ELSEVIER

European Journal of Operational Research 82 (1995) 176–189

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

Theory and Methodology

Tabu search for the multilevel generalized assignment problem

Manuel Laguna [a], James P. Kelly [a], José Luis González-Velarde [b], Fred Glover [a,*]

[a] *Graduate School of Business and Administration, Campus Box 419, University of Colorado at Boulder, Boulder, CO 80309-0419, USA*

[b] *Centro de Sistemas de Manufactura, ITESM Campus Monterrey, Sucursal de Correos J, Monterrey, NL 64849, Mexico*

Received July 1992
Communicated by Marc Salomon

Abstract

The multilevel generalized assignment problem (MGAP) differs from the classical GAP in that agents can perform tasks at more than one efficiency level. Important manufacturing problems, such as lot sizing, can be formulated as MGAPs; however, the large number of variables in the related 0–1 integer program makes the use of commercial optimization packages impractical. In this paper, we present a heuristic approach to the solution of the MGAP, which consists of a novel application of tabu search (TS). Our TS method employs neighborhoods defined by ejection chains, that produce moves of greater power without significantly increasing the computational effort.

Keywords: Tabu search; Generalized assignment problem; Lot sizing; Resource allocation

1. Introduction

The multilevel generalized assignment problem (MGAP) was first described by Glover, Hultz and Klingman (1979) in the context of large-scale task allocation in a major manufacturing firm. In general, the MGAP consists of assigning n tasks to m agents with a maximum of l efficiency levels. Each task j must be assigned to one and only one agent i at a level k. Each agent i has a limited amount b_i of a single resource. An agent i may have more than one task assigned to it, but the sum of the resource requirements for these tasks must not exceed b_i. The resource requirements of a particular task depend on the agent and level to which the task is assigned, and they are denoted by a_{ijk} (i.e., the resources used by task j assigned to agent i at level k). The cost of assigning task j to agent i at level k is represented by c_{ijk}. In real-world problems, the relationship between cost and resource utilization for any agent-level-task assignment is such that if $a_{ijk'} < a_{ijk''}$ then $c_{ijk'} > c_{ijk''}$. The objective of this combinatorial

* Corresponding author.

SSDI 0377-2217(93)E0174-V

optimization problem is to minimize the total assignment cost. A 0–1 integer programming formulation of the MGAP follows:

$$\text{Minimize} \quad Z(x) = \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{l} c_{ijk} x_{ijk} \tag{1}$$

$$\text{subject to} \quad \sum_{i=1}^{m} \sum_{k=1}^{l} x_{ijk} = 1, \quad j = 1, \ldots, n, \tag{2}$$

$$\sum_{j=1}^{n} \sum_{k=1}^{l} a_{ijk} x_{ijk} \leq b_i, \quad i = 1, \ldots, m, \tag{3}$$

$$x_{ijk} = 0 \text{ or } 1, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n, \quad k = 1, \ldots, l.$$

In the above model, the binary variable x_{ijk} is defined to be 1 if task j is assigned to agent i at the k-th level. In the application presented by Glover, Hultz and Klingman (1979), the objective is to minimize the combine cost of production and inventory holding by determining optimal product lot sizing and optimal assignment of production to machines. There are a maximum of l possible lot sizes, and the machines work in parallel at different rates and operational costs. Some general-purpose machines are capable of producing several (or all) of the products while others are more specialized. For this application, c_{ijk} represents the combined setup, production, and holding cost (per unit time) incurred when product j is assigned to machine i in the k-th possible lot size. Therefore, for a particular product-machine pair, a small lot size results in a large combined cost and vice versa.

The mathematical model of the MGAP may be graphically represented by means of a *network* (network-related formulation). The field of netforms has allowed OR theoreticians and practitioners to formulate problems as networks or generalized networks, even in contexts that do not immediately

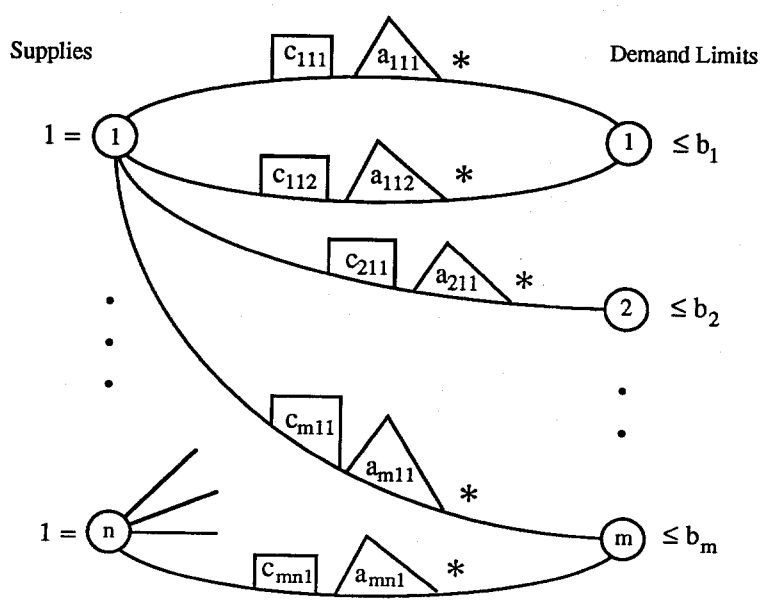

Fig. 1. Netform representation of the MGAP

suggest this possibility (see, e.g., the survey by Glover, Klingman and Phillips 1989). This field also introduces notational conventions that make possible the representation of problems in a network-related framework, including the identification of flow restrictions such as 'all or none' and 'multiple choice' and integer restrictions in generalized networks. The use of these conventions allows for a concise representation of network programming problems with 'side conditions'.

