Available online at www.sciencedirect.com

SCIENCE DIRECT®

European Journal of Operational Research 153 (2004) 184–190

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/dsw

Discrete Optimization

Effective algorithm and heuristic for the generalized assignment problem

Salim Haddadi *, Hacene Ouzia

Mathematics Department, Badji Mokhtar University, BP 12, Annaba 23000, Algeria

Received 6 March 2000; accepted 26 March 2002

Abstract

We present new Branch-and-Bound algorithm for the generalized assignment problem. A standard subgradient method (SM), used at each node of the decision tree to solve the Lagrangian dual, provides an upper bound. Our key contribution in this paper is a new heuristic, applied at each iteration of the SM, which tries to exploit the solution of the relaxed problem, by solving a smaller generalized assignment problem. The feasible solution found is then subjected to a solution improvement heuristic. We consider processing the root node as a Lagrangian heuristic. Computational comparisons are made with new existing methods.

Keywords: Branch-and-Bound; Generalized assignment; Knapsack; Solution improvement heuristic

1. Introduction

The generalized assignment problem (GAP) is to find a maximal profit assignment of n tasks (indexed by $j_1, j_2, \dots, j_n$) to m agents (indexed by $i_1, i_2, \dots, i_m$). Let $I = \{i_1, i_2, \dots, i_m\}$ and $J = \{j_1, j_2, \dots, j_n\}$. Let b_i be the resource availability of agent i. Let a_{ij} be the amount of resource required by agent i to perform task j, let p_{ij} be the profit so assigned, and let x_{ij} be a binary variable that indicates whether task j is assigned to agent i. Each task must be assigned to one agent without exceeding his resource availability. The mathematical model of GAP is

$$\begin{aligned} \max \quad & \sum_{i\in I}\sum_{j\in J} p_{ij}x_{ij}, \\ & \sum_{j\in J} a_{ij}x_{ij} \leqslant b_i, \quad i \in I, \\ & \sum_{i\in I} x_{ij} = 1, \quad j \in J, \\ & x_{ij} \in \{0,1\}, \quad i \in I,\ j \in J. \end{aligned}$$

The GAP may have no solution, and even the problem of finding a feasible assignment is NP-complete [8]. It has great practical significance [2,7–11,22,24] and leads to an extensive literature [1,4–9,12–14,17–21,23–26].

In this paper, we present a breadth-first Branch-and-Bound (BaB) algorithm for the GAP with largest-upper-bound-next branching strategy. A well-know Lagrangian relaxation is obtained by dualizing the second set of constraints (called

* Corresponding author.
E-mail address: s_haddadi@hotmail.com (S. Haddadi).

semi-assignment constraints). The relaxed problem separates into m independent knapsack problems. A standard subgradient method (SM) [15] is used at each node of the decision tree to solve the Lagrangian dual providing an upper bound. Our key contribution in this paper is a new heuristic, applied at each iteration of the SM, for constructing a feasible assignment by solving a smaller GAP. Every feasible assignment found is subjected to a solution improvement heuristic.

2. Decision tree structure

The decision tree is structured as a stack, and each node as a record containing (the items defined will be explained later):

- A set $F \subseteq J$ of 'free' tasks (not assigned by the branching strategy).
- $|F|$ subsets G_j $(\subseteq I)$, $j \in F$, where G_j is the set of agents allowed by the branching strategy to perform task j.
- One n-vector z of integers, where $z_j \in I$, $j \notin F$, indicates that task j is assigned to agent z_j.

  These items, in terms of the variables x_{ij}, signify that for $j \notin F$,

  $$x_{z_j j} = 1, x_{ij} = 0, \quad i \in I, \ i \neq z_j,$$

  and $x_{ij} = 0$ for $i \notin G_j$, $j \in F$.
- One $|F|$-vector of real numbers (Lagrangian multipliers) π_j, $j \in F$.
- m integer values β_i, $i \in I$.
- Three integer values, UB ($= -\infty$ if the node is not yet bounded), $P1$ and j^*.
- A set $H \subseteq I$.

