

Short Communication

# An exact algorithm for the fixed-charge multiple knapsack problem

Takeo Yamada <sup>\*</sup>, Takahiro Takeoka <sup>1</sup>

*Department of Computer Science, The National Defense Academy, Yokosuka, Kanagawa 239-8686, Japan*

Received 21 April 2007; accepted 11 October 2007

Available online 22 October 2007

## Abstract

We formulate the fixed-charge multiple knapsack problem (FCMKP) as an extension of the multiple knapsack problem (MKP). The Lagrangian relaxation problem is easily solved, and together with a greedy heuristic we obtain a pair of upper and lower bounds quickly. We make use of these bounds in the pegging test to reduce the problem size. We also present a branch-and-bound (B&B) algorithm to solve FCMKP to optimality. This algorithm exploits the Lagrangian upper bound as well as the pegging result for pruning, and at each terminal subproblem solve MKP exactly by invoking MULKNAP code developed by Pisinger [Pisinger, D., 1999. An exact algorithm for large multiple knapsack problems. *European Journal of Operational Research* 114, 528–541]. As a result, we are able to solve almost all test problems with up to 32,000 items and 50 knapsacks within a few seconds on an ordinary computing environment, although the algorithm remains some weakness for small instances with relatively many knapsacks.

© 2007 Elsevier B.V. All rights reserved.

*Keywords:* Integer programming; Multiple knapsack problem; Fixed-charge problem; Combinatorial optimization

## 1. Introduction

This paper is concerned with a variation of the *multiple knapsack problem* (MKP, [7,4,5]), where we are given a set of  $n$  items  $N = \{1, 2, \dots, n\}$  to be packed into  $m$  possible knapsacks  $M = \{1, 2, \dots, m\}$ . As in ordinary MKP, by  $w_j$  and  $p_j$  we denote the weight and profit of item  $j \in N$  respectively, and the capacity of knapsack  $i \in M$  is  $c_i$ . However, a fixed cost  $f_i$  is imposed if we use knapsack  $i$ , and the problem is to determine the set of knapsacks to be employed as well as to fill the adopted knapsacks with items such that the capacity constraints are all satisfied and the total net profit is maximized. Let  $y_i$  and  $x_{ij}$  be the decision variables such that  $y_i = 1$  if we use knapsack  $i$ , and  $y_i = 0$  otherwise. Also,  $x_{ij} = 1$  if item  $j$  is included in knapsack  $i$ , and  $x_{ij} = 0$  otherwise. Then, the problem is formulated as the following *fixed-charge multiple knapsack problem*.

$$\text{FCMKP: maximize } \sum_{i=1}^m \sum_{j=1}^n p_j x_{ij} - \sum_{i=1}^m f_i y_i, \quad (1)$$

$$\text{subject to } \sum_{j=1}^n w_j x_{ij} \leq c_i y_i, \quad i \in M, \quad (2)$$

$$\sum_{i=1}^m x_{ij} \leq 1, \quad j \in N, \quad (3)$$

$$x_{ij}, y_i \in \{0, 1\}, \quad i \in M, j \in N. \quad (4)$$

Throughout the paper we assume the following:

- A<sub>1</sub>. Problem data  $w_j, p_j (j \in N)$  and  $c_i, f_i (i \in M)$  are all positive integers.
- A<sub>2</sub>. Items are arranged in non-increasing order of profit per weight, i.e.,

$$p_1/w_1 \geq p_2/w_2 \geq \dots \geq p_n/w_n. \quad (5)$$

- A<sub>3</sub>. Knapsacks are numbered in non-increasing order of capacity per cost, i.e.,

$$c_1/f_1 \geq c_2/f_2 \geq \dots \geq c_m/f_m. \quad (6)$$

<sup>\*</sup> Corresponding author. Fax: +81 46 844 5911.

E-mail address: [yamada@nda.ac.jp](mailto:yamada@nda.ac.jp) (T. Yamada).

<sup>1</sup> Currently with Maritime Self-Defense Force, Japan.