The MGAP netform consists of $n+m$ nodes and at most $m \times l \times n$ arcs (i.e., there are at most l levels for each agent-task pair). For each source node j there is a fixed supply of one unit, and for each destination node i there is a demand that must not exceed b_i units. Every arc has a lower and an upper bound set to zero and one, respectively. The flows in every arc are restricted to be integers, therefore they can only be 0 or 1. There is also a cost c_{ijk} and multiplier a_{ijk} for every arc, which respectively identify the cost and capacity requirement for a task j that is being assigned to agent i at level k. The combination of arc multipliers and the 0-1 integer restriction results in a netform that is generally referred to as *integer network* or *0–1 generalized network*. A diagram of the MGAP netform appears in Fig. 1, where costs are enclosed in boxes, multipliers are enclosed in triangles, and integer restrictions for arc flows are indicated by asterisks.

MGAP formulations of 'real-world' problems contain a large number of binary variables, eliminating (for all practical purposes) the possibility of finding optimal solutions. In the following sections we present a heuristic approach to the solution of large MGAPs. Our method is based on the tabu search methodology, and incorporates a number of novel features, including the use of a highly effective move mechanism derived from the notion of ejection chains defined on the MGAP netform.

2. Ejection chains

A neighborhood structure for defining moves, based on the notion of ejection chains, was introduced by Glover (1991) in the context of the traveling salesman problem. The proposed method may be implemented at a variety of levels resulting in an interplay between simplicity of programming and sophistication of search. The neighborhoods defined by ejection chains provide the foundation for the more advanced levels of the solution method. Such neighborhoods are designed to produce moves of greater power with an efficient investment of computer effort.

An ejection chain is an embedded neighborhood construction that compounds the neighborhoods of simple moves to create more complex and powerful moves. A very simple move definition for the MGAP is one that changes the value of x_{ijk} from 0 to 1 and the value of $x_{i'jk'}$ from 1 to 0. In terms of the netform representation, this simple move is equivalent to deleting the k-th arc going from node j to node i, and inserting arc k' that goes from node j to node i'. (Note that in this context it is possible for node i' to be the same as node i representing a change of efficiency levels within the same agent.) An ejection chain results by allowing this move to 'eject' an element (or arc) at node i', which must then be 'repositioned' (replaced by another arc from the same source) to lead to some new node i''. The destination node i'' is given in the form of a pointer for each task, to avoid the computational cost of searching for a new agent to assign the task ejected by j. The process may continue through additional nodes until a suitable termination criterion is met. Our adaptation of the ejection chain idea to the present setting may be viewed as a heuristic generalization of a weighted alternating path approach, as applied in the solution of matching problems.

Before we present the different types of ejection chains used in our implementation, let us define s_i, the capacity slack of agent i, as follows. Let $\boldsymbol{x}$ be a given solution to the MGAP that satisfies the assignment constraints (2) but does not necessarily meet the capacity restrictions (3), then

$$s_i = b_i - \sum_{j=1}^{n} \sum_{k=1}^{l} a_{ijk} x_{ijk}.$$

Table 1
Ejection chains

Type	Leaving arcs	Entering arcs	Feasibility condition
Simple (a)	(i_1, j_1, k_1)	(i_1, j_1, k_2)	$s_{i_1} + a_{i_1 j_1 k_1} - a_{i_1 j_1 k_2} \geq 0$
(b)	(i_1, j_1, k_1)	(i_2, j_1, k_2)	$s_{i_2} - a_{i_2 j_1 k_2} \geq 0$
Double (a)	$(i_1, j_1, k_1)(i_1, j_2, k_3)$	$(i_1, j_1, k_2)(i_1, j_2, k_4)$	$s_{i_1} + a_{i_1 j_1 k_1} + a_{i_1 j_2 k_3} - a_{i_1 j_1 k_2} - a_{i_1 j_2 k_4} \geq 0$
(b)	$(i_1, j_1, k_1)(i_1, j_2, k_3)$	$(i_1, j_1, k_2)(i_2, j_2, k_4)$	$s_{i_1} + a_{i_1 j_1 k_1} + a_{i_1 j_2 k_3} - a_{i_1 j_1 k_2} \geq 0$
(c)	$(i_1, j_1, k_1)(i_2, j_2, k_3)$	$(i_2, j_1, k_2)(i_2, j_2, k_4)$	$s_{i_2} + a_{i_2 j_2 k_3} - a_{i_2 j_1 k_2} - a_{i_2 j_2 k_4} \geq 0$
(d)	$(i_1, j_1, k_1)(i_2, j_2, k_3)$	$(i_2, j_1, k_2)(i_3, j_2, k_4)$	$s_{i_2} + a_{i_2 j_2 k_3} - a_{i_2 j_1 k_2} \geq 0$
Circular	$(i_1, j_1, k_1)(i_2, j_2, k_3)$	$(i_2, j_1, k_2)(i_1, j_2, k_4)$	$s_{i_2} + a_{i_2 j_2 k_3} - a_{i_2 j_1 k_2} \geq 0$

Our search procedure assures that the assignment constraints (2) are always satisfied; however, infeasibility may occur due to the violation of the capacity constraints (3). Therefore, infeasible solutions are those for which s_i is strictly less than zero for at least one i. A measure of infeasibility, v, of an assignment-feasible solution $\boldsymbol{x}$, may be defined as the absolute value of the sum of all the negative capacity slacks, i.e., $v(\boldsymbol{x}) = \text{abs}(\sum_{i=1}^{m} \min(s_i, 0))$. Feasibility plays an important role in the selection of the ejection chains defined within our solution procedure.

Table 1 summarizes the types of ejection chains that are used to construct the candidate list of moves at each iteration of our solution approach. The first column of this table assigns a name to each type of chain. The next two columns show the entering and leaving arcs (or binary variables) within the chain. An arc is considered to be 'entering' when its flow changes from the lower limit 0 to the upper limit 1. A leaving arc is one for which its flow changes from 1 to 0. The feasibility condition of each chain assures that the capacity constraint of the 'ejection' node is satisfied. The ejection node is simply the index of the agent where an ejection occurs. For very small problems (i.e., $n < 10$ and $m \times l < 5$), there may be solution states for which no chain satisfies the feasibility condition. In these situations, the condition is relaxed, and the chains are selected in such way that the infeasibility in the ejection node is minimized. The entering arc associated with task j_2, e.g., arc (i_3, j_2, k_4), is selected such that the difference between its cost and the cost of the leaving arc, e.g., (i_2, j_2, k_3), is minimized (i.e., $c_{i_3 j_2 k_4} - c_{i_2 j_2 k_3}$ is minimum). Appendix A shows a graphical representation of the ejection chains summarized in Table 1.