Thus, a node consists at most of a set of n integers, $n+1$ sets of m integers, n real numbers and $n+m+3$ integer numbers. Depending on how a set is encoded, we can determine the encoding size of a node which does not exceed, even when we represent a set by a vector, $mn + 2(m+n) + 3$ integers and n real numbers. Since we need only z_j for $j \notin F$, and π_j for $j \in F$, in order to save space we might merge these two vectors into one of n real components.

We begin at the root node with: $F := J$, $G_j := I$, $j \in J$, $P1 := 0$, $UB := -\infty$, $\beta_i := b_i$, $i \in I$, $\pi_j := \max_{i \in I} p_{ij}$, $j \in J$, but H, j^*, z_j, $j \in J$, remain undefined until the node is bounded.

3. Upper bounding procedure

Given an unbounded live node, the upper bound of the underlying subproblem is $P1 + \{\min W(\pi) / \pi \in IR^{|F|}\}$, where $W(\pi)$ is the optimal value of the (relaxed) problem $R(\pi)$:

$$\begin{aligned} \max \quad & \sum_{j \in F} \sum_{i \in I | G_j \ni i} (p_{ij} - \pi_j) x_{ij} + \sum_{j \in F} \pi_j \\ & \sum_{j \in F | G_j \ni i} a_{ij} x_{ij} \leqslant \beta_i, \quad i \in I, \\ & x_{ij} \in \{0,1\}, \quad i \in I, \ j \in F, \end{aligned}$$

where

$$\beta_i = b_i - \sum_{j \notin F} a_{z_j j}, \quad i \in I,$$

$$P1 = \sum_{j \notin F} p_{z_j j}$$

are available at the node record. How these values are updated will be shown in Section 5 devoted to the branching strategy. A standard SM is used to solve the Lagrangian dual. Observe that for solving the knapsack problems only items ij such that $j \in F$, $i \in G_j$ and $p_{ij} - \pi_j > 0$ have to be considered. Let $S_i = \{ j \in F \,|\, G_j \ni i, p_{ij} - \pi_j > 0 \}$, $i \in I$, be the set of items involved in knapsack i. Since the number of iterations of the SM is fixed, the theoretical complexity of the upper bounding procedure is $\mathrm{O}\left(\sum_{i \in I} |S_i| \times \beta_i \right)$ using Dynamic Programming. However, we used a simple Backtracking algorithm [16] in our code.

At each iteration k of the SM, let $\pi^{(k)}$ be the vector of the Lagrangian multipliers. An optimal solution of problem $R(\pi^{(k)})$ is defined by a family of $|F|$ subsets of $I : X_j^{(k)} = \{ i \in I \,|\, x_{ij} = 1 \}$, $j \in F$. Let s be the last iteration of the SM. We select the task j^* (which will be used in the branching process) that contributes the most to the objective function value $W(\pi^{(s)})$.

$$j^* = \underset{j \in F/|X_j^{(s)}| \geqslant 2}{\operatorname{argmax}} \left\{ \sum_{i \in X_j^{(s)}} p_{ij} - |X_j^{(s)}|\pi_j \right\}, \qquad (1)$$

and we store in the current node: j^*, $H = X_{j^*}^{(s)}$ and $UB = P1 + \lceil W(\pi^{(s)}) \rceil$.