FCMKP is  $\mathcal{NP}$ -hard, since the special case of free knapsacks ( $f_i \equiv 0, \forall i \in M$ ) is simply an MKP which is already  $\mathcal{NP}$ -hard [4]. Yasuda [10] formulated FCMKP and gave a solution algorithm based on a branch-and-price approach. His algorithm solved problems with up to 200 items. In this article, we present an algorithm that solves larger FCMKPs to optimality in shorter computing time than previous algorithms.

## 2. Upper and lower bounds

This section derives an *upper bound* by applying the *Lagrangian relaxation* [3] to FCMKP. We also present a *greedy* algorithm to obtain a good approximate solution quickly, which necessarily gives a *lower bound* to FCMKP.

### 2.1. Lagrangian relaxation

With non-negative multipliers  $\lambda = (\lambda_i) \in R^m$  associated with (2), the Lagrangian relaxation to FCMKP is

$$\begin{aligned} \text{LFCMKP}(\lambda) : \text{maximize} \quad & \sum_{i=1}^m \sum_{j=1}^n (p_j - \lambda_i w_j) x_{ij} \\ & + \sum_{i=1}^m (\lambda_i c_i - f_i) y_i, \quad (7) \\ \text{subject to} \quad & (3), (4). \end{aligned}$$

For  $\lambda \geq 0$ , let  $\bar{z}(\lambda)$  denote the optimal objective value to LFCMKP( $\lambda$ ). Then, it is easily proved that  $\bar{z}(\lambda)$  gives an upper bound to FCMKP, and  $\bar{z}(\lambda)$  is a piecewise linear and convex function of  $\lambda$ . Moreover, if we consider the *Lagrangian dual*

$$\begin{aligned} \text{minimize} \quad & \bar{z}(\lambda), \\ \text{subject to} \quad & \lambda \geq 0, \end{aligned}$$

we have the following.

**Theorem 1.** *There exists an optimal solution  $\lambda^\dagger = (\lambda_i^\dagger)$  to the Lagrangian dual such that  $\lambda_1^\dagger = \lambda_2^\dagger = \dots = \lambda_m^\dagger (\equiv \lambda^\dagger)$ .*

**Proof.** Let  $\lambda = (\lambda_i) \geq 0$  be any feasible solution to the Lagrangian dual, and put  $k := \text{argmin}_{i \in M} \{\lambda_i\}$ . Then, since  $p_j - \lambda_i w_j \leq p_j - \lambda_k w_j$  for all  $i \in M$  and  $j \in N$ , the objective function (7) is maximized by setting  $x_{ij} = 1$  ( $y_i = 1$ , resp.) if and only if  $i = k$  and  $p_j - \lambda_k w_j > 0$  ( $\lambda_i c_i - f_i > 0$ , resp.), and we obtain

$$\bar{z}(\lambda) = \sum_{j=1}^n (p_j - \lambda_k w_j)^+ + \sum_{i=1}^m (\lambda_i c_i - f_i)^+, \quad (8)$$

where  $(\cdot)^+ := \max\{\cdot, 0\}$ . Since  $(\lambda_i c_i - f_i)^+$  is monotonically non-decreasing with respect to  $\lambda_i$ , under the condition  $\lambda_i \geq \lambda_k$ , (8) is minimized at  $\lambda_i \equiv \lambda_k (i \in M)$ .  $\square$

From this theorem, it suffices to consider the case of  $\lambda_i \equiv \lambda (\forall i \in M)$ , and thus LFCMKP( $\lambda$ ) is rewritten as

$$\begin{aligned} \text{LFCMKP}(\lambda) : \text{maximize} \quad & \sum_{j=1}^n (p_j - \lambda w_j) x_j \\ & + \sum_{i=1}^m (\lambda c_i - f_i) y_i, \quad (9) \\ \text{subject to} \quad & x_j, y_i \in \{0, 1\}, j \in N, i \in M, \quad (10) \end{aligned}$$

where we put

$$x_j := \sum_{i=1}^m x_{ij}.$$

Then, the optimal objective value is given as

$$\bar{z}(\lambda) = \sum_{j=1}^n (p_j - \lambda w_j)^+ + \sum_{i=1}^m (\lambda c_i - f_i)^+ \quad (11)$$

and thus we have

$$d\bar{z}(\lambda)/d\lambda = \sum_{\lambda c_i - f_i > 0} c_i - \sum_{p_j - \lambda w_j > 0} w_j,$$

provided that  $\bar{z}(\lambda)$  is differentiable at  $\lambda$ . The optimal  $\lambda^\dagger$  can be found by the *bisection method*. Let us introduce the *thresholds* as

$$\theta_j := p_j - \lambda^\dagger w_j, \quad \eta_i := \lambda^\dagger c_i - f_i. \quad (12)$$

