数理解析研究所講究録
第1682巻 2010年 168-175

An exact algorithm for the budget-constrained multiple knapsack problem

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

Byungjun You, Takeo Yamada
{g48095, yamada}@nda.ac.jp

1 Introduction

This paper is concerned with a variation of the *multiple knapsack problem* (MKP) [5, 6], where we are given a set of n items $N = \{1, 2, \ldots, n\}$ to be packed into m possible knapsacks $M = \{1, 2, \ldots, m\}$. 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 knapsacks are freely available within a fixed budget B. 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 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 put into knapsack i, and $x_{ij} = 0$ otherwise. Then, the problem is formulated as the following *budget-constrained multiple knapsack problem.*

BCMKP:

$$\text{maximize} \quad \sum_{i=1}^{m} \sum_{j=1}^{n} p_j x_{ij} \tag{1}$$

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

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

$$\sum_{i=1}^{m} f_i y_i \leq B, \tag{4}$$

$$x_{ij},\ y_i \in \{0, 1\}, \quad i \in M, j \in N. \tag{5}$$

Throughout the paper we assume the following.

$\mathbf{A}_1$. Problem data w_j, p_j $(j \in N)$, c_i, f_i $(i \in M)$ and B are all positive integers.

$\mathbf{A}_2$. Items are arranged in non-increasing order of profit per weight, i.e.,

$$p_1/w_1 \geq p_2/w_2 \geq \ldots \geq p_n/w_n. \tag{6}$$

$\mathbf{A}_3$. Knapsacks are numbered in non-increasing order of capacity per cost, i.e.,

$$c_1/f_1 \geq c_2/f_2 \geq \ldots \geq c_m/f_m. \tag{7}$$

BCMKP is $\mathcal{NP}$-hard [3], 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. Since BCMKP is a linear 0-1 programming problem, small instances can be solved using MIP (mixed integer programming) solvers. In this article, we present an algorithm that solves larger BCMKPs to optimality within a few minutes in an ordinary computing environment.

2 Upper bound

This section derives an *upper bound* by applying the *Lagrangian relaxation* [2] to BCMKP. With nonnegative multipliers $\boldsymbol{\lambda} = (\lambda_i) \in R^m$ and $\mu \in R$ associated with (2) and (4) respectively, the *Lagrangian relaxation* to BCMKP is

LBCMKP($\boldsymbol{\lambda}, \mu$):

$$\text{maximize} \quad \sum_{i=1}^{m}\sum_{j=1}^{n}(p_j - \lambda_i w_j)x_{ij} + \sum_{i=1}^{m}(\lambda_i c_i - \mu f_i)y_i + \mu B \tag{8}$$

$$\text{subject to} \quad (3), (5).$$

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

$$\text{minimize} \quad \bar{z}(\boldsymbol{\lambda}, \mu) \quad \text{subject to} \quad \boldsymbol{\lambda} \geq \mathbf{0}, \mu \geq 0,$$

we have the following.

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

Proof. For a fixed $\boldsymbol{\lambda} = (\lambda_i) \geq \mathbf{0}$, let $k := \arg\min_{i \in M}\{\lambda_i\}$. 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 (8) is maximized by setting $x_{ij} = 1$ if and only if $i = k$ and $p_j - \lambda_k w_j > 0$. Similarly, $y_i = 1$ if and only if $\lambda_i c_i - \mu f_i > 0$, and we obtain

$$\bar{z}(\boldsymbol{\lambda}, \mu) = \sum_{j=1}^{n}\left(p_j - \lambda_k w_j\right)^+ + \sum_{i=1}^{m}(\lambda_i c_i - \mu f_i)^+ + \mu B, \tag{9}$$

where $(\cdot)^+ := \max\{\cdot, 0\}$. Here, we note that $(\lambda_i c_i - \mu f_i)^+$ is monotonically non-decreasing with respect to λ_i ($i \neq k$). Thus, under the condition $\lambda_i \geq \lambda_k$, (9) is minimized at $\lambda_i \equiv \lambda_k$ ($i \in M$). ∎

