Home Problem 2

Konstantinos Zakkas
personnummer: 9206157795

October 2020

Problem 2.1, The traveling salesman problem (TSP)

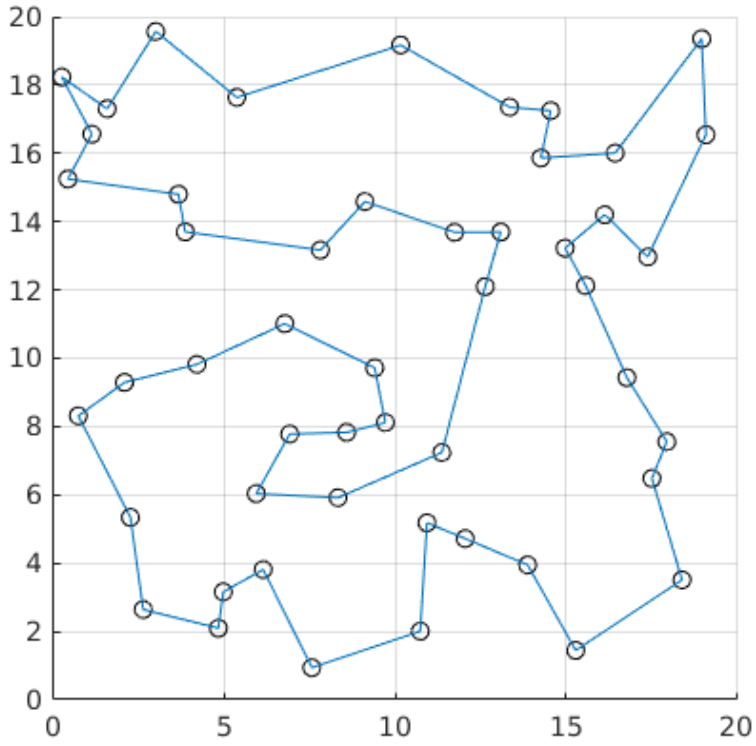
a) There are N! possible paths for N cities because factorial presents the number of different ways of arranging N distinct objects into a sequence. Since paths that start in different cities but they travel through the cities in the same order, it doesn't matter which city you start in. So we can remove the starting ciry and we get (N-1)! paths. We also divide by two because paths that go through a given sequence of cities in opposite order are also equivalent. Thus there are (N-1)!/2 distinct paths for N cities if N>2, because otherwise there can’t be a cycle in the graph and TSP is a Hamiltonian cycle.

b) Running the main script *GA21b.m* for 15 runs and 100 generations we get a path with length 271.75 which is not close to optimal path value 123. For 1000 generations it takes about half an hour to reach a value of 179.85 length units for the path. It doesn’t make any difference to use more runs because the programm usually gives the best result in the first 10-15 runs.

c) The main script is the *AntSystem.m.*

d) Running the script *NNPathLengthCalculator.m* we get a path length 146.23. The program chooses a starting city randomly so each time we run it we get a different path length that is close to the one we got this time. The best path for GA was found for 15 runs and 5000 generations and its length is 157.28. The best path found by the GA is longer than the nearest neighbour path. The GA doesn’t seem to perform well for this problem because the initial paths of the population are generated completely randomly, so they probably have big lengths and many runs are required to get shorter paths, while NNPathLengthCalculator easily finds a short path looking for the nearest neighbour of each node. Also, GA can get stuck to a local minimum and the only way to get out is through selection and mutations after many runs.

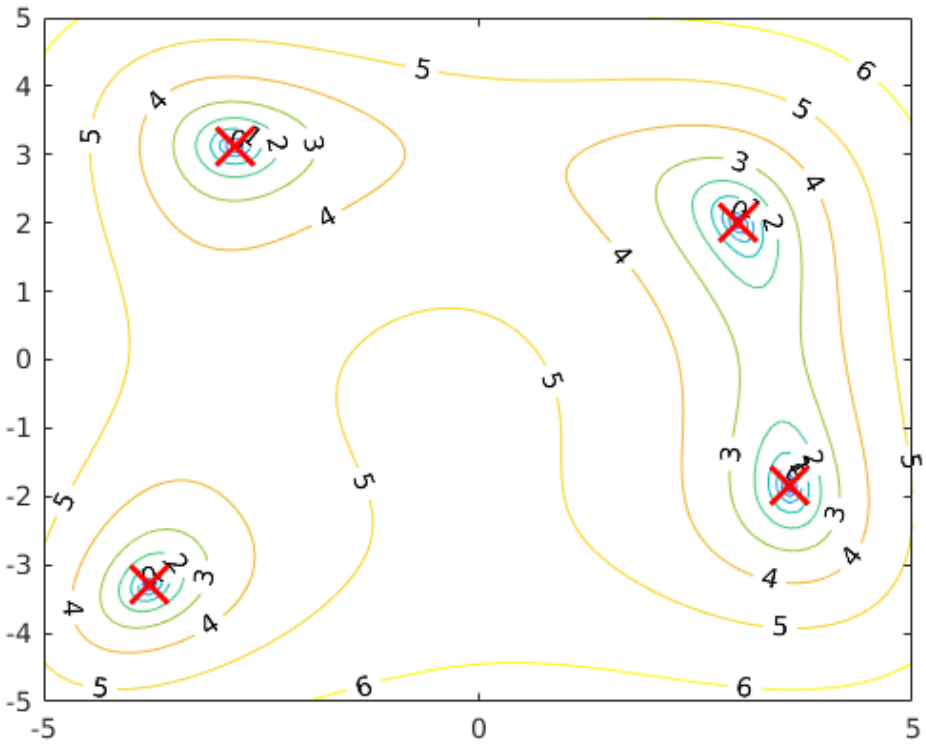
e) The shortest path found for both algorithms after a long run is 157.28 for GA and 122.32 for ACO. The length of the of the nearest-neighbour path from part d) is 146.23. ACO gave the shortest path because for every given step the most likely target is the nearest city. Thus ACO is very likely to find the nearest-neighbour path almost immediately, which is much shorter than randomly generated paths which used to initialize GA. The plot for the shortest path found with ACO is the following:



The shortest path found is included in the file *BestResultFound.m.*

Problem 2.2, Particle swarm optimization

Using the *contour* command in Matlab we plot the function log(0.01 + *f* (x, y)) :



Thus we can determine from the graph that the function has 4 minima, marked with red 'X's, over the range $(x,y) \in [-5,5]$. Running the program *PSO22.m* some times we get that the minima of the function and the corresponding function values are:

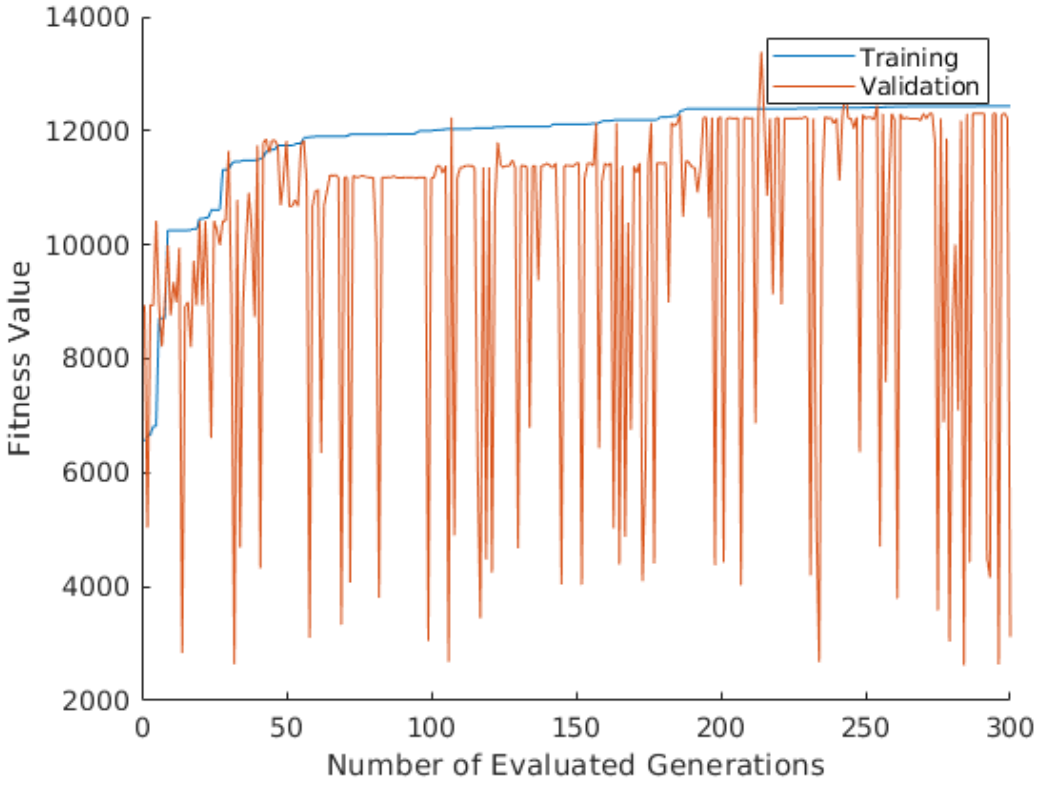
x	y	f(x,y)
3	2	0
3.584428	-1.848127	0
-3.779310	-3.283186	7.888609e-31
-2.805118	3.131313	7.888609e-31

Problem 2.3, Optimization of braking systems

The main script is *BS23.m*. The fitness measure for slope *i* is defined as:

$$F_i = \bar{u}_i d_i$$

where $\bar{u}_i$ is the average speed over the evaluation and d_i is the distance travelled before termination. The fitnesses of the best individual for both the training data nad the validation data during a run of the GA can be seen in the following figure:



From the plot we can see that fitnesses don't change drastically either for training data or validation data so there is no overfitting.