Then, the Lagrangian upper bound is given as

$$\bar{z} := \bar{z}(\lambda^\dagger) = \sum_{j \in N} \theta_j^+ + \sum_{i \in M} \eta_i^+. \quad (13)$$

### 2.2. A heuristic algorithm

For a standard 0–1 knapsack problem (KP) with knapsack capacity  $c$  and items being arranged according to (5), Pisinger [6] proposed the *forward* and *backward greedy* lower bounds as follows. First, let  $\bar{p}_j := \sum_{l=1}^j p_l$  and  $\bar{w}_j := \sum_{l=1}^j w_l$  be the *accumulated* profit and weight, respectively. The *critical* item  $b$  satisfies  $\bar{w}_{b-1} \leq c < \bar{w}_b$ . Then,

$$\underline{z}_f = \max_{j=b, \dots, n} \{\bar{p}_{b-1} + p_j : \bar{w}_{b-1} + w_j \leq c\}$$

and

$$\underline{z}_b = \max_{j=1, \dots, b-1} \{\bar{p}_b - p_j : \bar{w}_b - w_j \leq c\},$$

are both lower bounds to KP, and  $\underline{z} := \max\{\underline{z}_f, \underline{z}_b\}$  gives the Pisinger's lower bound.

We employ this method to solve FCMKP approximately but very quickly. For an arbitrary subset of items  $I \subseteq N$ , consider the knapsack problem with respect to knapsack  $i$  with capacity  $c_i$ . This is referred to as the knapsack problem  $\text{KP}(I, i)$ , and  $\text{Pis}(I, i)$  denotes the set of accepted items by the above stated Pisinger's algorithm. The total profit (and weight) of  $\text{Pis}(I, i)$  is denoted  $\bar{p}(I, i)$  (and  $\bar{w}(I, i)$ , resp.). Then, our heuristic algorithm for FCMKP is as follows. The objective value associated with this solution is henceforth denoted as  $\underline{z}$ .

**Algorithm Heuris**

- Step 1. Let  $I = N$ , and  $i = 1$ .
- Step 2. Apply the Pisinger’s greedy algorithm to  $\text{KP}(I, i)$  and obtain  $\text{Pis}(I, i)$ .
- Step 3. If  $\bar{p}(I, i) < f_i$ , go to Step 5. (Skip knapsack  $i$ )
- Step 4. Let  $I \geq ts \setminus \text{Pis}(I, i)$ . (Put  $\text{Pis}(I, i)$  into knapsack  $i$ )
- Step 5. If  $i \geq n$  stop. Else  $i \leftarrow i + 1$  and go to Step 2.

**3. Problem reduction**

A pegging test is well known for the standard 0–1 knapsack problem [2]. By applying this test, many variables are fixed either at 0 or 1, and removing these we obtain a problem of (often significantly) reduced size. In this section, we show that the similar pegging test can be applied to FCMKPs by introducing the Lagrangian relaxation first.

Assume that we have the optimal Lagrangian multiplier  $\lambda^\dagger$  and the corresponding upper bound  $\bar{z}$  given by (13), as well as the lower bound  $\underline{z}$  obtained in Section 2.2. We introduce  $\text{FCMKP}(y_k = \delta)$  for  $\delta \in \{0, 1\}$  as the subproblem of FCMKP with an additional constraint  $y_k = \delta$ , and  $\mathbf{P}(y_k = \delta)$  denotes the Lagrangian relaxation of  $\text{FCMKP}(y_k = \delta)$ . That is,  $\mathbf{P}(y_k = \delta)$ :

$$\begin{aligned} &\text{maximize} \quad \sum_{j=1}^n \theta_j x_j + \sum_{i=1}^m \eta_i y_i, & (14) \\ &\text{subject to} \quad (10), \quad y_k = \delta. \end{aligned}$$

Let  $(\mathbf{x}^*, \mathbf{y}^*)$  be an optimal solution to FCMKP with  $\mathbf{x}^* = (x_{ij}^*)$  and  $\mathbf{y}^* = (y_i^*)$ . Then, the following is immediate.

