A tabu search approach to the generalized assignment problem

Article *in* Journal of the Chinese Institute of Industrial Engineers · May 2004

DOI: 10.1080/10170660409509411

3 authors:

Tai-Hsi Wu
National Taipei University
61 PUBLICATIONS **1,579** CITATIONS
SEE PROFILE

Jinn-Yi Yeh
National Chiayi University
33 PUBLICATIONS **871** CITATIONS
SEE PROFILE

Yu-Ru Syau
National Formosa University
60 PUBLICATIONS **563** CITATIONS
SEE PROFILE

A TABU SEARCH APPROACH TO THE GENERALIZED ASSIGNMENT PROBLEM

Tai-Hsi Wu*, Jinn-Yi Yeh, and Yu-Ru Syau
Department of Industrial Engineering
Da-Yeh University
112 Shan-Jiau Rd., Da-Tsuen, Changhwa 515

ABSTRACT

The generalized assignment problem (GAP) determines the maximum profit or minimum cost assignment of *n* jobs to *m* agents, which is a problem embedded in the cell formation problem. In this paper, a tabu search heuristic, TSDL that consists of dynamic tabu tenure with long-term memory mechanism is presented to solve the GAPs. A standard set of 84 test problems adopted from the literature is used to evaluate the performance of the proposed algorithm and for comparison with other existing methods. The TSDL can very efficiently find solutions with good quality. The proposed algorithm should thus be useful to practitioners and researchers.

Keywords: assignment, generalized assignment problem, tabu search, cell formation problem.

1. INTRODUCTION

The generalized assignment problem (GAP) finds the maximum profit or minimum cost assignment of *n* jobs to *m* agents such that each job is assigned to exactly one agent and the capacity of each agent cannot be exceeded. The GAP can be formulated as:

(GAP)

$$Max \quad \sum_i \sum_j p_{ij} x_{ij} \tag{1}$$

s.t.

$$\sum_i x_{ij} = 1, \qquad \forall j \in J = \{1,...,n\} \tag{2}$$

$$\sum_j w_{ij} x_{ij} \le c_i, \quad \forall i \in I = \{1,...,m\} \tag{3}$$

$$x_{ij} \in \{0,1\}, \qquad \forall i \in I = \{1,...,m\}, \quad \forall j \in J = \{1,...,n\} \tag{4}$$

where i = index of agent, j = index of job; x_{ij} =1 if job j is processed by agent i, or 0 otherwise; p_{ij} = profit of assigning job j to agent i; w_{ij} = capacity assumption of job j processed by agent i; c_i = capacity of agent i. Equation (2) ensures that each job can be processed by only one agent. The capacity constraint for each agent is expressed in equation (3). Equation (4) is for the binary requirements on the decision variables.

GAP has wide applications, such as vehicle routing problems [3], capacitated warehouse location problems, design of communication networks, resource scheduling, and assignment of jobs to computers in a network [15]. Although GAP frequently appears in real life, it is an NP-hard combinatorial optimization problem [11], which has received considerable research in the past. Several algorithms that can effectively solve the GAP have been cited and compared as benchmarks many times in the literature. These algorithms are from Ross and Soland [25], Fisher et al. [11], Martello and Toth [20], Cattrysse et al. [6,7], Osman [23], Amini and Racer [1], Lorena and Narciso [19], Chu and Beasley [10], Narciso and Lorena [22]. Laguna et al. [18] presented a tabu search approach for solving the multilevel GAP, in which the agents are assumed to perform tasks at more than one efficiency level. Yagiura et al. [28] proposed a variable depth search (VDS) algorithm for solving the GAP. The main idea of VDS is to change the size of the neighborhood adaptively so that the algorithm can effectively traverse larger search space within reasonable computational time. Several branching rules are later incorporated to further improve the effectiveness of the VDS. In addition, Cattrysse and Van Wassenhove [8] have provided an extensive review of the methods for solving the GAP in exact or heuristic ways.

* Corresponding author: taiwu@mail.dyu.edu.tw

GAP is actually a problem embedded in the cell formation problem, one of the most important steps in cellular manufacturing. Cell formation [9, 26] acts to group parts with similar design features or processing requirements into families. Machine cells are then formed by assigning machines to manufacturing cells in order to minimize the intercellular moves or maximize the parts flow within cells. From the published results in the literature, it is known that exact methods and certain heuristic algorithms perform well only in small-sized GAPs. When the problem size increases and/or the problem becomes highly capacitated, it becomes difficult to obtain the optimal solution in a reasonable amount of time. To overcome the computational problem, this study proposes a heuristic method, TSDL, for solving the GAP optimally and efficiently. TSDL is based on the tabu search (TS) methodology, and incorporates the features of dynamic tabu tenure (tabu list size) and long-term memory function.

Tabu search methods have been applied for finding optimal solutions for many combinatorial optimization problems, and the reader is referred to [12, 13, 14] for a more thorough list of applications. For ease of implementation, most of the tabu-based algorithms adopt fixed-size tabu tenure as the primary design for short-term memory. To avoid being trapped at a local optimum, jump-out strategies are often developed for guiding the search to obtain a limited level of diversified solutions, thus increasing the probability of finding optimal or near optimal solutions. Recent studies [4, 24, 27] have indicated that both the diversification and intensification strategies must be used alternately and properly in the algorithm to fully explore the power of the TS method. Lately Yagiura et al. [29] proposed a tabu search algorithm for the GAP. This algorithm features an ejection chain approach, which is embedded in a neighborhood construction to create more complex and powerful moves. Computational comparisons on benchmark test instances indicate that the proposed algorithm obtain many best-known solutions among all heuristics tested especially for problems of larger sizes.

