W),\, U^1(W),$$

$$\max_{1\leq k\leq n}\{\ \max_{C_1,C_2:W-b_k\, w_k<C_1+C_2+s_k<W}\{\ V^{k-1}(C_1)+(W-C_1-C_2-s_k)\frac{p_k}{w_k}-f_k+U^{k+1}(C_2)\}\}\ \}\,.$$

These latter computations require another $O(nW^2)$ operations. The associated (x, y) solution is easy to obtain.

Under Assumption 1, $U^k(C) = 0$ for all k, C and the above maximization in C_2 is not needed. Then, the complexity is $O(n\,W)$. When the upper bounds b_i are not tight, i.e.

$b_i \geq \frac{W-s_i}{w_i}$ for all i, no items k can be taken at level $x_k = b_k$, hence $V^k(C) = 0$ for all k, C and the above maximization in C_1 is not needed. Then, the complexity is also $O(n\,W)$. When Assumption 1 holds and items are unbounded, solving the problem requires $O(n)$ (without sorting the items, one just needs to enumerate on which should be the critical item).

For the general case where some f_i may be negative, another dynamic programming recursion can be defined whose complexity is $O(n^2W)$ and does not require to sort items beforehand: Let $T^{k,i}(C)$ be the best value that can be accumulated with items 1 up to k, with $x_k \in \{0, b_k\}$ when $y_k = 1$, while not using i ($y_i = 0$) and using a capacity $C \in \{0, \ldots, W\}$:

$$T^{k,i}(C) = \max\{\underbrace{T^{k-1,i}(C)}_{x_k=y_k=0}, \underbrace{T^{k-1,i}(C-s_k)-f_k}_{x_k=0,y_k=1}, \underbrace{T^{k-1,i}(C-w_k\,b_k-s_k)+p_k\,b_k-f_k}_{x_k=b_k,\,y_k=1}\} \tag{24}$$

if $k \neq i$, or $T^{k,i}(C) = T^{k-1,i}(C)$ if $k = i$. Then, the optimal solution value can be obtained as

$$\max_{1\leq i\leq n} \max_{C}\{\underbrace{T^{n,i}(C)}_{x_i=y_i=0}, \underbrace{T^{n,i}(C) + \min\{\frac{(W-C-s_i)^+}{w_i}, b_i\}p_i - f_i\,\delta(W-C-s_i>0)}_{0\leq x_i\leq b_i,\; y_i=\delta(W-C-s_i>0)}, \tag{25}$$

$$\underbrace{T^{n,i}(C) - f_i\,\delta(W-C-s_i\geq 0)}_{x_i=0,y_i=\delta(W-C-s_i\geq 0)}\} \; . \tag{26}$$

where $\delta(*) = 1$ if $(*) = true$ and $(*)^+ = \max\{0, *\}$. Some computations can be saved by observing that $T^{k,i}(C) = T^{k,i+1}(C)$ for $k < i$, but this remark does not change the worst case complexity.

2 The multiple-class binary knapsack with setups

Model MBKS is defined by setting item bounds u_{ij} to 1 and restricting x_{ij} to be binary, i.e., the model takes the form:

$$\max\{\sum_{i=1}^{n}(\sum_{j=1}^{n_i} p_{ij}x_{ij} - \sum_{i=1}^{n} f_i y_i) : \sum_{i=1}^{n}((\sum_{j=1}^{n_i} m_{ij}w_i x_{ij}) + s_i y_i) \leq W, a_i y_i \leq \sum_{j=1}^{n_i} m_{ij}x_{ij} \leq b_i y_i \;\forall i,$$

$$x_{ij} \leq y_i \;\forall i,j \;,\;\; x_{ij} \in \{0,1\}\;\forall i,j \;,\;\; y_i \in \{0,1\}\;\forall i \;.\} \tag{27}$$

Here, Assumption 2 is restrictive: 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. Also note that we cannot assume positive profit p_{ij}. An item (i, j) with negative profit $p_{ij} < 0$ might be worth selecting to satisfy the lower bound a_i. However, if $a_i = 0$, one can make non restrictive assumptions:

Assumption 6 *(**without loss of generality**) Under Assumption 2, $p_{ij} \geq 0$ for all j.*

Otherwise, $x_{ij} = 0$ in any optimal solution. Similarly, Assumption 4 takes a weaker form:

Assumption 7 *(**without loss of generality**) Under Assumption 2, $f_i < \max\{\sum_j p_{ij}x_{ij} : \sum_j m_{ij}\, x_{ij} \leq b_i,\ x_{ij} \in \{0,1\}\ \forall j\}$.*

Otherwise, if the set-up cost is larger than the maximum profit that can be generated with the class i items, it is optimal to set $x_{ij} = 0\ \forall j$ and $y_i = 0$. Finally, eliminating rational class weights w_i as in Assumption 5 is feasible (by setting $m'_{ij} = m_{ij}w_i$, $a'_i = a_i w_i$, $b'_i = b_i w_i$ and multiplying by a constant sufficiently large to take this number integer); but it is not recommended as it introduces large numbers in knapsack constraints (3).

