Tabu Search with Reinforcement Learning for Location Problems

Kuidong Li,[1], Lydia Wallis,[2], and Rex K. Kincaid[3]

[1]Department of Mathematics, William & Mary
[2]Computer Science Department, William & Mary
[3]Department of Mathematics, William & Mary

August 15, 2025

1 Introduction

The similarities and differences between a tabu search heuristic coupled with unsupervised reinforcement (probabilistic) learning and a tabu search heuristic that incorporates a long-term memory function are explored. Computational experiments with two discrete location problems provide the test bed for the comparison. The first, the k-median problem, is a well studied location problem that seeks to determine the set of facility locations that minimize the average travel distance to the points served by the facilities. The second, the damper placement problem, is a novel location problem in which truss members of a flexible space structure are chosen so as to maximize the dampening effect on vibrational modes of interest.

2 Background Information

Tabu search is a high-level heuristic search strategy that attempts to effectively explore the search space of a discrete optimization problem by successively restraining (tabu moves) and freeing (aspiration criteria) of a local search ([2],[1]). Both discrete location problems studied here include a data matrix, D, which catalogs the effect of potential locations (column indices of D) on the *customers* (row indices of D) that are served by these locations. For the k-median problem, D is a shortest path distance matrix while for the damper placement problem, D is a modal strain energy matrix.

One of the earliest applications of tabu search, TS, that incorporates a long-term memory function, is found in [3] for the quadratic assignment problem. In [3], a constructive procedure uses D to generate an initial feasible solution. After a fixed number of iterations, TS is restarted using the same constructive procedure but with an altered D matrix. The distances in D increase for entries that were exchanged during the search. The long-term memory function implemented in the TS procedure for the k-median problem is quite similar to this approach.

Recently, several researchers, [6], [7], and [8], have successfully implemented tabu search heuristics coupled with a reinforcement learning mechanism to restart the search instead of a long-term memory function. There are a number of ways in which reinforcement learning attempts to guide a

search. Most incorporate a reward function when a location is added and a penalty function when a location is deleted from the current solution.

3 Descripton of Test cases

The first set of test cases address the k-median problem. The seminal work in modern location theory is generally attributed to Hakimi [4],[5]. Hakimi formulated the general problem of locating one or more facilities to minimize the sum of the distances (k-median) between facilities and customers on the network. The k-median problem is $\mathcal{NP}$-hard and, as a result, solution strategies focus on heuristic procedures. One difficulty in comparing the performace of heuristics is having a standard set of test problems. The OR-library ([10], [11]) provides 40 k-median test cases of varying sizes. We have chosen 4 problems from this libary that have proven difficult for other heuristics procedures to solve (i.e. determine the optimal solution [12]) as a way to benchmark the performance of our TS and reninforcement learning procedures.

Table 1: k-median OR-library test problems.

Problem	n	k	Opt Value
14	300	60	2968
15	300	100	1729
19	400	90	2845
34	700	140	3013

Two flexible space structures provide the data for damper placement problem test cases. Prior to the development of the International Space Station launched in 1998, the increasing demand for larger-sized space structures with lower mass led to the development of highly flexible truss structures where, in effect, every point can move relative to the next. The first test article was developed at NASA Langley during Phase I of the Control–Structures Interaction (CSI) Evolutionary Model (Figure 1). The second test article was developed for NASA at the MIT Space Engineering Research Center (Figure 1). Let m denote the number of modes and n denote the number of truss members in the structure. A normal modes analysis of a finite element model of these two structures yield modal strain energy matrices with dimensions of 10 by 1507 for the CSI truss and 34 by 222 for the MIT truss. The entries in the matrix have been normalized so that each entry denotes the percentage of the total modal strain energy imparted in mode i to truss member j. Let D denote the strain energy matrix.

