FFR105, Stochastic Optimization Algorithms
Home Problem 2

October 16, 2019

Ella Guiladi
930509-0822
guiladi@student.chalmers.se

1 Problem 2.1

(a) Since each city N is visited once, the paths can be generated by starting to visit one city, then visiting the next, and so on until all cities has been visited. This will give rise to $N!$ different ways to arrange the cities. In these $N!$ different paths of N elements, the reverse paths and the paths that starts in different cities, but go through the cities in the same order, are also included, i.e the paths that aren't distinct are included in this sets.

Since each route can start at any of the N cities, $N!$ can be divided by N in order to remove the possibility to go through the cities in the same order, i.e get the same path length. Considering that paths that go through a given sequence of cities in opposite order also are equivalent, it's necessary to divide $\frac{N!}{N}$ by 2 to exclude the reverse paths. Thus, the number of distinct paths in the general case of N cities is given by

$$\frac{N!}{2N} = \frac{N \cdot (N-1) \cdot (N-2) \cdot ... \cdot 1}{2N} = \frac{(N-1)!}{2} \tag{1}$$

(b) The main file GA21b.m and the function files that are used to write a Genetic Algorithm (GA), that searches for the shortest path between N cities is found in the file 2.1b. The function files that was used and not included in the TSPGraphics.zip are TournamentSelect.m, Mutate.m, InsertBestIndividual.m, InitializePopulation.m and EvaluateIndividual.m. The GA was implemented by running the main file GA21b.m a couple of times with the population size = 100, $p_{mut} = 0.02$ and tournament selection with tournament selection parameter = 0.75 for 5000 iterations.

(c) The main file AntSystem.m and the function files that are used to write the Ant colony optimization (ACO) in order to solve the routing problem, is found in file 2.1c. The function files that was used and not included in the TSPGraphics.zip are CalculateProbabilityParameters.m, ComputeDeltaPheromoneLevels.m, GeneratePath.m, GetNearestNeighbourPathLength.m, GetPathLength.m, GetVisibility.m, InitializePheromoneLevels.m, ProbabilityOfNextNode.m, RouletteWheelSelection.m and UpdatePheromoneLevels.m.

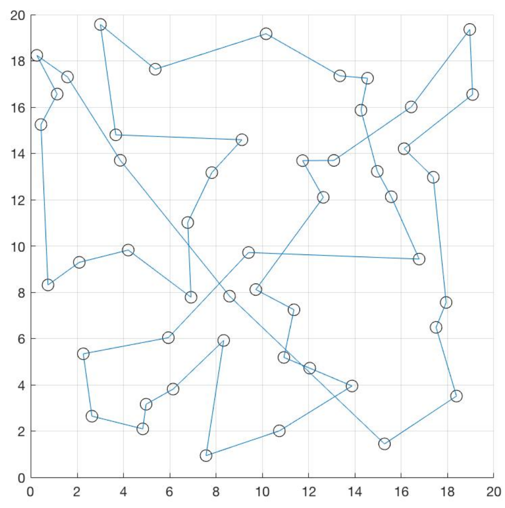
The funciton file GeneratePath.m that return the path for one ant uses CalculateProbabilityParameters.m to calculate the probability parameters. These parameters are used to compute the probability of choosing the next node with the function file RouletteWheelSelection.m. RouletteWheelSelection.m performs Roulette Wheel selection by setting the probabilities in the selection method equal to the probabilities to choose the next node that has not yet been visited, i.e nodes in the path that are not included in the tabu list. This is possible by calling the function file ProbabilityOfNextNode.m in RouletteWheelSelection.m.

In order to implement the Ant System algorithm one needs to run the main file AntSystem.m where the parameters $\alpha = 1$, $\beta = 3$ and $\rho = 0.5$ was chosen after recommendations from the lecture in the course. The number of ants was chosen to 50.