From this theorem, it suffices to consider the case of $\lambda_i \equiv \lambda$ ($\forall i \in M$), and thus $\bar{z}(\boldsymbol{\lambda}, \mu)$ is rewritten as

$$\bar{z}(\lambda, \mu) = (C_t - W_s)\lambda + (B - F_t)\mu + P_s. \tag{10}$$

Here s and t are the *critical values* defined as $s := \max\{j \mid p_j - \lambda w_j \geq 0\}$ and $t := \max\{i \mid \lambda c_i - \mu f_i \geq 0\}$ respectively, and W_s is the *accumulated* weight given by $W_s := \sum_{j=1}^{s} w_j$. P_s, C_t and F_t are analogously defined for (p_j), (c_i) and (f_i), respectively. For a fixed $\mu \geq 0$, (10) is a piecewise linear function of λ, and its gradient changes from $C_t - W_s$ to $C_t - W_{s-1}$ as λ increases from $p_s/w_s - 0$ to $p_s/w_s + 0$. Similarly, the gradient increases from $C_{t-1} - W_s$ to $C_t - W_s$ at $\lambda = (f_t/c_t)\mu$. Thus, we obtain the optimal λ as

$$\lambda^\dagger(\mu) = \begin{cases} p_s/w_s, & \text{if } W_{s-1} \leq C_t \leq W_s, \\ (f_t/c_t)\mu, & \text{if } C_{t-1} \leq W_s \leq C_t. \end{cases} \tag{11}$$

Putting this into (10), $\bar{z}(\lambda^\dagger(\mu), \mu)$ is also piecewise linear with respect to μ, and by *bisection method* we obtain an optimal $\mu^\dagger$ and $\lambda^\dagger := \lambda^\dagger(\mu^\dagger)$, and correspondingly an upper bound $\bar{z} = \bar{z}(\lambda^\dagger(\mu^\dagger), \mu^\dagger)$. Let us introduce the *thresholds* as

$$\theta_j := p_j - \lambda^\dagger w_j, \quad \eta_i := \lambda^\dagger c_i - \mu^\dagger f_i. \tag{12}$$

Then, from (9) the Lagrangian upper bound is given as

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

3 Problem reduction

Assume that we have the optimal Lagrangian multipliers $\lambda^\dagger$ and $\mu^\dagger$ with the corresponding upper bound $\bar{z}$ given by (13), as well as a lower bound $\underline{z}$ obtained by some heuristic algorithmes. Let δ be either 0 or 1, and we introduce $\mathrm{P}(y_k = \delta)$ as the subproblem of BCMKP with y_k fixed at δ. $\overline{\mathrm{P}}(y_k = \delta)$ denotes the Lagrangian relaxation of $\mathrm{P}(y_k = \delta)$ using the optimal $\lambda^\dagger$ and $\mu^\dagger$ in (8). That is,

$\overline{\mathrm{P}}(y_k = \delta)$:

$$\text{maximize} \quad \sum_{j=1}^{n} \theta_j x_j + \sum_{i=1}^{m} \eta_i y_i + \mu^\dagger B \tag{14}$$

$$\text{subject to} \quad x_j, y_i \in \{0,1\}, \forall j \in N, \forall i \in M, y_k = \delta, \tag{15}$$

where we introduce a new 0-1 variable x_j defined by $x_j = \sum_{i=1}^{m} x_{ij}$.

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

Theorem 2 (Pegging of knapsacks) *For every $k \in M$,*

(i) If $\eta_k > 0$ and $\bar{z} - \underline{z} \leq \eta_k \Rightarrow y_k^* = 1$,

(ii) If $\eta_k < 0$ and $\bar{z} - \underline{z} \leq -\eta_k \Rightarrow y_k^* = 0$.

Proof. (i) Note that the optimal objective value to $\overline{\mathrm{P}}(y_k = \delta)$ is $\bar{z}(y_k = \delta) := \sum_{j \in N} \theta_j^+ + \sum_{\substack{i \in M \\ i \neq k}} \eta_i^+ + \mu^\dagger B + \eta_k \delta$. Then, comparing this with (13) we have $\bar{z}(y_k = 0) = \bar{z} - \eta_k \leq \underline{z}$, which implies $y_k^* = 1$ in any optimal solution. (ii) is similarly proved. ∎

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 BCMKP.

Similarly, we can derive a pegging theorem for items, and applying this classify items into the disjoint sets $I_0 := \{j \in N | x_j^* = 0\}$, $I_1 := \{j \in N | x_j^* = 1\}$ and $N \setminus (I_0 \cup I_1)$. Removing K_0 and I_0, BCMKP is reduced (often significantly) in size. In what follows, we assume that these are already done, and thus $K_0 = I_0 = \emptyset$.