The goal is to select k truss members to be replaced by active dampers so that the *modal damping ratio* is maximized for all significant modes and is referenced as the Damper Placement Problem (DPP). Maximizing the modal damping ratio is a widely accepted goal in DPPs (see [9]). Both active and passive dampers dissipate forces that are internal to the structure and are most effective when replacing truss members with maximum extension or compression. The truss elements with maximum internal displacement are those with the largest strain energy over all modes. Following [13], a force-feedback control law (see also [14]) is used, yielding damping ratios that are directly proportional to the fraction of modal strain energy. Hence, the maximization of the modal damping ratio for all modes can be accomplished by selecting the k damper locations

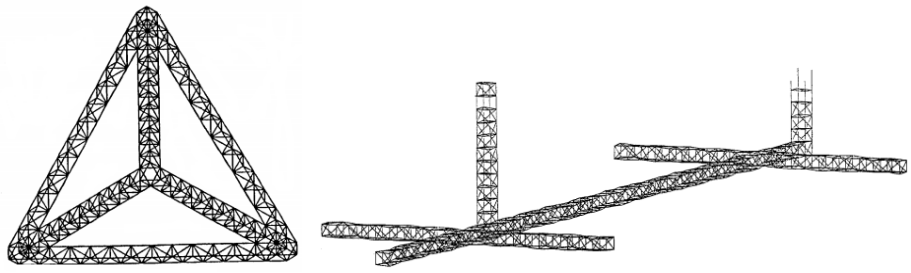

Figure 1: Flexible Space Structures: MIT and CSI

that maximize the minimum sum of modal strain energy over the k chosen locations. That is, given D the goal is to find the m by k submatrix whose smallest row sum is as large as possible.

4 Tabu Search with Reinforcement Learning

A high-level description of how a reinforcement learning procedure can be used instead of a more traditional long-term memory approach for TS is described. Inspired by the algorithm presented in [7] for an airline gate assignment problem, a probability learning procedure is implemented that rewards selected locations (columns of D) based on each location's contribution to solution quality, with the goal of escaping local optimum.

A probability vector $\vec{P}$ of size $n \times 1$, where n is the number of columns of the data matrix D, records the rewards and penalties. Each entry p_i of $\vec{P}$ denotes the selection probability of the associated location i. Initially, all p_i in P are set to $\frac{1}{n}$ for a fair start and are updated during the search.

TS proceeds by repeating a predetermined number of iterations (designate this parameter by *max_it*) of a local search in which all possible swaps between the k columns in the solution and the $n - k$ columns not in the current solution are examined. In each of the *max_it* iterations the *best* swap (which may be the best nonimproving swap) that is not tabu, or is tabu but meets the aspiration criterion is selected. The tabu status of the selected swap is updated, as well as the probability vector $\vec{P}$. Each collection of *max_it* iterations is called a *round*.

The initial solution for TS is generated by random selection or by a greedy add heuristic [15]. After the completion of each round (see Algorithm 1 below), a new starting solution is determined by making use of the probablity vector $\vec{P}$ before the next round of search iterations begins. During the search, an incumbent solution (the best solution found over all rounds), and temporal optima (the best solutions for each round) are recorded.

A matrix T of size $n \times n$, records the tabu status of each swap. Initially, each t_{ij} entry is set to 0. At the conclusion of each iteration, the tabu status t_{ij} and t_{ji} of the best swap are updated by adding the sum of the predetermined *tabu tenure* to the current iteration number. As a result, in subsequent iterations, exchanging columns i and j (or j and i) is deemed tabu if the current iteration number is less than $t_{i,j}$ (or $t_{j,i}$).

Algorithm 1 A Round of Tabu Search

Input: data matrix D(n,n); M selected columns of D; $NotM$ nonselected columns; tabu tenure t
Output: Updated selected set M, nonselected set $NotM$