Lagrangean relaxation heuristics have also been employed for solving the GAP [19, 22]. Usually the capacity constraint of the GAP is relaxed in the formulation, and the subgradient search method [2, 16, 17] is used to find a good set of multipliers. Having found the multipliers, the Lagrangean program is solved. However, at each iteration, the solution of the Lagrangean program may not be feasible, so it may be necessary to adjust such solution into a feasible solution for the original GAP.

Job	j1	j2	j3	j4	j5	j6	j7	j8	j9	j10
Agent	a3	a1	a2	a4	a1	a2	a3	a1	a4	a5

Figure 1. Configuration of a feasible solution of GAP

2. TABU SEARCH HEURISTIC

Tabu search (TS) employs flexible memory structures to store some information and attributes of solutions from the recent history of the search. TS gives some solutions (moves) recently or frequently visited a tabu restriction, to avoid the solution search process being trapped at a local optimum, unable to explore other regions of solution space. Traditional TS usually uses short-term memory to characterize certain moves as tabu for some iterations (tabu tenure, tabu list size) to avoid repetition of previously visited solutions. Several strategies such as the diversification and intensification of the solution searching and the adaptive tabu tenure have been utilized in designing the TS scheme, successfully exploiting the power of the TS method. In this section we describe the components of the proposed TS algorithm, TSDL.

2.1 Configuration

An easy way to represent a configuration of a feasible solution of the GAP is a string, whose size equals the number of jobs, as shown in Figure 1. Here, jobs 2, 5, and 8 are assigned to agent 1, while agent 5 processes only job 10. In such a configuration, the *jth* bit of the string stores the identifier of the agent to whom the job is assigned. Hence, the string corresponding to the example configuration in Figure 1 is (3, 1, 2, 4, 1, 2, 3, 1, 4, 5). Thus the problem can be viewed as a search for the best combinations of agent numbers, which results in the minimum assignment cost or maximum assignment profit.

2.2 Initial solution

Tabu search improves solution quality iteratively. An initial solution must be generated first before the solution improving process continues. The MTH algorithm by Martello and Toth [21] is one of the most widely used methods for finding a good feasible solution for the GAP. MTH uses f_{ij} to measure the "desirability" of assigning job j to agent i. In the first part of the algorithm it iteratively considers all jobs which have not been assigned, determining the job j' which has the maximum difference between the largest and the second largest f_{ij}. Then j' is assigned to the agent for which $f_{ij'}$ is a maximum. This step is repeated until all jobs have been assigned. In the second part of the algorithm the current solution is improved through local exchanges. In MTH, this improved step is executed only once, however, Lorena and Narciso [19] proposed a modified version of MTH, MMTH, to be included in the Lagrangean relaxation heuristic algorithm they proposed. In MMTH, the

improved step is executed iteratively until a better solution remains unchanged. Because Lorena and Narciso claimed that the results obtained by MMTH for the test problems are 5% on the average better than those by MTH, we hence adopted the MMTH for generating the initial feasible solution in this study. Martello and Toth [21] suggested four ways of defining f_{ij} : (1) $f_{ij} = p_{ij}$; (2) $f_{ij} = p_{ij} / w_{ij}$; (3) $f_{ij} = -w_{ij}$; (4) $f_{ij} = -w_{ij} / c_i$. In contrast, Lorena and Narciso [19] set $f_{ij} = p_{ij} /(\lambda_i w_{ij})$, where λ_i is the Lagrangean multiplier corresponding to the *ith* agent capacity limit constraint. For this study, we present the 6th setting for the f_{ij}: $f_{ij} = p_{ij}-w_{ij}$. The results from empirical testing of test problems show that applying the new setting of the f_{ij} leads to better solution quality than the four settings in [21].

2.3 Move and tabu restrictions

In TS methods, each iteration of the search process focuses on finding a good neighborhood solution with better quality than the current solution. The neighborhood of a given solution is defined as a set of all feasible solutions that can be reached by a single move/transition. In this study, three types of moves are implemented in the search procedure: (1) a single-move, (2) an exchange-move, and (3) a double-move. The single-move is an operation that moves a job *j* from its current agent *i* (source agent) to a new agent *i*' (destination agent). The new move made is denoted (*i*', *j*). The exchange-move is an operation that consists of two dependent single-moves. If a job *j* is moved from its source agent *i* to destination agent *i*' (first single-move), then one job *j*' ($j' \in j_{i'} = \{$ jobs processed by agent i' $\}$) from the destination agent *i*' of the first move has to be moved to the source agent *i* of the first move (second single-move) in exchange. Thus two moves, (*i*', *j*) and (*i*, *j*'), are generated. The double-move differs from the exchange-move only on the operation of the second single-move. Here, when a first move is made, one job *j*', *j*' $j' \in j_{i'}$, from the destination agent *i*' of the first move can be moved to any other agent except the source agent (called it *iC*) of the first move. Thus two moves, (*i*', *j*) and (*iC*, *j*'), are also generated here. The rule for choosing moves is described below. For the single-move, a move that results in the most improvement in the objective function value from the current solution is selected. That is,

$$M_1(i1', j1) =$$
$$\max\{obj^{(i',j)} - obj^{current}, \forall i' \in I, i' \neq i, \forall j \in J\}$$

where *I* and *J* are the sets for agents and jobs, respectively. For the exchange-move, the pair of moves selected is that which results in the most improvement in the objective function value from the current solution. That is,

$$M_2\{(i2', j2),(i2, j2')\} =$$
$$\max\{obj^{(i',j),(i,j')} - obj^{current}, \forall i' \in I, i' \neq i, \forall j \in J, \forall j' \in J_{i'}\}$$

For the double-move, the pair of moves selected is that which results in the most improvement in the objective function value from the current solution.

$$M_3\{(i3', j3),(i3^C, j3')\} =$$
$$max\{obj^{(i',j)(i^C,j')} - obj^{(i,j)}, \forall i' \in I, i' \neq i, \forall j \in J, \forall j' \in J_{i'}\}$$

where i^C represents the complement of agent *i*. Each of the three types of moves can be characterized as a pair / two pairs of identifiers, (agent identifier, job identifier).