**Theorem 2** (Pegging of knapsacks). *For every  $k \in M$ ,  $\bar{z} - \underline{z} < \eta_k \Rightarrow y_k^* = 1$ , and  $\bar{z} - \underline{z} < -\eta_k \Rightarrow y_k^* = 0$ .*

**Proof.** (i) Note that the optimal objective value to  $\mathbf{P}(y_k = \delta)$  is  $\bar{z}(y_k = \delta) := \sum_{j \in N} \theta_j^+ + \sum_{i \neq k} \eta_i^+ + \eta_k \delta$ . Then, comparing this with (13) we have  $\eta_k \geq 0 \Rightarrow \bar{z}(y_k = 0) = \bar{z} - \eta_k$ . Thus, from  $0 \leq \bar{z} - \underline{z} < \eta_k$ , we obtain  $\bar{z}(y_k = 0) = \bar{z} - \eta_k < \underline{z}$ , which implies  $y_k^* = 0$  in any optimal solution. This completes the proof. (ii) Similarly proved.  $\square$

Analogously, we can derive a pegging theorem for items. Applying Theorem 2, the knapsacks are classified into three disjoint subsets  $K_0 := \{i \in M | y_i^* = 0\}$ ,  $K_1 := \{i \in M | y_i^* = 1\}$  and the remaining  $M \setminus (K_0 \cup K_1)$ . Knapsack  $i \in K_0$  is never used, while  $i \in K_1$  is always used in any optimal solution to FCMKP. Similarly, by the pegging theorem for items we have disjoint sets of items  $I_0 := \{j \in N | x_j^* = 0\}$ ,  $I_1 := \{j \in N | x_j^* = 1\}$  and  $N \setminus (I_0 \cup I_1)$ . Note that  $I_1$  represents the set of items which are included in some knapsack in optimality. Removing  $K_0$  and  $I_0$ , FCMKP is reduced (often significantly) in size. In what follows, we assume that these are already removed, and thus  $K_0 = I_0 = \emptyset$ .

**4. A branch-and-bound algorithm**

To construct a branch-and-bound algorithm, we introduce a subproblem of FCMKP as follows. Let  $F_0$  and  $F_1$  be two subsets of  $M$  such that

$$K_0 \subseteq F_0, \quad K_1 \subseteq F_1 \quad \text{and} \quad F_0 \cap F_1 = \emptyset.$$

These represent the set of knapsacks which are fixed at 0 and 1, respectively. We consider the following.

$\mathbf{P}(F_0, F_1)$ : maximize (1) subject to (2)–(4), and

$$y_i = 0 \quad \forall i \in F_0; \quad y_i = 1 \quad \forall i \in F_1. \tag{15}$$

Using the optimal  $\lambda^\dagger$ , the Lagrangian relaxation to this problem is

$\bar{\mathbf{P}}(F_0, F_1)$ : maximize (14) subject to (10) and (15).

Let  $z^*(F_0, F_1)$  and  $\bar{z}(F_0, F_1)$  be the optimal objective values to these problems, respectively. Clearly,  $\bar{z}(F_0, F_1)$  gives an upper bound to  $z^*(F_0, F_1)$ , i.e.,  $z^*(F_0, F_1) \leq \bar{z}(F_0, F_1)$ ; and we have

$$\bar{z}(F_0, F_1) = \sum_{j=1}^n \theta_j^+ + \sum_{i \in F_1} \eta_i + \sum_{i \in U} \eta_i^+, \tag{16}$$

where  $U := M \setminus (F_0 \cup F_1)$  is the set of unfixed knapsacks. Then, if we have an incumbent lower bound  $\underline{z}$  satisfying  $\bar{z}(F_0, F_1) < \underline{z}$ , we can terminate subproblem  $\mathbf{P}(F_0, F_1)$ .