3. Tabu search

Early heuristic procedures based on the tabu search (TS) framework typically focused on short term memory issues. Most implementations consisted of simple fixed-size tabu lists as the primary design for short term memory, and some type of re-starting mechanism to achieve a limited level of diversification. However, recent TS applications for the solution of hard combinatorial optimization problems (see e.g., Kelly, Golden and Assad 1993; Glover and Laguna, 1993; Crainic et al., 1991) have shown the merit of designing procedures that contain more elaborate intensification and diversification strategies. In our TS method for the solution of the MGAP, we have designed a tabu structure that consists of multiple dynamic tabu lists with a long term memory component, and a strategic oscillation element that allows searching paths to cross the capacity–feasibility boundary.

After a move has been executed, the leaving arcs (or arc, in the case of a simple chain) become(s) tabu. The number of iterations that a leaving arc remains tabu is a function of three elements: (1) the

number of alternative arcs out of the same node as the leaving arc; (2) the difference between the ranks of the leaving and entering arcs; and (3) the frequency that the leaving arc has been a member of a previously selected ejection chain. Let the leaving and the entering arcs be (i_L, j, k_L) and (i_E, j, k_E), respectively. Then, the number of iterations that (i_L, j, k_L) will remain tabu (i.e., the time in which the leaving arc is not allowed to be part of the solution), is given by the following expression:

$$\text{tabu\_time}(i_L, j, k_L) = \kappa(j)\left(\frac{3}{2} + \frac{\Delta}{2(\kappa(j)-1)}\right) + \frac{m \times l \times \phi(i_L, j, k_L)}{\phi_{\max}}$$

where:

$\phi(i_L, j, k_L) =$ Number of times arc (i_L, j, k_L) has been part of an executed move.
$\phi_{\max} =$ Maximum $\phi(i_L, j, k_L)$, for all i, j, k.
$\kappa(j) =$ Number of arcs out of node j.
$\Delta =$ $\rho(i_L, j, k_L) - \rho(i_E, j, k_E)$.
$\rho(i, j, k) =$ The position of the arc (i, j, k) when all arcs out of node j are ordered by increasing value of the cost–resource ratio.

The minimum tabu_time value is $\kappa(j)$ and occurs when the entering arc has the worst cost–resource ratio (i.e., $\rho(i_E, j, k_E) = \kappa(j)$), the leaving arc has never before been part of an executed chain (i.e., $\phi(i_L, j, k_L) = 0$), and the leaving arc's cost–resource ratio is the best (i.e., $\rho(i_L, j, k_L) = 1$). The maximum tabu_time value is $2 \times \kappa(j) + m \times l$ and occurs when $\rho(i_E, j, k_E) = 1$, $\rho(i_L, j, k_L) = \kappa(j)$, and $\phi(i_L, j, k_L) = \phi_{\max}$.

The tabu structure defined above may be viewed as a form of multiple lists of dynamic size. There is one list for each task j and the size depends on the number of possible ways that the task can be assigned (i.e., the number of arcs out of node j). The long term memory element of the list is implemented in form of a frequency count. This component does not depend on the tasks and it is only a function of the history of the current search. The tabu status of a move is overridden if the move leads the search to the best feasible solution ever found.

The starting point for the TS procedure is the solution x_0 that results from selecting the arc with the lowest cost out of each node j. The objective function $Z(x_0)$ is then a lower bound to $Z(x^*)$, where x^* is the global optimum. Therefore, if $\nu(x_0) = 0$ then x_0 is the optimal solution. However, it has been shown (Fisher, Jaikumar and Van Wassenhove, 1986) that $Z(x_0)$ is generally a poor lower bound to the single-level GAP. In the tabu search methodology, it is important to create *best move* definitions that depend on the search state. In our approach, we establish the relationship between the solution state and the best neighbor of the current solution in the following manner. Let x be the current solution, x' the neighbor solution for which $Z(x') - Z(x)$ is minimum (hence, typically $\nu(x') > 0$), and x'' be the neighbor solution for which $\nu(x'') - \nu(x)$ is minimized (here, ties are broken with reference to the change on the objective function value), then the best neighbor is defined as follows:

Solution state	Best neighbor
$\nu(x) > 0$	x''
$\nu(x) = 0$ and $\nu(x'') > 0$	x'
$\nu(x) = 0$ and $\nu(x'') = 0$ and $Z(x'') \leq \tau$	x''
$\nu(x) = 0$ and $\nu(x'') = 0$ and $Z(x'') > \tau$	x'

When the current solution is infeasible, the best neighbor is the one that reduces infeasibility the most. When the search is inside the feasible region, but no neighbor exists within this region, the search moves to be infeasible neighbor with the smallest objective function value. Nonimproving moves within

the feasible region are accepted as long as the resulting objective function value does not exceed the threshold τ. The search is forced into infeasibility when the objective function values of the neighboring feasible solutions exceed the τ-value. The τ-value is defined as the objective function value of the first solution found during the last time the search entered the feasible region. A record is kept of the best solution (x_B) found every time the search enters feasibility. If $m \times l \times n$ iterations elapse without improving $Z(x_B)$ then τ is reset to $Z(x)$ causing the next nonimproving search state to select an infeasible neighbor.