4 A branch-and-bound algorithm

A characteristic feature of the branch-and-bound algorithm to be given below is that branchings are made with respect to variables (y_i) irrespective to (x_{ij}), and the latter is determined only at each *terminal subproblems*. To construct such a branch-and-bound algorithm, we introduce a *subproblem* of BCMKP as follows. Let F_0 and F_1 be two subsets of M such that

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

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

$\mathrm{P}(F_0, F_1)$: maximize (1) subject to (2) - (5), *and*

$$y_i = 0, \; \forall i \in F_0, \quad y_i = 1, \; \forall i \in F_1. \tag{16}$$

Using the optimal $\lambda^\dagger$ and $\mu^\dagger$ obtained in Section 2, the Lagrangian relaxation to this problem is

$\overline{\mathrm{P}}(F_0, F_1)$: maximize (14) subject to $x_j, y_i \in \{0,1\}, \forall j \in N, \forall i \in M$ and (16).

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

$$\bar{z}(F_0, F_1) = \sum_{j=1}^{n} \theta_j^+ + \sum_{i \in F_1} \eta_i + \sum_{i \in U} \eta_i^+ + \mu^\dagger B, \tag{17}$$

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 $\bar{z}(F_0, F_1) \leq \underline{z}$, we can *terminate* subproblem $\mathrm{P}(F_0, F_1)$.

Other conditions for pruning subproblems are *feasibility* and *dominance*. First of all, if the total cost of accepted knapsacks is larger than the budget, i.e., if $\sum_{i \in F_1} f_i > B$, $P(F_0, F_1)$ is infeasible, and thus terminated. Next, if we have knapsacks $(i_0, i_1) \in F_0 \times F_1$ such that $c_{i_0} \geq c_{i_1}$ and $f_{i_0} \leq f_{i_1}$, $P(F_0, F_1)$ is again terminated. Indeed, if such a pair (i_0, i_1) exists, we can define a subproblem $P(F_0', F_1')$ by exchanging the role of these knapsacks as $F_0' := F_0 \cup \{i_1\} \setminus \{i_0\}$ and $F_1' := F_1 \cup \{i_0\} \setminus \{i_1\}$. Then, $P(F_0, F_1)$ is dominated by $P(F_0', F_1')$, since all the feasible solutions of $P(F_0, F_1)$ are feasible to $P(F_0', F_1')$.

If $\mathrm{P}(F_0, F_1)$ 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 $\mathrm{P}(F_0, F_1)$ as $\mathrm{P}(F_0 \cup \{i\}, F_1)$ and $\mathrm{P}(F_0, F_1 \cup \{i\})$. On the other hand, if $U = \emptyset$, $\mathrm{P}(F_0, F_1)$ is a *terminal* subproblem. Here, $P(F_0, F_1)$ is an MKP with respect to the set of knapsacks F_1. An upper bound to this subproblem can be obtained by solving the following 0-1 knapsack problem, which is obtained by replacing the set of knapsacks with a single knapsack of capacity $\sum_{i \in F_1} c_i$. Let the *break item b* be given by $b = \min\{j : \sum_{i=1}^{j} w_i \leq \sum_{i \in F_1} c_i\}$. Linear relaxation gives an upper bound, called the *Dantzig bound*, to this problem as

$$\bar{z}_{\mathrm{term}}(F_0, F_1) = \lfloor P_{b-1} + \frac{c_b}{w_b}(\sum_{i \in F_1} c_i - W_{b-1}) \rfloor, \tag{18}$$

and if $\bar{z}_{\mathrm{term}}(F_0, F_1) \leq \underline{z}$, $\mathrm{P}(F_0, F_1)$ is also terminated. Otherwise, we solve this MKP by calling MULKNAP [8], and obtain the optimal $z^*(F_0, F_1)$. If this is better than the incumbent lower bound $\underline{z}$, we update this as $\underline{z} \leftarrow z^*(F_0, F_1)$.

Then, we can construct a branch-and-bound algorithm to solve $\mathrm{P}(F_0, F_1)$ as a recursive procedure. The algorithm starts with the initial lower bound $\underline{z}$ and $(F_0, F_1) := (K_0, K_1)$, and in termination produces an optimal solution to BCMKP. However, in implementing the branch-and-bound algorithm, we have to specify the *strategy* for the choice of the branching knapsack i, as well as the method to traverse the branch-and-bound tree. As for the branching knapsack, under assumption $\mathbf{A}_3$, we pick up the unfixed knapsack of the smallest index, i.e., $i := \min\{k | k \in U\}$, and call $P(F_0, F_1 \cup \{i\})$ recursively before calling $P(F_0 \cup \{i\}, F_1)$. This means that we examine subproblems with more knapsacks fixed at 1 earlier than others. The algorithm is as follows.

