Extending Dantzig's bound to the Bounded Multiple-class Binary Knapsack Problem*

François Vanderbeck

Mathématiques Appliquées Bordeaux (MAB), Université Bordeaux 1

351 Cours de la Libération, F-33405 Talence Cedex, France. Fax: +33 (05) 57 96 21 23

Email: fv@math.u-bordeaux.fr Url: http://www.math.u-bordeaux.fr/~fv

May 1998†(revised in March 1999 and in June 2001)

Abstract

The bounded multiple-class binary knapsack problem is a variant of the knapsack problem where the items are partitioned into classes and the item weights in each class are a multiple of a class weight. Thus, each item has an associated multiplicity. The constraints consists of an upper bound on the total item weight that can be selected and upper bounds on the total multiplicity of items that can be selected in each class. The objective is to maximize the sum of the profits associated with the selected items. This problem arises as a sub-problem in a column generation approach to the cutting stock problem. A special case of this model, where item profits are restricted to be multiples of a class profit, corresponds to the problem obtained by transforming an integer knapsack problem into a 0-1 form. However, the transformation proposed here does not involve a duplication of solutions as the standard transformation typically does. The paper shows that the LP-relaxation of this model can be solved by a greedy algorithm in linear time, a result that extends those of Dantzig (1957) and Balas and Zemel (1980) for the 0-1 knapsack problem. Hence, one can derive exact algorithms for the multi-class binary knapsack problem by adapting existing algorithms for the 0-1 knapsack problem. Computational results are reported that compare solving a bounded integer knapsack problem by transforming it into a standard binary knapsack problem versus using the multiple-class model as a 0-1 form.

Keywords: Knapsack problem, integer programming, linear programming relaxation.

*Mathematical Programming, Serie A, 2002. Digital Object Identifier (DOI) 10.1007/s10107-002-0300-7 - (c) Springer-Verlag. The original publication is available on LINK at http://link.springer.de

†Research Paper in Management Studies, WP14/98, University of Cambridge

1 Introduction

The knapsack problem is a linear integer program with a single constraint, and, as such, it is a core model in integer programming (arising as relaxation or separation sub-problem). In its 0-1 version, the variables are restricted to be binary. In the multiple-choice 0-1 knapsack problem, the item set is partitioned into subsets and a solution must include exactly one item in each subset. In the bounded multiple-choice 0-1 knapsack problem, there are restrictions on the number of items that can be selected in each subset. The bounded multiple-class 0-1 knapsack model, that is considered here, is a variation of the latter. The items are partitioned into classes and the item weights in each class are a multiple of a class weight. Thus, each item has an associated multiplicity. The constraints consists of an upper bound on the total item weight that can be selected and upper bounds on the total multiplicity of items that can be selected in each class. The objective is to maximize the sum of the profits associated with the selected items.

To formulate the Bounded Multiple-Class Binary Knapsack Problem (BMCBKP), we define n classes indexed by $i = 1, \ldots n$ and within each class the items are indexed by $j = 1, \ldots, n_i$. Each class, $i = 1, \ldots n$, is characterized by a weight, $w_i \in I\!R_+$, and a prescribed upper bound, $b_i \in I\!R_+$. Each item (i, j), for $i = 1, \ldots n$ and for $j = 1, \ldots, n_i$, is characterized by its profit $p_{ij} \in I\!R_+$ and its multiplicity $m_{ij} \in I\!R_+$. The capacity of the knapsack is denoted by $W \in I\!R_+$. Then, the bounded multiple-class 0-1 knapsack problem takes the form

$$\text{[BMCBKP]} \qquad \max \quad \sum_{i=1}^{n} \sum_{j=1}^{n_i} p_{ij}\, x_{ij} \tag{1}$$

$$\text{s.t.}$$

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

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

$$x_{ij} \in \{0, 1\} \qquad \text{for } i = 1, \ldots, n \text{ and for } j = 1, \ldots, n_i \tag{4}$$

It is assumed that (i) $b_i\, w_i \leq W \;\; \forall i$ (otherwise b_i can be decreased); (ii) $\sum_i b_i\, w_i > W$ (otherwise the problem is trivial); (iii) $m_{ij}\, w_i \leq W \;\; \forall (i, j)$ (otherwise $x_{ij} = 0$); and (iv) $m_{ij} \leq b_i \;\; \forall (i, j)$ (otherwise $x_{ij} = 0$).

The bounded multiple-choice 0-1 knapsack problem can be formulated in a similar way by letting $m_{ij} = 1 \; \forall (i, j)$ in (3), but replacing $m_{ij}\, w_i$ by w_{ij} in (2). The multiple-choice 0-1 knapsack problem is a special case of the latter where $b_i = 1 \; \forall i$. In the 0-1 knapsack problem, $n_i = 1 \; \forall i$ and constraints (3) are not needed. The bounded integer knapsack

problem can be polynomially transformed into a special case of BMCBKP as follows.