4. Computational experiments

A number of optimal procedures for the single-level GAP have been reported in the literature (see, e.g. Barcia and Jörnsten, 1990; Fisher, Jaikumar and Van Wassenhove, 1986; Guignard and Rosenwein, 1989; Jörnsten and Näsberg, 1986; Klastorin, 1979; Martello and Toth, 1990; Ross and Soland, 1975). A comprehensive review of these methods can be found in Osman (1991). One of the most referenced papers in this area is the work by Ross and Soland (1975). This paper describes a branch-and-bound procedure for the optimal solution of the classical single-level GAP. The computational experiments 'apparently' show that the method is able to optimally solved problems with up to 4000 binary variables (20 agents and 200 tasks). However, through the course of our research we have been able to determine that the problems used by Ross and Soland (1975) are easily solvable by commercial optimization packages using a 'generic' branch-and-bound procedure. Specifically, we generated 7 problems with 20 agents and 200 tasks using the random generator described in Ross and Soland (1975). (This generator was later referred to as A in Martello and Toth (1981).) We were able to solve these problems to optimality using the Cplex(MIP) optimization package. The median number of nodes generated by Cplex was 35, which compares favorable with the median of 75 obtained in a similar experiment reported in Ross and Soland (1975). The corresponding CPU times have a median value of 7.27 seconds using a DECstation 5000/120. Fisher, Jaikumar and Van Wassenhove (1986) have also experimentally shown that Ross and Soland's method is not able to solve quite small problem (5 agents and 20 tasks) when these problems are generated using a design that establishes an inverse relationship between cost and resources for every agent–task pair. From the four generators described in Martello and Toth (1981), only the one labeled D relates the cost and resource requirement values. After initial experimentation with problem generators A, B, and C, we determined that the problem instances did not represent a challenge for our procedure and decided to discontinue the use of these random problems.

The first set of experiments are focused on measuring the merit of our procedure in the single-level case. We generated 100 D-type problems with 5 agents and 20 tasks, and solved them to optimality using the branch-and-bound procedure described in Martello and Toth (1990). The D generator uses the following integer uniform distribution functions:

$$a_{ij} \in \text{Uniform}[1, 100],$$

$$c_{ij} = 111 - a_{ij} + \text{Uniform}[-10, 10],$$

$$b_i = 0.8 \times \left(\sum_{j=1}^{n} a_{ij} \right) / m.$$

The results of running 100 thousand iterations of the TS procedure (which corresponds to approximately 250 CPU seconds of the DECstation 5000/120) for each problem are presented in Table 2.

Table 2
Experiments with 100 D-type problems with 5 agents and 20 tasks

	Tabu Search	M&T Heuristic	M&T Exact
Optimal solutions	94	0	100
Percentage deviation from optimality:			
Minimum	0.000	0.567	0.000
Average	0.014	7.958	0.000
Maximum	0.715	15.028	0.000
CPU seconds to best (optimal):			
Average	24.75	0.00613	27.93
Maximum	244.98	0.00781	356.27

The column labeled 'M&T Heuristic' in Table 2 corresponds to the heuristic procedure used by Martello and Toth (1990) to find an upper bound at the top level of the branching tree. This heuristic is extremely fast, as indicated by its average and maximum running times, but its average deviation from optimality is approximately 8%. The TS procedure is able to find 94 out of 100 optimal solutions, and this number may be increased by allowing the search to continue beyond 100 000 iterations. However, for this kind and size of problem the exact method developed by Martello and Toth seems to be preferable, since it finds all optimal solutions within 6 minutes of CPU time.

Although the branch and bound procedure performs extremely well on relatively small problem instances, our conjecture is that as the problems increase in size, the TS method becomes more attractive in terms of the tradeoff between solution quality and computational effort. In order to investigate this issue, we created a set of 10 D-type problems with 5 agents and 25 tasks. Note that the problems in this set contain only 5 more tasks than those in the previous experiment (i.e., these problems are only 25%

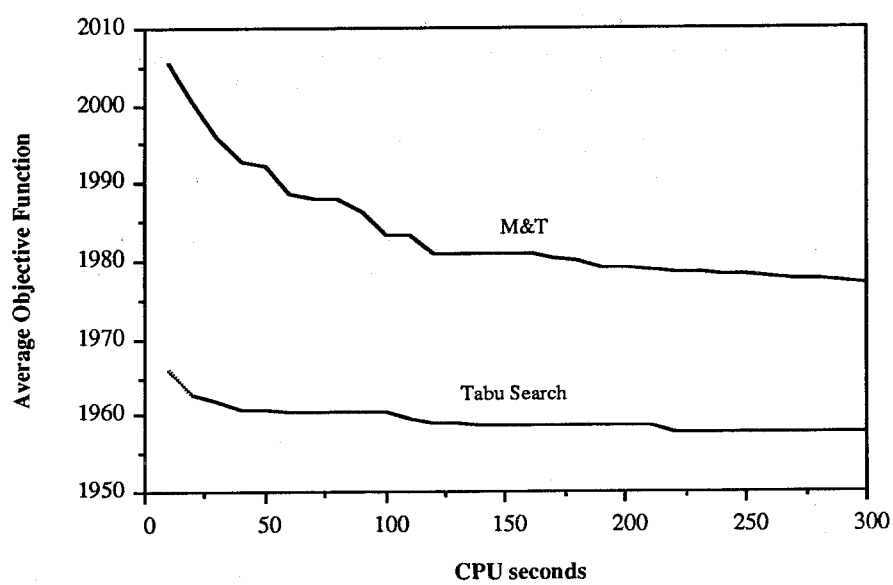

Fig. 2. D-type problems with 5 agents and 25 tasks

larger). Since finding and confirming optimal solutions to these problems requires an unreasonable amount of computer time, we compare the average solution quality of the M&T branch-and-bound and our TS method during the first 5 minutes of CPU time. A plot of the average solution value versus CPU seconds is presented in Fig. 2. This figure shows that the TS procedure is able to obtain a better average solution than the M&T branch-and-bound during the specified running time. It is reasonable to conclude that the TS method should be preferred over the M&T exact procedure for problems where the product of the number of agents and tasks (i.e., the number of binary variables in the integer programming model) exceeds 100.

