# Budgeting with Bounded Multiple-choice Constraints

David Pisinger
Dept. of Computer Science, University of Copenhagen,
Universitetsparken 1, DK-2100 Copenhagen, Denmark

March 1998

**Abstract**

We consider a budgeting problem where a specified number of projects from some disjoint classes has to be selected such that the overall gain is largest possible, and such that the costs of the chosen projects do not exceed a fixed upper limit. The problem has several application in government budgeting, planning, and as relaxation from other combinatorial problems.

It is demonstrated that the problem can be transformed to an equivalent Multiple-choice Knapsack Problem through dynamic programming. A naive transformation however leads to a drastic increase in the number of variables, thus we propose an algorithm for the continuous problem based on Dantzig-Wolfe decomposition: A master problem solves a Continuous Multiple-choice Knapsack Problem knowing only some extreme points in each of the transformed classes. The individual subproblems find extreme points for each given direction, using a median search algorithm.

An integer optimal solution is then derived by using the dynamic programming transformation to a Multiple-choice Knapsack Problem for an expanding core. The individual classes are considered in an order given by their gradients, and the transformation to a Multiple-choice Knapsack Problem is performed when needed. In this way, only a dozen of classes need to be transformed for standard instances from the literature.

Computational experiments are presented, showing that the developed algorithm is orders of magnitude faster than a general LP/MIP algorithm.

**Subject Classification:**
Programming, integer, algorithms: Dantzig-Wolfe decomposition;
Dynamic programming: Bounded Multiple-choice Knapsack Problem;

## 1 Introduction

Consider a budgeting problem with $k$ classes $N_1, \ldots, N_k$ of *projects*. It is demanded that *at most* (resp. *at least* or *exactly*) $a_i$ projects should be selected from class $N_i$. Each project $j \in N_i$ has an associated gain (*profit*) $p_{ij}$ and cost (*weight*) $w_{ij}$ and the problem is to choose the appropriate number of projects from each class such that the largest gain is obtained without exceeding a limit $c$ on the cost. Thus the *Budgeting Problem with Bounded Multiple-choice Constraints* (BBMC) may be formulated as

$$\begin{array}{ll} \text{maximize} & z = \sum_{i=1}^{k} \sum_{j \in N_i} p_{ij} x_{ij} \\ \text{subject to} & \sum_{i=1}^{k} \sum_{j \in N_i} w_{ij} x_{ij} \leq c, \\ & \sum_{j \in N_i} x_{ij} \leq a_i, \quad i \in L, \\ & \sum_{j \in N_i} x_{ij} \geq a_i, \quad i \in G, \\ & \sum_{j \in N_i} x_{ij} = a_i, \quad i \in E, \\ & x_{ij} \in \{0,1\}, \quad i = 1, \ldots, k, \quad j \in N_i. \end{array} \tag{1}$$

The sets $L, G, E$ must be disjunctive and $L \cup G \cup E = K$ where $K = \{1, \ldots, k\}$. All coefficients $p_{ij}, w_{ij}$, and $c$ are positive integers, and the classes $N_1, \ldots, N_k$ are mutually disjoint, class $N_i$ having *size* $n_i$. The *total* number of items is $n = \sum_{i=1}^{k} n_i$. If we relax the integrality constraint $x_{ij} \in \{0,1\}$ in (1) to $0 \leq x_{ij} \leq 1$ we obtain the *Continuous Budgeting Problem with Bounded Multiple-choice constraints* (*CBBMC*).

To avoid unsolvable or trivial situations we assume that

$$\sum_{i \in E \cup G} \underline{v}_i \leq c < \sum_{i \in E \cup L} \overline{v}_i + \sum_{i \in G} \sum_{j \in N_i} w_{ij} \tag{2}$$

where $\underline{v}_i$ is the weight sum of the $a_i$ lightest items in class $N_i$ and $\overline{v}_i$ is the weight sum of the $a_i$ heaviest items in the class.

The Budgeting Problem with Bounded Multiple-choice constraints is a generalization of the *Multiple-choice Knapsack Problem* in which $E = K$ and $a_i = 1$ for $i = 1, \ldots, k$. If each class has exactly two items, where one of these has $(p_{i1}, w_{i1}) = (0, 0)$, $i = 1, \ldots, k$ and additionally $a_i = 1$ for each class $K = E$, the problem (1) corresponds to the *0-1 Knapsack Problem (KP)*. The continuous relaxation of KP will be denoted by *CKP*.

The BBMC has a wide variety of applications in budgeting, where a number of projects from each class $N_i$ has to be selected, such that the overall gain is largest possible, and such that the costs demanded for the chosen projects do not exceed a fixed upper limit. It may be applied in Sequencing and Scheduling Problems as well as in Strategic Production Planning. BPBMC has several application in the social system, as it may be used for Hospital Planning with production constraints in each department, or Government Budgeting with demands in different sectors. Moreover BBMC contains several classical Knapsack Problems as special cases, among which we should mention the 0-1 Knapsack Problem, Subset-sum Problem, Bounded Knapsack Problem, Multiple-choice Knapsack

Problem. So all applications for these problems (see e.g. [6], and [8]) are immediately adaptable to this general model.

BBMC is $\mathcal{NP}$-hard as it contains KP as a special case. In this paper we will show that BBMC may be transformed to an ordinary MCKP through dynamic programming, which then is solved by use of a minimal-core dynamic programming algorithm from [8].