4. Lower bounding procedure

The crux of our method is a heuristic that tries to transform, at each iteration k of the SM, the solution $X^{(k)}$ into a feasible assignment. Let $X1 = \{j \in F/|X_j^{(k)}| = 1\}$. The heuristic operates as follows: Every task $j \in X1$ is assigned to the single agent i belonging to $X_j^{(k)}$. Setting $z_j = i$ for each $j \in X1$, $P2 = \sum_{j \in X1} p_{z_j j}$, $k_i = \beta_i - \sum_{j \in X1} a_{z_j j}$, $i \in I$, $U = U0 \cup U1$, where $U0 = \{j \in F/|X_j^{(k)}| = 0\}$ and $U1 = \{j \in F/|X_j^{(k)}| \geqslant 2\}$, we need solve a smaller GAP (called SGAP) with the remaining "unassigned" tasks (set U) and the remaining agents resource availability (k_i), but only the agents belonging to $X_j^{(k)}$ are allowed to perform a task j for which $|X_j^{(k)}| \geqslant 2$. This rule will be empirically justified later in Section 5. SGAP is the problem

$$\begin{aligned}
\max \quad & R = \sum_{i \in I} \sum_{j \in U} p_{ij} x_{ij} \\
& \sum_{i \in I} x_{ij} = 1, \quad j \in U0, \\
& \sum_{i \in X_j^{(k)}} x_{ij} = 1, \quad j \in U1, \\
& \sum_{j \in U0} a_{ij} x_{ij} + \sum_{j \in U1/X_j^{(k)} \ni i} a_{ij} x_{ij} \leqslant k_i, \quad i \in I, \\
& x_{ij} \in \{0,1\}, \quad i \in I,\ j \in U0, \\
& x_{ij} \in \{0,1\}, \quad i \in X_j^{(k)},\ j \in U1.
\end{aligned}$$

SGAP may have no solution. Assume it has an optimal solution with an objective function value $P3$. Set $z_j = i$, $j \in U$, whenever task $j \in U$ is assigned to some agent i. Then, we have a "complete" feasible assignment z_j, $j \in J$, inducing a profit $P1 + P2 + P3$.

Example (*from* [17]). The data are: $m = 3$, $n = 8$, and

$$(p_{ij}) = \begin{bmatrix} 27 & 12 & 12 & 16 & 24 & 31 & 41 & 13 \\ 14 & 5 & 37 & 9 & 36 & 25 & 1 & 34 \\ 34 & 34 & 20 & 9 & 19 & 19 & 3 & 34 \end{bmatrix},$$

$$(a_{ij}) = \begin{bmatrix} 21 & 13 & 9 & 5 & 7 & 15 & 5 & 24 \\ 20 & 8 & 18 & 25 & 6 & 6 & 9 & 6 \\ 16 & 16 & 18 & 24 & 11 & 11 & 16 & 18 \end{bmatrix},$$

$(b_i) = (26, 25, 34)$.

The solution of the relaxed problem ($\pi = 0$) is: $X_1 = \{3\}$, $X_2 = \{3\}$, $X_3 = \{1\}$, $X_4 = \{1\}$, $X_5 = \{1,2\}$, $X_6 = \{2\}$, $X_7 = \{1\}$, and $X_8 = \{2\}$, with $P1 = 0$ and $P2 = 196$. The problem SGAP is

$$\begin{aligned}
P3 = \max \quad & 24x_{15} + 36x_{25} \\
& 7x_{15} \leqslant 7, \\
& 6x_{25} \leqslant 13, \\
& x_{15} + x_{25} = 1, \\
& x_{15}, x_{25} \in \{0,1\},
\end{aligned}$$

whose optimal solution is $x_{15} = 0$ and $x_{25} = 1$. The global solution is optimal with profit $P1 + P2 + P3 = 232$.

The idea that led us to define SGAP to construct a feasible assignment came from observing the solution of problem $R(\pi)$ at successive iterations (see Table 1, for an example). There, it appears that the size of SGAP decreases as the SM progresses. Moreover, from iteration 113 on, all the tasks are assigned to one agent except a few which are either assigned to two agents or unassigned at all. So, during the last iterations, the successive solutions $X^{(k)}$ are "almost" feasible.