MBKS can be reformulated as a multiple choice knapsack, in the line of the work of Chajakis and Guignard [2]. 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 $(x_{ij})_{j=1,\cdots,n_i}$ that yield the same total multiplicity $M_i := \sum_j m_{ij}\ x_{ij}$. As a result, the optimization within each class can be done independently of the global optimization of the use of the knapsack capacity W. Thus, for each class, one can theoretically enumerate all pairs $(M_i := \sum_j m_{ij}x_{ij}, P_i := \sum_j p_{ij}\ x_{ij})$ that can be achieved by 0-1 solutions x_{ij}. Note that one only needs to consider undominated pairs (M_i, P_i): 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. To compute all undominated pairs, one needs to solve an "all-capacities" knapsack problem for each class i (a dynamic program is well suited for this), i.e. for each $M = a_i, \ldots, b_i$, we compute:

$$P_i(M) = \max\{\sum_j p_{ij}\ x_{ij} : \sum_j m_{ij}\ x_{ij} = M,\ x_{ij} \in \{0,1\}\ \forall j\} \tag{28}$$

Then, the multiple choice knapsack reformulation of MBKS is given by:

$$\max\{\sum_i \sum_{M=a_i}^{b_i} (P_i(M) - f_i)\lambda_M^i : \quad \sum_i \sum_{M=a_i}^{b_i} (M\ w_i + s_i)\ \lambda_M^i \leq W,$$
$$\sum_{M=a_i}^{b_i} \lambda_M^i \leq 1\ \forall i,\ \lambda_M^i \in \{0,1\} \forall i, M\} \tag{29}$$

This reformulation involves a pseudo-polynomial number of variables. As for CKS, we use this multi-choice knapsack reformulation approach to benchmark our numerical results.

2.1 Characterizations of optimal solutions

In analyzing the structure of optimal solutions, first 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 5 *Under Assumptions 1 and 2, there exists an optimal solution where* $y_i = 0$ *when* $\sum_j x_{ij} = 0$.

The characterisation of LP solution to MBKS also relies on a decomposition per item class:

Observation 6 *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:*

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

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

$$\frac{P_i^{LP}(a_i) - f_i}{w_i\, a_i + s_i} \leq \frac{P_i^{LP}(b_i) - f_i}{w_i\, b_i + s_i} \tag{30}$$

where $P_i^{LP}(M)$ *is the solution to the LP relaxation of (28), 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* $f_i < 0$.

This observation results from the analysis of extreme solutions for the single class knapsack problems define by constraints (3). Figure 1 illustrates both the case where the slope $\frac{P_i^{LP}(a_i)-f_i}{w_i\, a_i+s_i} > \frac{P_i^{LP}(b_i)-f_i}{w_i\, b_i+s_i}$ and vice versa.

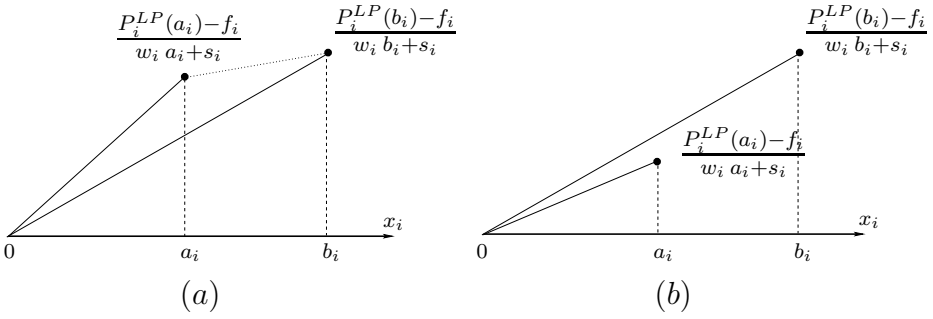

Figure 1: Ratio of class i profit per unit of knapsack capacity consumption

Similarly to what we did for model CKS, we can derive from Observation 6 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 6 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

$$\max \sum_{i \in I^a} \sum_{j=1}^{n_i} (p_{ij} - \frac{f_i}{a_i} m_{ij}) z_{ij}^a + \sum_{i \in I} \sum_{j=1}^{n_i} (p_{ij} - \frac{f_i}{b_i} m_{ij}) z_{ij}^b - \sum_{i \in I^f} f_i z_i^f \tag{31}$$

$$\text{s.t.} \sum_{i \in I^a} \sum_{j=1}^{n_i} (w_i + \frac{s_i}{a_i}) m_{ij} z_{ij}^a + \sum_{i \in I} \sum_{j=1}^{n_i} (w_i + \frac{s_i}{b_i}) m_{ij} z_{ij}^b + \sum_{i \in I^f} s_i z_i^f \leq W \tag{32}$$

$$\sum_{j=1}^{n_i} [\frac{m_{ij}}{a_i} z_{ij}^a + \frac{m_{ij}}{b_i} z_{ij}^b] \leq 1 \quad \forall i \in I^a \tag{33}$$

$$\sum_{j=1}^{n_i} \frac{m_{ij}}{b_i} z_{ij}^b + z_i^f \leq 1 \quad \forall i \in I^f \tag{34}$$

$$\begin{aligned} z_{ij}^a + z_{ij}^b &\leq 1 \quad \forall i \in I^a, j \\ z_{ij}^a &\in [0,1] \; \forall i \in I^a, j \\ z_{ij}^b &\in [0,1] \; \forall i \in I, j \\ z_i^f &\in [0,1] \; \forall i \in I^f \end{aligned} \tag{35}$$

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

$$x_{ij} = z_{ij}^a + z_{ij}^b \quad \text{and} \quad y_i = \sum_j (\frac{m_{ij}}{a_i} z_{ij}^a + \frac{m_{ij}}{b_i} z_{ij}^b) + z_i^f \tag{36}$$

Constraints (33-34) 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:

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

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 (31-35).*

On one hand, it is known that the LP relaxation of binary knapsack problem with SOS constraints admits a greedy solution [6]. On the other hand, [15] shows that the LP

relaxation of a binary knapsack with class bounds can also be solved using a greedy procedure. But, solving problem (31-35) 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 [15] 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)*

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

This assumption corresponds to the case illustrated by part (b) of Figure 1. 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 a little less restrictive than Assumption 1.