Section 2 will present the transformation of BBMC to MCKP, and show several fundamental properties for MCKP. Section 3 shows how Dantzig-Wolfe decomposition may be used to solve the continuous BBMC without transforming classes to MCKP. Finally Section 4 describes how gradients in each class may be used to define a core of the problem, and how the same gradients may be used for reductions of states in a dynamic programming algorithm. Some computational experiments are presented in Section 5.

A first version of this paper was presented at AIRO'96 [9].

# 2 Transformation to MCKP

Every BBMC can be put on a form with equalities in all the cardinality constraints:

$$\begin{array}{ll} \text{maximize} & z = \sum_{i=1}^{k} \sum_{j \in N_i} p_{ij} x_{ij} \\ \text{subject to} & \sum_{i=1}^{k} \sum_{j \in N_i} w_{ij} x_{ij} \leq c, \\ & \sum_{j \in N_i} x_{ij} = a_i, \quad i = 1, \ldots, k, \\ & x_{ij} \in \{0,1\}, \quad i = 1, \ldots, k, \quad j \in N_i. \end{array} \tag{3}$$

The transformation is a stepwise modification of the constraints and item profits/weights:

**step 1** A class $N_i$ with a constraint $\sum_{j \in N_i} x_{ij} \geq a_i$ is modified to a constraint of the form

$$\sum_{j \in N_i} x_{ij} \leq n - a_i \tag{4}$$

by adding $\sum_{j \in N_i} p_{ij}$ to the objective function and subtracting $\sum_{j \in N_i} w_{ij}$ from the capacity $c$. In addition, all items $j \in N_i$ change sign to $(-p_{ij}, -w_{ij})$. In this way we have changed the problem of choosing at least $a_i$ items to the problem of not choosing at most $n - a_i$ items.

**step 2** A class $N_i$ with a constraint $\sum_{j \in N_i} x_{ij} \leq a_i$ is easily modified to equality constraint

$$\sum_{j \in N_i} x_{ij} = a_i \tag{5}$$

by adding $a_i$ new items to the class which have profit and weight equal to zero.

**step 3** Negative profits and weights are handled by adding a sufficiently large constant to all items in a class $N_i$. Let $p_i^m = \min_{j \in N_i} p_{ij}$ and $w_i^m = \min_{j \in N_i} w_{ij}$. Replace all item profits and weights by $(p_{ij} - p_i^m, w_{ij} - w_i^m)$. This does not affect the optimal solution since a specific number of items is chosen in each class. We must however add the constant $a_i p_i^m$ to the objective function, and subtract $a_i w_i^m$ from the capacity for each class $N_i$.

In the rest of the paper we will assume that BBMC is on the standard form (3).

A BBMC on the standard form may be transformed to an equivalent MCKP as follows: In each class $N_i$ we construct all undominated sets of $\alpha$ items through dynamic programming for $\alpha = 1, \ldots, a_i$. Thus let $f^i_{h,\alpha}(\tilde{c})$; $h = 1, \ldots, n_i$; $\alpha = 1, \ldots, a_i$; $\tilde{c} = 0, \ldots, c$ be an optimal solution to the following cardinality constrained knapsack problem defined on the first $h$ items of class $N_i$, and with exactly $\alpha$ chosen items:

$$f^i_{h,\alpha}(\tilde{c}) = \max \left\{ \begin{array}{l} \sum_{j=1}^{h} p_{ij} x_{ij} : \sum_{j=1}^{h} w_{ij} x_{ij} = \tilde{c};\ \sum_{j=1}^{h} x_{ij} = \alpha; \\ x_{ij} \in \{0,1\},\ j = 1, \ldots, h \end{array} \right\}. \tag{6}$$

Initially we set $f^i_{0,0}(\tilde{c}) = 0$ for all $\tilde{c} = 0, \ldots, c$, while subsequent values of $f^i_{h,\alpha}$ are found by using the recursion:

$$f^i_{h,\alpha}(\tilde{c}) = \begin{cases} f^i_{h-1,\alpha}(\tilde{c}) & \text{if } \alpha = 0 \text{ or } \tilde{c} - w_{ih} < 0\ , \\ \max\{f^i_{h-1,\alpha}(\tilde{c}),\ f^i_{h-1,\alpha-1}(\tilde{c} - w_{ih}) + p_{ih}\} & \text{if } \alpha > 0,\ \tilde{c} - w_{ih} \geq 0\ . \end{cases} \tag{7}$$

We may define an equivalent MCKP by for each class $N_i$, and for each weight $j = 0, \ldots, c$ setting (see Fig. 1):

$$\overline{w}_{ij} = j; \qquad \overline{p}_{ij} = f^i_{n_i,a_i}(j)\ , \tag{8}$$

In this way we obtain a MCKP with classes $\overline{N}_i$, and the problem is to maximize the profit sum by choosing exactly one item from each class without exceeding the capacity constraint [6]:

$$\begin{array}{ll} \text{maximize} & z = \sum_{i=1}^{k} \sum_{j \in \overline{N}_i} \overline{p}_{ij} \overline{x}_{ij} \\ \text{subject to} & \sum_{i=1}^{k} \sum_{j \in \overline{N}_i} \overline{w}_{ij} \overline{x}_{ij} \leq c, \\ & \sum_{j \in \overline{N}_i} \overline{x}_{ij} = 1, \quad i = 1, \ldots, k, \\ & \overline{x}_{ij} \in \{0,1\}, \quad i = 1, \ldots, k, \quad j \in \overline{N}_i. \end{array} \tag{9}$$

Class $\overline{N}_i$ has size $\overline{n}_i = c$, thus the transformation runs in $O(n_i a_i c)$ in each class of BBMC, using totally $O(c \sum_{i=1}^{k} n_i a_i)$ time. There are $c$ items in each class of (9), so the MCKP can be solved in $O(kc^2)$ through dynamic programming [8]. This gives a total solution time for BBMC of $O(c \sum_{i=1}^{k} n_i a_i + kc^2)$.