In our code, instead of solving SGAP, we approximated it by applying Martello and Toth's heuristic [19] with two agent/task selection functions $(p_{ij} - \pi_j)/a_{ij}$ and $(p_{ij} - \pi_j)/(a_{ij} \times k_i)$. The complexity of approximating SGAP is therefore $O(m|U|^2)$. Fortunately, as we have seen, the cardinality of U decreases as the SM proceeds. In fact, we performed the lower bounding procedure only when $|U| \leqslant 30$.

Every feasible assignment found, identified by z_j, $j \in J$, is subjected to a solution improvement heuristic. The method, sketched in [1,21], is

Table 1
Size of SGAP at successive iterations of the SM at the root node (type D problem of size 20 × 200)

Iteration	$\|U0\|$	$\|U1\|$	$\|U\|$	$\text{Max}\{\|X_j\|\}$	Size of SGAP	
					Number of constraints	Number of variables
1	200	0	200	0	220	4000
2	50	106	156	13	176	
3	71	96	167	11	187	
...						
50	37	36	73	3	93	
...						
100	16	20	36	2	56	360
...						
150	9	8	17	2	37	196

detailed in [13,14]. The feasible assignment is improved repeatedly until a local optimal solution is reached, by selecting the best improvement among all those maintaining feasibility, either by shifting a task to another agent, or by interchanging two agent/task assignments. Assuming bounded profits, the complexity of the solution improvement heuristic is $O(n \max\{m, n\})$.

5. Branching strategy

The ultimate goal is to solve the GAP by bounding as few nodes as possible. Thus a satisfactory branching rule is one that generates few successors of a node. The choice of j^* in (1) is motivated by the fact that infeasibility stemming from an unassigned task leads to processing m successor nodes near the tree root, while with our choice we generate at most $|H| + 1$ nodes, where, very often, $|H| = 2$. Once we select a node with the largest value *UB*, we find the task j^* and the set H. Let $H = \{h_1, \dots, h_r\}$, $2 \leqslant r \leqslant m$. Then we create r successor nodes and in each node record "t", $t = 1, \dots, r$, we assign task j^* to agent h_t and make the necessary updates:

- $UB^{(t)} \leftarrow -\infty$;
- $F^{(t)} \leftarrow F - \{j^*\}$;
- $G_j^{(t)} \leftarrow G_j,\ j \in F^{(t)}$;
- $\pi_j^{(t)} \leftarrow \pi_j,\ j \in F^{(t)}$;
- $P1^{(t)} \leftarrow P1 + p_{h_t j^*}$;
- $\beta_i^{(t)} \leftarrow \beta_i,\ i \in I$;
- $\beta_{h_t}^{(t)} \leftarrow \beta_{h_t} - a_{h_t j^*}$ (assuming $\beta_{h_t} > a_{h_t j^*}$);
- $z_j^{(t)} \leftarrow z_j,\ j \notin F^{(t)}$;
- $z_{j^*}^{(t)} \leftarrow h_t$.

Now, we look at the set $S = G_{j^*} \setminus H$. If it is empty, we just delete the parent node, otherwise we put in the parent node: $UB \leftarrow -\infty$, $G_{j^*} \leftarrow S$ (for reference, let us call such a node a S-node).

We observed during the computational experience that S-nodes were often fathomed because they provided upper bounds that were not greater than the global lower bound. This observation suggests that task j^* is "probably" assignable to one agent in H, and explains why, while defining SGAP, we restricted agents belonging to $H = X_j^{(s)}$ to perform a task j with $|H| \geqslant 2$.

6. Computational experience

The BaB algorithm is coded in Turbo-Pascal 7.0 on a Compatible IBM (Pentium MMX 200 MHz). The code is available from the first author simply via e-mail. We consider the treatment of the root node (without branching) as a Lagrangian heuristic. Seventy-two benchmark test problems, taken from OR-library [3], were run. Sixty of them of type C are small sized. The other twelve are medium sized, six of type C and six of type D. These problems were used in [7,14,26]. How they were generated is explained in [26]. Type D are known as the hardest instances since the coefficients p_{ij} and a_{ij} are inversely correlated. Instances of type E, used in [26], are unavailable in OR-library. However, from the results reported in [26] they seem not to be harder than type D instances.