One of the goals of our experimentation is to show that the TS method is robust and maintains a good level of performance across experiments with structurally different problem instances of various sizes. We emphasize that our procedure requires no adjustment of parameters, hence eliminating the need for preliminary experiments to 'tune' the method. A structurally different set of problems may be obtained by allowing the resource requirements to be distributed according to an exponential distribution. This probability function more accurately captures the disparity among equipment in actual production facilities, where the setup time for highly specialized machines greatly differs from the time taken to prepare general purpose machinery. The following is a description of a random problem generator that we would like to label with the letter E. This generator may be used to create random instances of full or sparse single and multilevel GAPs.

$$a_{ijk} = 1 - 10 \times \ln(\text{Uniform}(0, 1]) \qquad \text{with probability } p, \text{ and with probability } 1-p \text{ arc } (i, j, k) \text{ is not included,}$$

$$c_{ijk} = 1000/a_{ijk} - 10 \times \text{Uniform}[0, 1],$$

$$b_i = \max\left(\frac{0.8}{m} \times \sum_{j=1}^{n} \sum_{k=1}^{\kappa_i(j)} \frac{a_{ijk}}{\kappa_i(j)}, \max_{\text{all } j,k} a_{ijk}\right).$$

In the equation to calculate the b_i-values, $\kappa_i(j)$ denotes the number of arcs going from node j to node i. For single-level problems with $p = 1$, $\kappa_i(j)$ has a value of 1 for all values of i and j.

We used the E generator to create single-level GAP instances with 5 agents and number of tasks ranging from 15 to 20. For each size, 10 problems were generated and optimally solved with Martello and Toth's procedure (labeled M&T in Fig. 3). The TS method was able to match the optimal solutions in every case (i.e., 60 instances). The average CPU time (in seconds) to optimality was computed for each problem size and the results are shown in Figure 3. Note that the vertical axis is a logarithmic scale. Hence, it is clear from this figure that the TS procedure consistently finds optimal solutions in several orders of magnitude less time than the M&T approach, and that this difference becomes more extreme as the problem size increases (e.g., the average time to optimality for problems with 5 agents and 20 tasks is 1.28 seconds for the TS method and 4944.81 seconds for Martello and Toth branch-and-bound). This figure also illustrates the difficulties encountered by the branch-and-bound designed by Martello and Toth in solving problem instances where the coefficients are not uniformly distributed.

The next set of experiments focuses on assessing the performance of the TS method on instances of the MGAP for different p-values. For these experiments we use the E generator to create 9 sets of 10 problems with numbers of agents and levels ranging from 3 to 5, numbers of tasks ranging from 20 to 40, and p-values of 1, 0.75, and 0.5. The best (upper-bound) solutions for these problems were found using Cplex(MIP), since the Martello and Toth method does not apply to this problem and, to the best of our knowledge, no specialized optimal procedure exists for the solution of the MGAP.

Table 3 summarizes the results from solution trials of 180 CPU seconds for every problem in each set. The time limit was chosen to allow Cplex to run under the 'best-bound' branch-and-bound strategy without exhausting the random access memory of our computer (16 megabytes). The two 'Time to Best'

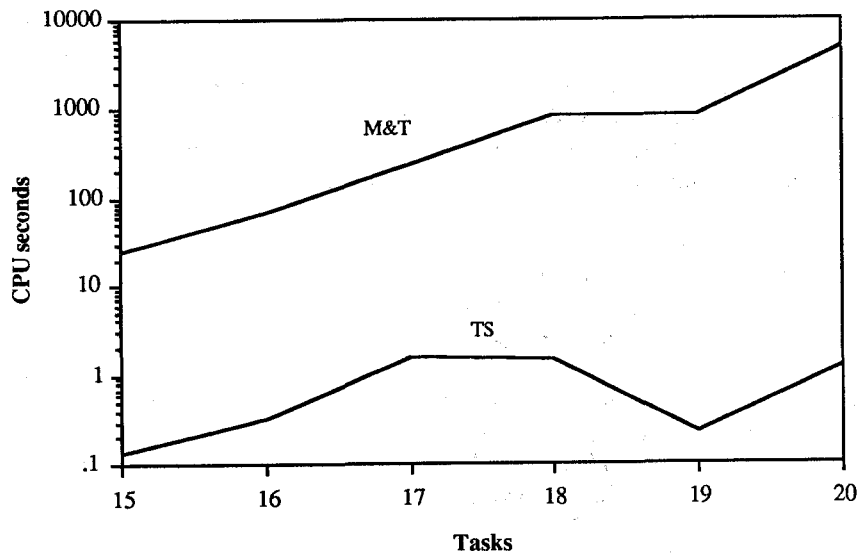

Fig. 3. Comparison of average time to optimal

columns in Table 3 contain the average CPU seconds that each procedure required to find its best solutions to each problem. The two 'Objective Value' columns identify the average value of the best solutions found, within the specified time limit, by the corresponding methods. The 'Model size' column gives the average number of binary variables in the MIP formulation. The values in the column labeled 'Optima' indicate the number of problems for which Cplex confirmed optimality. The TS solutions are on the average at least as good as the Cplex solutions in 7 out of 9 cases. This experiment disclosed the that the TS procedure experienced more difficulty with sparse problems where the number of levels is larger than the number of agents. However, the average deviation from optimality for the problems in the (0.75: 3, 5, 25) set is only 0.33%.