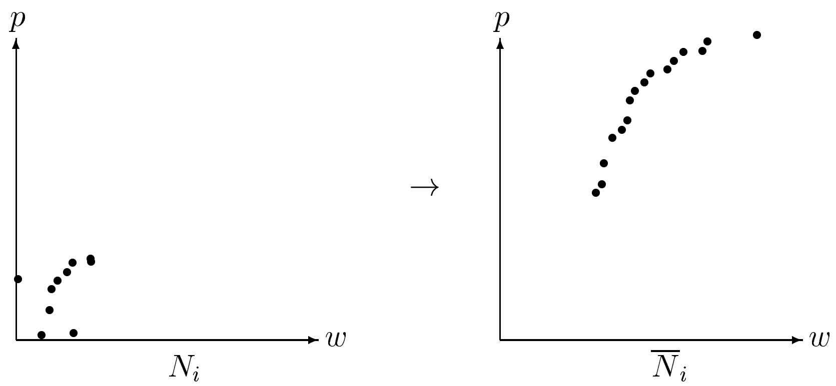


Figure 1: Transformation $N_i$ to $\overline{N}_i$. Each entry in $\overline{N}_i$ corresponds to the sum of $a_i$ items from $N_i$, dominated entries are deleted. In the present example $a_i = 4$

## 2.1 Properties of MCKP and BBMC

Having transformed the BBMC to an equivalent MCKP, well-known properties of the latter problem can be used to further restrict the solution space. We have:

**Definition 1** If two items $r$ and $s$ in the same class $\overline{N}_i$ satisfy that

$$\overline{w}_{ir} \le \overline{w}_{is} \text{ and } \overline{p}_{ir} \ge \overline{p}_{is} , \tag{10}$$

then we say that item $r$ *dominates* item $s$. Similarly if some items $r, s, t \in \overline{N}_i$ with $\overline{w}_{ir} \le \overline{w}_{is} \le \overline{w}_{it}$ and $\overline{p}_{ir} \le \overline{p}_{is} \le \overline{p}_{it}$ satisfy

$$\frac{\overline{p}_{it} - \overline{p}_{ir}}{\overline{w}_{it} - \overline{w}_{ir}} \ge \frac{\overline{p}_{is} - \overline{p}_{ir}}{\overline{w}_{is} - \overline{w}_{ir}} , \tag{11}$$

then we say that item $s$ is *continuously dominated* by items $r$ and $t$.

**Proposition 1** (Sinha and Zoltners [10]) Given two items $r, s \in \overline{N}_i$. If item $r$ dominates item $s$ then an optimal solution to MCKP with $\overline{x}_{is} = 0$ exists. If two items $r, t \in \overline{N}_i$ continuously dominate an item $s \in \overline{N}_i$ then an optimal solution to CMCKP with $\overline{x}_{is} = 0$ exists.

**Proposition 2** (Sinha and Zoltners [10]) An optimal solution to CMCKP satisfies the following: (a) The solution has at most two fractional variables $\overline{x}_{fb}$ and $\overline{x}_{fb'}$. (b) If it has two fractional variables they must be in the same class $\overline{N}_f$.

To characterize solutions to CMCKP further, let us consider some results from 0-1 Knapsack Problems. Balas and Zemel [1] showed that the CKP may be solved in $O(n)$ through a partitioning technique which seeks a parameter $\lambda$ such that when we partition the items $\{1, \ldots, n\}$ in $S = \{j : p_j/w_j > \lambda\}$, $T = \{j : p_j/w_j = \lambda\}$ and $U = \{j : p_j/w_j < \lambda\}$ then $\sum_{j \in S} w_j < c \le \sum_{j \in S \cup T} w_j$. The objective value of the continuous solution is then $\sum_{j \in S} p_j + \lambda(c - \sum_{j \in S} w_j)$.

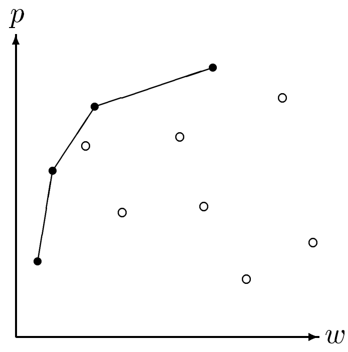


Figure 2: A class $\overline{N}_i$, where the white items are continuously dominated by the black items. The undominated items $\overline{R}_i$ (black) form the upper convex boundary of $\overline{N}_i$.