while *iteration* $<$ *max_it* **do**
  Set $T_{i,j} = 0$ for all $i, j \in T$, Set $p_i = 1/n$ for all $i \in \vec{P}$.
  for *all* $M_i \in M$ **do**
    for *all* $\in NotM_j$ **do**
      let $M_i = NotM_j$, $NotM_j = M_i$
      compute objective value for M, let $S_{(i,j)}$ = new obj value
      let $M_i = NotM_j$, $NotM_j = M_i$
    end
  end
  while *best swap is None* **do**
    let (i, j) be $argmax(S)$ or $argmin(S)$
    if $T_{(M_i,NotM_j)} >= iteration$ **then**
      if $S_{(i,j)}$ *is better the current best objective value* **then**
        Best Swap $\leftarrow (i, j)$, incumbent solution = $S_{(i,j)}$
      end
      else
        $S_{(i,j)} = 0$ or ∞
      end
    end
    Best Swap $\leftarrow (i, j)$
  end
  Set $M_i = NotM_j$, $NotM_j = M_i$; $T_{(M_i,NotM_j)}$, $T_{(NotM_j,M_i)}$ = iteration $+ t$
  if $S_{(i,j)} >$*temporal optimum* **then**
    temporal optimum=$S_{(i,j)}$
  end
end
return M,
return M, $NotM$

5 K-Median Computational Experiments

The k-median computational experiments were conducted with three models; one that uses standard TS; one that uses LTM for its restartsl; and one that uses reinforcement learning for its restarts. These two models were coded with Python and run on four datasets from the OR library [11], each with a known optimal solution. Knowledge of the optimal solutions allowed us to compare the efficiency and correctness of the models. Each test run used the same or similar TS search parameters to allow adequate comparison. If the baseline model (no LTM) found the optimal solution to the data set, then the second model (w/ LTM) was not run. The findings are summarized in the table below.

Problem	n	p	Opt Value	No LTM	w/ LTM	w/ RL
14	300	60	2968	2968	n/a	n/a
15	300	100	1729	1736	1732	1729
19	400	80	2845	2848	2848	2846
34	700	140	3013	3022	3016	3018

5.1 Model 1: Baseline Tabu Search with no LTM

The baseline version of TS did not utilize any kind of long-term memory component, relying only on the local search procedure outlined in [2] and [1], which includes a tabu tenure of 60 as well as aspirational moves. All starting solutions were generated with a simple greedy-add algorithm [15]. Each search consisted of 200 to 350 iterations. These parameters were determined during the exploratory phase of this research and were not rigorously tested for optimality. Graphs for each data set were generated to visualize the algorithm's progress. The baseline model found the optimal solution for only one of the four data sets, which indicates that LTM has the potential to improve these initial results. The successful search results for data set 14 without an LTM component are given in Figure 2.