Given a bounded integer knapsack problem

$$\max\{\sum_{i=1}^{n} p_i\, y_i :\ \sum_{i=1}^{n} w_i\, y_i \leq W,\ y_i \leq b_i \text{ for } i = 1, \ldots n,\ y \in I\!N^n\}\ , \tag{5}$$

where $b_i \in I\!N$ for each $i = 1, \ldots n$, let

$$n_i = \lfloor \log_2 b_i \rfloor + 1\,,\ m_{i\,j} = 2^{j-1} \ \text{ and } \ p_{i\,j} = m_{i\,j}\, p_i \text{ for } j = 1, \ldots, n_i \ \text{ and } i = 1, \ldots n. \tag{6}$$

Then, a solution y of (5) corresponds to a *unique* solution x of (1-4) and

$$y_i = \sum_{j=1}^{n_i} m_{i\,j}\, x_{i\,j} \qquad \text{for } i = 1, \ldots, n. \tag{7}$$

Observe that the above 0-1 transformation of a bounded integer knapsack problem is different from the *standard* 0-1 transformation, as described in Martello and Toth (1990) for instance. In the latter, the multiplicity of the last 0-1 variable associated to item i is computed so as to insure that the item upper bound is satisfied. I.e.,

$$m_{i\,n_i} = b_i - \sum_{j=1}^{n_i - 1} m_{i\,j} \qquad \text{for } i = 1, \ldots n \tag{8}$$

Then, constraints (3) are not necessary. Thus, the *standard* 0-1 transformation gives rise to a pure 0-1 knapsack problem

$$\max\{\sum_{i=1}^{n}\sum_{j=1}^{n_i} p_{i\,j}\, x_{i\,j} :\ \sum_{i=1}^{n}\sum_{j=1}^{n_i} m_{i\,j}\, w_i\, x_{i\,j} \leq W,\ x\ \in \{0,1\}^{\sum_{i=1}^{n} n_i}\}\ , \tag{9}$$

whose LP solution can be computed efficiently by a greedy algorithm (Dantzig, 1957).

In Section 3, we show that the LP-relaxation of problem BMCBKP can also be solved by a greedy algorithm after ordering of the items by non-increasing profit-per-weight ratio, thereby extending Dantzig's result (1957) for the 0-1 knapsack problem. Moreover, we show that the LP solution can be obtained in linear time by extending the median search procedure proposed by Balas and Zemel (1980) for the 0-1 knapsack problem. These results imply that one can derive exact algorithms for the bounded multiple-class 0-1 knapsack problem by replicating what has been done for the 0-1 knapsack problem. We exemplify this in Section 4. In Section 2, we give some motivations for our interest in model BMCBKP.

2 Motivations

The main interest of model BMCBKP is that it provides a 0-1 form of the integer knapsack problem that does not involve a duplication of solutions. Indeed, with the standard transformation (8), there are typically various ways of decomposing a solution y of (5) into a binary solution x that satisfies (7). Hence, the standard transformation introduces redundant representations of some solutions.

Exact algorithms for the bounded knapsack problem are implicit enumeration approaches such as branch-and-bound or dynamic programming where the way (order) in which the enumeration is carried out is a very crucial element in the success of the method. The formulation of the problem that one uses leads to some "natural" choice of enumeration strategy. In this context, there is an open debate on whether it is better to solve bounded integer knapsack problems directly (based on formulation 5) or through a transformation to a 0-1 form. However, in that debate, the 0-1 form of the bounded knapsack problem that was considered is that of a standard binary knapsack problem because one knew how to solve its LP-relaxation (Dantzig, 1957) in linear time (Balas and Zemel, 1980). But, by introducing redundant representation of solutions, the *standard* 0-1 transformation (8) will impair the efficiency of the implicit enumeration approach. The alternative 0-1 transformation of the bounded knapsack problem to model BMCBKP does not suffer from the same drawback while the LP-relaxation remains solvable in linear time.

Let us briefly summarize the current state of the art for the binary and integer knapsack problem, mentioning a few issues where the 0-1 form proposed here might be useful. To our knowledge the current best algorithm for the 0-1 knapsack problem is that of Martello, Pisinger, and Toth (1997): the *combo* algorithm combines the better of dynamic programming and branch-and-bound approaches; it is fast and robust, solving most classes of 0-1 knapsack problem (including strongly correlated problems) with thousands of items in a fraction of a second. The algorithmic research on bounded integer knapsack problems follows the progresses on 0-1 problems but with some time lag. The latest algorithm to our knowledge is that of Pisinger (2000). We note that some of Pisinger's motivations for developing a direct approach to the integer knapsack problem stem from the drawbacks of the standard 0-1 transformation which do not arise if one uses the transformation proposed here. For instance, Pisinger (2000) notes that the standard transformation does not permit the use of bound reduction techniques in solving the problem, since bounds b_i on class i items are not explicitly in the model. Moreover, even in a direct approach to the integer problem, one might exploit the structure of the 0-1 form. For instance, Pisinger (2000) finds it more efficient to enumerate on the core problem using a dynamic programming recursion for the 0-1 form, because adding 0-1 components one at the time

allows to eliminate some intermediate states by dominance.