Observation 7 *Under Assumption 8, problem (31-35) admits a solution where all variables z^a_{ij} and z^f_i have value zero.*

Indeed, if Assumption 8 holds and $z^a_{ij} > 0$, one can modify the solution by setting $z^{a\,\prime}_{ij} = 0$ and $z^{b\,\prime}_{ij} = z^b_{ij} + \frac{(w_i + \frac{s_i}{a_i})\,m_{ij}}{(w_i + \frac{s_i}{b_i})\,m_{ij}}\,z^a_{ij}$. This solution modification is feasible with regard to knapsack constraint (32) by construction but also with regard to constraint (33) as it can be easily checked. Moreover, the profit value of the modified solution is not less than the original. Similarly, if $z^f_i > 0$, decreasing its value allows to increase some z^b_{ij} 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:*

$$\frac{(p_{ij} - \frac{f_i}{b_i}\,m_{ij})}{(w_i + \frac{s_i}{b_i})\,m_{ij}} \tag{37}$$

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

$$\sum_{k \in K^i,\, k < c_i} m_k \leq b_i \quad but \quad \sum_{k \in K^i,\, k \leq c_i} m_k > b_i \; . \tag{38}$$

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

$$W(l) = \sum_i \sum_{k \in K^i(l)} (w_i + \frac{s_i}{b_i})\, m_k + \sum_{i \in I^b(l)} (w_i + \frac{s_i}{b_i})\, (b_i - \sum_{k \in K^i(l)} m_k) \; . \tag{39}$$

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

$$W(c) \leq W \quad but \quad W(c) + (w_{i_c} + \frac{s_{i_c}}{b_{i_c}})\, m_c > W \tag{40}$$

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

$$\begin{aligned}
x_k &= 1 && \text{for } k \in K^i(c) \text{ and } i = 1, \ldots, n, && (41)\\
x_{c_i} &= \frac{1}{m_{c_i}} (b_i - \sum_{k \in K^i(c)} m_k) && \text{for } i \in I^b(c), && (42)\\
x_c &= \frac{1}{(w_{i_c} + \frac{s_{i_c}}{b_{i_c}})\, m_c} (W - W(c)) && \text{if } c \in K^{i_c} && (43)\\
x_k &= 0 && \text{otherwise} && (44)\\
y_i &= 1 && \text{for } i \in I^b(c) && (45)\\
y_{i_c} &= \frac{\sum_{k \in K^{i_c}(c)} m_k + m_c\, x_c}{b_{i_c}} && \text{for } i : c \in K^i && (46)\\
y_i &= 0 && \text{otherwise.} && (47)
\end{aligned}$$

Proof: Observation 7 implies that in the LP formulation (31-35) 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 [15]. 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 (31-35) 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

$$\max \sum_{i=1}^{n} \sum_{j=1}^{n_i} (p_{ij} - \frac{f_i}{b_i}\, m_{ij})\, z_{i\,j} \tag{48}$$

$$\text{s.t.} \sum_{i=1}^{n} \sum_{j=1}^{n_i} (w_i + \frac{s_i}{b_i})\, m_{ij}\, z_{ij} \leq W \tag{49}$$

$$\sum_{j} m_{ij}\, z_{ij} \leq b_i \qquad \forall i \tag{50}$$

$$z_{ij} \in \{0,1\} \qquad \forall i,\, j\,. \tag{51}$$

Problem MBKS and problem (48-51) 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 (x, y) for MBKS, by applying the mapping (36) 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

$$\frac{p_{ij}}{w_i m_{ij}}. \tag{52}$$

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

$$\max \sum_{i \in I^0} \sum_{j \in J_i} (p_{ij} - \frac{f_i}{b_i}\, m_{ij})\, z_{ij} + \sum_{i \in I^1} \sum_{j \in J_i} p_{ij}\, z_{ij} \tag{53}$$

$$\text{s.t.} \sum_{i \in I^0} \sum_{j \in J_i} (w_i + \frac{s_i}{b_i})\, m_{ij}\, z_{ij} + \sum_{i \in I^1} \sum_{j \in J_i} w_i\, m_{ij}\, z_{ij} \leq C \tag{54}$$

$$\sum_{j \in J_i} m_{ij}\, z_{ij} \leq C_i \qquad \forall i \tag{55}$$

$$z_{ij} \in \{0,1\} \qquad \forall i,\, j\,. \tag{56}$$

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 is 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 (53-56) and the residual MBKS problem in its (x, y)-formulation have the same LP value and therefore computing the LP-solution of (53-56) provides a valid dual bound for that node. Finally, observe that a preprocessing can be applied to (53-56). 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 (53-56) before computing its LP value.

The LP-relaxation of (53-56) admits a greedy solution. Items whose class is in I^0 are sorted by decreasing ratio (37), while those whose class is in I^1 are sorted by decreasing ratio (52) 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 (52) instead of (37), 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 new y_i variable, as it modifies the partition I^0, I^1, and hence the partition of the items.