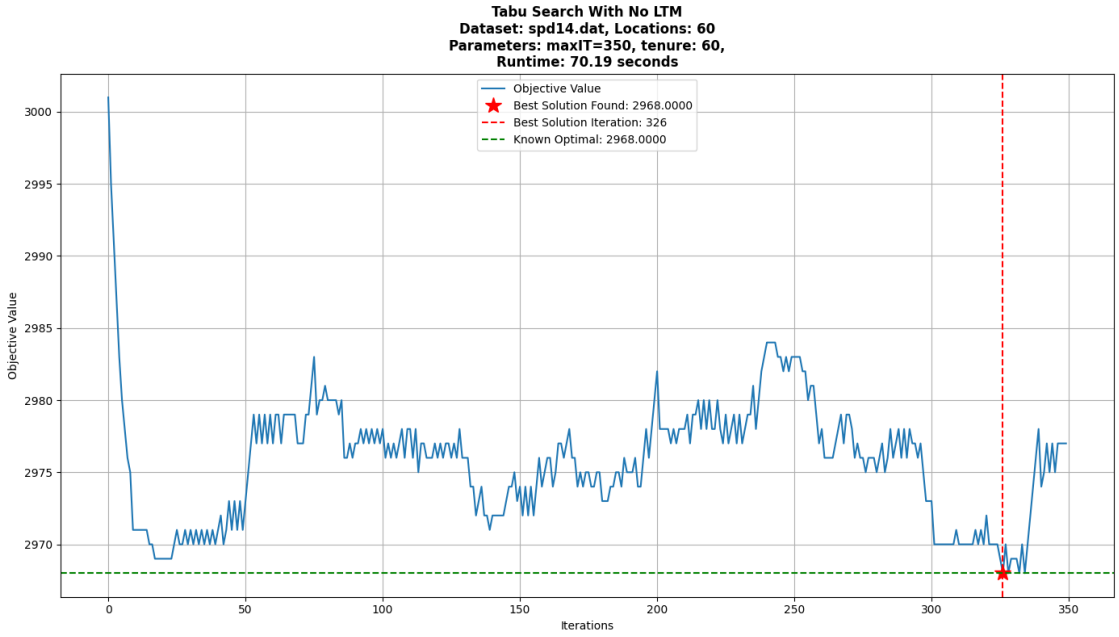

Figure 2: Baseline Tabu Search Visualization

5.2 Model 2: Adjusted Distance Matrix for LTM Restarts

The LTM procedure makes use of the previous round's search history by adjusting the value of the entries in a copy of D, the shortest path distance matrix. Let $D_{LTM} = D$ and let d_{max} be the maximum entry in D_{LTM}. A fraction, μ of d_{max} is added, entry-wise, to the matrix D_{LTM}.

After some experimentation, the parameter μ was assigned the value 0.03. Each column j of D_{LTM} corresponding to a location in the best solution found in the previous round is altered by adding $\mu * d_{max}$ to $d_{LTM}(i, j)$ for all rows i. This method encourages the model to restart with lesser-used locations, as shortest path distances associated with locations found in the best local solution are penalized in the distance matrix D_{LTM}. (Note that D_{LTM} is used only in the greedy-add procedure, the original D is used in the improvement phase of TS.) For this model, the number of iterations for each run was 350 to allow time for the run to decrease the likely higher objective value of its restart solution.

This restart method was successful in finding a better solution than the baseline model for two of the data sets, 15 and 34. For the remaining data set, 19, this model did not find a better solution than the baseline model. In fact, only in the first run did the model find a solution as good as the baseline model. The inefficiency of subsequent runs for this data set indicates that after a certain point, the $\mu * d_{max}$ value lead the search into non-improving areas of the solution space. In the future, it could be beneficial to decrease this parameter for locations that have not been a part of the solution for several runs. This smoothing could reduce the chance that a location that belongs to the optimal solution is perpetually excluded from the restart solutions.