Thus, we argue that the debate on whether it is better to solve a bounded integer knapsack based on its integer formulation or based on an equivalent 0-1 formulation should be reviewed in the light of the 0-1 transformation proposed here. The "third way" for solving a bounded integer knapsack problem that arise for this discussion is: *take your favorite algorithm for the 0-1 knapsack problem, adapt it for solving problem BMCBKP, and use it to solve your integer knapsack problem.* Indeed, the result of this paper is that the LP-relaxation of BMCBKP can be solved just as efficiently as that of the standard 0-1 knapsack problem. Thus, one can potentially adapt the other features of an exact algorithm (branch-and-bound / dynamic programming) for the standard binary knapsack problem to derive an efficient exact algorithm for BMCBKP. We have tested this method by adapting the standard branch-and-bound algorithm of Horowith and Sahni. Then, we compared this approach to using a transformation to a standard binary knapsack that we also solved using the algorithm of Horowith and Sahni. We emphasize that the purpose of this test was not to develop a competitive algorithm for the bounded integer knapsack, but simply to compare the solution of an integer knapsack problem through both 0-1 transformations using the same algorithm. Our computational results reported in Section 4 show that the size of the branch-and-bound tree obtained when using the BMCBKP 0-1 form can between 10 and 60 % smaller than that obtained when using the standard 0-1 form.

The applicability of model BMCBKP also concerns the unbounded integer knapsack problem since in many practical case there are implicit bounds that remain small enough to allow a 0-1 transformation. Indeed, in the unbounded version of problem (5), we can set $b_i = \lfloor \frac{W}{w_i} \rfloor$. In the instances for the unbounded integer knapsack problem considered by Martello and Toth (1990), these implicit bounds increase with the number of items, n, because the knapsack capacity is defined as a fraction of the sum of the item weights. However, in many practical applications, such as bin packing, cutting stock, vehicle routing, or other partitioning problems with capacity restrictions, the knapsack capacity does not increase with the number of items involved.

We came to consider model BMCBKP in our work on the development of efficient branching rules for use in a branch-and-price algorithm for the cutting stock problem. When a column generation procedure is used to solve the Gilmore-Gomory LP formulation of the cutting stock problem, the sub-problem is an integer knapsack problem with implicit bounds. To obtain an integer solution to the cutting stock problem one needs to combine the column generation procedure with a branch-and-bound procedure. However, the branching scheme must preserve the tractability of the column generation

sub-problem while yielding a balanced branch-and-bound tree (Vanderbeck, 2000). We found that transforming the integer knapsack problem into a 0-1 problem allows to implement efficient branching rules that do not modify the structure of the sub-problem. The branching scheme used consists in enforcing that the number of cutting patterns with $x_{ij} = 1$ for some chosen item (i, j) is integer. After adding a branching constraint that bounds the number of such cutting patterns used in the solution, the profit of item (i, j) in the column generation sub-problem is modified to account for the dual variable associated with the branching constraint (Vanderbeck, 1999). Thus, in the modified sub-problems that arise in the course of the branch-and-price algorithm, the item profits are no longer a multiple of the class profits, i.e., typically $p_{ij} \neq m_{ij}\, p_i$.

There again, the standard binary knapsack model could be used as a 0-1 form, but we recommend using model BMCBKP instead. Indeed, in this case, the benefits of reducing the redundancies in the solution space can be much more significant: the branching scheme used in the branch-and-price algorithm amounts to defining a partitioning of sub-problem solution space and enforcing the integrality of the number of columns chosen in each parts of this partition; thus, symmetries in the sub-problem solution space also affects the efficiency of the branching scheme for the master problem. The little extra time that might be needed to solve BMCBKP sub-problem rather than standard binary knapsack sub-problems should be largely compensated by the time saving that results from having to consider much fewer branch-and-price nodes. In our study of the cutting stock problem (Vanderbeck, 1999) we did not explicitly compare the two approaches (using the standard binary knapsack versus the BMCBKP model for the sub-problem), we only used model BMCBKP. But, we expect that the reduction in the number of nodes could be similar to that observed for the integer knapsack problem in the computational test that are reported in Section 3.

3 Greedy Algorithm for Computing the LP Bound

Let $m = \sum_{i=1}^{n} n_i$ be the total number of items (i, j) in problem (1-4). Let us order these items according to non increasing values of the profit per unit of weight, and let us renumber the items in that order, using index $k = 1, \ldots, m$. Let us define $\tilde{p}_k = p_{ij}$, $\tilde{w}_k = m_{ij}\, w_i$ and $\tilde{m}_k = m_{ij}$ to denote respectively the profit, the weight, and the multiplicity of the item (i, j) that is the k^{th} position in our ordering. Thus,

$$\frac{\tilde{p}_1}{\tilde{w}_1} \geq \frac{\tilde{p}_2}{\tilde{w}_2} \geq \ldots \geq \frac{\tilde{p}_m}{\tilde{w}_m}\,. \tag{10}$$

Moreover, let $K^i \subset \{1,\ldots,m\}$ denote the set of positions occupied by class i items in that ordering, i.e.,

$$K^i = \{k : \text{ the } k^{th} \text{ item in our ordering is some item } (i,j) \text{ for } j \in \{1,\ldots,n_i\}\} .$$

The following proposition shows that LP relaxation of BMCBKP is solved by a greedy algorithm that consists in considering items in the order (10) and inserting them in the solution at their maximum feasible level. Intuitively, the result follows from the observation that, because of the assumption $w_{ij} = m_{ij}\, w_i$, the greedy ordering (10) is also greedy with regard to the class upper bound constraints (3), i.e., $\frac{\tilde{p}_{k_1}}{\tilde{m}_{k_1}} \geq \frac{\tilde{p}_{k_2}}{\tilde{m}_{k_2}}$ for any two items $k_1 < k_2$ that belong to the same class.