Other conditions for pruning subproblems are feasibility and dominance. Let the total possible knapsack capacities at  $\mathbf{P}(F_0, F_1)$  be  $\bar{c}(F_0, F_1) := \sum_{i \in M \setminus F_0} c_i$ , and  $\bar{w}_{\text{fix}} := \sum_{j \in I_1} w_j$  denotes the total weight of accepted items. Then, if  $\bar{w}_{\text{fix}} > \bar{c}(F_0, F_1)$  subproblem  $\mathbf{P}(F_0, F_1)$  is infeasible and thus terminated. Furthermore, we say that a knapsack  $i$  dominates  $i'$  if  $c_i \geq c_{i'}$ ,  $f_i \leq f_{i'}$  and  $(c_i, f_i) \neq (c_{i'}, f_{i'})$ .  $\mathbf{P}(F_0, F_1)$  is inconsistent if there exist  $i \in F_0$  and  $i' \in F_1$  such that  $i$  dominates  $i'$ . Such a subproblem is also terminated.

If  $\mathbf{P}(F_0, F_1)$  is not terminated by any of these criteria, and in addition if  $U$  is non-empty, we pick up a knapsack  $i \in U$  and generate two subproblems of  $\mathbf{P}(F_0, F_1)$  as  $\mathbf{P}(F_0 \cup \{i\}, F_1)$  and  $\mathbf{P}(F_0, F_1 \cup \{i\})$ . On the other hand, if  $U = \emptyset$ ,  $\mathbf{P}(F_0, F_1)$  is a terminal subproblem. Terminal  $\mathbf{P}(F_0, F_1)$  is actually an MKP, which can be solved using Pisinger’s MULKNAP code [7], and we obtain optimal  $z^*(F_0, F_1)$ . If this is better than the incumbent lower bound  $z$ , we update this as  $z \geq ts \ z^*(F_0, F_1)$ . Then, we can construct a recursive B&B algorithm to solve  $\mathbf{P}(F_0, F_1)$  in a standard way. The algorithm start with the initial lower bound  $\underline{z}$  (obtained in Section 2.2) and  $(F_0, F_1) := (K_0, K_1)$ , and in termination produces an optimal solution to FCMKP.

However, in implementing the B&B algorithm, we have to specify the strategy for the choice of the branching knapsack  $i$ , as well as the method to traverse the B&B tree of subproblems. For the latter we employ the depth-first search, since other methods (such as the breadth-first or best-first methods) are usually heavy in memory requirement. After some preliminary tests, we consider the following two strategies.

*Strategy 1:*

We pick up knapsack  $i \in U$  sequentially, i.e., under assumption  $A_1$ , in non-increasing order of  $c_i/f_i$ . Out of two generated subproblems, we first examine  $P(F_0 \cup \{i\}, F_1)$  and then  $P(F_0, F_1 \cup \{i\})$ .

*Strategy 2:*

We pick up knapsack  $i \in U$  in non-decreasing order of  $|\eta_i|$ . If  $\eta_i > 0$ , we examine  $P(F_0 \cup \{i\}, F_1)$  first and then  $P(F_0, F_1 \cup \{i\})$ ; otherwise we go to  $P(F_0, F_1 \cup \{i\})$  and then to  $P(F_0 \cup \{i\}, F_1)$ .

**5. Numerical experiments**

We evaluate the performance of the B&B algorithm of the previous section through a series of numerical experiments. We implemented the algorithm in ANSI C language and conducted computation on an IBM RS/6000 Model 270 workstation (CPU: POWER3-II SMP 2way  $\times$  4, 375 MHz).

Table 1  
Comparison against IP solvers ( $m = 5$ )

Type	$n$	CPLEX		XPRESS-MP		B&B	
		#solved	CPU	#solved	CPU	#solved	CPU
UNCOR	20	30	0.20	30	0.10	30	0.01
	40	25	64.18	23	36.68	30	0.01
	60	18	52.73	12	69.29	30	0.00
WEAK	20	30	19.95	30	9.19	30	0.05
	40	9	55.40	13	117.67	30	0.00
	60	3	9.77	4	70.79	30	0.00
STRONG	20	30	3.78	30	2.84	30	0.02
	40	20	33.75	20	23.26	30	0.01
	60	17	59.02	14	17.45	30	0.01