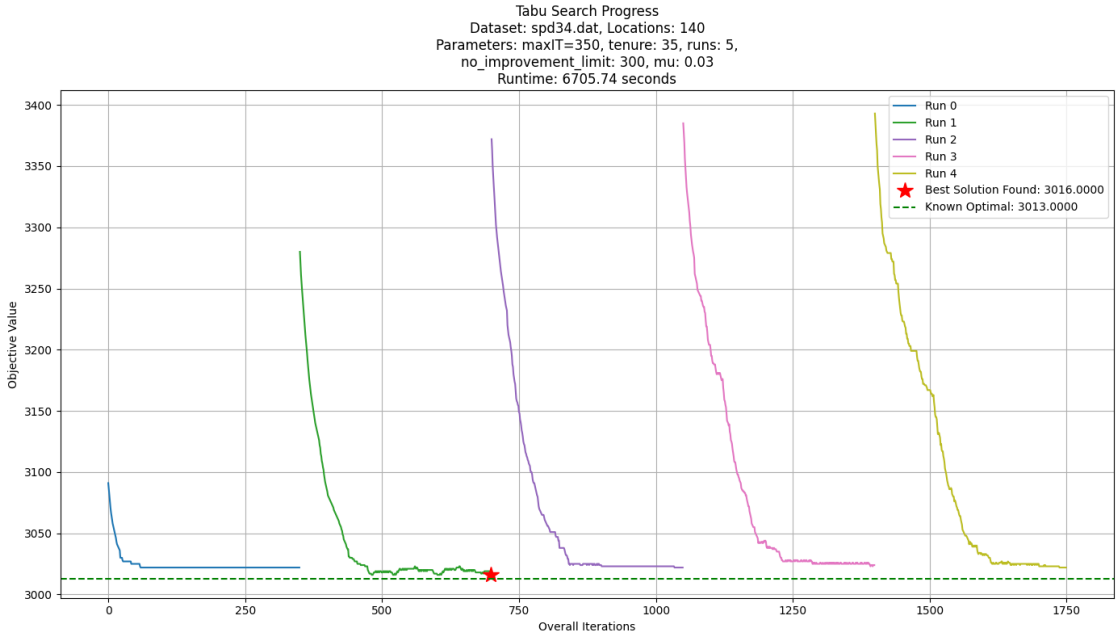

Figure 3: Visualization of Tabu Search with LTM Restarts

5.3 Model 3: Reinforcement Learning with Reward and Penalty Matrix

The third version of the model uses reinforcement learning techniques to gather information for restarting a new round of TS. Taking inspiration from [7], each row i of the probability vector is initialized with $p_j = 1/n$ for all j, where n is the number of potential locations. When a location i remains selected in the best solution at the end of iteration t, p_j is rewarded as follows:

$$p_j(t+1) = \begin{cases} \alpha + (1-\alpha)p_j(t) & \text{if } j = l \\ (1-\alpha)p_j(t) & \text{otherwise} \end{cases}$$

where α $(0 < \alpha < 1)$ is a reward factor. If location l is swapped for location m, location l's probability vector entry is penalized as follows:

$$p_j(t+1) = \begin{cases} (1-\gamma)(1-\beta)p_j(t) & \text{if } j = l \\ \gamma + (1-\gamma)\frac{1}{k-1}\beta + (1-\gamma)(1-\beta)p_j(t) & \text{if } j = m \\ (1-\gamma)\frac{1}{k-1}\beta + (1-\gamma)(1-\beta)p_j(t) & \text{otherwise} \end{cases}$$

where β $(0 < \beta < 1)$ is a penalization factor and γ $(0 < \gamma < 1)$ is a compensation factor. After each update, the probability vector is normalized to ensure that it sums to 1. A smoothing factor ρ is used if any probability value gets too large. In the computational experiments smoothing is done whenever a probability threshold of 0.3 is reached. When restarting, the locations with the k largest probabilities are selected (intensification), and the remaining $n - k$ locations are chosen at random (diversification). k is set to $n/2$, half of the number of potential locations in the data set.

This approach was moderately successful, finding the optimal solution for data set 15 and getting closer to the optimal value for data set 19 than the other two models. However, this model only matched, not improved, the best value found by the second model. As such, our reinforcement learning model shows promise but may need further fine-tuning of its parameters.