Algorithm SOLVE-BCMKP(F_0, F_1)

Input: $F_0, F_1 \subseteq M$, $F_0 \cap F_1 = \emptyset$, and an incumbent lower bound $\underline{z}$.

Step 1 (Check feasibility and dominance).
If $\sum_{i \in F_1} f_i > B$, or there exists a pair $(i_0, i_1) \in F_0 \times F_1$ such that $c_{i_0} \geq c_{i_1}$ and $f_{i_0} \leq f_{i_1}$, return.

Step 2 (Generalized pegging test).
If $\bar{z}(F_0, F_1) \leq \underline{z}$, return.

Step 3 (Terminal node).
If $U := M \setminus (F_0 \cup F_1) \neq \emptyset$, go to step 4. Otherwise do:

(i) (Dantzig bound) If $\bar{z}_{\mathrm{term}}(F_0, F_1) \leq \underline{z}$, return.

(ii) ($P(F_1, F_1)$ is an MKP).

- Solve this problem using MULKNAP and obtain the optimal objective value $z^*(F_0, F_1)$.
- If $z^*(F_0, F_1) > \underline{z}$, update the incumbent by $\underline{z} \leftarrow z^*(F_0, F_1)$.
- return.

Step 4 (Branching).

(i) Pick up the smallest $i \in U$.

(ii) Call SOLVE-BCMKP$(F_0, F_1 \cup \{i\})$.

(iii) Call SOLVE-BCMKP$(F_0 \cup \{i\}, F_1)$.

(iv) return.

5 Numerical experiments

We evaluate the performance of the branch-and-bound algorithm of the previous section through a series of numerical experiments. We implement the algorithm in ANSI C language and conduct computation on an Dell Precision T7400 workstation (CPU: Xeon X5482 Quad-Core×2, 3.20GHz).

5.1 Design of experiments

Instances are prepared within the range of $200 \leq n \leq 160000$ and $5 \leq m \leq 150$ according to the following scheme. The weight w_j is distributed uniformly random over the integer interval $[10, 1000]$, and profit p_j is related to w_j in the following way.

- Uncorrelated case (UNCOR): uniformly random over $[10, 1000]$, 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 $c_i = \lfloor 500n \cdot \alpha \cdot \xi_i \rfloor$, where (ξ_i) is uniformly distributed over the simplex $\{(\xi_1, \ldots, \xi_m) | \sum_{i=1}^m \xi_i = 1, \xi_i \geq 0\}$, and α is a parameter to control the ratio of items that can be accepted into the knapsacks. Since average weight of items is approximately 500, $\alpha = 0.50$ means that about a half of all the items can be accommodated in the knapsacks. Knapsack cost is given by $f_i := \rho_i c_i$, where ρ_i is uniformly random over $[0.5, 1.5]$, and the budget B is chosen as $B = \beta \sum_{i=1}^m f_i$. Here β is another parameter that controls the ratio of the budget B over the total cost of knapsacks.

5.2 Comparison against MIP solvers

Table 1: Comparison against MIP solvers.

Correlation	n	m	branch-and-bound		CPLEX 11.1	
			#solved	CPU_{sec}	#solved	CPU_{sec}
UNCOR	10	3	10	0.00	10	0.01
	20	6	10	0.00	10	0.08
	30	9	10	1.66	10	8.81
	40	12	10	14.31	6	845.00
WEAK	10	3	10	0.00	10	0.01
	20	6	10	0.00	10	0.48
	30	9	10	0.34	10	107.03
	40	12	10	16.84	0	1800.00
STRONG	10	3	10	0.00	10	0.01
	20	6	10	0.00	10	2.54
	30	9	10	0.04	9	374.06
	40	12	10	0.66	2	1637.38

Table 1 summarizes the computation of small instances with parameters $\alpha = 0.5$ and $\beta = 0.6$, $n = 10 \sim 40$ and $m = 3 \sim 12$ using MIP solver CPLEX 11.1 [4] and the branch-and-bound method of section 4. For each correlation type and values of n and m, 10 random instances are generated and solved. Here shown are the number of instances solved to optimality (#solved) within a fixed CPU time, and the average CPU time in seconds. We set the time-limit of computation at 1800 CPU seconds, and if computation is truncated due to this time-limit, the CPU time for this instance is interpreted as 1800 seconds in computing averages.