Our branch-and-bound algorithm is presented in Table 3. Items k denotes the next item in the greedy ordering for the current residual problem (53-56). 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 Table 3). Each list is appropriately sorted by decreasing ratio $\frac{p_k}{w_k}$. One moves forward in list L^0 (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*.

The other notations used in Table 3 are analogous to that of Table 1, except for K_i that stands for the remaining capacity/multiplicity within class i. Moreover, i_k denotes the class index of item k and j_k its index within the class. In "Forward Moves", we set z_k's to one in formulation (53-56), which amount to fixing $x_{(i_k,j_k)}$ to 1 in MBKS. If $i_k \in I^0$, we set $y_{i_k} = 1$, 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 recomputed. 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 Table 4. 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.

Table 3: Branch-and-Bound for MBKS when $f_i \geq 0$ and $a_i = 0\ \forall i$

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 (37). 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 (52). 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 (53-56) 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 B&B node.

Compute UB: Let $U = Z$; $K = C$; $K_i = C_i$, $\forall i$; and let $l = k$.
Step A: While $(l \leq N)$ and $(w_l \leq K)$ and $(m_l \leq K_{i_l})$ {
If $(w_{i_l}\, m_{i_l,j_l} + s_{i_l}(1 - y_{i_l}) > C)$, $\{l = next(l)$, goto Step A. $\}$
$U \mathrel{+}= p_l$; $K \mathrel{-}= w_l$; $K_{i_l} \mathrel{-}= m_l$; $l = next(l)$. $\}$
Step B: If $(l > N)$, goto Test Pruning.
Step C: If $((m_l > K_{i_l})$ and $(w_l \frac{K_{i_l}}{m_l} \leq K))$,
$\{U \mathrel{+}= p_l \frac{K_{i_l}}{m_l}$; $K \mathrel{-}= w_l \frac{K_{i_l}}{m_l}$; $K_{i_l} = 0$; $l = next(l)$; go to Step A$\}$.
Step D: /* $(w_l > K)$ or even $(w_l \frac{K_{i_l}}{m_l} > K)$ */ $U \mathrel{+}= p_l \frac{K}{w_l}$.

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

Forward Move: While $(k \leq N)$ and $(w_{i_k}\, m_k + s_{i_k}\,(1 - y_{i_k}) \leq C)$ and $(m_k \leq C_{i_k})$, do {
$x_{(i_k,j_k)} = 1$; $C \mathrel{-}= w_k$; $C_{i_k} \mathrel{-}= m_k$; $Z \mathrel{+}= p_k$;
if $(y_{i_k} == 1)$, then $k = next(k)$;
else $\{y_{i_k} = 1$; $C \mathrel{-}= s_{i_k}$; $Z \mathrel{-}= f_{i_k}$; *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 $(k \geq 1))$.
If $(k == 0)$, STOP.
/* $(x_{(i_k,j_k)} == 1)$ */ **WithdrawItem**(k); $k = next(k)$.
Go to Compute UB.

Table 4: subroutine **WithdrawItem**(k) of the Branch-and-Bound for MBKS

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*; $\}$.

2.2.1 Dynamic programs for model MBKS

A solution by dynamic programming assumes integer data: s_i, w_i, and $W \in I\!N$. Let us first consider the unbounded case where $a_i = 0$ and $b_i \geq \left\lfloor \frac{W-s_i}{w_i} \right\rfloor$ for all i. Then, one can write a dynamic programming recursion where $V^i(C)$ defines the best value that can be achieved using items from class $k = 1, \ldots, i$ with a capacity consumption C and $V^{ij}(C)$ defines the best value that can be achieved using items from class $k = 1, \ldots, i-1$ plus at least one item among the first j items of class i, with a capacity consumption C. The $V^{ij}(C)$ and $V^i(C)$ values can be computed recursively as follows:

$$\begin{aligned} V^{ij}(C) &= \max\{ \quad V^{i,j-1}(C), \;\; V^{i,j-1}(C - w_i\, m_{ij}) + p_{ij}, \\ &\qquad\qquad V^{i-1}(C - w_i\, m_{ij} - s_i) - f_i + p_{ij}\} \\ V^{i}(C) &= \max\{ \quad V^{i-1}(C), \; V^{i,n_i}(C), \; V^{i-1}(C - s_i) - f_i\} \; . \end{aligned} \tag{57}$$

Such dynamic program requires $O(\sum_i n_i\, W)$ operations. [2]

For the bounded case, one must first solve a knapsack sub-problem within each class before solving the overall problem: Let $U^i(M)$ be the optimal value that can be achieved with class i items using a multiplicity of **exactly** M units. $U^{ij}(M)$ is defined by analogy with the above $V^{ij}(C)$ definition. One can compute $U^i(M)$ and $U^{ij}(M)$ by dynamic programming: Initially, $U^{i,j}(0) = 0\ \forall j$ and $U^{i,0}(M) = -\infty$ for $M = 1, \ldots, b_i$; then, one sets $U^{ij}(M) = U^{i,j-1}(M)$ for $M = 1, \ldots, m_{ij} - 1$ and one computes

$$U^{ij}(M) = \max\{U^{i,j-1}(M), U^{i,j-1}(M - m_{ij}) + p_{ij}\} \tag{58}$$

$M = m_{ij}, \ldots, b_i$ and for $j = 1, \ldots, n_i$. Then

$$U^i(M) = U^{i,n_i}(M) \;\; \forall M \; . \tag{59}$$

These computations requires $O(n_i\, b_i)$ operations for each class i. Therefore the overall complexity for computing all the $U^i(M)$ is $O(\sum_i n_i b_i)$ (which is bounded by $O(\sum_i n_i W)$