The number of iterations of the SM allowed is 150 (resp. 80) at the root node and 75 (resp. 40) elsewhere for type D (resp. C) instances. The relaxation coefficient is halved every 8 (resp. 12) iterations for type D (resp. C) instances. Surprisingly, it was easier to get convergence to the Lagrangian bound for type D instances.

Table 2 shows the experimental results obtained. Computing times are measured in seconds (including I/O operations) using the DOS function *gettime*.

Concerning the sixty small instances of type C, the Lagrangian heuristic (results at the root node) was effective in accuracy as well as in speed. Fifty-seven of the sixty solutions found were optimal, and 60% of them were proved optimal at the root node ($LB = UB$) and needed no branching. The BaB algorithm needed never more than 25 nodes nor more than 11 seconds to confirm optimality. The results obtained on type C instances of small size suggest that they cannot be used for comparing algorithms on recent computers.

We now compare the BaB algorithm as well as the Lagrangian heuristic with the tabu search heuristic of Yagiura et al. [26] and the BaB algorithm of Nauss. All we know about the latter was

Table 2
Computational results

Type	Size	Results at root node			BaB algorithm					
		Best	Time to best	Total time	Best	Node to best	Time to best	Number of nodes	Total time	Worst local bound
C	5×100	1931	0.77	1.48	1931	1	0.77	13	7.52	
	10×100	1402	1.15	2.47	1402	1	1.15	37	20.60	
	20×100	1246	1.26	3.74	1243	14	22.19	19	26.80	
	5×200	3457	4.39	5.05	3456	15	27.68	23	34.88	
	10×200	2808	6.86	7.74	2806	50	176.97	59	193.28	
	20×200	2392	11.65	12.91	2391	32	94.42	34	96.01	
D	5×100	6362	5.49	8.95	6353	14	65.47	215	471.04	
	10×100	6368	16.26	17.96	6349	48	371.57	346	Time limit[a]	6345
	20×100	6196	17.79	24.71	6196	1	17.79	231	Time limit	6179
	5×200	12750	12.41	33.23	12742	36	535.69	155	1481.50	
	10×200	12453	26.81	43.12	12436	96	2068.11	111	Time limit	12427
	20×200	12250	55.58	58.39	12250	1	55.58	90	Time limit	12231

[a] Time limit = 2400 seconds.

Table 3
Comparison of the Lagrangian heuristic and the BAB algorithm with Yagiura et al.'s tabu search heuristic

Type	Size	Yagiura et al. (5 runs) 300 MHz		Lagrangian heuristic 200 MHz		BaB algorithm 200 MHz	
		Best	Average time to best	Best	Time to best	Best	Time to best
C	5×100	1931[a]	0.6	1931[a]	0.77	1931[a]	0.77
	10×100	1402[a]	3.0	1402[a]	1.15	1402[a]	1.15
	20×100	1243[a]	22.5	1246	1.26	1243[a]	22.19
	5×200	3456[a]	3.7	3457	4.39	3456[a]	27.68
	10×200	2806[a]	403.8	2808	6.86	2806[a]	176.97
	20×200	2391[a]	301.8	2392	11.65	2391[a]	94.42
D	5×100	6353[a]	649.2	6362	5.49	6353[a]	65.47
	10×100	6349[a]	2440.7	6368	16.26	6349[a]	371.57
	20×100	6206	1591.9	6196[a]	17.79	6196[a]	17.79
	5×200	12743	3564.8	12750	12.41	12742[a]	535.69
	10×200	12440	5829.9	12453	26.81	12436[a]	2068.11
	20×200	12277	1757.7	12250[a]	55.58	12250[a]	55.58

[a] The best objective function value attained by the three.

