Stochastic optimization algorithms, 2016
Report, Home Problems set2
Jose Esteban Pérez-Hidalgo, 880416T670

Problem 2.1 TSP

a) Here we are dealing with permutations of N elements. So, the number of ways in which one can arrange N (different) elements, say, the number of permutations, is the factorial of the number of elements, e.g: $N!$
Now, in this particular case, we are looking for the number of *distinct paths*. Taking into account the paths one can form starting in different cities but running through the cities in the same order or the reverse order; for every distinct path (d_p) there are $2N$ ways to reorder the path keeping its path length.
Hence:

$$d_p \times 2N = N!$$

$$\Rightarrow d_p = \frac{N!}{2N} = \frac{(N-1)!}{2}$$

b) As discussed during the lectures, the GA without modifications is not the best way to find the best solution for the TSP. In this case the algorithm was initialized with a population of 5000 individuals.

c) The ASO seems to be the proper way to attack the TSP. It takes many runs to find the best set of parameters to find the best solution. In this case the algorithm was initialized with a population of 50 ants, $\alpha = 4$, $\beta = 3$ and $\rho = 0.5$.
The parameters set was chosen based on the performance of the algorithm. A small number of ants was better than a large one, a medium value of ρ worked well, and for α and β it was noted that the algorithm reached a lower best path length faster with this values.
Additional functions named: `GetProbability.m`, `GetNode.m` and `EuclideanPathLength.m`; were used to solve this problem.

d) The modifications made to the GA actually made the algorithm better at trying to find the best path, but the program gets stuck very early in the runs with a good but not optimal solution.
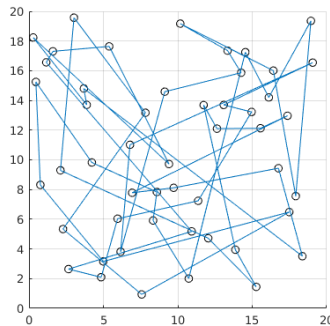Plots:

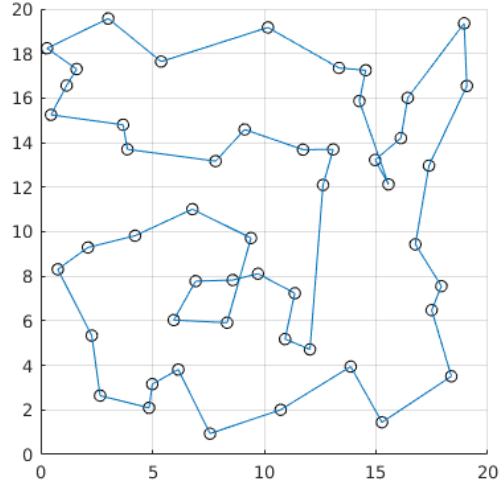Figure 1: Best path found using `GA21b.m`. Path length $= 351$ length units.

Figure 2: Best path found using `AntSystem.m`. Path length = 128 length units.
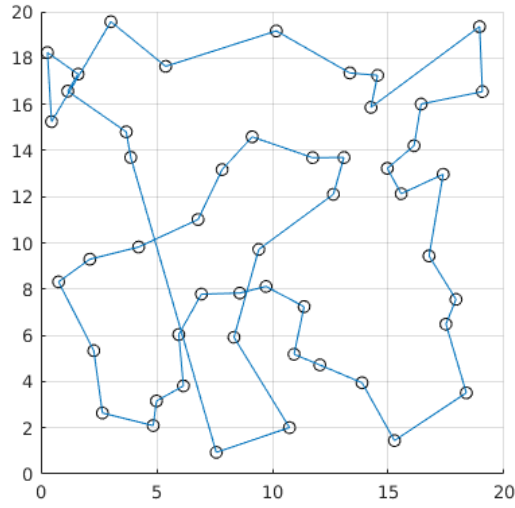

Figure 3: Best path found using `GA21d.m`. Path length = 134 length units.

The conclusion is that the ASO works better for this kind of problem, but it has to be noted that a lot of runs are needed to set the algorithm with the right choice of parameters.

Problem 2.2 PSO

a) The algorithm handled this problem very well as the solution is reached very quickly (after 1000 generations). The minimum of the function is:

$$f(x^*, y^*) = 1$$

Where:

$$(x^*, y^*)^T = (5, 4)^T.$$

b) As in the previous case, the solution is reached very quickly (after 1000 generations). The minimum of the function is:

$$f^* = -737$$

This global minimum is reached in two points:

$$\mathbf{x}_a^* = (0, 12, 23, 17, 6)^T.$$

$$\mathbf{x}_b^* = (0, 11, 22, 16, 6)^T.$$

In both cases the PSO algorithm found the solution very quickly, proving the power of this method for this kind of problems