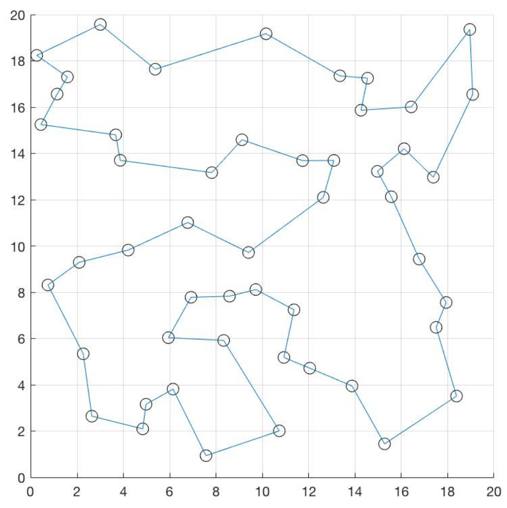
(d) The main file NNPathLengthCalculator.m and the function files that are used in order to compute the length of the path from a randomised starting city to its nearest neighboring city, are found in the file 2.1d. By running the main file NNPathLengthCalculator.m one gets the nearest neighbourhood path length to be ≈ 139.226 length units after one iteration, when starting in city 35. The path length ranges from approximately 135 to 160 lenght units, since the starting city is randomised in each iteration.

This shows that the NNPathLengthCalculator.m performs better than the GA, where the best path was found to be about 163.403 length units after 5000 iterations. The reason for this result is a consequence of the fact that GA will start with a completely random path, in contrast to NNPathLengthCalculator.m that starts in a random city from which the nearest neighboring path is generated. The GA will require a lot of swaps in order to find the shortest path, which increases the running time. In order to get a shorter path one needs to increase the number of iterations notably. The initialized random path also increases the risk for the GA to get stuck in a local minimum, which will most likely result in a longer path.

(e) The shortest path for the GA and the ACO are presented in figure (1a) respectively figure (1b). The path lengths from the GA, ACO and the nearest-neighbour path (from 2.d) are summarised in table (1) for clarity. The shortest path found does not exceed 123 length units and is the path found by the ACO, which is clearly visualised in figure (1b). The best path is included as a vector in file 2.1e. This is due to ACO easily finding the nearest-neighbor path.



(a) The best path found by the GA for the TSP with the path length ≈ 163.403 length units after 5000 iterations.



(b) The best path found by the ACO for the TSP with the path length ≈ 121.105 length units for ant number 2 after 18 iterations.

Figure 1: The path for the TSP found by the GA in (a) and by the ACO in (b)

Table 1: Length of the path found by GA, ACO and nearest-neighbour path

Algoritm	Path length
GA	163.403
ACO	121.105
Nearest-neighbour path	139.226

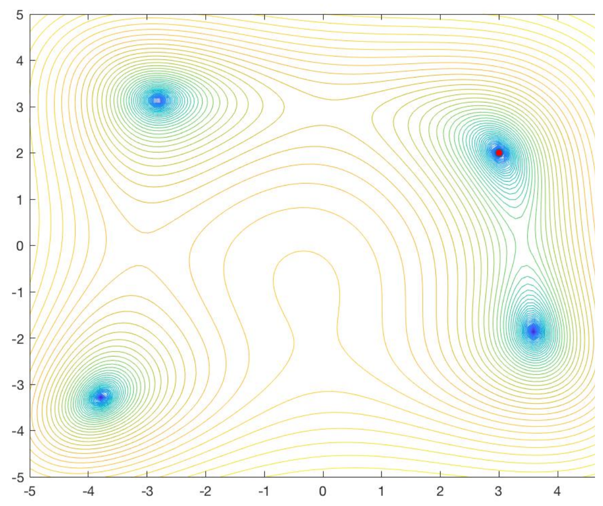
2 Problem 2.2

The main file PSO22.m and the function files that are used in order to implement the Particle swarm optimization (PSO), are found in the file 2.2. By running the main file PSO22.m a couple of times one gets the positions of the best global particle, as well as their function value, i.e each minima for the function $f(x, y)$. The main file and function files are written through the Algorithm 5.1 in the course litterature, including the varying inertia weight.

