Metaheuristics for Optimization

Series 4 : Ant System and Traveling Salesman Problem

Return no later than November 10, 2019

The goal of this exercise is to implement the *Ant System* (AS) algorithm and use it to solve different instances of the *Traveling Salesman Problem* (TSP) we already encountered in the previous TP.

1 The AS algorithm

In the AS algorithm for the TSP problem, at each iteration t ($t = 1, \ldots, t_{max}$), each ant k ($k = 1, \ldots, m$) moves on a weighted graph that encodes the TSP problem and constructs a path of n nodes (i.e. cities).

In the construction of a solution, ants select consecutive cities to be visited through a stochastic mechanism. For each ant k which is in city i the probability of going to city j in iteration t is given by :

$$p_{ij}^k(t) = \begin{cases} \frac{(\tau_{ij}(t))^\alpha (\eta_{ij})^\beta}{\sum_{l \in J} (\tau_{il}(t))^\alpha (\eta_{il})^\beta} & \text{if } j \in J \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where J is the set of cities not yet visited by ant k , η_{ij} is the inverse of the distance [1] between cities i and j (i.e. $\eta_{ij} = \frac{1}{d_{ij}}$), $\tau_{ij}(t)$ is the intensity of the path between i and j at iteration t. The parameter α and β control the relative importance of the pheromone versus the heuristic information η_{ij}.

At each iteration the pheromones are updated by all the m ants that have built a solution. The pheromone τ_{ij} associated with the edge joining cities i and j is updated as following :

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^k(t) \quad (2)$$

where ρ is the evaporation rate and $\Delta\tau_{ij}^k(t)$ is the quantity of pheromone laid on edge (i, j) by ant k :

$$\Delta\tau_{ij}^k(t) = \begin{cases} \frac{Q}{L^k(t)} & \text{if ant } k \text{ used edge } (i,j) \text{ in its tour} \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

where Q is a constant, and L^k is the length of the tour constructed by ant k. Initially, a small quantity $\tau_0 \geq 0$ is assigned to each edge joining a pair of cities. The best path in terms of the length is returned as a final solution. The algorithm is summarized in the pseudo-code below.

1. The relative distances should be computed using the euclidean metric.

Algorithm 1

```
1: for all t = 1, ..., t_max do
2:     for all ant k = 1, ..., m do
3:         choose a city at random
4:         while there exists a city not visited do
5:             choose a city j according to (1)
6:         end while
7:         mark a path according to (3)
8:     end for
9:     update all paths according to (2)
10:    Keep the best of solutions obtained at last iteration
11: end for
```

1.1 Choice of the parameters

The parameters α, β and ρ should have values 1, 5 and 0.1, respectively. The parameters Q and τ_0 have values L_{nn} and $\frac{1}{L_{nn}}$, respectively, where L_{nn} is a possible solution determined by using the nearest neighbour algorithm (see Section 2). You are free to assign the value to the m and t_{max} parameters, i.e. experiment with a few values of m and t_{max} and select the one which gives the best results. You are allowed to change the values of the above parameters as long as you mention it clearly in the report and argument on the way it improves the performance of the metaheuristic.

2 Work to do

In all the experiments assume that you start and end your path in the first city, i.e. the first city might be a warehouse from where the goods are delivered to all the facilities (it means that $c_{n+1} = c_1$ must hold in (4).

Run your AS algorithm on the two problems attached to this exercise [2] and report the solutions as in (4).

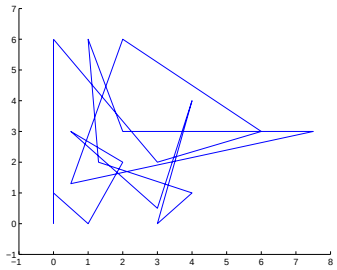
For the above problems analyze the influence of the m and t_{max} parameters to the quality of the solutions.

Compare the results with those obtained by using the simple greedy algorithm called *nearest neighbour algorithm* [3], repeated 10 times.

For the two TSP instantiations provide a visualization of the obtained optimal solutions by plotting the positions of the cities and the shortest paths obtained by the AS and the greedy algorithm. The following picture displays an example of such a visualization.

2. These problems represent facility placements (e.g. shops) in a city, where more facilities are in a center of a city, and less are in a suburb.

3. This algorithm simply requires to pick a city randomly, then move to the nearest one, then to the nearest one that has not been visited yet, etc



Moreover, randomly generate five TSP problems of size 50, 60, 80 and 100. For these problems report and comment on the means and standard deviations of the execution times and lengths of the paths of the AS and greedy algorithm (the latter averaged over 10 runs).

Compare and discuss the performance of the Ant System (AS) and the Simulated Annealing (SA) algorithms applied on the TSP in terms of their solution quality and execution time.

Your code should return the solution of the TSP in the form of a sequence of cities corresponding to the shortest path and its length, i.e.

$$[c_1, \ldots, c_n] : \sum_{i=1}^{n} d(c_i, c_{i+1}) \tag{4}$$

where $c_1, \ldots, c_n$ are cities and d is a distance measure.

Please include in your report a short description of the meaning of the various parameters used in the AS algorithm, as well as a discussion on its advantages and drawbacks.

3 Report

Each student is required to give back a *personal* work consisting of a code and a concise but precise report in PDF format 4-5 pages) displaying an introduction to the optimization problem to be solved, a description of the employed metaheuristic, the experiments carried through with the corresponding results and discussion. Both report and code have to be uploaded on Moodle (TP4).