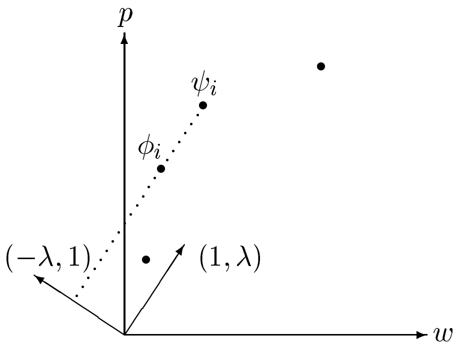


Figure 3: Projection of $(\overline{w}_{ij}, \overline{p}_{ij}) \in \overline{R}_i$ on $(-\lambda, 1)$. In this case $\overline{M}_i = \{\phi_i, \psi_i\}$, $\phi_i$ is the lightest item in $\overline{M}_i$, and $\psi_i$ is the heaviest item in $\overline{M}_i$.

Now for the MCKP assume that all continuously dominated items have been removed according to Proposition 1, leaving back the undominated items $\overline{R}_i$ (Fig. 2). Then due to the convexity of each class one can see the continuous problem as a search for a $\lambda$ such that

$$\sum_{i=1}^{k} \overline{w}_{i\phi_i} < c \leq \sum_{i=1}^{k} \overline{w}_{i\psi_i} \tag{12}$$

where

$$\begin{aligned} &\phi_i = \arg\min\nolimits_{j \in \overline{M}_i} \overline{w}_{ij}, \qquad \psi_i = \arg\max\nolimits_{j \in \overline{M}_i} \overline{w}_{ij}, \qquad i = 1, \ldots, k \\ &\overline{M}_i = \left\{ j \in \overline{R}_i : (\overline{p}_{ij} - \lambda \overline{w}_{ij}) = \max\nolimits_{\ell \in \overline{R}_i} (\overline{p}_{i\ell} - \lambda \overline{w}_{i\ell}) \right\}, \quad i = 1, \ldots, k \end{aligned} \tag{13}$$

Thus $\overline{M}_i$ is the items in $\overline{R}_i$ which have the largest projection of $(\overline{w}_{ij}, \overline{p}_{ij})$ at $(-\lambda, 1)$ and $\phi_i$ resp. $\psi_i$ are the lightest/heaviest among the items in $\overline{M}_i$ (Fig. 3). The objective value of the CMCKP is then defined as

$$z_{\text{CMCKP}} = \sum_{i=1}^{k} \overline{p}_{i\phi_i} + \lambda \left( c - \sum_{i=1}^{k} \overline{w}_{i\phi_i} \right). \tag{14}$$

We have previously seen that the transformation of BBMC to MCKP is expensive, thus the formulation (12) to (13) cannot directly be used for solving BBMC. For a given value of $\lambda$ it is however easy for each class $N_i$ to find the corresponding extreme point in $\overline{N}_i$ which has the largest projection on $(-\lambda, 1)$. The items in $\overline{M}_i$ are determined by

$$\max_{\ell \in \overline{R}_i} (\overline{p}_{i\ell} - \lambda \overline{w}_{i\ell}) = \max_{B \subset N_i, |B| = a_i} \left( \sum_{j \in B} p_{ij} - \lambda \sum_{j \in B} w_{ij} \right) = \max_{B \subset N_i, |B| = a_i} \sum_{j \in B} (p_{ij} - \lambda w_{ij}) \ , \quad (15)$$

i.e. an extreme point $(\overline{p}_{i\ell}, \overline{w}_{i\ell}) \in \overline{N}_i$ in direction $(-\lambda, 1)$ is the sum of the $a_i$ items in $N_i$ which have largest projection on $(-\lambda, 1)$. To find the $a_i$ items in $N_i$ having the largest projection, it is sufficient to order the items in $N_i$ according to

$$\Pi_{ij}(\lambda) = p_{ij} - \lambda w_{ij} \ , \quad (16)$$

choosing the $a_i$ first entries in this list.

To find the extreme points with smallest/largest weights sums, i.e. $(\overline{p}_{i\phi_i}, \overline{w}_{i\phi_i})$ resp. $(\overline{p}_{i\psi_i}, \overline{w}_{i\psi_i})$ we simply use the same ordering (16), breaking ties such that the smallest resp. largest weights are chosen first. Using a median search algorithm to find the $a_i$ first items with the given ordering, the extreme points in $\overline{N}_i$ can be found in $O(n_i)$ time.

Let us end this section with a characterization of solutions to CBBMC:

**Proposition 3** There exists a LP-optimal solution which satisfies the following: (a) There are zero or two fractional variables. (b) If there are two fractional variables then they belong to the same class $\overline{N}_f$.

**Proof** Transform the BBMC to an equivalent MCKP through dynamic programming. continuously dominated states will never lead to an LP-optimal solution, thus they may be removed. Solve the continuous MCKP. Due to Proposition 2, zero or two decision variables in CMCKP will be fractional. If there are zero fractional variables, then there will neither be any fractional variables in the LP-optimal solution to BBMC.

Thus assume that there are two fractional variables in the LP-optimal solution to MCKP. The variables will be found in the same set $\overline{N}_f$. Let $\overline{x}_{fr}$ and $\overline{x}_{fs}$ be those two variables and let $X_r = \{x_{f,r_1}, \ldots, x_{f,r_{a_f}}\}$ resp. $X_s = \{x_{f,s_1}, \ldots, x_{f,s_{a_f}}\}$ be the variables in BBMC which correspond to the two chosen states in MCKP. Obviously $|X_r| = |X_s| = a_f$ and $X_r \subset N_f$, $X_s \subset N_f$.