[2] Observe that this complexity $O(\sum_i n_i\, W)$ does not imply that the multiple class problem requires a higher complexity than the integer knapsack problem treated in Section 3.1 (for which the unbounded problem can be solved in $O(nW)$). Indeed, the input data file is of length proportional to $\sum_i n_i$ since it includes the description of the profit values p_{ij}.

as $b_i \leq \lfloor \frac{W}{w_i} \rfloor$). As an aside, observe that when $m_{ij} = 2^{j-1}\ \forall i, j$, a given multiplicity M can only be obtained from a single combination of 0-1 items (i, j) and $U^i(M)$ can be computed directly, although this does not change the computational complexity. >From $U^i(M)$ values, one can compute $V^i(C)$, the best value that can be achieved with items of class 1 up to i and capacity C:

$$V^i(C) = \max\{\underbrace{V^{i-1}(C)}_{y_i=0}, \underbrace{\max_{a_i \leq M \leq b_i} \{V^{i-1}(C - w_i\, M - s_i) + U^i(M) - f_i\}}_{y_i=1}\} \ . \qquad (60)$$

This requires $O(nW \max_i\{b_i\})$ operations (which is bounded by $O(nW^2)$ but can be much smaller than $O(nW^2)$ in practice).

When an integer knapsack problem is transformed into a binary multiple class knapsack problem one can treat the class bounds a_i and b_i implicitly and use the dynamic recursion (57) for the unbounded case to benefit from the lower complexity $O(\sum_i n_i\, W)$. Indeed, in such case, the profit is defined for the class and not for the 0-1 items, therefore Assumption 2 is valid (quantity a_i can be incorporated to the fixed cost and weight). To eliminate the upper bound b_i, one just needs to amend the 0-1 transformation defined in the introduction: set $n_i = \lfloor \log_2 b_i \rfloor + 1$ and $m_{ij} = 2^{j-1}$ for $j = 1, \ldots, n_i - 1$ but $m_{i,n_i} = b_i - \sum_{j=1}^{n_i-1} m_{ij}$.

3 Extensions to non-binary models

We now examine possible extensions of the analysis of model CKS and MBKS to their integer counterpart.

3.1 The integer knapsack problem with setups

Consider the problem with integer variables x given by (7-11), where $a_i, b_i \in I\!N$. Reformulation (13) is now valid without having to make an extra assumption on the integrality of data. Assumptions 2, 3, and 4 remain valid without loss of generality. But the proof of validity of Assumption 4 must be adapted now that bounds a_i and b_i are assumed integer: If $0 < f_i < p_i\, b_i$ for some i, then, in any optimal solution, either $x_i = y_i = 0$ or $x_i \geq \lceil \frac{f_i}{p_i} \rceil$, because a solution where $0 < x_i \leq \lfloor \frac{f_i}{p_i} \rfloor \leq \frac{f_i}{p_i}$ can be improved by setting $x_i = y_i = 0$. Thus, $\lceil \frac{f_i}{p_i} \rceil$ can be interpreted as a lower bound a_i and eliminated as in the proof of validity of Assumption 2. However, the nice characterization of optimal solution of Observation 2 does not carry on to IKS: case (ii) can now arise for more than one class.

The LP solution of IKS is obviously that of CKS. Hence, Proposition 1 and Observation 4 extend to IKS. The primal greedy heuristic of Table 2 can be adapted for IKS.

In Step D, instead of solving $CKP(y)$, one can use a standard primal greedy procedure for the integer knapsack problem in x (y being fixed) that is defined by (??-??) with additional constraints $x_i \in I\!N \ \forall i \in I$. However, the branch-and-bound procedure for CKS cannot be simply extended for IKS because the weaker characterization of optimal solution implies that there are more than two branches to be considered for each class. Note that the static greedy ordering (23) assumes $x_i = b_i$ while branching decisions on x_i can lead to considering $x_i \in \{1, \ldots, b_i - 1\}$. Therefore, after making branching decisions that define new bounds on x_i's, the greedy ordering of the classes changes for the residual problem. However, another approach that exploits the results of this paper is possible. It consists in transforming the IKS model into a MBKS model using the 0-1 transformation of [15] and apply the branch-and-bound procedure proposed above for model MBKS.

Assuming integer data (s_i, w_i, and $W \in I\!N$), a dynamic programming solution can easily be obtained by generalizing the standard dynamic program for the integer knapsack problem. Let $V^i(C)$ be the best value that can be accumulated using items 1 up to i and using a capacity C: i.e. $V^i(C) = \max\{\sum_{k=1}^{i}(p_k x_k - f_k y_k) : \sum_{k=1}^{i}(w_k x_k + s_k y_k) \leq C\,, x_k \leq b_k\, y_k,\ x_k \in I\!N, y_k \in \{0,1\}\ \text{ for } k = 1, \ldots i\}$. Setting $V^0(C) = 0$ for all $0 \leq C \leq W$, the values $V^i(C)$ can be computed recursively:

$$V^i(C) = \max\{V^{i-1}(C), \max_{x_i=0,\ldots,b_i} V^{i-1}(C - w_i\, x_i - s_i) + p_i\, x_i - f_i\}$$

for all $0 \leq C \leq W$ and $k = 1, \ldots, n$. This requires $O(nW^2)$ operations in the worth case. Then, the optimal solution value is given by $V^n(W)$. A dynamic program of a better complexity $O(n\,W\,\log W)$ can be achieved by transforming the integer knapsack problem in a 0-1 model and applying the dynamic program presented at the end of Section 2.2.1.