Table 4
Comparison of the BaB algorithm with Nauss's

Type	Size	Nauss 300 MHz time limit = 3600 seconds				BaB 200 MHz time limit = 2400 seconds			
		Best	Time to best	Total time	Opt?	Best	Time to best	Total time	Opt?
C	5×100	1931[a]	0.3	7.1	Yes	1931[a]	0.77	7.52	Yes
	10×100	1402[a]	15.5	33.2	Yes	1402[a]	1.15	20.60	Yes
	20×100	1243[a]	39.7	69.5	Yes	1243[a]	22.19	26.80	Yes
	5×200	3456[a]	36.4	45.8	Yes	3456[a]	27.68	34.88	Yes
	10×200	2806[a]	631.1	1081.0	Yes	2806[a]	176.97	193.28	Yes
	20×200	2392	926.0	Time limit	No	2391[a]	94.42	96.01	Yes
D	5×100	6353[a]	170.7	174.7	Yes	6353[a]	65.47	471.04	Yes
	10×100	6349[a]	1023.7	Time limit	?	6349[a]	371.57	Time limit	?
	20×100	6196[a]	2122.9	Time limit	?	6196[a]	17.79	Time limit	?
	5×200	12745	102.1	Time limit	No	12742[a]	535.69	1481.50	Yes
	10×200	12436[a]	1309.9	Time limit	?	12436[a]	2068.11	Time limit	?
	20×200	12264	3464.4	Time limit	No	12250[a]	55.58	Time limit	?

[a] The best objective function value attained by the two algorithms.

taken from the previous paper where the size of the decision tree was omitted. Yagiura et al. used a SunUltra2 Model 2300 (300 MHz) and Nauss, a Dell XPS D300 (300 MHz).

From [26, Table 5], it appears that the heuristic of Yagiura et al. is the most accurate among all known heuristics. But nothing is said about their relative speed. Table 3 gives the comparison of the BaB algorithm and the Lagrangian heuristic with Yagiura et al.'s. The latter found (after five runs) better solution for eight (four of type C and four of type D) of the instances, while the Lagrangian heuristic found better solution on two (of type D) of the instances. However, the Lagrangian heuristic has a crucial advantage: it is very fast and provides a gap in which the optimal solution must lie. The BaB algorithm (with limited running time) clearly outperforms the tabu search heuristic in accuracy as well as in speed.

We see in Table 4 that the BaB algorithm found better solutions for three instances, two of which were proved optimal. At the contrary, Nauss's algorithm never found better. Furthermore, the optimal (or best) solution was often found earlier by the BaB algorithm. In Table 2, for each instance which was not confirmed, we provide the worst local bound (among all the remaining live nodes) to strengthen the optimum. For example, for the second instance of type D we know that the optimal profit must lie between 6345 and 6349.

To finish, we give some suggestions drawn from the computational experience. When confronted to GAP instances, two possible situations may arise:

1. We do not insist on optimality, and good approximate solutions are satisfactory. Then, we can use the BaB algorithm, either without branching at all, or by fixing a maximum number of nodes or a maximum running time.
2. An optimal solution is essential. This case depends on the size of the instances. If the instances are of small size ($m \times n \leqslant 500$) or medium sized ($1000 \leqslant m \times n \leqslant 4000$) but of type C, we can use the BaB algorithm. Otherwise, an optimal solution cannot be guaranteed.

References

[1] M.M. Amini, M. Racer, A rigorous computational comparison of alternative solution methods for the generalized assignment problem, Management Science 40 (1994) 868–890.

[2] V. Balachandran, An integer generalized transportation model for optimal job assignment of computer networks, Operations Research 24 (1976) 742–759.

[3] J.E. Beasley, OR-library: Distributing test problems by electronic mail, Journal of Operational Research Society 41 (1990) 1069–1072.

[4] D.G. Cattrysse, L.N. Van Wassenhove, A survey of algorithms for the generalized assignment problem,

European Journal of Operational Research 60 (1992) 260–272.