Table 2  
Bounding and pegging results

$\delta$	$m$	$n$	UNCOR			WEAK			STRONG		
			gap	PegI	PegK	gap	PegI	PegK	gap	PegI	PegK
0.25	30	2000	273.9	40.1	85.3	178.2	8.5	90.7	1547.2	0.0	50.0
		8000	180.6	47.9	94.7	61.7	39.7	98.0	1385.4	0.0	85.3
		32000	230.7	50.5	97.0	21.6	54.1	99.7	1532.2	0.0	93.7
	50	2000	550.6	22.3	69.4	317.3	0.3	76.8	2379.0	0.0	15.8
		8000	224.8	43.9	92.4	107.5	23.8	96.2	2236.9	0.0	64.0
		32000	103.0	53.5	98.0	35.1	48.9	100.0	2160.7	0.0	85.8
0.50	30	2000	190.2	35.5	91.0	147.4	4.8	96.3	1735.9	0.0	61.3
		8000	128.1	42.2	97.0	71.7	20.0	97.0	1776.1	0.0	90.7
		32000	193.8	39.2	97.3	50.0	31.2	99.0	1785.7	0.0	96.0
	50	2000	328.4	25.8	84.4	256.2	0.0	88.4	3172.4	0.0	29.2
		8000	163.5	39.3	95.0	103.2	12.0	97.8	3046.4	0.0	72.0
		32000	434.3	30.3	97.8	38.9	33.0	99.4	3232.1	0.0	89.2
0.75	30	2000	183.4	31.8	93.7	134.1	1.3	98.0	2439.7	0.0	64.0
		8000	279.8	30.1	94.3	51.4	16.9	99.3	2237.3	0.0	91.3
		32000	165.7	36.7	98.3	21.4	28.7	99.3	2430.2	0.0	96.3
	50	2000	273.3	25.7	88.8	196.4	0.0	93.0	4247.3	0.0	34.0
		8000	228.1	29.9	96.2	89.7	8.2	98.2	3472.7	0.0	76.0
		32000	170.4	36.4	99.0	52.0	22.0	99.4	3629.0	0.0	92.4

5.1. Design of experiments

The size of the instances tested is  $n = 20-16,000$  and  $m = 5-50$ , and instances are prepared according to the following scheme. The weight  $w_j$  is distributed uniformly random over the integer interval  $[1, 1000]$ , and profit  $p_j$  is related to the weight in the following ways:

- Uncorrelated case (UNCOR): Uniformly random over  $[1, 1000]$ , independent of  $w_j$
- Weakly correlated case (WEAK): Uniformly random over  $[w_j, w_j + 200]$ .
- Strongly correlated case (STRONG):  $p_j := w_j + 20$

Knapsack capacity  $c_i$  is determined by  $\lfloor 500n \cdot \delta \cdot \xi_i \rfloor$ , where  $(\xi_i)$  is uniformly distributed over  $\{(\xi_1, \dots, \xi_m) \mid \sum_{i=1}^m \xi_i = 1, \xi_i \geq 0\}$ , and  $\delta$  is either 0.25, 0.50 or 0.75. Knapsack cost is given by  $f_i := \rho_i c_i$ , where  $\rho_i$  is uniformly random over  $[0.5, 1.5]$ . Here the parameter  $\delta$  in the knapsack capacity controls the ratio of items that can be accepted into the knapsacks. Since average weight of items is approximately 500,  $\delta = 0.50$  means that approximately a half of all the items can be accommodated in the knapsacks. An ANSI C program to generate these instances is available from the author’s web site [9].

5.2. Comparison against IP solvers

Table 1 summarizes the computation of small instances with  $n = 20-60$  and  $m = 5$  using IP solvers CPLEX 10.1.0 [1], XPRESS-MP Ver. 1-16-20 [8], and B&B method of Section 4. For each correlation type and  $n$ , 30 random instances are generated and solved. Here shown are the number of instances solved to optimality within 600 CPU seconds (#solved), and the average CPU time in seconds (CPU), which is taken over solved instances.