When the item upper bounds are not tight, i.e. $b_i \geq \left\lfloor \frac{W - s_i}{w_i} \right\rfloor$ for all i, the recursion can take a simpler form. The complexity gets down to $O(nW)$ operations as is the case for the standard unbounded integer knapsack problem. Let $V^i(C, y)$ be the best value that can be accumulated using items 1 up to i, with setup variable $y_i = y$ and using a capacity C. Then,

$$V^i(C,0) = \max\{V^{i-1}(C,0), V^{i-1}(C,1)\} \qquad \text{and}$$

$$V^i(C,1) = \max\{\underbrace{V^i(C - s_i, 0) - f_i}_{x_i=0}, \underbrace{V^i(C - w_i, 1) + p_i}_{x_i \geq 1}\}$$

The optimum is given by $\max\{V^n(W,0),\ V^n(W,1)\}$.

3.2 The multiple-class integer knapsack with setups

Consider model MIKS, whose formulation is (1-6), as a generalization of model MBKS where variables x_{ij} take integer value. Then, $u_{ij} \in I\!N$ defines an upper bound on x_{ij}.

Due to constraints (3), u_{ij} can be redefined as $\min\{u_{ij}, \lfloor \frac{b_i}{m_{ij}} \rfloor\}$. Results derived for MBKS can be extended for MIKS, basically by introducing the multiplicative factor u_{ij} where appropriate. Subproblem (28) becomes $P_i(M) = \max\{\sum_j p_{ij}\, x_{ij} : \sum_j m_{ij}\, x_{ij} = M,\ x_{ij} \in \{0, \ldots, u_{ij}\}\ \forall j\}$. The LP solution of MIKS have the same properties as that of MBKS: Observation 6 and Proposition 2 are still valid. Proposition 3, that provides a greedy LP solution to MBKS under Assumption 8, can be generalized for the integer case. The greedy ordering of the items remains to sort item by non-increasing ratio (37). The definition of intra class critical items (38) takes a modified form: c_i is such that $\sum_{k \in K^i,\, k<c_i} m_k\, u_k \leq b_i$ but $\sum_{k \in K^i,\, k \leq c_i} m_k\, u_k > b_i$. While (39) becomes

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

and the definition (40) of the critical item becomes:

$$W(c) \leq W \quad \text{but} \quad W(c) + (w_i + \frac{s_i}{b_i})\, m_c\, u_c > W \tag{61}$$

The greedy LP solution (x, y) is given by (41-47) where each multiplicity m_k must be replaced by $m_k\, u_k$. Thus, we have shown that

Proposition 4 *Under Assumption 8, the LP relaxation of MIKS can be solved by a greedy algorithm.*

A dynamic programming solution assumes integer data. Recursion (57) can be adapted for MIKS in the case bounds are loose, i.e. $a_i = 0$, $b_i \geq \left\lfloor \frac{W - s_i}{w_i} \right\rfloor$, $u_{ij} \geq \left\lfloor \frac{b_i}{m_{ij}} \right\rfloor$ for all i: let

$$V^{ij}(C) = \max\{V^{i,j-1}(C),\ \ V^{i,j}(C - w_i\, m_{ij}) + p_{ij}, V^{i-1}(C - w_i\, m_{ij} - s_i) - f_i + p_{ij}\}$$

while the computation of $V^i(C)$ remains the same. Its complexity is $O(\sum_i n_i W)$.

For the bounded case, recursion (60) can easily be adapted by redefining (58) as

$$U^{ij}(M) = \max_{0 \leq x \leq u_{ij}} \{U^{i,j-1}(M - m_{ij}\, x) + p_{ij}\, x\} \tag{62}$$

The overall complexity becomes $O(\sum_i n_i W^2)$.

When only bounds u_{ij} are tight but a_i and b_i's are loose (i.e. $a_i = 0,\ b_i \geq \left\lfloor \frac{W - s_i}{w_i} \right\rfloor\ \forall i$), a better complexity can be achieved using binary decomposition of the integer variables x_{ij}. Let

$$n_{ij} = \lfloor \log_2 u_{ij} \rfloor + 1 \tag{63}$$

and define n_{ij} items (i, j, k) with

$$m_{ijk} = m_{ij}\, 2^{k-1} \text{ for } k = 1, \ldots, n_{ij} - 1 \text{ and } m_{ijn_{ij}} = u_{ij} - \sum_{k=1}^{n_{ij}-1} 2^{k-1} \tag{64}$$

and

$$p_{ijk} = p_{ij}\, 2^{k-1} \text{ for } k = 1, \ldots, n_{ij} - 1 \text{ and } p_{ijn_{ij}} = p_{ij}\, (u_{ij} - \sum_{k=1}^{n_{ij}-1} 2^{k-1}\, m_{ij}) \tag{65}$$

Constraint (4) is replaced by $x_{ijk} \leq y_i \; \forall (i, j, k)$. In this way, bounds u_{ij} are built into the reformulation and recursion (57) for the unbounded problem can be used with complexity $O(\sum_{ij} n_{ij} W)$.

