Second homework assignment

Jurek Eisinger *

Stochastic optimization algorithms

October 9, 2023
Student ID (GU): 8508561538

*jurek.jonathan.eisinger@vub.be

Problem 2.1 - The travelling salesman problem

Find the shortest path for the Travelling Salesman Problem (TSP) using Ant Colony Optimization (ACO).

The length of the best path found was 93.9754 lenght units long. It is depicted in figure **??**.

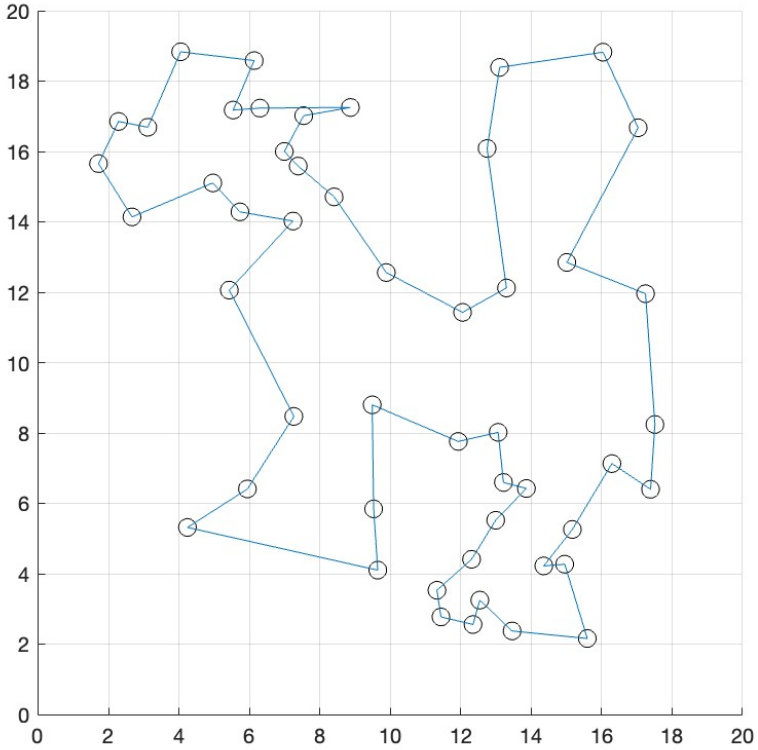

Figure 1: Best path found with ACO for the TSP

Problem 2.2 - Particle swarm optimization

Implement a PSO algorithm and with this algorithm, find all the minima of the following function:

$$f(x_1, x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$$

over the range $(x_1, x_2) \in [-5, 5]$.

The contourplot that was used to make out the rough positions of the minima is shown in the following figure.

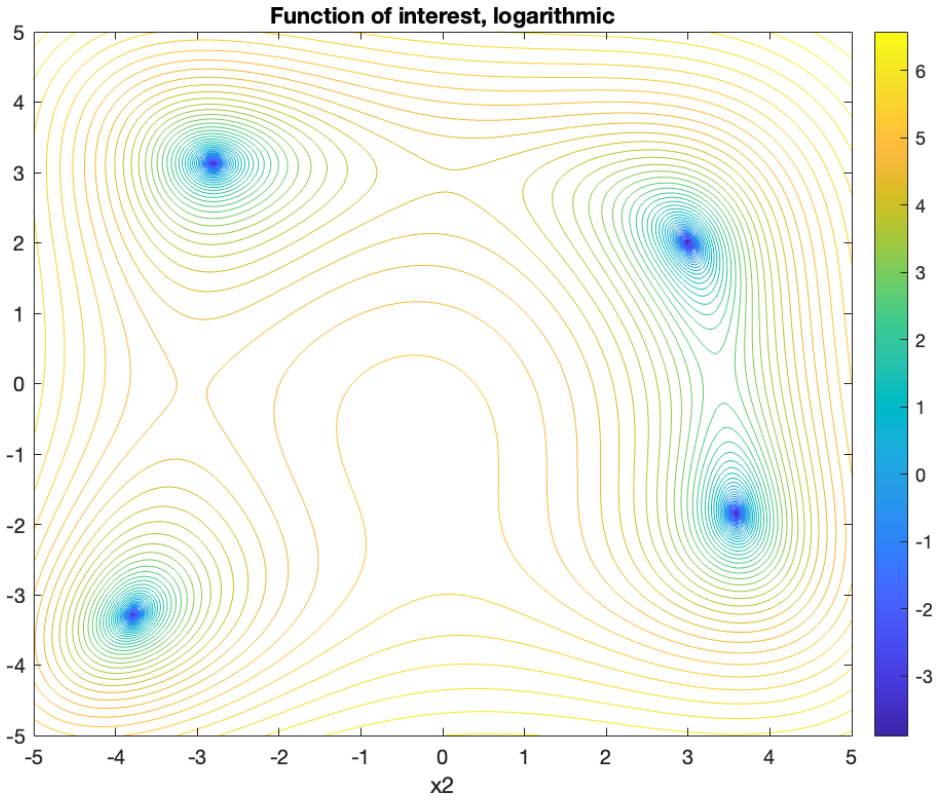

Figure 2: Enter Caption

The table containing the values for all four minimas found using PSO is:

x_1	x_2	$f(x_1, x_2)$
3.5844	-1.8481	0
3	2	0
-2.8051	3.1313	7.8886×10^{-31}
-3.7793	-3.2832	7.8886×10^{-31}