Assume that the items are ordered according to their projection (16), and let $\gamma$ be the projection of the last item $x_{f,r_{a_f}}$. Those items $x_{r_k}$ which have a larger projection than $\gamma$ will be present in both sets and thus $x_{r_k} = 1$ in an optimal solution to CBBMC. The remaining items will all have the same projection $\gamma$. Then an LP-optimal solution may be constructed by repeatedly exchanging an lighter item $x_{r_k}$ with a heavier item $x_{s_\ell}$. When two items are met where an exchange will lead to an overfilled knapsack, these are the fractional variables, and the continuous solution is a convex combination of $x_{r_k}$ and $x_{s_\ell}$ such that $x_{r_k} + x_{s_\ell} = 1$ and such that the total weight is applied. □

# 3 Dantzig-Wolfe decomposition for the CBBMC

The continuous BBMC can be solved through a dynamic programming transformation to MCKP. The continuous MCKP can then be solved in linear time (measured in the number of items in the transformed classes) by applying the algorithms by Dyer [4] or Zemel [11]. Since the number of items introduced by the dynamic programming transformation is $\sum_{i=1}^{k} \overline{n}_i = kc$, the CMCKP will be solved in $O(kc)$ and thus the continuous BBMC is solved in $O(c \sum_{i=1}^{k} n_i^2 + kc)$ when we take into account the time used for the transformation.

Better solution times can be obtained through Dantzig-Wolfe decomposition (Dantzig and Wolfe [3]). A master problem solves a continuous MCKP, while the individual subproblems find extreme points for each class $\overline{N}_i$. The algorithm is a kind of binary search for the parameter $\lambda$ which satisfies (12) to (13).

As seen in the proof of Proposition 3, we know that $\lambda$ will correspond to a quotient

$$\lambda = \frac{\alpha}{\beta} = \frac{p_{ir} - p_{is}}{w_{ir} - w_{is}} \tag{17}$$

where $r, s$ are two items in the same class $N_i$. If $P$, $W$ are the largest profit resp. weight of an item, then we know that $\lambda = \frac{\alpha}{\beta}$ should be found among values $0 \leq \alpha \leq P$ and $1 \leq \beta \leq W$, thus we may use binary search to determine the optimal value of $\lambda$. Let initially $[\Lambda_1, \Lambda_2] = [0, P]$ be the interval in which $\lambda$ should be found and consider the following algorithm for the master problem:

**step 1** Let $\lambda = \frac{\alpha}{\beta}$ be a rational number in the interval $[\Lambda_1, \Lambda_2]$. Such a value may be derived efficiently using the function `small_rational_between` from [5]. Derive $\phi_i, \psi_i$ and $\overline{M}_i$ as given by (13) in each class by solving the subproblem (15). If the constraint (12) is satisfied, we may stop since $\lambda$ defines the optimal solution.

**step 2** Let $\lambda = (\Lambda_1 + \Lambda_2)/2$. For each class $N_i$ derive $\phi_i, \psi_i$ and $\overline{M}_i$ as given by (13) for this value of $\lambda$. If constraint (12) is satisfied, then we may terminate with an optimal value of $\lambda$. Otherwise, if $\sum_{i=1}^{k} \overline{w}_{i\phi_i} > c$ we know that $\lambda$ is too small thus setting $\Lambda_1 = \lambda$. In a similar way, if $\sum_{i=1}^{k} \overline{w}_{i\psi_i} < c$ we set $\Lambda_2 = \lambda$. Go to step 1.

In order to analyze the time complexity of the algorithm, we have previously noticed that there are $O(PW)$ possible values of $\lambda = \alpha/\beta$. A binary search among these values will demand $O(\log(PW))$ iterations of the master problem. Solving the subproblems takes $\sum_{i=1}^{k} O(n_i) = O(n)$ time for each value of $\lambda$, thus the whole algorithm runs in $O(n \log P + n \log W)$ which is polynomial in the input size.

The decomposition shows that we are able to solve the continuous BBMC without knowing the items in each class $\overline{N}_i$ and in particular we do not need to transform $N_i$ to $\overline{N}_i$. Indeed, we do not even need to know all continuously undominated items (extreme points) as these are generated "on the run" in $O(n_i)$ time. The solution times for the continuous BBMC have been sketched in Figure 4. Problems with $n_i = 100$ items in each of the $k$ classes can be solved in a couple of seconds, and the solution times seem to grow linearly with the size of the problem.

| $k$ | *time* | log *time* |
|---:|---:|---:|
| 10 | 0.004 | -2.40 |
| 30 | 0.01 | -2.00 |
| 100 | 0.03 | -1.52 |
| 300 | 0.09 | -1.05 |
| 1000 | 0.32 | -0.49 |
| 3000 | 1.05 | 0.02 |
| 10000 | 3.61 | 0.56 |
| 30000 | 14.78 | 1.17 |
| 100000 | 48.79 | 1.69 |