Table 3  
B&B result

CORR	$\delta$	$m$	$n$	$K_{\text{used}}$	Strategy 1		Strategy 2			
					#MKP	CPU	#MKP	CPU		
UNCOR	0.25	30	2000	79.3	3.7	0.044	1.4	0.029		
			8000	79.0	1.5	0.105	1.0	0.100		
			32000	78.3	1.1	0.427	1.0	0.420		
		50	2000	80.4	10.0	0.108	1.3	0.032		
			8000	80.4	2.6	0.142	1.3	0.112		
			32000	76.8	1.2	0.494	1.1	0.490		
		0.50	30	2000	60.0	1.5	0.027	1.2	0.025	
				8000	58.3	1.5	0.100	1.0	0.090	
				32000	56.3	1.0	0.430	1.0	0.428	
	50		2000	58.8	4.2	0.052	1.8	0.036		
			8000	57.8	2.0	0.125	1.3	0.114		
			32000	55.4	1.0	0.523	1.0	0.521		
	0.75		30	2000	47.7	1.2	0.025	1.3	0.026	
				8000	46.3	1.3	0.105	1.1	0.098	
				32000	46.3	1.0	0.411	1.0	0.412	
		50	2000	47.4	3.0	0.041	1.4	0.031		
			8000	47.0	1.7	0.124	1.1	0.117		
			32000	44.4	1.0	0.480	1.0	0.475		
		WEAK	0.25	30	2000	74.7	1.9	0.034	1.0	0.029
					8000	75.3	1.3	0.110	1.0	0.102
					32000	76.0	1.0	0.418	1.0	0.417
	50			2000	77.0	6.7	0.062	1.1	0.035	
				8000	77.4	2.0	0.144	1.0	0.123	
				32000	73.8	1.0	0.497	1.0	0.498	
0.50	30			2000	70.3	1.6	0.034	1.2	0.029	
				8000	69.3	1.2	0.114	1.0	0.106	
				32000	68.7	1.0	0.442	1.0	0.438	
	50		2000	70.0	3.7	0.048	1.1	0.033		
			8000	70.0	1.4	0.130	1.0	0.124		
			32000	63.8	1.0	0.508	1.0	0.506		
	0.75		30	2000	67.0	1.2	0.029	1.0	0.029	
				8000	64.0	1.0	0.104	1.0	0.099	
				32000	65.0	1.0	0.402	1.0	0.400	
50			2000	65.6	2.7	0.044	1.0	0.032		
			8000	65.2	1.3	0.124	1.1	0.121		
			32000	60.8	1.0	0.495	1.0	0.494		
STRONG			0.25	30	2000	60.0	10.3	1.522	1.3	0.252
					8000	53.7	2.9	2.300	1.0	0.931
					32000	53.7	2.2	7.941	1.0	3.142
	50			2000	57.4	28.1	14.591	2.6	0.479	
				8000	52.4	10.5	6.126	1.5	0.983	
				32000	52.4	3.9	11.820	1.0	4.732	
	0.50	30		2000	57.3	7.3	2.584	1.0	0.160	
				8000	53.0	2.7	3.778	1.0	1.569	
				32000	52.3	1.6	8.881	1.0	5.557	
		50	2000	56.8	21.7	6.800	1.7	0.702		
			8000	48.4	7.0	10.746	1.1	1.468		
			32000	51.0	3.0	16.865	1.0	7.829		
		0.75	30	2000	56.0	6.8	2.732	1.0	0.255	
				8000	52.7	2.6	3.962	1.1	1.350	
				32000	51.0	1.4	10.242	1.0	6.301	
	50		2000	56.2	19.8	8.397	1.2	0.710		
			8000	48.0	6.0	10.621	1.0	1.582		
			32000	50.2	2.4	26.669	1.0	9.124		

From this table we see that commercial solvers are able to solve only very small instances within this time limit, while B&B solved all these problems within 0.1 seconds.

5.3. Bounds and reduction

Table 2 gives the results of computation of upper and lower bounds as well as pegging for larger instances with  $n = 2000\text{--}32000$  and  $m = 30\text{--}50$ . CPU time for these are always less than 1.0 second, and thus negligible. The table shows the *gap* between  $\bar{z}$  and  $\underline{z}$ , the ratio of items fixed at 0 ( $\text{PegI} := 100 \cdot |I_0|/n$ ), and the ratio of the fixed knapsacks ( $\text{PegK} := 100 \cdot (|K_0| + |K_1|)/m$ ). Each row is the average over 10 randomly generated instances.