Proposition 1 *A solution of the linear programming relaxation of problem (1-4) is obtained as follows. For $i \in \{1,\ldots,n\}$, let the critical item for class i, $c_i \in K^i$, be such that*

$$\sum_{k \in K^i,\, k<c_i} \tilde{m}_k \leq b_i \quad but \quad \sum_{k \in K^i,\, k \leq c_i} \tilde{m}_k > b_i .$$

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

$$W(l) = \sum_i \sum_{k \in K^i(l)} \tilde{w}_k + \sum_{i \in I(l)} \frac{(b_i - \sum_{k \in K^i(l)} \tilde{m}_k)}{\tilde{m}_{c_i}}\, \tilde{w}_{c_i} .$$

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

$$W(c) \leq W \quad but \quad W(c) + \tilde{w}_c > W \tag{11}$$

and set

$$\begin{aligned}
x_k &= 1 \qquad \text{for } k \in K^i(c) \text{ and } i = 1,\ldots,n,\\
x_{c_i} &= \frac{1}{\tilde{m}_{c_i}} (b_i - \sum_{k \in K^i(c)} \tilde{m}_k) \qquad \text{for } i \in I(c),\\
x_c &= \frac{1}{\tilde{w}_c} (W - W(c)),\\
x_k &= 0 \qquad \text{otherwise.}
\end{aligned}$$

Proof: First, observe that defining c to be the *highest index* item satisfying (11) ensures that if $c = c_i$ for some i, then $\frac{1}{\tilde{w}_{c_i}} (W - W(c_i)) \leq \frac{1}{\tilde{m}_{c_i}} (b_i - \sum_{k \in K^i(c)} \tilde{m}_k)$. Now, we show that the solution x defined above is optimal for the LP relaxation of (1-4) by exhibiting a complementary dual solution of the same value. The dual problem takes the form

$$\min\{W\,\lambda + \sum_{i=1}^{n} b_i\,\mu_i + \sum_{k=1}^{m} \nu_k \ :$$

$$\tilde{w}_k\,\lambda + \tilde{m}_k\,\mu_i + \nu_k \geq \tilde{p}_k \;\; \forall k \in K^i \text{ and } \forall i$$

$$(\lambda, \mu, \nu) \in I\!R_+^{1+n+m} \;\} \; .$$

A dual solution is

$$\lambda = \frac{\tilde{p}_c}{\tilde{w}_c}$$

$$\mu_i = \frac{1}{\tilde{m}_{c_i}} \, (\tilde{p}_{c_i} - \tilde{w}_{c_i}\,\lambda) \quad \text{for } i \in I(c)$$

and zero otherwise, and

$$\nu_k = \tilde{p}_k - \tilde{w}_k\,\lambda - \tilde{m}_k\,\mu_i \quad \text{for} \quad k \in K^i(c) \quad \text{and} \quad i = 1, \ldots, n$$

and zero otherwise. Let us verify that those values are non-negative, that they satisfy the dual constraints, and that they yield an objective value equal to the primal objective value that results from setting x as defined above. Clearly we have $\lambda \geq 0$, $\mu_i \geq 0 \;\; \forall i$, since $\frac{\tilde{p}_{c_i}}{\tilde{w}_{c_i}} \geq \frac{\tilde{p}_c}{\tilde{w}_c}$ for $c_i < c$, and $\nu_k \geq 0 \;\; \forall k$, since $\tilde{m}_{c_i}\,\tilde{w}_k = \tilde{m}_k\,\tilde{w}_{c_i}$ and $\frac{\tilde{p}_k}{\tilde{m}_k} \geq \frac{\tilde{p}_{c_i}}{\tilde{m}_{c_i}}$ for $k \in K^i(c)$ and $i = 1, \ldots n$. One can also verify case by case the inequality $\tilde{w}_k\,\lambda + \tilde{m}_k\,\mu_i + \nu_k \geq \tilde{p}_k$ for each $k \in K^i$ and $i = 1, \ldots n$:

- If $k \geq c$ and $c_i \geq c$, then $\nu_k = \mu_i = 0$ and $\frac{\tilde{p}_c}{\tilde{w}_c} \geq \frac{\tilde{p}_k}{\tilde{w}_k}$.
- If $k \geq c_i$ and $c_i < c$, then $\nu_k = 0$, $\tilde{m}_{c_i}\,\tilde{w}_k = \tilde{m}_k\,\tilde{w}_{c_i}$ and $\frac{\tilde{p}_{c_i}}{\tilde{m}_{c_i}} \geq \frac{\tilde{p}_k}{\tilde{m}_k}$.
- If $k < c$ and $k < c_i$, then $\nu_k = \tilde{p}_k - \tilde{w}_k\,\lambda - \tilde{m}_k\,\mu_i$.