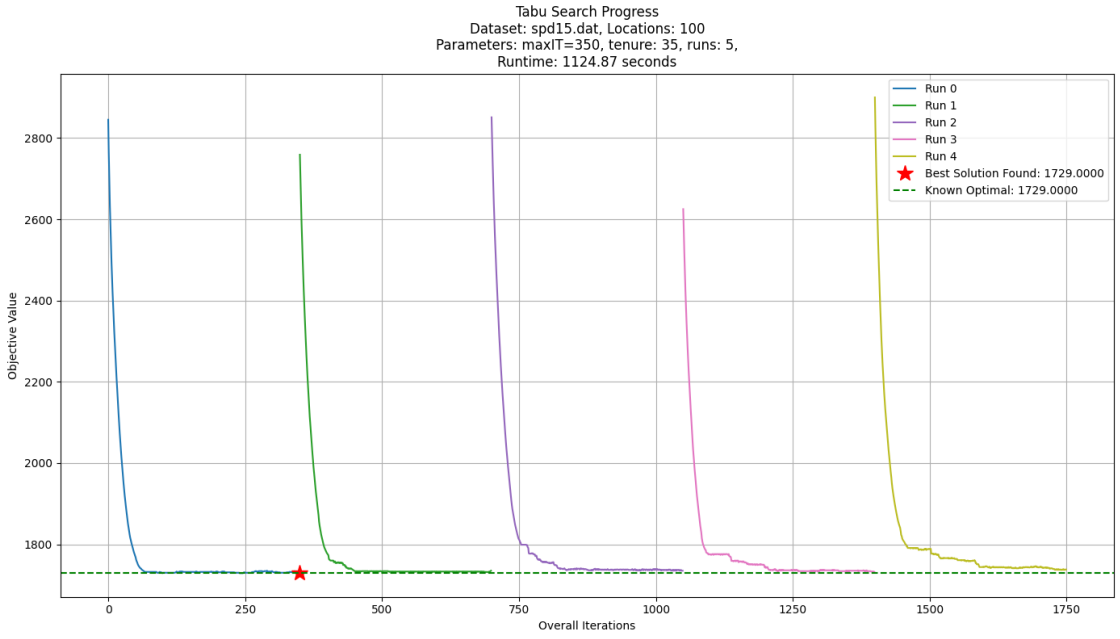

Figure 4: Visualization of Tabu Search with Reinforcement Learning

6 DPP Computational Experiments

In this section, the performance of the LTM-Restarted Tabu Search(LTM) and Probability Learning-Restarted Tabu Search(PLT) are examined through computational experiments. The search space (problem size) of the DPP grows exponentially as the number of location chosen (k) increases. After considering computation time, the search parameters are doubled each time k is doubled. For example, $(25, 25, 10)$ which denotes 25 rounds of tabu search with 25 as the maximum number of iterations, and a tabu_tenure of 10 iterations for $k = 8$, would be doubled to $(50, 50, 20)$ when $k = 16$. To compare the performance of the heuristics, experiments are carried out such that the basic search parameters and random seeds are fixed for each value of k tested. By running numerous repetitions of plain Tabu Search (TS), 4 sets of search parameters were identified, that is, 4 different combinations of maximum round, maximum iteration, and tabu_tenure length.

For PLT, two different methods for updating the probability vector are proposed, see Algorithm 2. If `Dual_Update` is set to false, a pre-determined value α is used to update the probability vector when an improving solution is found; if it is set to true, and a new global optimum is found, another pre-determined value β is introduced to update the probability vector. The probability vector will reset when each iteration is complete.

For LTM, a parameter μ is used to adjust the data matrix. Depending on whether the search is in an intensify (diversify) phase, multiply $\vec{f}$, the recorded frequency vector by μ, and add (subtract) the product to each row of the temporal data matrix. This temporal matrix is then presented to a Greedy Add heuristic to generate a new starting solution; see Algorithm 3.

In the experiments, the four sets of basic search parameters, three random seeds, and two choices of the special search parameter are cross-computed, resulting in 24 unique trials for LTM and PLT. For the plain tabu search(TS), the size of the basic search parameters are doubled to make up for the special search parameters to make an even number of 24 trials. The initial solutions for TS and PLT are randomly generated, while the initial solution for LTM is obtained using the greedy add heuristic.

After extensive exploration with different values, the following parameter values were selected: $\mu = 0.0003$ and $\mu = 0.0005$ as special search parameters for LTM. Both the average objective values of all 24 trials, as well as the best objective achieved are recorded. The number in parentheses is the number of times the observed best solution was found in the 24 trials. The results are shown in Table 1.

k	Tabu Mean	**Tabu Best**	PLT Mean	**PLT Best**	LTM Mean	**LTM Best**
8	.014208	**.014371(1)**	.014247	**.014371(4)**	.014064	.014291(11)
16	.029554	.029646(3)	.029565	**.029718(1)**	.029582	.029647(18)
32	.057719	**.057943(2)**	.057814	**.057943(8)**	.057718	**.057943(1)**

Table 1: Performance Metrics for csi Data

k	Tabu Mean	**Tabu Best**	PLT Mean	**PLT Best**	LTM Mean	**LTM Best**
10	23.043	23.13207(1)	23.091	**23.34815(3)**	22.890	23.13207(2)
20	46.708	46.98334 (1)	46.869	**47.06370(1)**	46.535	46.76752(2)

Table 2: Performance Metrics for mit truss Data

Algorithm 2 Probability Vector Update

Input: Probability Vector $\vec{P}$, two reward/penalty parameters α and β
Output: Updated ProbDability Vector $\vec{P}$

if $S_{(i,j)} >$ *temporal optimum* **then**
  if *Dual Probability Update* **and** $S_{(i,j)}$ *is better the current best objective value* **then**
    Set $p_i = p_i \times (1 - \beta)$ for all $i \neq M_i$
    Set $p_i = p_i \times (1 - \beta) + \beta$ for $i = M_i$
  end
  else
    Set $p_i = p_i + \alpha$ for $i = M_i$
    Set $p_i = p_i - \alpha$ for $i = NotM_j$
  end
end
return $\vec{P}$