Our final experiment consists of using the TS method to obtain the best known solution to a real-world lot sizing problem. The problem is to find the optimal lot sizes for 30 jobs that are to be processed on 7 machines with equal capacity of 2106 units. Each job may be processed in up to 3 different lot sizes. This problem results in an integer programming formulation with 338 binary variables (i.e., the coefficient matrix is approximately 54% full) and 37 constraints. Appendix B contains the data

Table 3
Summary of results for MGAP instances

Problem set (p: m, l, n)	TS Method		Cplex (MIP)			
	Time to Best	Objective Value	Time to Best	Objective Value	Model Size	Optima
(1.0: 3, 5, 20)	65.028	2280.6	77.121	2280.6	300	4
(1.0: 5, 3, 20)	82.079	1664.2	79.429	1705.8	300	0
(1.0: 4, 4, 20)	62.508	1985.6	106.156	2020.6	320	0
(0.75: 3, 5, 25)	98.098	3005.1	68.865	2995.2	281	10
(0.75: 5, 3, 25)	89.459	2660.5	91.889	2734.2	280	0
(0.75: 4, 4, 25)	79.752	2889.1	115.176	2909.9	298	1
(0.5: 3, 5, 40)	87.391	5184.8	67.239	5159.3	299	8
(0.5: 5, 3, 40)	108.575	5406.3	76.547	5591.8	300	1
(0.5: 4, 4, 40)	90.424	5174.8	103.717	5195.0	320	2

for this problem as expressed in a netform representation. The problem was run on Cplex(MIP) for more than 50 hours, this time using the 'depth-first' strategy to avoid computer memory problems. Cplex(MIP) was unable to find a single feasible solution during this time. The best solution found by our TS approach, and therefore the best known solution to this problem, has an objective function value of 691 634, and it was found in 120.07 CPU seconds.

5. Conclusions

We have presented a TS method with search neighborhoods defined by ejection chains for the solution of MGAP instances. Our procedure has been also tested on difficult single-level GAP instances and has shown a consistently high level of performance. Furthermore, the procedure does not require parameter adjustments to maintain its high performance level when solving a variety of structurally different problems. This represents a notable departure from most heuristic approaches. The TS method was also successful in finding a feasible solution to a real-world problem for which a state-of-the-art and commercially available optimization package was unable to find any feasible solutions.

There are several avenues for refining and extending our work. Strategies of scaling the capacity requirements (and hence the arc multipliers) have been proved useful in settings employing surrogate constraints (Glover, 1977; Freville and Plateau, 1991). In addition, infeasibility measures based on accentuating the scaled constraint violations (as by squaring) may also be relevant where problem coefficients are not congenially distributed. Beyond this, our restriction to ejection chains of at most two links is not at all essential, and the incorporation of larger ejection chains may yield additional benefits.

Appendix A

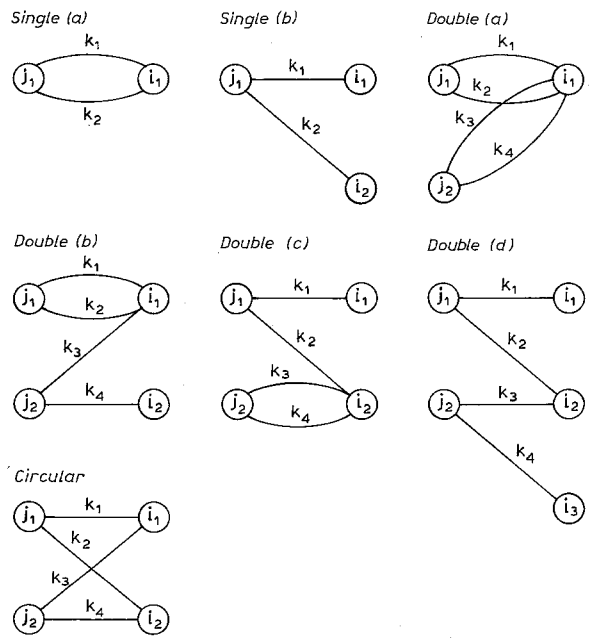

Fig. A.1. Ejection chains.

Appendix B

Table B.1

Arc No.	Agent	Task	Level	Cost	Resources	Arc No.	Agent	Task	Level	Cost	Resources
1	1	1	1	46743	1176	56	1	28	2	54007	152
2	1	1	2	48745	1020	57	1	28	3	57630	1100
3	1	1	3	52644	968	58	1	29	1	49072	1211
4	1	2	1	33871	654	59	1	29	2	51696	1031
5	1	2	2	36358	598	60	1	29	3	56507	971
6	1	3	1	5627	165	61	1	30	1	6029	174
7	1	4	1	45488	686	62	2	1	1	36984	1004
8	1	5	1	35757	575	63	2	1	2	39161	848
9	1	6	1	12809	332	64	2	1	3	43261	796
10	1	6	2	13493	272	65	2	2	1	26475	725
11	1	7	1	35008	692	66	2	2	2	26477	557
12	1	7	2	37535	624	67	2	2	3	29020	501
13	1	8	1	5748	168	68	2	10	1	23961	478
14	1	9	1	6268	181	69	2	10	2	25636	414
15	1	10	1	25717	550	70	2	11	1	22547	539
16	1	10	2	27324	486	71	2	11	2	24256	479
17	1	11	1	27610	629	72	2	13	1	47780	1094
18	1	11	2	29271	569	73	2	13	2	51446	902
19	1	12	1	26139	748	74	2	13	3	58008	838
20	1	12	2	27209	568	75	2	15	1	15039	365
21	1	12	3	30466	508	76	2	15	2	16194	313
22	1	13	1	54914	1272	77	2	18	1	35016	837
23	1	13	2	58356	1080	78	2	18	2	36525	645
24	1	13	3	64657	1016	79	2	18	3	40893	581
25	1	14	1	13828	353	80	2	19	1	28299	815
26	1	14	2	14581	293	81	2	19	2	28790	659
27	1	15	1	18003	417	82	2	19	3	31177	607
28	1	15	2	19112	365	83	2	21	1	33153	874
29	1	16	1	5967	163	84	2	21	2	33915	718
30	1	17	1	21478	640	85	2	21	3	36599	666
31	1	17	2	21811	484	86	2	23	1	4845	150
32	1	17	3	24040	432	87	2	24	1	6243	176
33	1	18	1	39658	950	88	2	28	1	41039	1109
34	1	18	2	41083	758	89	2	28	2	42972	953
35	1	18	3	45368	694	90	2	28	3	46802	901
36	1	19	1	37927	940	91	2	29	1	39506	1041
37	1	19	2	38342	784	92	2	29	2	42346	861
38	1	19	3	40653	732	93	2	29	3	47374	801
39	1	20	1	19894	451	94	3	1	1	36984	1004
40	1	20	2	21249	399	95	3	1	2	39161	848
41	1	21	1	41669	1015	96	3	1	3	43261	796
42	1	21	2	42338	859	97	3	2	1	26475	725
43	1	21	3	44931	807	98	3	2	2	26477	557
44	1	22	1	42341	1063	99	3	2	3	29020	501
45	1	22	2	44401	883	100	3	10	1	23961	478
46	1	22	3	48684	823	101	3	10	2	25636	414
47	1	23	1	5733	162	102	3	11	1	22547	539
48	1	24	1	7623	193	103	3	11	2	24256	479
49	1	25	1	9606	280	104	3	13	1	44940	1094
50	1	25	2	9696	220	105	3	13	2	48606	902
51	1	26	1	11003	300	106	3	13	3	55168	838
52	1	26	2	11389	248	107	3	15	1	15039	365
53	1	27	1	12211	324	108	3	15	2	16194	313
54	1	27	2	12748	264	109	3	18	1	33206	837
55	1	28	1	52281	1308	110	3	18	2	34715	645