Having obtained the best solution for each type of move, the best neighborhood solution of the current solution is selected by picking the best from the above three solutions, provided that it is not a tabu move (as explained in the next section). That is, *best_move* =

$$max\{ M_1(i1', j1), M_2\{(i2', j2),(i2, j2')\},$$
$$M_3\{(i3', j3),(i3^C, j3')\}\}$$

The attributes of certain moves, which were recently or frequently visited, are stored in a data structure called a tabu list to prevent moving back to these moves and avoid being trapped at a local optimum or causing cycling. Suppose the number of elements in the tabu list is fixed and denoted as N^T, each element in the tabu list stores the information of agent and job identifiers of the move just selected. Any move having the (agent, job) identifiers appearing in the tabu list is classified as a tabu move. The tabu list is updated with each move in a search procedure. For an element in the tabu list to be released from the tabu status, there is a tabu tenure (measured in terms of iterations) that must elapse. This tenure is actually the size of the tabu list, N^T.

2.4 Aspiration criteria

The aspiration criteria in a tabu search provide a mechanism for releasing the tabu status for a move which would have led to better solutions. In this study, we use the typical criteria stating that if a move produces a feasible solution better than the best-known solution, then the tabu status is disregarded and the move is accepted.

2.5 Intensification and diversification strategies

The TS method usually employs the short-term memory function to store attributes of certain recently visited moves and give them a tabu status. In addition, two strategies are used alternately and periodically to search more efficiently for good solutions: intensification (intermediate memory) and diversification (long-term memory) functions. Intensification strategy first finds a region with

potentially prosperous solutions, and the search process focuses on this region locally. Diversification is used to escape from local optima and is achieved by invoking the long-term memory function. In our searching procedure, if no better solution is found after a certain number of moves, the intensification strategy is implemented by modifying the measure of attractiveness of a move $M(i, j)$ to $M'(i, j)$.

$$M'(i,j) = M(i,j) + para1 \times \frac{\begin{array}{c}cumulative\ improvement\ to\\ the\ objective\ value\ of\ move\ (i,j)\end{array}}{\begin{array}{c}number\ of\ times\ move\ (i,j)\\ was\ selected\ since\ starting\end{array}} \quad (5)$$

where *para*1 is a weight parameter to control this strategy. From the above formulation, it can be observed that moves with good performance over the past iterations become more attractive, and the search is intensified among these moves. This information on attractiveness of moves are stored in a data structure with dimension $m \times n$, similar to the tabu list, which is referred to the "intermediate memory".

The diversification strategy guides the search toward solution regions not yet explored. When repetitions keep occurring on certain number of moves, the diversification strategy is implemented by modifying the measure of attractiveness of a move, in a similar manner to the intensification strategy.

$$M'(i,j) = M(i,j) - para2 \times \begin{array}{c}number\ of\ times\ the\ move\\ was\ selected\ since\ starting\end{array} \quad (6)$$

where *para*2 is a weight parameter to control the strategy. From the above formulation, it can be observed that moves frequently visited become less attractive. Therefore moves that appeared infrequently over the past iterations are now more likely to be selected. This strategy hence guides the searching to unexplored solution space. Information on attractiveness of moves at this stage is stored in a data structure similar to the intermediate memory and tabu list, and is referred to the "long-term memory".

2.6 Adaptive control of tabu tenure

Setting the size of tabu list (tabu tenure) is one of the most important tasks in designing the tabu scheme. Smaller values of tabu tenure may still cause cycling, while larger values may forbid some good moves and thus slow down the convergence of the algorithm. In addition to a fixed value, the tabu tenure may also be dynamic or random. In this study, the tabu tenures of the proposed TS algorithm are adjusted automatically based on a function of solution process attributes. A simple mechanism for adapting the tabu tenure is proposed. The tabu tenure was initially set to 1. The moves visited during the search and the corresponding iteration numbers are stored in records so that, after the last move is selected, one can check for the repetition of moves and calculate the *Interval* between two visits. The information of *Interval* for each move is also stored in records. Tabu tenure is increased by a ratio, *incr_ratio*, when repetition of moves is discovered; and decreased by a ratio, *decr_ratio*, when no repetition was found during a certain number of iterations (*No_Rep*). Regardless of how the tabu tenure decreases or increases, the upper and lower limits for the tabu tenure are set at 15 and 1, respectively. Note that there may be cases, especially in problems with small sizes, in which all feasible moves are given a tabu status. In that case, the tabu tenure would then be decreased by the ratio, *decr_ratio*.

2.7 Tabu search heuristic procedure TSDL

Before describing the complete TS heuristic algorithm, TSDL, we first explain how the algorithm judges the timing for using the diversification and intensification strategies. For the intensification strategy, when no better solution was obtained in *No_Impr* iterations, this strategy is invoked. However, things are somewhat more complicated for the diversification strategy. If a move appears again after a very large number of iterations, this condition should not be treated as repetition. A number representing this big time interval is denoted *Max_Interval*. Any move with an *Interval* greater than the *Max_Interval* would not be considered as a repetition. When the number of repetitions for a move is greater than the *Rep_Max*, this move is added to the set of often-repeated ones, and the counter *Chaotic* is increased by 1. When *Chaotic* is greater than *Chaos*, it implies that repetitions of solutions keep occurring, and the diversification strategy has to be invoked to escape from the local solution region. The complete TSDL procedure is given below.

