Metaheuristics for Optimization

Series 2 : The Quadratic Assignment Problem

Part 1 : Return no later than October 3, 2022 (16h00)
Part 2 : Return no later than October 10, 2022 (16h00)

1 Quadratic Assignment Problem

The *Quadratic Assignment Problem* (QAP) is one of the fundamental combinatorial optimization problems, and it is important both in theory and practice. Intuitively, QAP can best be described as the problem of assigning a set of facilities (e.g. factories) to a set of locations (e.g. cities) with given distances between the locations and given flows (e.g. the amount of supplies transported) between the facilities. The goal then is to place all the facilities on different locations in such a way that the sum of the products between flows and distances is minimal. This problem is known to be NP-hard.

More formally, given n facilities and n locations together with two (symmetric) matrices $D = [d_{rs}]$ and $W = [w_{ij}]$ where d_{rs} is a distance between locations r and s and w_{ij} is the flow between facilities i and j, the QAP can be stated as follows :

$$argmin_{\psi \in S(n)} I(\psi) = \sum_{i,j=1}^{n} w_{ij} d_{\psi_i \psi_j}$$

where $S(n)$ is the set of all permutation (corresponding to the assignments) of the set of integers $\{1, \ldots, n\}$; ψ_i gives the location of facility i in the current solution $\psi \in S(n)$.

2 Tabu Search

Tabu Search (TS) uses a local or neighborhood search procedure to iteratively move from a solution ψ to a solution ψ' in the neighborhood of ψ, until some stopping criterion has been satisfied. To explore regions of the search space that would be left unexplored by the local search procedure, tabu search (TS) modifies the neighborhood structure of each solution as the search progresses. The solutions admitted to $N^*(\psi)$, i.e. the neighborhood, are determined through the use of special memory structures. The search then progresses by iteratively moving from a solution ψ to a solution ψ' in $N^*(\psi)$. In this work we focus on the "tabu list" memory structures. In its simplest form, a tabu list contains the solutions that have been visited in the recent past (less than l moves ago, where l is the tabu tenure time). Solutions in the tabu list are excluded from $N^*(\psi)$.

3 Tabu Search for QAP

This section describes the specific design of the tabu search algorithm for the QAP :

Fundamental concepts :

1. The *neighborhood* $N(\psi)$ of a given solution ψ is given by the classical 2-exchange which amounts to swapping two locations.
2. The *tabu list* contains assignment of facilities to specific locations, i.e. the elements of tabu list are in the form (i, r) which refers to the fact that it might be forbidden to assign facility i to a location r. More precisely, a neighboring solution that places facilities i and j to locations r and s, respectively, is tabu if in the past l iterations local search moves were done that removed facility i from location r *and* facility j from location s.
3. The algorithm starts by randomly generating an initial assignment solution. At each iteration the best non tabu solution in $N^*(\psi)$ (the neighborhood of the current solution) is chosen as the new solution, even if it is worse than the current solution. The algorithm terminates after t_{max} iterations. Then the overall solution is the best assignment among the t_{max} iterations.
4. Here TS has one additional rule which introduces a type of *diversification mechanism* to the local search and which might be important to achieve good computational results in the long run : If a facility i has not been placed on a specific location r during the last u iterations, any move that does not place facility i on location r is forbidden. In fact, in such a situation, the algorithm forces to place a facility on one particular location. Fix $u = n^2$.
5. Optimizations and improvements :

   The TS algorithm utilizes a matrix to store the cost associated with each swap that may be executed for the current permutation. These "partial costs" can then be added to the original cost of the permutation to obtain the value associated with the new permutation. In this manner, the costs of possible moves can be quickly evaluated and once a move is chosen, the matrix can be efficiently updated to reflect the costs associated with the newly formed permutation. The objective function difference $\Delta(\psi, i, j)$ obtained by exchanging locations ψ_i and ψ_j can be computed in $O(n)$ time using :

$$\Delta(\psi, i, j) = 2 \sum_{k \neq i,j} (w_{jk} - w_{ik})(d_{\psi_i,\psi_k} - d_{\psi_j,\psi_k})$$

4 Work to Do

4.1 Part 1

Work on implementing the functions that will allow you to :

— Import data
— Compute the fitness
— Perform two-swap
— Evaluate the Delta function
— Evaluate if a point is tabu
— Update the tabu list
— Filter the diversification mechanism swaps

— Update the diversification matrix

To start, you can take advantage of the given skeleton code.

4.2 Part 2

Run your algorithm 10 times on the example provided with this exercise (the "1.dat" file) ; make sure that you start each iteration with a different initial state. Vary the l parameter (tabu tenure) over $\{1, 0.5n, 0.9n\}$ and for each value of l report the best, the mean and the standard deviation of the obtained values of I. For this problem check whether the diversification mechanism (Point 4 in section 3) helps.

Optional : Randomly generate 4 QAPs problems (i.e. symmetrical distance and weight matrices) of size $n = \{40, 50, 80, 100\}$. Fix $l = 0.5n$ and run your algorithm 10 times on these problems. Report mean and standard deviation of the obtained values of I. Here it is up to you to decide whether to use the diversification mechanism.

You may set t_{max} to 20000 [1].

4.3 Task definition file

The task definition file should have the following format :

```
n D W
```

where n is the number of locations (and facilities) and $\boldsymbol{D}$ and $\boldsymbol{W}$ are distance and weight matrices, respectively. The tabs (0x09) or spaces (0x20) should be used as fields separators. Every line starting with characters #, ! or ; should be ignored. Empty lines should be ignored as well.

5 Work to return

For this series, submit your code, results, and comments, and upload them on moodle, by :

— Monday, October 3, 2022 at 16h00 for Part 1.
— Monday, October 10, 2022 at 16h00 for Part 2.

The use of python notebooks is highly recommended, to be able to include code, results, and comments in the same file. Graphs must include a title and a legend for the axes when needed.

Make sure to test and discuss the effect of :

— varying the tenure time
— diversification mechanism
— **OPTIONAL :** varying t_{max} (for eg. 500 to 20,000)

In this specific case, what is the neighborhood of an element of the research space ? In terms of n (number of facilities/locations), what is the size of the neighborhood ?

1. If your algorithm is too slow decrease the value of the t_{max} parameter.