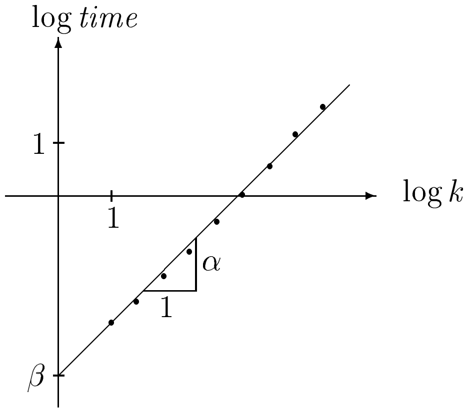


Figure 4: Solution times (in seconds) for solving the continuous problem through Dantzig-Wolfe decomposition. The graph depicts log *time* as a function of $\log k$ for uncorrelated instances with $n_i = 100$ items. Profits and weights are randomly distributed in $[1, 1000]$, and $a_i$ is randomly distributed in $[1, 20]$. Times are average values of 100 instances. The solution times seem to grow linearly with $k$ as $\alpha = 1.0$ and $\beta = -3.4$, thus the solution times can be approximated by $t(k) = 10^{\beta} k^{\alpha} = 0.000398k$

# 4 Solving BBMC to integer optimality

In the previous section we saw that Dantzig-Wolfe decomposition may be used to solve the continuous BBMC efficiently. Obtaining an integer solution is however more difficult, since this step somehow demands the transformation (8) of classes $N_i$ to $\overline{N}_i$.

If a *core* is used for solving the transformed BBMC, only the classes in the core need to be transformed according to (8) and hopefully the size of the core is much smaller than $k$. We will apply the core algorithm presented in Pisinger [8]. The algorithm is based on dynamic programming, where the search is focused on a small number of classes where there is a large probability for finding an optimal solution. Since it is difficult to say in advance how large a core should be chosen, the algorithm initially start with one class $\overline{N}_f$ in the core, and gradually expands the core according to some greedy rules. These greedy rules are based on the concept of *forward* and *backward gradients* $\lambda_i^+, \lambda_i^-$ defined as follows:

$$\lambda_i^+ = \max_{j \in \overline{N}_i,\ \overline{w}_{ij} > \overline{w}_{ib_i}} \frac{\overline{p}_{ij} - \overline{p}_{ib_i}}{\overline{w}_{ij} - \overline{w}_{ib_i}}, \qquad i \neq f, \tag{18}$$

$$\lambda_i^- = \min_{j \in \overline{N}_i,\ \overline{w}_{ij} < \overline{w}_{ib_i}} \frac{\overline{p}_{ib_i} - \overline{p}_{ij}}{\overline{w}_{ib_i} - \overline{w}_{ij}}, \qquad i \neq f, \tag{19}$$

where item $b_i \in \overline{N}_i$ is the continuously optimal choice in the set, and class $\overline{N}_f$ is the fractional class with up to two fractional variables.

The forward gradient $\lambda_i^+$ is a measure of the largest possible gain per weight unit by choosing a heavier items in class $\overline{N}_i$ instead of the continuously optimal item $b_i$. Similarly the backward gradient $\lambda_i^-$ is a measure of the smallest possible loss per weight unit by

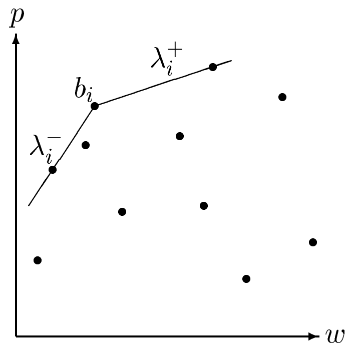


Figure 5: Gradients $\lambda_i^+, \lambda_i^-$ in class $\overline{N}_i$.

choosing a lighter item in class $\overline{N}_i$. The gradients can be derived efficiently without transforming classes $N_i$ to $\overline{N}_i$. Again we use the technique of deriving extreme points from previous section using an iterative algorithm to derive $\lambda_i^+$:

Set $\lambda^* = 0$.
**repeat**
  Set $\lambda_i^+ = \lambda^*$.
  Find an extreme point $(P, W) \in \overline{N}_i$ in direction $\lambda = \lambda_i^+$ according to (15).
  Set $\lambda^* = (P - \overline{p}_{ib_i})/(W - \overline{w}_{ib_i})$.
**until** $(\lambda_i^+ = \lambda^*)$

A similar approach may be used for $\lambda_i^-$, where we iterate from the initial value $\lambda_i^- = \infty$.

Now the main algorithm for the exact solution of BBMC runs as follows: First, we derive the gradients for all classes, and define the core as $C = \overline{N}_f$, where class $f$ is the fractional class. In each iteration we choose the class with largest $\lambda_i^+$ resp. smallest $\lambda_i^-$ and add it to the core. The chosen class $N_i$ first must be transformed to the corresponding set $\overline{N}_i$, and then all combinations are enumerated through dynamic programming, using the recursion from [8].

The gradients are also used for fathoming states in the dynamic programming. Let $\overline{N}_s$ be the class with smallest $\lambda_i^-$ among the classes not yet enumerated, and let $\overline{N}_t$ be the class with largest value $\lambda_i^+$. By this assumption we get the following upper bound on a state in the dynamic programming with profit sum $\pi$ and weight sum $\mu$:

$$u(\pi, \mu) = \begin{cases} \pi + (c - \mu)\lambda_t^+ & \text{if } \mu \leq c, \\ \pi + (c - \mu)\lambda_s^- & \text{if } \mu > c. \end{cases} \tag{20}$$

The state is fathomed if $\lfloor u(\pi, \mu) \rfloor \leq z$, where $z$ is the so far best solution found in the dynamic programming recursion. For conveniency we set $\lambda_t^+ = 0$ and $\lambda_s^- = \infty$ when all classes have been enumerated to fathom states which cannot be improved further.