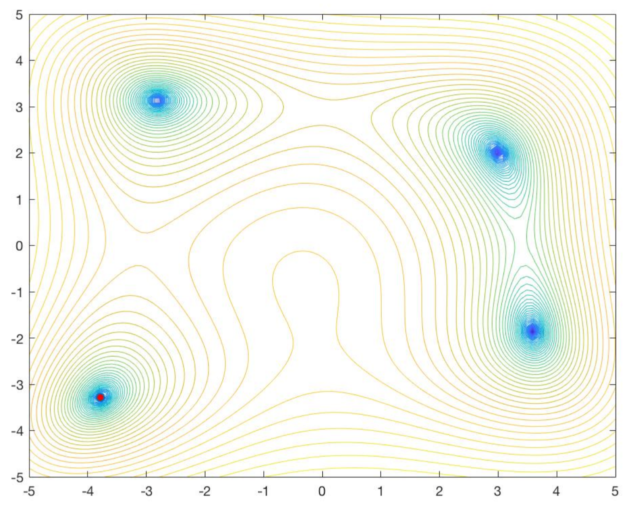
By analyzing the contour plots in figure (2) it is clear that there are four local minima over the range $(x, y) \in$ [-5,5]. The four minima are presented in table (2). From table (2) and figure (2) one can draw the conclusion that the PSO finds all local minima that was identified by the contour plots.

Table 2: The four local minima and corresponding function value found by PSO

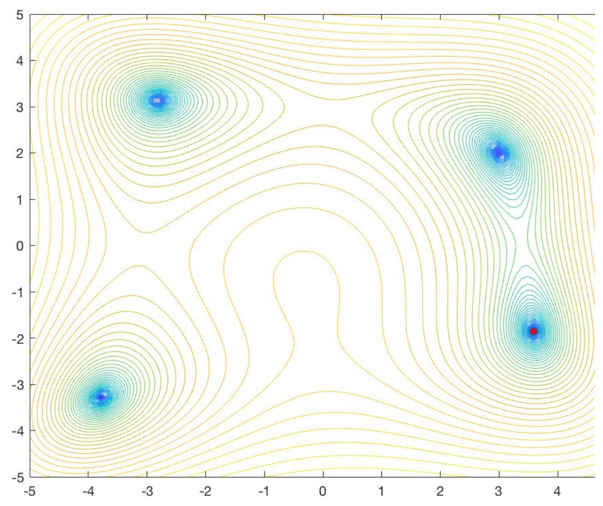
x	y	$f(x, y)$
2.999998	2.000000	0.000000
-3.779299	-3.283156	0.000000
3.584427	-1.848127	0.000000
-2.805102	3.131293	0.000000



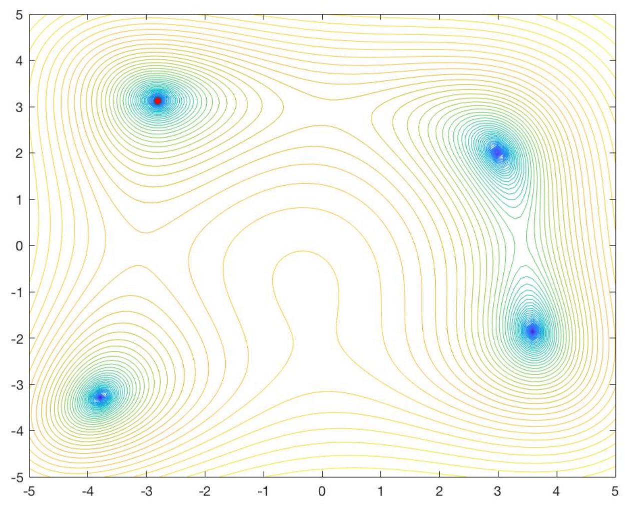
(a) Minima for x=2.999998 and y=2.000000 marked with a red dot



(b) Minima for x=-3.779299 y=-3.283156 marked with a red dot.



(c) Minima for x=3.584427 y=-1.848127 marked with a red dot.



(d) Minima for x=-2.805102 y=3.131293 marked with a red dot.

Figure 2: The contour plot of $f(x, y)$ for $(x, y) \in$ [-5,5]. Each of the four minima is marked with a red dot in the plot.