From Table 1 we see that commercial solvers are able to solve only very small instances within the time-limit, while branch-and-bound solved all these problems within 30 seconds.

5.3 Large instances

Table 2 gives the results of computation of upper and lower bounds as well as the pegging test for larger instances with $n = 20000 \sim 160000$ and $m = 50 \sim 150$. The table shows the upper and lower bounds ($\bar{z}$ and $\underline{z}$, respectively), and the column of 'Error(%)' gives their relative errors defined by $100 \cdot (\bar{z} - \underline{z})/\underline{z}$. Applying the pegging test, some knapsacks are fixed either at 0 or 1, and m' shows the number of unfixed knapsacks, i.e., $m' := |M \backslash (K_0 \cup K_1)|$. '$\mathrm{CPU}_1$' is the CPU time in seconds to compute upper and lower bounds, as well as to carry out the pegging test. From this table, we observe that the pegging works effectively in reducing the number of knapsacks. However, we found that it is less effective for reducing the size of n.

Table 3 is the results of the branch-and-bound algorithm for various instances. This table shows the optimal value (z^*), the number of the generated nodes in the branch-and-bound algorithm (#nodes), the number of pruned subproblems due to infeasibility (#infeas), or due to upper bounds (#ub), the number of terminal MKPs solved by calling MULKNAP (#Pis), and the CPU time in seconds (CPU_2). Each row is again the average over 10 random instances.

Except for some cases, the branch-and-bound method solved BCMKPs exactly by invoking MULKNAP only a few times and within a few minutes. We conclude that the branch-and-bound algorithm is very successful for these large instances.

6 Conclusion

We have formulated the budget-constrained multiple knapsack problem, and presented an 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 calling MULKNAP, we were able to solve almost all BCMKPs with up to $n = 160000$ items and $m = 150$ knapsacks within a few minutes in an ordinary computing environment. However instances with smaller n and larger m remain hard to be solved exactly.

References

[1] R.S. Dembo, P.L. Hammer, A reduction algorithm for knapsack problems, Methods of Operations Research 36 (1980) 49-60.

[2] M. Fisher, The Lagrangian relaxation method for solving integer programming problems, Management Science 27 (1981) 1-18.

[3] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman and Company, San Francisco, 1979.

[4] ILOG, CPLEX 11.1, http://www.ilog.com/products/cplex, 2009.

[5] H. Kellerer, U. Pferschy, D. Pisinger, Knapsack Problems, Springer, Berlin, 2004.

[6] S. Martello, P. Toth, Knapsack Problems: Algorithms and Computer Implementations, John Wiley & Sons, Chichester, 1990.

[7] D. Pisinger, An expanding-core algorithm for the exact 0-1 knapsack problem, European Journal of Operational Research 87 (1995) 175-187.

[8] D. Pisinger, An exact algorithm for large multiple knapsack problems, European Journal of Operational Research 114 (1999) 528-541.

[9] L. Wolsey, Integer Programming, John Wiley & Sons, New York, 1998.

[10] T. Yamada, T. Takeoka, An exact algorithm for the fixed-charge multiple knapsack problem, European Journal of Operational Research 192 (2009) 700-705.

Table 2: Bounding and pegging results for large instances.