Table B.1 (continued)

Arc No.	Agent	Task	Level	Cost	Resources	Arc No.	Agent	Task	Level	Cost	Resources
111	3	18	3	39083	581	166	4	21	3	44931	807
112	3	19	1	28299	815	167	4	22	1	42341	1063
113	3	19	2	28790	659	168	4	22	2	44401	883
114	3	19	3	31177	607	169	4	22	3	48684	823
115	3	21	1	33153	874	170	4	23	1	5733	162
116	3	21	2	33915	718	171	4	24	1	7623	193
117	3	21	3	36599	666	172	4	25	1	9606	280
118	3	23	1	5029	150	173	4	25	2	9696	220
119	3	24	1	6243	176	174	4	26	1	11003	300
120	3	28	1	41039	1109	175	4	26	2	11389	248
121	3	28	2	42972	953	176	4	27	1	12211	324
122	3	28	3	46802	901	177	4	27	2	12748	264
123	3	29	1	39506	1041	178	4	28	1	52281	1308
124	3	29	2	42346	861	179	4	28	2	54007	1152
125	3	29	3	47374	801	180	4	28	3	57630	1100
126	4	1	1	46743	1176	181	4	29	1	49072	1211
127	4	1	2	48745	1020	182	4	29	2	51696	1031
128	4	1	3	52644	968	183	4	29	3	56507	971
129	4	2	1	33871	654	184	4	30	1	6029	174
130	4	2	2	36358	598	185	5	1	1	36984	1004
131	4	3	1	5807	165	186	5	1	2	39161	848
132	4	4	1	47288	686	187	5	1	3	43261	796
133	4	5	1	36729	575	188	5	2	1	26475	725
134	4	6	1	13417	332	189	5	2	2	26477	557
135	4	6	2	14100	272	190	5	2	3	29020	501
136	4	7	1	35008	692	191	5	10	1	23961	478
137	4	7	2	37535	624	192	5	10	2	25636	414
138	4	8	1	5558	168	193	5	11	1	22547	539
139	4	9	1	6268	181	194	5	11	2	24256	479
140	4	10	1	24287	550	195	5	13	1	47780	1094
141	4	10	2	25894	486	196	5	13	2	51446	902
142	4	11	1	27610	629	197	5	13	3	58008	838
143	4	11	2	29271	569	198	5	15	1	15039	365
144	4	12	1	26139	748	199	5	15	2	16194	313
145	4	12	2	27209	568	200	5	18	1	35016	837
146	4	12	3	30466	508	201	5	18	2	36525	645
147	4	13	1	54914	1272	202	5	18	3	40893	581
148	4	13	2	58356	1080	203	5	19	1	28299	815
149	4	13	3	64657	1016	204	5	19	2	28790	659
150	4	14	1	13828	353	205	5	19	3	31177	607
151	4	14	2	14581	293	206	5	21	1	30905	874
152	4	16	1	5957	163	207	5	21	2	31667	718
153	4	17	1	21478	640	208	5	21	3	34351	666
154	4	17	2	21811	484	209	5	23	1	5029	150
155	4	17	3	24040	432	210	5	24	1	6243	176
156	4	18	1	39658	950	211	5	28	1	41039	1109
157	4	18	2	41083	758	212	5	28	2	42972	953
158	4	18	3	45368	694	213	5	28	3	46802	901
159	4	19	1	37927	940	214	5	29	1	39506	1041
160	4	19	2	38342	784	215	5	29	2	42346	861
161	4	19	3	40653	732	216	5	29	3	47374	801
162	4	20	1	18714	451	217	6	1	1	46743	1176
163	4	20	2	20069	399	218	6	1	2	48745	1020
164	4	21	1	41669	1015	219	6	1	3	52644	968
165	4	21	2	42338	859	220	6	2	1	33871	654

Table B.1 (continued)