Problem 2.3 - Optimization of braking system

We will approach the optimization of the braking system with a Neural Network. The parameters of the Neural Network will be determined by a GA that we shall implement in this exercise. The NN will have three inputs:

- v
- α
- T_b

and two outputs:

- P_P
- Δ_{gear} - choice whether to in- or decrease gear, gear changes limited to one per two seconds

The evaluation of the Neural network with the weights we created will be done using a fitness function that looks like:

$$F_i = \bar{v}_i d_i \tag{1}$$

where it is clear that the fitness is proportional to the average speed of the truck, and d_i is the covered distance. A crucial step for understanding this algorithm is that we evaluate the neural network at a certain step in time. And at every step in time, the neural network tells us how to adjust P_P and how to shift. Then - using these new values for P_P and the new F_{eb} (because we are in a new gear) - we evaluate $v(t)$ again.

Step 2: Truck model

We want to implement the truck model. This means that we want to solve this equation:

$$M\dot{v} = F_g - F_b - F_{eb} \tag{2}$$

For every step in time, we evaluate the derivative of the velocity as:

$$\dot{v} \approx \frac{v(t + \Delta t) - v(t)}{\Delta t} \tag{3}$$

After this evaluation, we have all the values to plug into our neural network - T_b, α and now also v.

Step 3: Optimisation program

The evaluation of the neural network simply amouts to multiplying the weight matrices with the corresponding input vectors, applying the sigmoid function to all the elements in the vector and then again, multiplying the second weight matrix w_{HO} to this new vector corresponding to the values of the nodes of the hidden layer. We apply the

Step 4: Test and run the program

Due to no time left, I was not able to determine why the program stopped after a distance of 6m for every run. The results are not really interpretable.
However, I think that it is not missing much to work.

Problem 2.4 - Function fitting using Linear Genetic Programming

Use linear genetic programming (LGP) to find the parameters of a function $g(x)$.

We define a LGP programme with M variable registers and N constant registers. The operator set is $\{+, -, \times, /\}$

Essentially, we are doing something that we have done before - we execute a GA. But this time, the evaluation looks a little different:
We set only the first spot in the variable register equal to the value of x_k. The remaining $M - 1$ spots are set to zero:

$$\mathcal{R}_{\text{var}} = [x_k, 0, ..., 0] \qquad \mathcal{R}_{\text{const}} = [c_1, ..., c_N] \tag{4}$$

And then we follow the instructions given by the chromosome. The obtained value after the instructions have been executed is approximating the corresponding y_k, which we will denote $\hat{y_k}$. This procedure we do for all k data points. Then we form the error over all these. Now, our model has been evaluated. This we do for all the individuals.

The procedure that is done is actually quite simple and very close to the procedure of a GA.

- Initialize population randomly - *take into account the varying ranges for all the parts of the chromosome*
- Assign fitness values to individuals - *procedure described above*
- Tournament selection
- Mutation - *assign random value in allowed range to some value in the chromosome - with certain probability*
- Crossover - *non length-preserving and only at instructions*
- Proceed until function is found

I used three constant and three variable registers, so that $M = N = 3$. The values of the elements in the constant register were set to:

$$c_1 = 1, \quad c_2 = 3, \quad c_3 = -1$$

The values found by the program are:
Best cost: 3.5113×10^8 which corresponds to an error of 2.848×10^{-9}. The true function is depicted in figure **??** and the approximated function is displayed in figure **??** As we can see, these two functions look completely alike. That makes sense, given that the error is very small. We obtain the final function to be:

$$g(x) = \frac{x * (x^2 - x) + 1}{x * (x * x - x) + 1 + x^2 * (x^2 - x)} \tag{5}$$

which we can simplify to:

$$g(x) = \frac{x^3 - x^2 + 1}{x^4 - x^2 + 1} \tag{6}$$

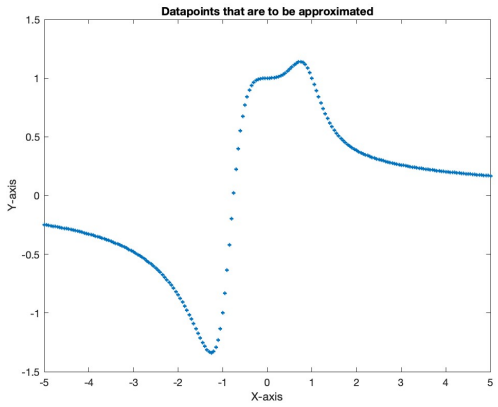

Figure 3: True function

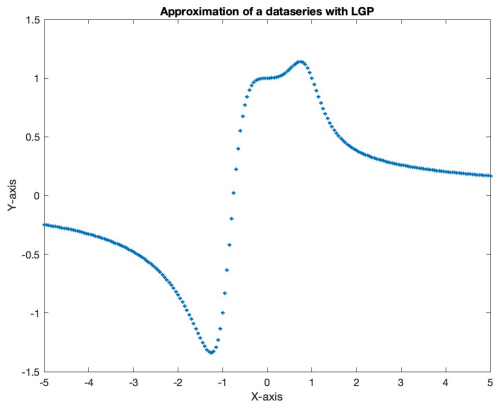

Figure 4: Approximated function using LGP

So that we find $a_0 = 1$, $a_1 = 0$, $a_2 = -1$, $a_3 = 1$ and $b_0 = 1$, $b_1 = 0$, $b_2 = -1$, $b_3 = 0$, $b_4 = 1$ and all other parameters are zero.