Correlation	m	$n(\times 10^4)$	$\bar{z}(\times 10^7)$	$\underline{z}(\times 10^7)$	Error(%)	m'	CPU_1
UNCOR	50	2	0.67839	0.67776	0.09	13.6	0.01
		4	1.36165	1.36010	0.11	16.2	0.02
		8	2.71760	2.71492	0.10	14.9	0.05
		16	5.43095	5.42441	0.12	16.0	0.10
	100	2	0.68241	0.68223	0.03	19.3	0.01
		4	1.36177	1.36125	0.04	23.6	0.02
		8	2.71905	2.71781	0.05	26.7	0.05
		16	5.42959	5.42713	0.05	23.4	0.10
	150	2	0.68306	0.68299	0.01	17.8	0.01
		4	1.36191	1.36173	0.01	21.1	0.02
		8	2.71811	2.71749	0.02	31.9	0.05
		16	5.44194	5.44055	0.02	31.2	0.10
WEAK	50	2	0.48252	0.48176	0.16	13.6	0.01
		4	0.97220	0.97032	0.20	16.2	0.02
		8	1.93985	1.93660	0.17	14.9	0.05
		16	3.87370	3.86576	0.21	16.0	0.11
	100	2	0.48740	0.48718	0.04	19.3	0.01
		4	0.97232	0.97168	0.07	23.6	0.02
		8	1.94149	1.93998	0.08	26.7	0.05
		16	3.87202	3.86903	0.08	23.4	0.11
	150	2	0.48819	0.48810	0.02	17.8	0.01
		4	0.97251	0.97229	0.02	21.1	0.03
		8	1.94033	1.93959	0.04	31.9	0.06
		16	3.88695	3.88525	0.04	31.3	0.11
STRONG	50	2	0.37426	0.37326	0.27	17.1	0.48
		4	0.74238	0.74094	0.19	14.2	1.08
		8	1.49757	1.49432	0.22	16.2	2.09
		16	2.98705	2.98142	0.19	14.9	6.76
	100	2	0.37287	0.37264	0.06	20.7	0.61
		4	0.75081	0.75034	0.06	23.0	1.57
		8	1.49777	1.49652	0.09	26.1	2.89
		16	2.98985	2.98719	0.09	27.2	8.63
	150	2	0.37527	0.37520	0.02	16.2	0.59
		4	0.75219	0.75203	0.02	18.5	0.84
		8	1.49811	1.49746	0.04	31.2	2.49
		16	2.98785	2.98645	0.05	34.8	3.45

Table 3: Branch-and-bound results for large instances.

Correlation	m	$n(\times 10^4)$	$z^*(\times 10^7)$	#nodes	#infeas	#ub	#Pis	CPU_2
UNCOR	50	2	0.67793	1525.8	178.1	583.7	2.1	0.42
		4	1.36067	2986.0	476.9	1014.6	2.5	1.57
		8	2.71614	769.0	140.5	243.1	1.9	0.82
		16	5.42675	1086.0	160.2	380.9	2.9	2.36
	100	2	0.68226	7414.8	763.9	2942.8	1.7	2.01
		4	1.36153	7081.0	970.5	2567.8	3.2	3.75
		8	2.71849	8054.8	1059.9	2965.1	3.4	8.58
		16	5.42826	4840.4	939.5	1479.1	2.6	9.48
	150	2	0.68300	3254.4	490.5	1136.1	1.6	0.74
		4	1.36178	4353.0	431.4	1744.1	2.0	2.46
		8	2.71782	9085.8	858.2	3683.4	2.3	10.12
		16	5.44151	11524.8	1443.7	4315.9	3.8	24.38
WEAK	50	2	0.48197	1525.2	177.9	583.6	2.1	0.40
		4	0.97101	2988.8	477.9	1015.0	2.5	1.56
		8	1.93808	769.0	140.5	243.1	1.9	0.82
		16	3.86860	1088.8	160.5	382.0	2.9	2.37
	100	2	0.48722	7416.2	763.8	2943.6	1.7	1.94
		4	0.97202	7085.8	972.3	2568.4	3.2	3.71
		8	1.94080	8049.0	1061.3	2960.8	3.4	8.49
		16	3.87040	4841.4	939.2	1479.9	2.6	9.45
	150	2	0.48812	3260.2	491.5	1138.0	1.6	0.81
		4	0.97235	4370.4	431.7	1752.5	2.0	2.44
		8	1.94000	9121.8	863.4	3696.2	2.3	10.05
		16	3.88642	11559.4	1444.2	4332.7	3.8	24.37
STRONG	50	2	0.37365	1894.6	356.1	589.8	2.4	1.53
		4	0.74142	1586.0	193.9	597.7	2.4	2.75
		8	1.49551	2946.8	475.4	996.5	2.5	8.91
		16	2.98398	767.4	140.5	242.3	1.9	13.17
	100	2	0.37271	3317.6	436.3	1221.0	2.5	1.83
		4	0.75050	10080.6	1173.8	3863.7	3.8	8.81
		8	1.49727	7365.6	1027.8	2652.3	3.7	17.10
		16	2.98867	8526.4	1093.5	3167.2	3.5	42.15
	150	2	0.37520	3685.8	325.7	1517.1	1.1	1.53
		4	0.75206	4666.6	613.7	1718.6	2.0	4.30
		8	1.49783	7787.2	965.1	2925.4	4.1	21.62
		16	2.98725	9468.6	901.8	3830.8	2.7	31.81