The primal heuristic that can be derived from the first plunge depth into branch-and-bound tree of Table 3 can be extended to the integer case: one must set x_k to

$$x_k = \min\{u_k, \left\lfloor \frac{(C - S)}{w_{i_k}\, m_k} \right\rfloor, \frac{C_{i_k}}{m_k}\} \, .$$

To solve MIKS by branch-and-bound, one can convert it to a MBKS problem using a binary decomposition and use the procedure of Table 3. When the item upper bounds are loose, i.e., $u_{ij} \geq \left\lfloor \frac{b_i}{m_{ij}} \right\rfloor$, the 0-1 transformation can be done defining $m_{ijn_{ij}} = m_{ij}\, 2^{n_{ij}-1}$ as suggested in [15] in order to avoid multiple representations of the same solution. However, when bounds u_{ij} are tight, we use the 0-1 transformation (63-64) where $m_{ijn_{ij}} = u_{ij} - \sum_{k=1}^{n_{ij}-1} 2^{k-1}$ even though it suffers from symmetry. Indeed, enforcing the u_{ij} bounds by adding explicit constraints would yield a model that has a structure from MBKS.

4 Numerical tests

As for the standard knapsack problem, whether dynamic programming or branch-and-bound is a more efficient approach will typically depend on the size of numbers (such as the capacities) and the correlation between item profits and weights: dynamic programs have pseudo-polynomial complexity but tend to reach their worst case bound; while branch-and-bound procedures have exponential worst case complexity but, in practice, they tend to use fewer iterates than DP to prove optimality when instances are not too correlated (the greedy ordering is not really discriminant for correlated instances). Numerically comparing branch-and-bound and dynamic programming for knapsack problem with setups would merely reproduce this standard analysis. Anyway, these pure approaches are made obsolete nowadays by hybrid approaches. The best performing approaches for knapsack problems are those combining the advantages of both branch-and-bound and dynamic programming paradigm, truncating the enumeration based on dominance principle and dual bounds. They are dynamic programs enumerating solution starting from the greedy solution and progressively extending the so called core around that greedy solution and making extensive use of dual bound pruning and other form of preprocessing (see [7]).

As it was made clear from the introduction, 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. In particular, our main algorithmic contribution was to extend the branch-and-bound paradigm (greedy enumeration scheme and greedy dual bounds) 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 the branch-and-bound algorithms proposed in Table 1 and in Table 3 to a standard MIP solver (Xpress-MP [17], 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 [12]. This makes comparison to specialized solver for multiple choice knapsack somewhat bias.

For the tests on model CKS, we generate 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 $(s_i + b_i w_i) = \frac{W}{0.2n}$. For each class i, $s_i + w_i$ are generated from a uniform distribution in interval $[\frac{0.35W}{0.2n}, \frac{0.65W}{0.2n}]$; thus, $s_i + w_i$ 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 we 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 Table 5. The first column indicates the chosen parameters of 10 random instances for the 18 combination of parameters n, α, and β. The following columns give the average computation time for the standard MIP solver "Xpress-MP", the average time obtained using the specialized multiple-choice knapsack dynamic program solver of Pisinger [11] ("MCKP") after application of transformation (13) and the average time of our Branch-and-Bound algorithm of Table 1 ("BB"). Time units are seconds on a PC bi-pro. Xeon 3GHz, 2Go. We also run tests using a basic dynamic program (which would be more in line with our basic Branch-and-Bound solver). For illustration, our computation times for $n1000 - \alpha 2 - \beta 0.5$ was then 695.16s and for $n5000 - \alpha 2 - \beta 0.5$ it was 13226.49s which is more than 250 time slower that the solver of Pisinger. The last line of Table 5 provides average times over all classes. It shows that our basic BB algorithm is about 100 time faster than transforming the problem to a multiple choice knapsack and applying an efficient solver to the reformulated problem.

For the tests on model MBKS, we generate 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

Table 5: Computation time for CKS on random instances

parameters	Xpress	MCKP	BB
$n1000 - \alpha 2 - \beta 0.2$	20.84	2.24	0.05
$n1000 - \alpha 2 - \beta 0.5$	11.11	2.17	0.05
$n1000 - \alpha 2 - \beta 0.8$	11.67	2.70	0.04
$n1000 - \alpha 4 - \beta 0.2$	11.66	1.63	0.04
$n1000 - \alpha 4 - \beta 0.5$	8.99	1.62	0.03
$n1000 - \alpha 4 - \beta 0.8$	9.34	1.64	0.04
$n5000 - \alpha 2 - \beta 0.5$	388.25	48.77	0.55
$n5000 - \alpha 2 - \beta 0.2$	504.80	50.68	1.01
$n5000 - \alpha 2 - \beta 0.8$	301.29	47.70	0.38
$n5000 - \alpha 4 - \beta 0.2$	601.20	37.60	0.60
$n5000 - \alpha 4 - \beta 0.5$	225.82	36.63	0.44
$n5000 - \alpha 4 - \beta 0.8$	301.23	36.33	0.31
$n10000 - \alpha 2 - \beta 0.2$	1644.59	195.96	2.84
$n10000 - \alpha 2 - \beta 0.5$	1214.14	192.85	1.28
$n10000 - \alpha 2 - \beta 0.8$	1053.59	186.00	0.98
$n10000 - \alpha 4 - \beta 0.2$	959.92	147.29	2.34
$n10000 - \alpha 4 - \beta 0.5$	938.30	143.40	0.98
$n10000 - \alpha 4 - \beta 0.8$	796.34	143.87	0.80
average	500.17	71.06	0.71

setup ($f_i > 0$, $s_i > 0$). For each class i, $s_i + w_i$ are generated from a 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 large constant and we set $r_{ij} = 1 - \beta + \beta\, g_{ij}\, \frac{2}{M}$. Then, we compute $p_{ij} = \lfloor m_{ij} w_i r_{ij} \rfloor$.