Procedure TSDL

Step 1. Initialization.

Step 2. Generate a feasible solution through MMTH algorithm with new setting for f_{ij} in Section 2.2.

Step 3. Judge timing for invoking intensification strategy. If no better solution was obtained in *No_Impr* iterations, go to Step 5.

Step 4. Use short-term memory to find the next move. Go to Step 6.

Step 5. Use intermediate memory to find next move based on modified equation (5).

Step 6. If all neighborhood solutions are in tabu status, reduce tabu tenure by ratio, *decr_ratio*, and go to Step 3; otherwise accept neighborhood solution found.

Step 7. Update tabu list and memory structure.

Step 8. If the solution found is better than current best solution, make the replacement.
Step 9. Update tabu tenure, if necessary.
Step 10. Check timing for invoking diversification strategy. If yes, use long-term memory and run for 30 iterations to determine the next move, based on modified equation (6); otherwise go to Step 13.
Step 11. If solution found is better than the current best solution, make the replacement.
Step 12. Clear the tabu list, intermediate and long-term memory.
Step13. If the iteration limit is exceeded, stop the run; otherwise, go to Step 3.

Step 6 is added mainly for solving small-sized problems in which the number of neighborhood solutions is very limited. It is, hence, very possible that all neighborhood solution are in tabu status. For problems with larger sizes, Step 6 may be ignored. In Step 10, instead of running the program using long-term memory function for just one iteration, 30 iterations are implemented to increase the probability of finding more "diversified" solutions. In Step 12, the tabu list, intermediate and long-term memories are cleared every time after a diversification strategy is performed to bring the searching process to a new stage. This derives from our empirical testing experience, which indicates that clearing this information usually produces better results.

In this study, only one set of values for the parameters for the computational experiment is used. The settings of parameters used in the proposed procedure TSDL are determined based on the experience of intensive testing. Solutions with better quality may be obtained if these values for the parameters are tailored to each problem of different sizes and types. The values of parameters are given below: *Max_Interval*: 100, *Rep_Max*: 5, *Chaos*: 5, *No_Impr*: 100, *No_Rep*:10, *incr_ratio*: 0.1, *decr_ratio*: 0.1, *para*1: 0.4 or 0.5, *para*2: 0.4 or 0.5.

3. COMPUTATIONAL RESULTS

The TSDL algorithm presented in this study was coded in C and run on a personal computer (Pentium III 667, 256 Mb RAM). TSDL was tested on 84 test problems adopted from the literature, ranging from 5 agents / 15 jobs to 20 agents / 200 jobs. All of these test problems are available on **http://mscmga.ms.ic.ac.uk/jeb/orlib/gapinfo.html** [5].

Of the 84 problems used, 60 problems are categorized as "small-sized" problems, and have been used by Cattrysse et al. [7], Osman [23], and Chu and Beasley [10]. These 60 problems have the following characteristics: $m \in \{5, 8, 10\}$ and the ratio $(n/m) \in \{3, 4, 5, 6\}$. For each of the 12 possible combinations, 5 test sets were generated, yielding 60 problems in total. The data were all drawn from a discrete uniform distribution with p_{ij} ~U(15, 25), w_{ij} ~U(5, 25), and $c_i \sim (0.8/m)\sum_{j=1}^{n} w_{ij}$.

The remaining 24 problems are categorized as "large-sized" problems. These problems were generated by Chu and Beasley [10] as a standard set of test problems, mainly for the convenience of the research community. In these 24 large-scaled instances, $m \in \{5, 10, 20\}$ and $n \in \{100, 200\}$. They were classified into four types: A, B, C and D. The settings for each type of test problems are as follows[10]:

Type A: w_{ij} are integers from U(5, 25), p_{ij} are integers from U(10, 50), and $c_i = 0.6(n/m)15+0.4R$, where $R = max_{i \in I} \sum_{j \in J, I_j = i} w_{ij}$, and $I_j = min\left[i \middle| p_{ij} \le p_{kj}, \forall k \in I\right]$.

Type B: w_{ij} and p_{ij} are the same as Type A, and c_i is set to 70% of the value given in Type A.

Type C: w_{ij} and p_{ij} are the same as Type A, and $c_i = 0.8 \sum_{j \in J} w_{ij} / m$.

Type D: w_{ij} are integers from U(1, 100), $p_{ij} = 111 - w_{ij} + e$, where e are integers from U(-10, 10) and $c_i = 0.8 \sum_{j \in J} w_{ij} / m$.

For each problem type, one problem is generated for each agent / job combination, giving a total of 24 problems.

Optimal solutions for the 60 small sized problems are known from Cattrysse [7], and Chu and Beasley [10], obtained through a branch-and-bound procedure with run times ranging from a few seconds to a few hours. These optimal solutions were obtained when the GAPs were solved in maximization form. For the large-scale problems, we use CPLEX (MIP solver) to solve them. During the process, optimal solutions were obtained for 15 problems out of 24, and the process of the remaining 9 problems was terminated by the computer due to running out of memory. In most of the test instances, the optimal/best solutions of the TSDL algorithm were found on a smaller number of iterations than the iterations limit for each run.

3.1 Computational results of TSDL for small-sized problems

To test the effects of adopting both dynamic tabu tenure and long-term memory mechanism, the computational results of the proposed TSDL are compared with those from a traditional tabu search method, TTS.