The primal objective is

$$\sum_i \sum_{k \in K^i(c)} \tilde{p}_k + \sum_{i \in I(c)} \tilde{p}_{c_i} \, \frac{1}{\tilde{m}_{c_i}} \, (b_i - \sum_{k \in K^i(c)} \tilde{m}_k) + \tilde{p}_c \, \frac{1}{\tilde{w}_c} \, (W - W(c)) \; .$$

After using $W(c) = \sum_i \sum_{k \in K^i(c)} \tilde{w}_k + \sum_{i \in I(c)} \tilde{w}_{c_i} \, \frac{1}{\tilde{m}_{c_i}} \, (b_i - \sum_{k \in K^i(c)} \tilde{m}_k)$ and regrouping terms, the primal objective takes the form

$$\frac{\tilde{p}_c}{\tilde{w}_c}\,W + \sum_{i \in I(c)} (\tilde{p}_{c_i} - \frac{\tilde{p}_c}{\tilde{w}_c}\,\tilde{w}_{c_i}) \, \frac{1}{\tilde{m}_{c_i}} \, (b_i - \sum_{k \in K^i(c)} \tilde{m}_k) + \sum_i \sum_{k \in K^i(c)} (\tilde{p}_k - \frac{\tilde{p}_c}{\tilde{w}_c}\,\tilde{w}_k)$$

which is equal to $W\,\lambda + \sum_{i=1}^{n} b_i\,\mu_i + \sum_{k=1}^{m} \nu_k$. ∎

Clearly, the greedy solution can be obtained in $O(m \log m)$ because the sorting of the m items is the bottleneck step. But, it can also be computed in linear time by adapting the procedure proposed by Balas and Zemel (1980) for the 0-1 knapsack problem. The key observation is that the optimum value of λ can be determined independently of the μ_i's. Indeed, for each class i, either the class upper bound constraint is active and the knapsack capacity used by class i items is exactly $w_i\,b_i$, or it is not active and it can be ignored. For

a given (tentative) value λ, class i upper bound is active if $\sum_{k\in\{k\in K^i:\frac{\tilde{p}_k}{\tilde{w}_k}>\lambda\}} \tilde{m}_k > b_i$. If class i bound is active, the associated dual value μ_i and primal solution can be determined in $O(n_i)$ by applying Balas and Zemel's procedure to the sub-problem

$$\max\{\sum_{j=1}^{n_i} p_{ij}\, x_{ij} : \sum_{j=1}^{n_i} m_{ij}\, x_{ij} \le b_i,\ 0 \le x_{ij} \le 1\ \forall j\}\ . \tag{12}$$

Hence, one can determine the optimum dual values λ and μ_i's and an associated primal solution to the LP relaxation of BMCBKP using the following procedure, where W denotes the remaining knapsack capacity and b_i denotes the slack to the upper bound for class i.

Step 0: Let $I = \{1, \ldots, n\}$.
Let $K = \{1, \ldots, m\}$.
Let $(x_1, \ldots, x_m) = (0, \ldots, 0)$.
Set W and b_i for all i equal to the rhs of constraints (2) and (3) respectively.

Step 1: Compute the median λ of the values $\{\frac{\tilde{p}_k}{\tilde{w}_k} :\ k \in K\}$.

Step 2: Compute $G^i = \{k \in K^i \cap K :\ \frac{\tilde{p}_k}{\tilde{w}_k} > \lambda\}$ for $i \in I$,
$A(G) = \{i \in I : \sum_{k\in G^i} \tilde{m}_k > b_i\}$, and
$W(G) = \sum_{i\in A(G)} w_i\, b_i + \sum_{i\in I\setminus A(G)} \sum_{k\in G^i} \tilde{w}_k$.
If $W(G) \ge W$ (λ is too small), **then** let $K = \cup_i G^i$;
goto Step 1.
Else, let $K = K \setminus \cup_i G^i$, $W = W - W(G)$,
$x_k = 1$ and $b_i = b_i - \tilde{m}_k$ for $k \in G^i$ and $i \in I \setminus A(G)$,
$I = I \setminus A(G)$, and, for each class $i \in A(G)$, add the solution of (12)
for items $k \in G^i$ to the current primal solution x.

Step 3: Compute $E^i = \{k \in K^i \cap K :\ \frac{\tilde{p}_k}{\tilde{w}_k} = \lambda\}$ for $i \in I$,
$A(E) = \{i \in I : \sum_{k\in E^i} \tilde{m}_k > b_i\}$, and
$W(E) = \sum_{i\in A(E)} w_i\, b_i + \sum_{i\in I\setminus A(E)} \sum_{k\in E^i} \tilde{w}_k$.
If $W(E) < W$ (λ is too large), **then** let $K = K \setminus \cup_i E^i$, $W = W - W(E)$,
$x_k = 1$ and $b_i = b_i - \tilde{m}_k$ for $k \in E^i$ and $i \in I \setminus A(E)$,
$I = I \setminus A(E)$, and, for each class $i \in A(E)$, add an arbitrary
selection of (fraction of) items $k \in E^i$ to the current primal solution x,
so as to satisfy the class upper bound at equality;
goto Step 1.
Else, add an arbitrary selection of (fraction of) items $k \in \cup_i E^i$ to the
current primal solution x, so as to fill the remaining knapsack
capacity W while satisfying the class upper bounds, b_i's.