# 5 Computational Results

To test the actual performance of the presented BBMC algorithm, it has been implemented in C. The algorithm assumes that BBMC is on the standard form (3).

Table I: Time used for solving continuous problem. Average of 100 instances

| | | Uncorrelated | | Weakly corr. | | Subset-sum | | Zig-zag | |
|---|---|---|---|---|---|---|---|---|---|
| $k$ | $n_i$ | $R:100$ | 1000 | $R:100$ | 1000 | $R:100$ | 1000 | $R:100$ | 1000 |
| 10 | 10 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 100 | 10 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 1000 | 10 | 0.03 | 0.03 | 0.04 | 0.04 | 0.02 | 0.02 | 0.04 | 0.04 |
| 10000 | 10 | 0.38 | 0.37 | 0.47 | 0.45 | 0.24 | 0.24 | 0.47 | 0.48 |
| 10 | 100 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 100 | 100 | 0.03 | 0.03 | 0.04 | 0.03 | 0.01 | 0.01 | 0.03 | 0.04 |
| 1000 | 100 | 0.32 | 0.33 | 0.38 | 0.38 | 0.14 | 0.15 | 0.34 | 0.39 |
| 10000 | 100 | 3.53 | 3.61 | 4.08 | 4.02 | 1.56 | 1.56 | 3.89 | 4.32 |

Table II: Number of sets which have been enumerated (core size). Average of 100 instances

| | | Uncorrelated | | Weakly corr. | | Subset-sum | | Zig-zag | |
|---|---|---|---|---|---|---|---|---|---|
| $k$ | $n_i$ | $R:100$ | 1000 | $R:100$ | 1000 | $R:100$ | 1000 | $R:100$ | 1000 |
| 10 | 10 | 3 | 3 | 4 | 7 | 1 | 1 | 4 | 5 |
| 100 | 10 | 6 | 12 | 6 | 11 | 1 | 2 | 6 | 16 |
| 1000 | 10 | 6 | 17 | 5 | 9 | 1 | 2 | 6 | 14 |
| 10000 | 10 | 11 | 10 | 5 | 9 | 1 | 2 | 10 | 12 |
| 10 | 100 | 3 | 5 | 2 | 3 | 0 | 0 | 1 | 6 |
| 100 | 100 | 3 | 12 | 2 | 3 | 0 | 0 | 1 | 8 |
| 1000 | 100 | 4 | 8 | 2 | 3 | 0 | 0 | 2 | 6 |
| 10000 | 100 | 6 | 10 | 2 | 4 | 0 | 0 | 2 | 8 |

We will consider four different instance types taken from the literature [8,10]. Each instance is tested with coefficients in the *range* $R = 100$ or 1000 and different numbers of classes $k$ and sizes $n_i$:

- *Uncorrelated instances:* In each class we generate $n_i$ items by choosing $w_{ij}$ and $p_{ij}$ randomly in $[1, R]$.
- *Weakly correlated instances:* In each class, $w_{ij}$ is randomly distributed in $[1, R]$ and $p_{ij}$ is randomly distributed in $[w_{ij} - 10, w_{ij} + 10]$, such that $p_{ij} \geq 1$.
- *Subset-sum instances:* $w_{ij}$ is randomly distributed in $[1, R]$ and $p_{ij} = w_{ij}$.
- *Zig-zag:* Sinha and Zoltners [10] constructed some special instances which have very few dominated items. For each class $i$, $n_i$ items $(w'_j, p'_j)$ are constructed as in the uncorrelated case, and the profits and weights are oredered in increasing order. Finally we set $w_{ij} = w'_j$ and $p_{ij} = p'_j$ for $j = 1, \ldots, n_i$.

The cardinalities $a_i$ in (1) are randomly chosen in $[1, A]$, where $A = 5$ when $n_i = 10$, and $A = 20$ when $n_i = 100$. The capacity $c$ is set to $c = \sum_{i=1}^{k} (\underline{v}_i + \overline{v}_i)/2$, where $\underline{v}_i$ and $\overline{v}_i$ are defined as in (2). All tests were run on a hp9000/C200 200 Mhz, and the entries are average values of 100 instances.

First, Table I gives the average time for solving the continuous problem. Notice that the solution times grow linearly with the problem size, and that even problems with