Table 1. Computational results for small-sized problems using TTS and TSDL

Problem	No. of agent	No. of job	Optimal	**TTS** algorithm		**TSDL** algorithm	
				Best sol'n	Exec. Time (sec)	Best sol'n	Exec. Time (sec)
Gap1-1	5	15	336	**336** *	0.112	**336** *	0.342
Gap1-2			327	**327** *	0.181	**327** *	0.332
Gap1-3			339	**339** *	0.292	**339** *	0.346
Gap1-4			341	**341** *	0.221	**341** *	0.347
Gap1-5			326	**326** *	0.183	**326** *	0.370
Gap2-1	5	20	434	**434** *	0.579	**434** *	0.426
Gap2-2			436	**436** *	0.606	**436** *	0.425
Gap2-3			420	**420** *	0.550	**420** *	0.447
Gap2-4			419	**419** *	0.570	**419** *	0.452
Gap2-5			428	**428** *	0.670	**428** *	0.435
Gap3-1	5	25	580	579	0.693	**580** *	0.495
Gap3-2			564	**564** *	0.569	**564** *	0.509
Gap3-3			573	**573** *	0.721	**573** *	0.503
Gap3-4			570	**570** *	0.494	**570** *	0.510
Gap3-5			564	**564** *	0.683	**564** *	0.542
Gap4-1	5	30	656	**656** *	0.736	**656** *	0.648
Gap4-2			644	643	0.781	**644** *	0.754
Gap4-3			673	671	0.747	**673** *	0.610
Gap4-4			647	646	0.792	**647** *	0.587
Gap4-5			664	**664** *	0.741	**664** *	0.640
Gap5-1	8	24	563	**563** *	0.744	**563** *	0.639
Gap5-2			558	**558** *	0.776	**558** *	0.585
Gap5-3			564	**564** *	0.747	**564** *	0.632
Gap5-4			568	**568** *	0.756	**568** *	0.649
Gap5-5			559	558	0.670	**559** *	0.619
Gap6-1	8	32	761	**761** *	0.976	**761** *	0.985
Gap6-2			759	**759** *	0.774	**759** *	0.881
Gap6-3			758	757	0.923	**758** *	0.910
Gap6-4			752	**752** *	0.934	**752** *	0.897
Gap6-5			747	746	0.935	**747** *	0.907
Gap7-1	8	40	942	**942** *	1.130	**942** *	1.323
Gap7-2			949	**949** *	1.156	**949** *	1.245
Gap7-3			968	**968** *	1.180	**968** *	1.205
Gap7-4			945	**945** *	1.056	**945** *	1.138
Gap7-5			951	**951** *	1.164	**951** *	1.232
Gap8-1	8	48	1133	1132	1.317	**1133** *	1.601
Gap8-2			1134	1133	1.320	**1134** *	1.439
Gap8-3			1141	1140	1.310	**1141** *	1.527
Gap8-4			1117	1116	1.379	**1117** *	1.660
Gap8-5			1127	1126	1.340	**1127** *	1.667
Gap9-1	10	30	709	**709** *	0.924	**709** *	0.939
Gap9-2			717	716	0.964	**717** *	1.312
Gap9-3			712	**712** *	0.975	**712** *	0.944
Gap9-4			723	**723** *	1.015	**723** *	0.914
Gap9-5			706	704	0.817	**706** *	0.880
Gap10-1	10	40	958	**958** *	1.328	**958** *	1.386
Gap10-2			963	**963** *	1.308	**963** *	1.379
Gap10-3			960	956	1.297	**960** *	1.389
Gap10-4			947	**947** *	1.378	**947** *	1.488
Gap10-5			947	946	1.270	**947** *	1.463
Gap11-1	10	50	1139	1136	1.815	**1139** *	2.070
Gap11-2			1178	1177	1.762	**1178** *	2.068
Gap11-3			1195	**1195** *	2.147	**1195** *	3.053
Gap11-4			1171	**1171** *	1.804	**1171** *	2.221
Gap11-5			1171	**1171** *	1.621	**1171** *	2.029
Gap12-1	10	60	1451	**1451** *	2.165	**1451** *	2.724
Gap12-2			1449	**1449** *	2.220	**1449** *	2.830
Gap12-3			1433	**1433** *	2.134	**1433** *	2.589
Gap12-4			1447	**1447** *	2.331	**1447** *	3.635
Gap12-5			1446	**1446** *	2.260	**1446** *	3.867

*：optimal solution value

Except for the employment of only short-term memory and a tabu tenure fixed at 7, TTS has the same structure as the TSDL. Table 1 presents computational results of TTS and TSDL algorithms for the 60 small-sized problems, obtained with a limit of 1500 iterations.

For each problem, the known optimal solutions, the values of the best solution for each algorithm, and the corresponding run times are reported.

Examining Table 1, we observe that the TSDL performs very well and finds the optimal solution for all 60 problems, while the TTS finds the optimum in 42 out of 60 problems. The run times for both approaches are very short. The longest run in 60 problems only takes the TSDL 3.867 seconds, while the TTS spends only 2.331 seconds to finish the longest run. The effects of employing dynamic tabu tenure and long-term memory function are confirmed by the better solution quality of the TSDL than the TTS.

To compare the proposed algorithm with existing methods, we adopt the table summarizing the computational results appeared in [10, 23] and present Table 2 in this paper. Table 2 lists the computational results of the TSDL algorithm, and 11 other existing heuristic algorithms. Ten trials were implemented for each problem in Chu and Beasley [10], with the best and average solution values reported. Thus the average percentage deviation (δ) from optimal for each problem was defined as $\delta = \sum_{i=1}^{10} \frac{S_0 - S_i}{10 S_0}$, where S_i is the best solution value of the i-th trial and S_0 is the optimal solution value. In addition to the average percentage deviation (δ) from optimal for each problem, the average percentage deviation for all problems (δ_{all}) (Gap1 to Gap12), the average percentage deviation of the best solution (δ_{best}), and the number of optimal solutions obtained in 60 problems are reported as well.