Observe that, in Step 3, items $k \in E^i$ share the same value of $\frac{\tilde{p}_k}{\tilde{m}_k}$, which is why one can select items arbitrarily in E^i. In Step 2, solving sub-problems of type (12) can

be done $O(\sum_i |G^i|)$ (Balas and Zemel, 1980). In Step 1, finding the median of a set of values can be done in linear time. Every other operation can be done in constant or linear time (Balas and Zemel, 1980). At each iteration of the procedure, (i.e for each *goto Step 1* statement), the number of remaining items, $|K|$, is at least halved. Hence, if we denote by $T(m)$ the running time of our procedure for an instance with m items, we have that $T(m) \leq c_1\, m + c_2 + T(\frac{m}{2})$ for some constants c_1 and c_2. By iteratively applying this inequality, we obtain $T(m) \leq c_1\, m + c_2 + \frac{c_1}{2}\, m + c_2 + T(\frac{m}{4}) \leq \ldots \leq c_1(1 + \frac{1}{2} + \frac{1}{4} + \ldots)\, m + c_2\, \log_2 m + T(1) \leq 2\, c_1\, m + c_2\, \log_2 m + T(1)$. Thus, we have shown that

Proposition 2 *A solution of the linear programming relaxation of problem (1-4) can be obtained in time $O(m)$, where m is the total number of items over all classes.*

In its study of the bounded multiple choice knapsack problem, Pisinger (1996) considers a formulation involving class lower bound constraints or class cardinality constraints for some classes. The result of Proposition 1 can be extended to such a generalization of model BMCBKP, where constraints (3) are replaced by

$$\begin{aligned} \sum_{j=1}^{n_i} m_{i\,j}\, x_{i\,j} &\leq b_i \qquad \text{for } i \in U \\ \sum_{j=1}^{n_i} m_{i\,j}\, x_{i\,j} &= b_i \qquad \text{for } i \in E \\ \sum_{j=1}^{n_i} m_{i\,j}\, x_{i\,j} &\geq b_i \qquad \text{for } i \in L \end{aligned}$$

and the sets U, E, and L are disjoint subsets of the ground set $\{1, \ldots n\}$. Then, it is assumed that $\sum_{i \in E \cup L} b_i\, w_i \leq W$ and $\sum_{i \in U \cup E} b_i\, w_i + \sum_{i \in L} \sum_{j=1}^{n_i} m_{i\,j}\, w_i > W$.

In the case $U \subset \{1, \ldots, n\}$, and $E = L = \emptyset$, one can let $b_i = \infty$ for $i \notin U$, redefine $U = \{1, \ldots, n\}$, and apply Proposition 1. When $E \neq \emptyset$, the optimal LP solution for the class i items with $i \in E$ can be obtained independently of the other classes because their capacity consumption will be $b_i\, w_i$ in any case. For class $i \in E$, the LP solution is given by applying the greedy algorithm to

$$\max\{\sum_{j=1}^{n_i} p_{i\,j}\, x_j : \sum_{j=1}^{n_i} m_{i\,j}\, x_j = b_i\,,\ 0 \leq x_j \leq 1\ \forall j\}\,. \tag{13}$$

Then, items from classes $i \in E$ are removed, the knapsack capacity is reduced by $\sum_{i \in E} b_i\, w_i$. Similarly, if $L \neq \emptyset$, one can solve a sub-problem (13) a priory for classes $i \in L$. Then, the LP relaxation of remaining problem where items of classes $i \in L$ that are in the greedy solution of (13) have been removed (the unused fraction of eventual critical items remain in the problem), the items of classes $i \in E$ have all been removed, and the knapsack capacity has been updated, is solved according to Proposition 1.

4 Computational Experiment

We argued that model BMCBKP is useful in providing an alternative 0-1 form for the bounded integer knapsack problem that does not yield a duplication of solutions and hence should lead to fewer branch-and-bound nodes (or dynamic programming states). Here, we proceed to test that claim, showing that using model BMCBKP instead of the standard 0-1 form of a bounded integer knapsack yields fewer branch-and-bound nodes and LP bound evaluations.

In that purpose, we have generated random instances of the bounded integer knapsack as in Pisinger (2000) and we have proceeded to solve them in two ways: firstly, by transforming them to a standard binary knapsack problem and solving them using the branch-and-bound algorithm of Horowith and Sahni (see Martello and Toth, 1990, pages 30-31 and Nemhauser and Wolsey, 1988, pages 455-456); secondly, by transforming them to a BMCBKP and solving them using an adapted version of Horowith and Sahni branch-and-bound algorithm. We chose the classic algorithm of Horowith and Sahni for it being well-known and for its simplicity (it is easy to adapt it for BMCBKP). Our purpose is clearly not to develop a competitive algorithm for the bounded knapsack problem but just to test the impact of choosing a 0-1 form rather than another.