Table III: Total time used. Average of 100 instances

| $k$ | $n_i$ | Uncorrelated $R: 100$ | Uncorrelated 1000 | Weakly corr. $R: 100$ | Weakly corr. 1000 | Subset-sum $R: 100$ | Subset-sum 1000 | Zig-zag $R: 100$ | Zig-zag 1000 |
|---|---|---|---|---|---|---|---|---|---|
| 10 | 10 | 0.00 | 0.00 | 0.00 | 0.01 | 0.00 | 0.01 | 0.00 | 0.00 |
| 100 | 10 | 0.01 | 0.02 | 0.01 | 0.02 | 0.00 | 0.01 | 0.01 | 0.02 |
| 1000 | 10 | 0.05 | 0.09 | 0.06 | 0.07 | 0.03 | 0.03 | 0.06 | 0.08 |
| 10000 | 10 | 0.52 | 0.52 | 0.69 | 0.68 | 0.34 | 0.35 | 0.67 | 0.69 |
| 10 | 100 | 0.05 | 0.09 | 0.07 | 1.10 | 0.07 | 1.09 | 0.09 | 1.15 |
| 100 | 100 | 0.08 | 0.68 | 0.08 | 0.37 | 0.08 | 0.95 | 0.12 | 2.55 |
| 1000 | 100 | 0.49 | 0.62 | 0.57 | 0.86 | 0.25 | 1.24 | 0.58 | 2.38 |
| 10000 | 100 | 4.81 | 5.05 | 5.76 | 6.18 | 2.11 | 3.07 | 5.69 | 7.59 |

1 000 000 variables are solved in less than 5 seconds. Thus the Dantzig-Wolfe decomposition for the continuous problem is very successful.

Next Table II shows the average core size for each instance type. The core size is the number of classes which were transformed $N_i \to \overline{N}_i$ and enumerated in the dynamic programming algorithm for the MCKP. Most problems are solved with only a dozen of classes enumerated, meaning that a very limited number of the expensive transformations $N_i \to \overline{N}_i$ need to be performed. This demonstrates that the gradients are well suited for defining a core.

Finally Table III gives the total computational times for solving the problems to integer optimality. Even very large sized instances are solved within 8 seconds, and the solution times are very stable for all sizes and types of instances.

A few experiments with CPLEX 5.0 [2] were run to compare the performance of the developed algorithm with that of a good LP/MIP solver. The first instance in each class of the uncorrelated instances with $n_i = 100$ and $R = 1000$ were run for different values of $k$. The obtained results are reported in Table IV, demonstrating that the specialized algorithm for BBMC is able to solve the continuous problem two orders of magnitude faster than the best of the LP algorithms. The integer problems seem to be very degenerated, since CPLEX is not able to find the optimal solution in reasonable time for large instances.

Table IV: Time used by CPLEX for solving uncorrelated instances with $n_i = 100$, $R = 1000$. The first three columns give the solution time for solving the LP-relaxation by using primal simplex, dual simplex or interior point methods. The next two columns give the solution times and the number of branch-and-bound nodes generated for solving the problem to integer optimality.

| $k$ | LP-primal (seconds) | LP-dual (seconds) | LP-barrier (seconds) | integer opt (seconds) | branch-and-bound (nodes) | |
|---|---|---|---|---|---|---|
| 10 | 0.03 | 0.04 | 0.09 | 2.96 | 1302 | |
| 100 | 0.88 | 2.14 | 1.38 | 587.83 | 19654 | (*) |
| 1000 | 72.21 | 316.70 | 27.26 | 8761.26 | 20523 | (†) |
| | (*) terminated with solution 0.00962 % from optimum | | | | | |
| | (†) terminated with solution 0.02185 % from optimum | | | | | |

# 6 Conclusion

The BBMC is a very general model which has concrete applications in budgeting, manufacturing and packing/loading. In particular, it contains all "classic" knapsack problems as a special case. It has been shown that the BBMC is solvable in pseudopolynomial time $O(c\sum_{i=1}^{k} n_i a_i + kc^2)$. For the continuous BBMC, Dantzig-Wolfe decomposition lead to an efficient division of the problem: A master problem solves a MCKP problem, while subproblems find extreme points of the transformed sets $\overline{N}_i$. This results in a polynomial algorithm running in $O(n \log P + n \log W)$. Computational experiments have demonstrated that the continuous as well as integer BBMC can be solved orders of magnitude faster than by using a general LP/MIP solver.

# References


[1] E. Balas and E. Zemel (1980), "An Algorithm for Large Zero-One Knapsack Problems", *Operations Research*, **28**, 1130–1154.

[2] "CPLEX base system with barrier and mixed integer solver options" (1997), ILOG inc, NV, USA.

[3] G.B. Dantzig and P. Wolfe (1960), "Decomposition principle for linear programs", *Operations Research*, **8**, 101–111.

[4] M. E. Dyer (1984), "An $O(n)$ algorithm for the multiple-choice knapsack linear program", *Mathematical Programming*, **29**, 57-63.

[5] "LEDA — Library of Efficient Datatypes and Algorithms" (1997), LEDA Software GmbH, Saarbrücken, Germany.

[6] S. Martello and P. Toth (1990), *Knapsack Problems: Algorithms and Computer Implementations,* Wiley, Chichester, England.

[7] G.L. Nemhauser, L.A. Wolsey (1988), *Integer and Combinatorial Optimization,* Wiley, Chichester, England. Chapter I.4, 83–113.

[8] D. Pisinger (1995), "A minimal algorithm for the Multiple-choice Knapsack Problem," *European Journal of Operational Research*, **83**, 394–410.

[9] D. Pisinger (1996), "Budgeting with Bounded Multiple-choice Constraints", Proceedings *AIRO'96*, Perugia, September 17-20, 1996.

[10] A. Sinha and A. A. Zoltners (1979), "The multiple-choice knapsack problem", *Operations Research*, **27**, 503–515.

[11] E. Zemel (1984), "An $O(n)$ algorithm for the linear multiple choice knapsack problem and related problems", *Information Processing Letters*, **18**, 123–128.