Unlike methods such as the GA algorithm in Chu and Beasley [10], in which the average and the best results from 10 replicates are calculated, the proposed TSDL algorithm obtains results in a single execution. This explains why in Table 2 the δ_{all} equals to the δ_{best} in the column headed "TSDL". It can be observed that the proposed TSDL heuristic performs the best among all heuristics in terms of the solution quality, being capable of finding the optimal solutions for all 60 problems. Although the GA heuristic also found all the optimums, however, it performs slightly worse than the TSDL on the average percentage deviation for all problems. The computational times for other existing algorithms are not given in the tables below, since it does not make much sense to compare run times obtained from different computers.

3.2 Computational results of TSDL for large-sized problems

Chu and Beasley [10] and Yagiura et al. [29] successively provide benchmarks for the large-sized GAPs with minimization objectives. To compare with their results, we changed our GAP formulation with maximization objective to be a minimization problem by changing the profit data in the objective function to be negative. Table 3 summarizes the results of the CPLEX, the proposed TSDL, the GA heuristic [10] and the TS algorithm [29]. Chu and Beasley [10] provides the results of problem types A to D, while Yagiura et al. [29] aiming at harder problem types, and hence ignoring problem types A and B. Since problems of Type A are easier to solve, a limit of 300 iterations was set for the implementation of the TSDL. For problem types B, C, and D, the iteration limit for each run was 10000. Note that in Table 3, the best overall solutions of the GA were obtained from 10 replicates of the GA heuristic, the best overall solutions of the TS were obtained from 5 replicates of the TS algorithm for each problem, while TSDL obtained the best solutions in a single execution. Also note that the TS algorithm were implemented on Sun Ultra2 Model 2300 with time limits 150 seconds for n = 100, and 300 seconds for n = 200.

Table 2. Average percentage deviation (δ) from optimal of existing algorithms for small-sized problems

Problem set	MTH	FJVBB	MTBB	SPH	LTIFA	RSSA	TS6	TS1	GA	CDT	**TSDL**
Gap1	5.43	0.00	0.00	0.08	1.74	0.00	0.00	0.00	0.00	0.00	0.00
Gap2	5.02	0.00	0.00	0.11	0.89	0.00	0.24	0.10	0.00	0.05	0.00
Gap3	2.14	0.00	0.00	0.09	1.26	0.00	0.03	0.00	0.00	0.00	0.00
Gap4	2.35	0.83	0.18	0.04	0.72	0.00	0.03	0.03	0.00	0.06	0.00
Gap5	2.63	0.07	0.00	0.35	1.42	0.00	0.04	0.00	0.00	0.04	0.00
Gap6	1.67	0.58	0.52	0.15	0.82	0.05	0.00	0.03	0.01	0.00	0.00
Gap7	2.02	1.58	1.32	0.00	1.22	0.02	0.02	0.00	0.00	0.17	0.00
Gap8	2.45	2.48	1.32	0.23	1.13	0.10	0.14	0.09	0.05	0.19	0.00
Gap9	2.18	0.61	1.06	0.12	1.48	0.08	0.06	0.06	0.00	0.03	0.00
Gap10	1.75	1.29	1.15	0.25	1.19	0.14	0.15	0.08	0.04	0.17	0.00
Gap11	1.78	1.32	2.01	0.00	1.17	0.05	0.02	0.02	0.00	0.00	0.00
Gap12	1.37	1.37	1.55	0.10	0.81	0.11	0.07	0.04	0.01	0.01	0.00
Avg % δ_{all}	n/a	n/a	n/a	n/a	1.15	0.21	0.10	0.07	0.01	0.06	0.00
Avg % δ_{best}	2.56	0.84	0.78	0.13	1.15	0.04	0.06	0.03	0.00	n/a	0.00
# optimal	0	26	24	40	3	39	40	45	60	42	60

Examining Table 3, we can observe that for the 15 problems with optimal solutions generated by the CPLEX before the computer ran out of memory and terminated, the TSDL was capable of finding them all, while the GA found only 10. Considering all problems, the TSDL obtains the best solutions (which may or may not be optimal) in 16 out of 24 instances, while the GA found 10. All the runs finished in less than 350 seconds for the TSDL. If we focus on the results of test problems of types C and D, the TS algorithm of Yagiura et al. [29] obtains the best solutions in all the 12 problems tested. In problem type C, the TSDL performed almost as good as the TS algorithm. Dominance of the TS algorithm in problem type D reveals its superiority. Solutions of problem types C and D obtained by the TSDL deviate from best solutions of the TS algorithm [29] by 0.55%.

Narciso and Lorena [22] also ran these 24 large-scaled problems as maximization problems. The computational results of the TSDL are reported and listed in Table 4 in comparison with those from the RH algorithm of Narciso and Lorena [22]. They obtained the optimal solutions (signed by (*)) by running the RH algorithm without the limit of iterations, searching the optimality condition (ub-lb) < 1, where ub and lb are the upper and lower bounds, respectively.

It can be seen from Table 4 that both the RH and TSDL algorithms found 15 optimal solutions. However, among the 9 problems with unknown optimums, in 8 instances the TSDL resulted in solutions better than or equal to the RH algorithm. That is, TSDL improves best-known solutions in all but one case (5×200 in problem type C). In comparison with the running without an iteration limit to obtain the optimal solutions for the RH, TSDL solves all instances more effectively and efficiently than the RH algorithm. This again confirms the superior performance of the proposed TSDL algorithm.