Algorithm 3 Greedy Add using LTM

Input: Frequency vector $\vec{f}$, data matrix D, ratio μ, column index set N, row index set M
Output: $X \subseteq N$ (set of k columns of D)

if *Intensify/Diversify* **then**
  $D'_i = D_{i,:} \pm (\vec{f} \times \mu),\ \forall\ rows\ i$
  Greedy Add Heuristic
  X$\leftarrow \{\}$
  while $|X| < k$ **do**
    for $j \in N - X$ **do**
      for $i \in M$ **do**
        $rowsum(i) = D'_{i,j} + \sum_{l \in X} D'_{i,l}$
      end
      $rowsumMax(j) = \max_{i \in M} rowsum(i)$
    end
    $j* = argmin_{j \in N-X} rowsumMax(j)$
    $X = X + j^*$
  end
end
return X

References

[1] Glover, F., "Tabu Search: A Tutorial," *INTERFACES*, **20** (1990) 74-94.

[2] Glover, F., "Tabu Search–Part I," *ORSA J. on Computing*, **1** (1989) pp. 190-206.

[3] Skorin-Kapov, J., "Tabu Search Applied to the Quadratic Assignment Problem," *ORSA J. on Computing*, **2** (1990) pp. 33-42.

[4] Hakimi, S.L., "Optimal Locations of Switching Centers and the Absolute Centers and Medians of a Graph," *Operations Research*, **12**, 450-459 (1964).

[5] Hakimi, S.L., "Optimum Distribution of Switching Centers in a Communication Network and Some Related Graph Theoretic Problems," *Operations Research*, **13**, 462-475 (1965).

[6] Li, L., Z. Wei, J. Hao, and K. He, "Probability Learning Based Tabu Search for the Budgeted Maximum Coverage Problem," *Expert Systems with Appications*, Vol. 183, November (2021) article 115310.

[7] Li, M., J. Hao, and Q. Wu, "Learning-driven Feasible and Infeasible Tabu Search for Airport Gate Assignment," *European Journal of Operational Research*, Vol. 302, Issue 1, October (2022) pp. 172-186.

[8] Sun, Z., U. Benlic, M. Li, and Q. Wu, "Reinforcement Learning Based Tabu Search for the Minimum Load Coloring Problem," *Computers and Operations Research*, Vol. 143 (2022) article 105745.

[9] Anderson, E., M. Trubert, J. Fanson, "Testing and Application of a Viscous Passive Damper for Use in Precision Truss Structures," *Proceedings of 32nd Structures, Structural Dynamics and Materials Conference,* Baltimore, MD., April (1991) pp. 2795-2807.

[10] Beasley, J.E., "A note on solving large p-median problems," *European Journal of Operational Research*, Vol. 21 (1985) pp. 270-273.

[11] Beasley, J.E., "OR-Library: distributing test problems by electronic mail", *Journal of the Operational Research Society*, Vol. 41(11) (1990) pp. 1069-1072. (last updated 2018) Available for download at `https://people.brunel.ac.uk/ mastjjb/jeb/info.html`

[12] Chiyoshi, F., and R.D. Galvao, "A Statistical Analysis of Simulated Annealing Applied to the p-Median Problem," *Annals of Operations Research*, Vol. 96, (2000) pp. 61-74.

[13] Padula, S., and C.A. Sandridge, "Passive/Active Strut Placement by Integer Programming," *Topology Design of Structures,* Martin P. Bledsoe and Carlos A. Mota Soares, eds., Kluwer Academic Publisher (1993) pp. 145-156.

[14] Preumont, A., "Active Damping by a Local Force Feedback with Piezoelectric Actuators," *Proceedings of 32nd Structures, Structural Dynamics and Materials Conference,* Baltimore, MD., April, 1991, 1879-1887.

[15] Daskin, M., *Network and Discrete Location: Models, Algorithms, and Applications*, John Wiley & Sons, 2013. `ISBN`: 978-0-470-90536-4