Table 6: Computation time for MBKS on random instances

parameters	$\sum_i n_i$	Xpress	MCKP	BB
$n10 - \alpha 2 - \beta 0.2$	3256.4	168.24	3.18	1.01
$n10 - \alpha 2 - \beta 0.5$	3256.4	177.24	3.34	0.14
$n10 - \alpha 2 - \beta 0.8$	3256.4	282.20	3.29	0.15
$n10 - \alpha 4 - \beta 0.2$	4781.2	280.79	8.29	5.59
$n10 - \alpha 4 - \beta 0.5$	4781.2	781.66	8.86	1.15
$n10 - \alpha 4 - \beta 0.8$	4781.2	1125.58	8.58	0.36
$n50 - \alpha 2 - \beta 0.2$	60832.9		386.78	596.27
$n50 - \alpha 2 - \beta 0.5$	60832.9		395.64	155.48
$n50 - \alpha 2 - \beta 0.8$	60832.9		395.36	88.19
$n50 - \alpha 4 - \beta 0.2$	84524.7		1130.54	808.13
$n50 - \alpha 4 - \beta 0.5$	84524.7		1130.70	930.23
$n50 - \alpha 4 - \beta 0.8$	84524.7		1142.29	174.03
$n100 - \alpha 2 - \beta 0.5$	231698.0		4201.33	1204.17
$n100 - \alpha 2 - \beta 0.2$	231698.0		4120.22	4012.15
$n100 - \alpha 2 - \beta 0.8$	231698.0		4163.15	1338.06
$n100 - \alpha 4 - \beta 0.2$	335019.0		11032.00	6506.93
$n100 - \alpha 4 - \beta 0.5$	335019.0		11161.46	5387.78
$n100 - \alpha 4 - \beta 0.8$	335019.0		11161.13	5838.55
average	120018.7		2803.12	1502.69

The results for MBKS are presented in the Table 6. The column "$\sum_i n_i$" gives the total number of items. The last three columns give the average computation time on 10 random instances for the standard MIP solver "Xpress-MP" [17], the average time to apply transformation (29) and then the specialized multiple-choice knapsack dynamic program solver of Pisinger [11] ("MCKP"), and the average time taken by our Branch-

and-Bound algorithm of Table 3 ("BB"). Time units are seconds. "Xpress-MP" could not solved any instances with $n = 50$ and $n = 100$ within our time limit of respectively 3h and 10h. 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 can take more than 5h for one instance and less than 20s for another). 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 (28) 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 [7], section 1.3). To do this, there is no much better algorithm than a basic dynamic program (which is what we use). The average times reported at the bottom of Table 6 shows that, over all instance classes, BB is approximately 2 time faster than MCKP on average.

5 Conclusion

The paper provides a review of the literature on knapsack problems with setups, discusses various reformulation, 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. Dynamic programming recursion are provided for the sake of establishing the complexity of the knapsack variants with setups. The greedy enumeration scheme and greedy dual bounds of our branch-and-bound procedures could be exploited to develop hybrid dynamic programming approaches in future work in the line of the best performing approaches for the standard knapsack problem. The assumption that all item weights are a multiple of their class weight is essential for solving the LP relaxation by a greedy procedure. However, in the above multiple choice reformulations as well as for the the dynamic programs presented herein, this assumption was not used. Hence, the latter approaches extend to the case where this assumption does not hold. Extentions to the integer models IKS and MIKS have also been considered.

References

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

[2] E.D. Chajakis and M. Guignard. xact algorithms for the setup knapsack problem. *INFOR*, 32(3):124–142, 1994.

[3] G.B. Dantzig. Discrete variable extremum problems. *Operations research*, 5:266–277, 1957.

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

[5] R. Jans and Z. Degraeve. Improved lower bounds for the capacitated lot sizing problem with setup times. *Oper. Res. Letters*, 32:185–195, 2004.

[6] E.L. Johnson and M.W. Padberg. A note on the knapsack problem with special ordered sets. *Operations Research letters*, 1:18–22, 1981.

[7] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems.* Springer, Berlin, Germany, 2004.

[8] S. Martello and P. Toth. *Knapsack Problems: Algorithms and computer Implementations.* Wiley-Interscience Series in Discrete Mathematics and Optimization, 1990.

[9] G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization.* John Wiley & Sons, Inc, 1988.

[10] N. Perrot. *Advanced IP column generation strategies for the cutting stock stock problem and its variants.* PhD thesis, University Bordeaux 1, 2005.

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

[12] D. Pisinger and P. Toth. Knapsack problems. *Handbook of Combinatorial Optimization*, 1:299–428, 1998.

[13] H. Sural, L.N.Van Wassenhove, and C.N. Potts. The bounded knapsack problem with setups. *INSEAD working paper series*, 97-71-TM, 1997.

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

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

[16] F. Vanderbeck. *Column Generation*, chapter 12. Springer, 2005.

[17] XPress-MP. User guide and reference manual. *Dash Optimization*, Release 12.