4. CONCLUDING REMARKS

A tabu structure consisting of dynamic tabu tenure with a long-term memory mechanism, TSDL, has been presented in this study. Due to the alternate use of diversification and intensification strategies in

Table 3. Computational results for large-sized problems (minimization objectives)

Type	Problem size	CPLEX			TSDL			GA[10]		TS[29]	
		Optimal sol'n		Exec. Time (sec)	Best sol'n		Exec. Time (sec)	Best overall sol'n		Best overall sol'n	
A	5×100	1698		0.49	**1698**	*	1.852	**1698**	*	-	-
	5×200	3235		0.34	**3235**	*	5.273	**3235**	*	-	-
	10×100	1360		0.33	**1360**	*	3.801	**1360**	*	-	-
	10×200	2623		0.38	**2623**	*	18.274	**2623**	*	-	-
	20×100	1158		0.50	**1158**	*	6.362	**1158**	*	-	-
	20×200	2339		0.52	**2339**	*	30.233	**2339**	*	-	-
B	5×100	1843		291.74	**1843**	*	39.444	**1843**	*	-	-
	5×200	3552		6355.07	**3552**	*	81.326	3553		-	-
	10×100	1407		13.33	**1407**	*	51.256	**1407**	*	-	-
	10×200	3083	?	-	**2828**		142.204	2831		-	-
	20×100	1166		8.52	**1166**	*	83.408	**1166**	*	-	-
	20×200	2339		81.89	**2339**	*	347.846	2340		-	-
C	5×100	1931		20.45	**1931**	*	32.971	**1931**	*	**1931**	*
	5×200	3456		2114.45	**3456**	*	92.963	3458		**3456**	*
	10×100	1402		221.71	**1402**	*	52.824	1403		**1402**	*
	10×200	3057	?	-	2807		155.343	2814		**2806**	*
	20×100	1243		309.20	**1243**	*	70.155	1244		**1243**	*
	20×200	2512	?	-	2398		255.551	2397		**2392**	
D	5×100	6487	?	-	6371		45.451	6373		**6357**	
	5×200	12895	?	-	12784		121.102	12796		**12746**	
	10×100	6445	?	-	6410		67.653	6379		**6358**	
	10×200	12643	?	-	12649		188.333	12601		**12446**	
	20×100	6401	?	-	6310		69.361	6269		**6221**	
	20×200	12574	?	-	12534		242.548	12452		**12284**	

? = solution obtained before the run was terminated due to running out of memory
* = optimal solution confirmed on [29]

Table 4. Computational results for large-sized problems (maximization objectives)

Type	Problem set	RH[22] Best sol'n		TSDL Best sol'n		TSDL Exec. time
A	5×100	**4456**	*	**4456**	*	0.294
	5×200	**8788**	*	**8788**	*	1.515
	10×100	**4700**	*	**4700**	*	0.703
	10×200	**9413**	*	**9413**	*	2.965
	20×100	**4857**	*	**4857**	*	1.381
	20×200	**9666**	*	**9666**	*	6.160
B	5×100	4008		4026	#	21.193
	5×200	8502		8505	#	45.109
	10×100	**4633**	*	**4633**	*	26.560
	10×200	9255		9255	#	96.454
	20×100	**4817**	*	**4817**	*	54.167
	20×200	9670		9682	#	172.911
C	5×100	**4411**	*	**4411**	*	16.711
	5×200	8347		8346		62.357
	10×100	4528		4535	#	27.583
	10×200	9247		9258	#	119.564
	20×100	4784		4790	#	45.550
	20×200	9611		9625	#	164.062
D	5×100	**9147**	*	**9147**	*	0.001
	5×200	**18750**	*	**18750**	*	0.014
	10×100	**10349**	*	**10349**	*	0.005
	10×200	**20562**	*	**20562**	*	0.022
	20×100	**10839**	*	**10839**	*	0.006
	20×200	**21733**	*	**21733**	*	0.034

* = optimal solution value
\# = solution value of the TSDL better than or equal to Narciso and Lorena [22]

the TSDL, extremely good quality results have been obtained very efficiently for the 60 small-sized test problems, and comparable solutions have been obtained for the rest 24 large-sized test problems.

Further research may be performed using different tabu search strategies to improve the performance of the TS-based heuristic, especially the problem-solving efficiency. These include (1) different intermediate and long-term memory functions; (2) a faster escaping strategy from local optima; and (3) a more elite candidate list [14].