In brief, the algorithm Horowith and Sahni for the standard binary knapsack problem is a depth first search branch-and-bound procedure. The knapsack items are sorted beforehand in order of non increasing ratio of profit over weight (breaking ties using a non increasing order of the weights). The LP bound is computed at the root node and re-computed only when the branching decisions implies that it may have changed. The branching strategy consists in plunging depth by setting variables to one as long as the remaining capacity allows it (a forward move), then the next variable is set to zero before re-computing the LP-bound. If the node can be pruned, backtracking takes place by returning to the last variable set to one and setting it to zero. Every time the bottom of the tree is reached (all variables being set) the incumbent is updated.

The same procedure can be used to solve the BMCBKP. The only adaptations required consist of computing the LP-bound according to Proposition 1 and keeping track of the remaining class capacities as well as the global capacity. However, in the case of a BMCBKP obtained as the 0-1 form of a bounded knapsack problem, the item profit are a multiple of the class profit, i.e. $p_{ij} = m_{ij}\, p_i \quad \forall (i,j)$. Hence, all items in a class share the same ratio $\frac{p_{ij}}{w_{ij}}$ and can be placed in successive position in the greedy ordering. Then, computing the LP bound is done as for the bounded integer knapsack problem which requires less operations than the method of Proposition 1 valid for a general BMCBKP.

The resulting algorithm is presented in Table 1. We also use this simplification of the LP bound computation when solving the standard 0-1 form.

We generated random instances of the bounded integer knapsack problem with n classes (products), for $n = 100$, 300, and 1000. The weights, w_i, are integer and uniformly distributed in $[1, R]$ with $R = 1000$ or 10000. *Uncorrelated* instances are obtained by generating profits in the same way as the weights, while *correlated* instances are obtained by generating profits, p_i, uniformly distributed in $[w_i - R/10, w_i + R/10]$ such that $p_i > 1$. The upper bounds, b_i, are uniformly distributed in $[5, 10]$, and the knapsack capacity is set equal to a fraction of the weight sum: $W = \frac{\sum_i w_i}{4}$. To compare the two solution methods, we measure the number of branch-and-bound nodes (this number is incremented each time we set a variable to 1 or 0), the number of times we compute the LP-bound, and the time it takes to solve the 0-1 problem. For a given choice of n, R, and correlation, we generate and solve 100 random instances. Then, we compare the total number of B-a-B nodes, bound evaluations, and CPU time for both method and compute the relative difference as a percentage according to: $percentage = \frac{b-s}{b} * 100$, where b is the bigger number and s the smaller.

Table 1: *Pseudo-code of Horowith and Sahni algorithm adapted for the solution of the BMCBKP 0-1 form of a bounded integer knapsack problem*

Step 0: Let $i = 1, \ldots, n$ be the class index in the order of non increasing ratio $\frac{p_i}{w_i}$.
Let $k = 1, \ldots, m$ be the item index in the order of increasing class index and decreasing multiplicity $\tilde{m}_k$ within each class i.
Let $i(k) = i : k \in K^i$ denote the class associated to item $k = (i, j)$.
Let $(x_1, \ldots, x_m) = (y_1, \ldots, y_m) = (0, \ldots, 0)$ (x records the incumbent solution).
Set W and b_i for all i be the rhs of constraints (2) and (3) respectively and $u_i = \sum_{j=1}^{n_i} m_{ij}$. Set $z = 0$, $inc = 0$, $k = 1$.

Step 1: Compute UB: $ub = z;\ c = W;$
for $(i = i(k);\ i \leq n;\ i{+}{+})$ *if* $(\min\{b_i, u_i\} > 0)\{$
if $(\min\{b_i, u_i\}w_i \leq c)$ $\{\ c{-}{=} \min\{b_i, u_i\}w_i;\ ub{+}{=} \min\{b_i, u_i\}p_i;\ \}$
else $\{ub{+}{=} \frac{c}{w_i}p_i;\ break;\ \}\ \}$
if $(ub \leq inc)$ *goto* Step 4;

Step 2: Forward Move:
while $((k \leq n) and\ (\tilde{w}_k <= W) and\ (\tilde{m}_k <= b_{i(k)}))$
$\{\ y_k = 1;\ z{+}{=} \tilde{p}_k;\ W{-}{=} \tilde{w}_k;\ b_{i(k)}{-}{=} \tilde{m}_k;\ u_{i(k)}{-}{=} \tilde{m}_k;\ k{+}{+};\ \}$
if $((k > n) or\ (W == 0)$ /* **leaf node reached** */ $\{$
if $(z > inc)$ $\{\ inc = z;\ x = y;\}$
goto Step 4; $\}$

Step 3: Setting to Zero:
$y_k = 0;\ u_{i(k)}{-}{=} \tilde{m}_k;\ k{+}{+};$ /* **setting an item to 0 reduces** $u_{i(k)}$ */
if $(k > n)$ /* **leaf node reached** */ $\{$
if $(z > inc)$ $\{\ inc = z;\ x = y;\}$
goto Step 4; $\}$
goto Step 1;