From this table, we observe that pegging works effectively for large instances in fixing knapsacks. Although it is somewhat weak for STRONG instances with large  $m$ , usually more than 90% of knapsacks are fixed either at 0 or 1. In UNCOR case, often nearly a half of items are fixed at 0, but in STRONG case pegging becomes less effective in removing items. This appears to be due to a wide gap remaining between the upper and lower bounds in this case.

5.4. Exact solutions

Table 3 gives the results of B&B for various instances. These tables show the percentage (%) of the knapsacks used in optimality ( $K_{\text{used}}$ ), the number of MKPs solved by calling MULKNAP code [7] (#MKP), and CPU time in seconds under strategies 1 and 2. Each row is again the average over 10 random instances.

Except for some cases, B&B solved FCMKP to optimality by invoking MULKNAP only a few times and within a few seconds. Here note that if all knapsacks are fixed, the reduced problem is MKP and thus we have #MKP = 1. For large  $n$ , this is approximately the case, and thus we have small #MKP. Strategy 2 is usually superior to Strategy 1, and this is especially the case for STRONG instances with larger  $m$ . B&B with Strategy 2 solved almost all instances within 10 CPU seconds in our computing environment.

However, in multiple knapsack problems smaller instances are often harder to solve [4]. Table 4 investigates the performance of our B&B method for such a case. Here, we see that #MKP and CPU time increase, and the problem becomes harder to solve, as  $m$  becomes larger, even for small  $n$ .

Table 4  
Result for small size instances ( $\delta = 0.50$ )

$n$	$n$	UNCOR		WEAK		STRONG	
		#MKP	CPU	#MKP	CPU	#MKP	CPU
100	8	12.9	0.00	5.9	0.01	18.6	0.08
	12	71.2	0.02	79.4	0.04	85.6	0.25
	16	1648.7	3.63	386.1	0.29	445.3	1.28
150	8	6.9	0.00	3.5	0.00	6.4	0.06
	12	20.1	0.01	21.6	0.01	105.6	0.84
	16	357.2	0.16	96.2	0.08	836.6	5.26
200	8	4.1	0.00	3.7	0.00	19.9	0.26
	12	31.2	0.02	19.0	0.01	23.4	0.24
	16	41.0	0.02	55.4	0.05	386.5	3.13

## 6. Conclusion

We have formulated the fixed-charge multiple knapsack problem, and presented a solution algorithm to solve this problem to optimality. By combining the Lagrangian relaxation and pegging test with the branch-and-bound method that solves MKP at each terminal nodes by invoking MULKNAP, we were able to solve almost all FCMKPs with up to  $n = 32,000$  items and  $m = 50$  knapsacks within 10 seconds on an ordinary computing environment. However, the algorithm remains weakness for small instances with relatively many knapsacks.

## References

- [1] CPLEX 10.1.0, ILOG, <http://www.ilog.com/products/cplex>, 2007.
- [2] R.S. Dembo, P.L. Hammer, A reduction algorithm for knapsack problems, *Methods of Operations Research* 36 (1980) 49–60.
- [3] M. Fisher, The Lagrangian relaxation method for solving integer programming problems, *Management Science* 27 (1981) 1–18.
- [4] H. Kellerer, U. Pferschy, D. Pisinger, *Knapsack Problems*, Springer, Berlin, 2004.
- [5] S. Martello, P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*, John Wiley & Sons, Chichester, 1990.
- [6] D. Pisinger, An expanding-core algorithm for the exact 0–1 knapsack problem, *European Journal of Operational Research* 87 (1995) 175–187.
- [7] D. Pisinger, An exact algorithm for large multiple knapsack problems, *European Journal of Operational Research* 114 (1999) 528–541.
- [8] XPRESS-MP Ver. 1.16.20, Dash Associates (<http://www.dashoptimization.com>), 2006.
- [9] T. Yamada, <http://www.nda.ac.jp/~yamada/paper/fcmkp>, 2007.
- [10] R. Yasuda, An algorithm for the fixed-charge multiple knapsack problem (in Japanese). Unpublished MS Thesis, National Defense Academy, 2005.