REFERENCES

1. Amini, M. M. and M. Racer, "A hybrid heuristic for the generalized assignment problem," *European Journal of Operational Research*, **87,** 343-348 (1995).
2. Baker, B. M. and J. Sheasby, "Accelerating the convergence of subgradient optimization," *European Journal of Operational Research*, **117,** 136-144 (1999).
3. Baker, M. and J. Sheasby, "Extensions to the generalized assignment heuristic for vehicle routing," *European Journal of Operational Research*, **119**, 147-157 (1999).
4. Battiti, R. and G. Tecchiolli, "The reactive tabu search," *ORSA Journal on Computing*, **6**, 126-140 (1994).
5. Beasley, J. E. "OR-Library: Distributing test problems by electronic mail," *Journal of Operational Research Society*, **41**, 1069-1072 (1990).
6. Cattrysse, D., Z. Degraeve, and J. Tistaert, "Solving the generalized assignment problem using polyhedral," *European Journal of Operational Research*, **108**, 618-628 (1998).
7. Cattrysse, D. M., L. N. Salomon, and Van Wassenhove, "A set partitioning heuristic for the generalized assignment problem," *European Journal of Operational Research*, **72**, 167-174 (1994).
8. Cattrysse, D. G. and L. N. Van Wassenhove, "A survey of algorithms for the generalized assignment problem," *European Journal of Operational Research*, **60**, 260-272 (1992).
9. Caux, C., R. Bruniaux, and H. Pierreval, "Cell formation with alternative process plans and machine capacity constraints: A new combined approach," *International Journal of Production Economics*, **64**, 279-284 (2000).
10. Chu, P. C. and J. E. Beasley, "A genetic algorithm for the generalised assignment problem," *Computers & Operations Research*, **24,** 17-23 (1997).
11. Fisher, M. L., R. Jaikumar, and L. N. Van Wassenhove, "A multiplier adjustment method for the generalized assignment problem," *Management Science*, **32**, 1095-1103 (1986).
12. Glover, F., "Tabu search – Part I," *ORSA Journal on Computing*, **1**, 190-206 (1989).
13. Glover, F., "Tabu search–Part II," *ORSA Journal on Computing*, **2**, 4-32 (1990).
14. Glover, F. and M. Laguna, *Tabu Search*, Kluwer Academic Publishers, Boston, (1997).
15. Grigoriadis, M. D., D. J. Tang, and L. S. Woo, "Considerations in the optimal synthesis of some communications networks," *paper presented at 45th joint National ORSA/TIMS Meeting*, Boston, MA, U. S. A. (1975).
16. Haddadi, S., "Simple Lagrangian heuristic for the set covering problem," *European Journal of Operational Research***, 97,** 200-204 (1997).
17. Held, M. and R. M. Karp, "The traveling-salesman problem and minimum spanning trees: Part II," *Mathematical Programming*, **1**, 6-25 (1971).
18. Laguna, M., J. P. Kelly, J. L. González-Velarde, and F. Glover, "Tabu search for the multilevel generalized assignment problem," *European Journal of Operational Research*, **82,** 176-189 (1995).
19. Lorena, L. A. N. and M. G. Narciso, "Relaxation heuristics for a generalized assignment problem," *European Journal of Operational Research*, **91**, 600-610 (1996).
20. Martello, S. and P. Toth, "An algorithm for the generalized assignment problem," *In Operations Research: Brans*, J.P. (ed.) North-Holland, Amsterdam, 589-603 (1981).
21. Martello, S. and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*, John Wiley and Sons, New York, 1990.

22. Narciso, M. G. and L. A. N. Lorena, "Lagrangean / surrogate relaxation for generalized assignment problems," *European Journal of Operational Research*, **114**, 165-177 (1999).
23. Osman, I. H., "Heuristics for the generalized assignment problem: Simulated annealing and tabu search approaches," *OR Spektrum*, **17**, 211-225 (1995).
24. Rolland, E., D. A. Schilling, and J. R. Current, "An efficient tabu search procedure for the p-Median problem," *European Journal of Operational Research*, **96**, 329-342 (1996).
25. Ross, G. T. and R. M. Soland, "A branch-and-bound algorithm for the generalized assignment problem," *Mathematical Programming*, **8,** 91-103 (1975).
26. Sun, D., L. Lin, and R. Batta, "Cell formation using tabu search," *Computers & Industrial Engineering*, **28**, 485-494 (1995).
27. Sun, M., J. E. Aronson, P. G. McKeown, and D. Drinka, "A tabu search heuristic procedure for the fixed charge transportation problem," *European Journal of Operational Research*, **106**, 441-456 (1998).
28. Yagiura, M., T. Yamaguchi, and T. Ibaraki, "A variable depth search algorithm with branching search for the generalized assignment problem," *Optimization Methods and Software*, **10**, 419-441 (1998).
29. Yagiura, M., T. Ibaraki, and F. Glover, "An ejection chain approach for the generalized assignment problem," *INFORMS Journal on Computing* (to appear)

ABOUT THE AUTHORS

Tai-His Wu is a professor in the department of Industrial Engineering at Da-Yeh University. He received his master and Ph.D. degrees in Industrial Engineering from Texas A&M University, USA, in 1994. His research and teaching interests include developing models and algorithms for production and transportation problems.

Jinn-Yi Yeh received the B.S. degree from Chung-Yuan Christian University, Taiwan, in 1987 and the M.S. and Ph.D. degrees from the University of Texas at Arlington in 1993 and 1997, respectively. He is presently an Assistant Professor in the Department of Industrial Engineering, Da-Yeh University, Taiwan. His research interests focus on semiconductor manufacturing quality control, medical image processing, and heuristic algorithms.

Yu-Ru Syau received the B.S. degree in Mathematics and the M.S. degree in Mathematics, both from the National Central University, Chung-Li, Taiwan, in 1987 and 1989, respectively, and the Ph.D. degree in Mathematics from the University of Illinois at Chicago, Illinois, U.S.A., in 1994. Currently, she is a professor in the department of Industrial Engineering at Da-Yeh University.

(Received July 2003; revised October 2003; accepted November2003)

以禁忌搜尋演算法求解一般化指派問題

吳泰熙* 葉進儀 蕭育如
大葉大學工業工程學系
515 彰化縣大村鄉山腳路 112 號

摘要

一般化指派問題尋求最大利潤或最小成本之工作指派計畫，其應用面非常廣泛，單元形成問題即是一例。本文提出一禁忌搜尋演算法TSDL來求解一般化指派問題，該演算法利用了動態禁忌名單及長期記憶體機制。為驗證本演算法之效用及效率，我們採用了84題文獻範例，並與現存文獻標竿演算結果做一比較後，發現TSDL可以於極短之演算時間內求得品質甚佳之演算結果。

關鍵詞：一般化指派問題，禁忌搜尋演算法，動態禁忌名單
(聯絡人: **taiwu@mail.dyu.edu.tw**)