Step 4: Backtracking:
$l = k - 1;$
/* **if the last item is set to 1, go back one level** */
if $(y_l == 1)$ $\{\ y_l = 0;\ z{-}{=} \tilde{p}_l;\ W{+}{=} \tilde{w}_l;\ b_{i(l)}{+}{=} \tilde{m}_l;\ u_{i(l)}{+}{=} \tilde{m}_l;\ l{-}{-};\ \}$
/* **go back to last item set to 1** */
for $(; l \geq 1; l{-}{-})$ $\{$ *if* $(y_l == 1)$ *break*; *else* $u_{i(l)}{+}{=} \tilde{m}_l;\ \}$
if $(l == 0)$ *return* $(inc, x);$
/* **set it to 0** */
$y_l = 0;\ z{-}{=} \tilde{p}_l;\ W{+}{=} \tilde{w}_l;\ b_{i(l)}{+}{=} \tilde{m}_l;$
$k = l + 1;$ *goto* Step 1;

In Table 2, we report these percentages. A *negative* percentage, say -50%, indicates that solving the BMCBKP 0-1 form rather than solving the standard 0-1 form lead to a 50% *reduction* of that counter. Thus, our computational results are that across correlated and uncorrelated instances with 100 to 1000 products, using BMCBKP rather than the standard 0-1 form can give rise to a decrease in the size of the branch-and bound tree (and thus the number of bound evaluations) that varies between 10 and 58 percent, while the computation time decreases are a little less significant. The time comparison is not exactly in line with that of the counters because for each item taken in or off the knapsack solution, there is an additional operation to update the class capacity for the BMCBKP; moreover, extra tests must be implemented to check whether an item fits in the remaining class capacity.

Table 2: *Comparison of the resolution of integer knapsack problems by transformation to a binary knapsack versus a multiple class knapsack problem: percentage differences in number of branch-and-bound nodes, upper bounds evaluations, and CPU time when using Horowith and Sahni branch-and-bound algorithm for both binary and multiple class knapsack problems.*

		Uncorrelated		Correlated	
n		R=1000	R=10000	R=1000	R=10000
	BB nodes	-39.0	-46.9	-58.4	-54.5
100	bound eval	-37.8	-46.2	-58.2	-54.3
	CPU time	-27.7	-45.8	-53.6	-54.1
	BB nodes	-25.6	-27.3	-44.2	-43.0
300	bound eval	-24.1	-25.9	-44.1	-42.8
	CPU time	-13.0	-23.5	-39.6	-44.2
	BB nodes	-10.3	-11.6	-15.8	-26.7
1000	bound eval	-5.81	-7.81	-13.3	-25.8
	CPU time	-10.6	-8.84	-12.3	-17.4

Thus, our computational test show that using the BMCBKP 0-1 form rather than the standard 0-1 form can yield a significant reduction in the size of the branch-and-bound tree. The difference get more significant as the problems get harder. In particular, reducing the knapsack capacity yields bigger differences: for example, if we set $W = \frac{\sum_i w_i}{8}$, the results for correlated instance $n = 1000$ and $R = 1000$ are BB nodes = -32.2, bound eval = -31.7, CPU time = -23.2. The difference between the two approaches becomes less significant as n increases. We believe that this is due to item profit and weight being generated within constant intervals: thus, as n increases, there are more items with similar

characteristics which results in both methods having more difficulties and their difference vanishing. The fact that the difference between the 2 approaches is more significant for $R = 10000$ seems to support this analysis.

This comparison between the two 0-1 forms is just indicative as it would probably be somewhat different had we used/adapted another classic algorithm for the binary knapsack problem. In any case, this computational experiment also demonstrates our claim that model BMCBKP can be solved to optimality as easily as a binary knapsack problem: algorithms can be derived by adapting those developed for the binary knapsack and computation time can be expected to be of the same order of magnitude. BMCBKP instances with item profits $p_{i\,j} \neq m_{i\,j}\,p_i$ where solved just as easily in our study of the cutting stock problem (Vanderbeck, 1999).

Acknowledgment

We thank Stefan Scholtes for reviewing the initial draft of this paper and making helpful comments. We are also most grateful to the anonymous referees for their constructive comments. The author was supported in part by a Management Research Fellowship granted by the British Economic and Social Research Council (ESRC).

References

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

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

Martello S., D. Pisinger, and P. Toth (1997). Dynamic Programming and Tight Bounds for the 0-1 Knapsack Problem, *working paper*, DEIS, University of Bologna.

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

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

Pisinger, D. (1996). Dantzig-Wolfe Decomposition for the Bounded Multiple-choice Knapsack Problem, *working paper* (pisinger@diku.dk).

Pisinger, D. (2000). A minimal algorithm for the bounded knapsack problem, *INFORMS journal of computing*, 12 75-84.

Vanderbeck, F. (1999). Computational Study of a Column Generation algorithm for Bin Packing and Cutting Stock problems, *Mathematical Programming*, Ser. A, **86**, pp565-594.

Vanderbeck, F. (2000). On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm, *Operations Research*, Vol. **48**, No. 1., pp111-128.