[5] D.G. Cattrysse, M. Salomon, L.N. Van Wassenhove, A set partitioning heuristic for the generalized assignment problem, European Journal of Operational Research 72 (1994) 167–174.

[6] L. Chalmet, L. Gelders, Lagrangian relaxation for a generalized assignment-type problem, in: M. Roubens (Ed.), Advances in Operations Research, Amsterdam, 1977, 103–109.

[7] P.C. Chu, J.E. Beasley, A genetic algorithm for the generalized assignment problem, Computers and Operations Research 24 (1997) 17–23.

[8] M.L. Fisher, R. Jaïkumar, A generalized assignment heuristic for vehicle routing, Networks 11 (1981) 109–124.

[9] M.L. Fisher, R. Jaïkumar, L.N. Van Wassenhove, A multiplier adjustment method for the generalized assignment problem, Management Science 32 (1986) 1095–1103.

[10] F. Gheysens, B. Golden, A. Assad, A new heuristic for determining fleet size and composition, Mathematical Programming Study 26 (1986) 233–236.

[11] M.D. Grigoriadis, D.J. Tang, L.S. Woo, Considerations in the optimal synthesis of some communications networks, 45th Joint National Meeting of ORSA/TIMS, Boston, MA, 1974.

[12] M. Guignard, M. Rosenwein, An improved dual based algorithm for the generalized assignment problem, Operations Research 37 (1989) 658–663.

[13] S. Haddadi, Lagrangian decomposition based heuristic for the generalized assignment problem, Information Systems and Operational Research 37 (1999) 392–402.

[14] S. Haddadi, H. Ouzia, Effective Lagrangian heuristic for the generalized assignment problem, Information Systems and Operational Research 39 (2001) 351–356.

[15] M.P. Held, P. Wolfe, H.P. Crowder, Validation of subgradient optimization, Mathematical Programming 6 (1974) 62–88.

[16] E. Horowitz, S. Sahni, Fundamentals of computer algorithms, Computer Science Press, Rockville, MD, 1978.

[17] K. Jornsten, M. Nasberg, A new Lagrangian relaxation approach to the generalized assignment problem, European Journal of Operational Research 27 (1986) 313–323.

[18] T.D. Klastorin, An effective subgradient algorithm for the generalized assignment problem, Computers and Operations research 6 (1979) 155–164.

[19] S. Martello, P. Toth, An algorithm for the generalized assignment problem, in: J.P. Brans (Ed.), Operational Research '81, North-Holland, Amsterdam, 1981, pp. 589–603.

[20] S. Martello, P. Toth, Linear assignment problems, in: S. Martello, G. Laporte, M. Minoux, C. Ribeiro (Eds.), Surveys in Combinatorial Optimization, Annals of Discrete Mathematics, vol. 31, North-Holland, Amsterdam, 1987, pp. 103–109.

[21] J.B. Mazzola, Generalized assignment with nonlinear capacity interaction, Management Science 35 (1989) 923–941.

[22] J.B. Mazzola, A.W. Neebe, C.V.R. Dunn, Production planning of a flexible manufacturing system in a material requirements planning environment, International Journal of Flexible Manufacturing Systems 1/2 (1989) 115–142.

[23] G.T. Ross, R.M. Soland, A Branch and Bound algorithm for the generalized assignment problem, Mathematical Programming 8 (1975) 91–103.

[24] G.T. Ross, R.M. Soland, Modeling facility location problems as generalized assignment problems, Management Science 24 (1977) 345–357.

[25] M. Savelsbergh, A Branch-and-Price algorithm for the generalized assignment problem, Operations Research 45 (1997) 831–841.

[26] M. Yagiura, T. Ibaraki, F. Glover, An Ejection Chain Approach for the Generalized Assignment Problem, Technical Report #99013, Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, 1999. Available at http://www-or.amp.i.kyoto-u.ac.jp/~yagiura/papers/.