Arc No.	Agent	Task	Level	Cost	Resources	Arc No.	Agent	Task	Level	Cost	Resources
221	6	2	2	36358	598	276	6	29	3	56507	971
222	6	3	1	5807	165	277	6	30	1	6029	174
223	6	4	1	47288	686	278	7	1	1	50201	1176
224	6	5	1	36729	575	279	7	1	2	52203	1020
225	6	6	1	13417	332	280	7	1	3	56101	968
226	6	6	2	14100	272	281	7	2	1	33871	654
227	6	7	1	33055	692	282	7	2	2	36358	598
228	6	7	2	35582	624	283	7	3	1	5627	165
229	6	8	1	5748	168	284	7	4	1	45488	686
230	6	9	1	6268	181	285	7	5	1	35757	575
231	6	10	1	25717	550	286	7	6	1	12809	332
232	6	10	2	27324	486	287	7	6	2	13493	272
233	6	11	1	27610	629	288	7	7	1	33055	692
234	6	11	2	29271	569	289	7	7	2	35582	624
235	6	12	1	26139	748	290	7	8	1	5748	168
236	6	12	2	27209	568	291	7	9	1	6268	181
237	6	12	3	30466	508	292	7	10	1	24287	550
238	6	13	1	51364	1272	293	7	10	2	25894	486
239	6	13	2	54806	1080	294	7	11	1	27610	629
240	6	13	3	61107	1016	295	7	11	2	29271	569
241	6	14	1	13828	353	296	7	12	1	26139	748
242	6	14	2	14581	293	297	7	12	2	27209	568
243	6	15	1	18003	417	298	7	12	3	30466	508
244	6	15	2	19112	365	299	7	13	1	54914	1272
245	6	16	1	6195	163	300	7	13	2	58356	1080
246	6	17	1	22788	640	301	7	13	3	64657	1016
247	6	17	2	23121	484	302	7	14	1	13828	353
248	6	17	3	25350	432	303	7	14	2	14581	293
249	6	18	1	37396	950	304	7	15	1	18003	417
250	6	18	2	38821	758	305	7	15	2	19112	365
251	6	18	3	43105	694	306	7	16	1	5957	163
252	6	19	1	37927	940	307	7	17	1	21478	640
253	6	19	2	38342	784	308	7	17	2	21811	484
254	6	19	3	40653	732	309	7	17	3	24040	432
255	6	20	1	21074	451	310	7	18	1	39658	950
256	6	20	2	22429	399	311	7	18	2	41083	758
257	6	21	1	41669	1015	312	7	18	3	45368	694
258	6	21	2	42338	859	313	7	19	1	37927	940
259	6	21	3	44931	807	314	7	19	2	38342	784
260	6	22	1	39531	1063	315	7	19	3	40653	732
261	6	22	2	41591	883	316	7	20	1	21074	451
262	6	22	3	45874	823	317	7	20	2	22429	399
263	6	23	1	5733	162	318	7	21	1	41669	1015
264	6	24	1	7623	193	319	7	21	2	42338	859
265	6	25	1	9606	280	320	7	21	3	44931	807
266	6	25	2	9696	220	321	7	22	1	42341	1063
267	6	26	1	11003	300	322	7	22	2	44401	883
268	6	26	2	11389	248	323	7	22	3	48684	823
269	6	27	1	12211	324	324	7	23	1	5733	162
270	6	27	2	12748	264	325	7	24	1	7623	193
271	6	28	1	52281	1308	326	7	25	1	9606	280
272	6	28	2	54007	1152	327	7	25	2	9696	220
273	6	28	3	57630	1100	328	7	26	1	11865	300
274	6	29	1	49072	1211	329	7	26	2	12251	248
275	6	29	2	51696	1031	330	7	27	1	12211	324

Table B.1 (continued)

Arc No.	Agent	Task	Level	Cost	Resources	Arc No.	Agent	Task	Level	Cost	Resources
331	7	27	2	12748	264	335	7	29	1	49072	1211
332	7	28	1	52281	1308	336	7	29	2	51696	1031
333	7	28	2	54007	1152	337	7	29	3	56507	971
334	7	28	3	57630	1100	338	7	30	1	6029	174

References

Barcia, P., and Jörnsten, K. (1990), "Improved Lagrangean decomposition: An application to the Generalized Assignment Problem", *European Journal of Operational Research* 46, 84–92.

Crainic, T.G., Gendreay, M., Soriano, P., and Toulouse, M. (1991), "A Tabu Search procedure for multicommodity location/allocation with balancing requirements", Publication No. 797, Centre de Recherche sur les transports, Université de Montreal.

Fisher, M., Jaikumar, R., and Van Wassenhove, L. (1986), "A multiplier adjustment method for the Generalized Assignment Problem", *Management Science* 32, 1095–1103.

Freville, A., and Plateau, G. (1991), "An efficient preprocessing procedure for the solution of 0–1 Multiknapsack Problems to optimality", *Mathematical Programming* 31, 78–105.

Glover, F. (1977), "Heuristic for Integer Programming using surrogate constraints", *Decision Sciences* 8, 156–166.

Glover, F. (1991), "Multilevel tabu search and embedded search neighborhoods for the Traveling Salesman Problem", to appear in *ORSA Journal on Computing*.

Glover, F., Hultz, J., and Klingman, D. (1979), "Improved computer-based planning techniques – Part II", *Interfaces* 9/4, 12–20.

Glover, F., Klingman, D., and Phillips, N. (1990), "Netform modeling and applications", *Interfaces* 20/4, 7–27.

Glover, F., and Laguna, M. (1993), "Bandwidth packing: A Tabu Search approach", *Management Science* 39/4, 492–500.

Guignard, M., and Rosenwein, M.B. (1989), "An improved dual based algorithm for the Generalized Assignment Problem", *Operations Research* 37/4, 658–663.

Jörnsten, K.O., and Näsberg, M. (1986), "A new Lagrangean relaxation approach to the Generalized Assignment Problem", *European Journal of Operational Research* 27, 313–323.

Kelly, J.P., Golden, B.L., and Assad, A.A. (1993), "Large-scale controlled rounding using Tabu Search with strategic oscillation", *Annals of Operations Research* 41, 69–84.

Klastorin, T.D. (1979), "An effective sub-gradient algorithm for the Generalized Assignment Problem", *Computers & Operations Research* 6, 155–164.

Martello, S., and Toth, P. (1981), "An algorithm for the Generalized Assignment Problem", in: *Proceedings of the 9-th IFORS Conference*, Hamburg, Germany.

Martello, S., and Toth, P. (1990), *Knapsack Problems: Algorithms and Computer Implementations*, Wiley, New York.

Osman, I.H. (1991), "Metastrategy Simulated Annealing and Tabu Search for combinatorial optimization", The Management School, Imperial College, London, UK.

Ross, G.T., and Soland, R.M. (1975), "A branch and bound algorithm for the Generalized Assignment Problem", *Mathematical Programming* 8, 91–103.