FFR105 Stochastic Optimization: Problem Set 2 Report

John Tember
19980731-5255

October 16, 2024

Contents

1 Problem 2.1: The Traveling Salesman Problem (TSP)

In this problem, we implemented a Ant System (AS) algorithm to solve the Traveling Salesman Problem (TSP) problem. The following path was obtained, with a length of = 99.5401975891768.

```
bestPath = [47 34 42 28 48 33 38 45 27 39 44 40 37 49 30 36 8
16 21 19 7 14 22 17 24 23 4 5 18 1 13 10 9 2 12 6 15 3 11 20 29 50
32 25 26 31 43 35 41 46];
```

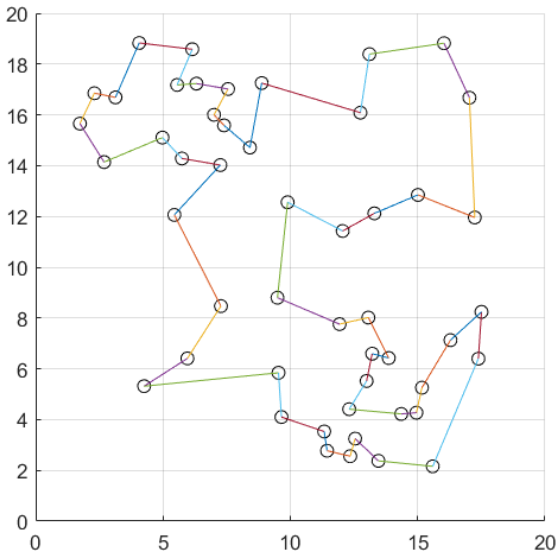

Figure 1: Plot of the best path found by the AS algorithm.

2 Problem 2.2: Particle Swarm Optimization (PSO)

In this problem, we implemented a Particle Swarm Optimization (PSO) algorithm to find the local minima of the function:

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

The following minima were found using the PSO algorithm, which are also reflected in the contour plot shown below.

x	y	$f(x, y)$
3.0	2.0	0.0
3.5844	-1.8481	0.0
-2.8051	3.1313	0.0
-3.7793	-3.2832	0.0

Table 1: Local minima and their corresponding function values, all numbers rounded at four decimals.

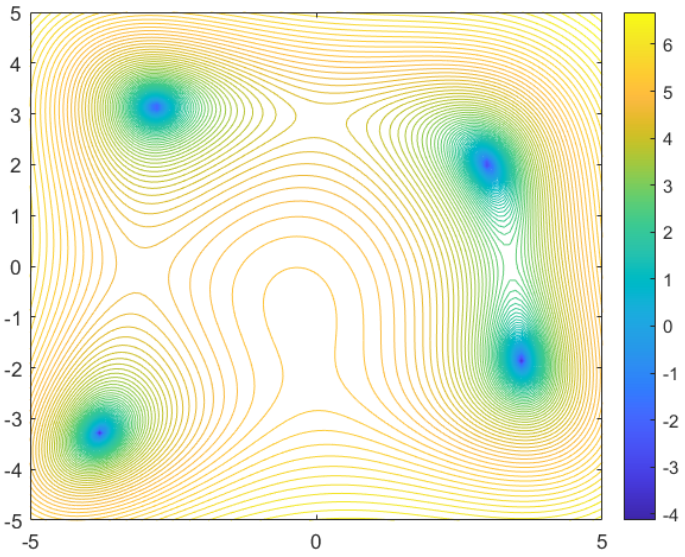

Figure 2: Contour plot of the function with local minima marked.

3 Problem 2.3: Optimization of Braking Systems (Voluntary)

3.1 Step 1: Encoding and Decoding

The first task involved encoding the provided neural network into a chromosome, and then decoding it back into the network's weights and biases. I discovered that each variable required a 32-bit sequence representation in the chromosome in order to pass the test. As a result, 32-bit binary sequences were used throughout the entire task to represent the weights and biases within the chromosomes.

The network dynamics were implemented using simple matrix multiplication, along with the sigmoid activation function, where the sigmoid constant was set to 2.

3.2 Step 2: Truck Model

The truck model was implemented as a function that simulates the truck running down a slope until either the truck reaches the end of the slope or violates any of the constraints. To ensure the correct calculation of the final state when the truck surpasses the slope's boundary, I implemented an additional function called `UpdateTimeDelta`. This function is called at most once, and adjusts the `timeDelta` to properly reflect the last state of the simulation.

3.3 Step 3: Optimization Program

My implementation included four main files:

- `GAOptimizer`: The main function that calls the `RunFFNNOptimization` function.
- `RunFFNNOptimization`: This function orchestrates the optimization process using genetic algorithms and holdout validation.
- `EvaluateIndividual`: This function takes a chromosome, decodes it into a neural network, and returns the fitness and plot data by running the simulation in `RunTruckSimulation`.
- `RunTruckSimulation`: Simulates the truck's behavior on the given slope using the decoded neural network.

For the truck's gear system, I implemented a cooldown mechanism with a duration of two seconds. The gear cooldown continuously decreases over time, and gears can only be shifted when the cooldown reaches zero.

3.4 Step 4: Holdout Validation

Holdout validation was implemented with a patience of 10 generations, meaning the optimization process was terminated if no improvement in validation fitness was observed for 10 consecutive generations.

3.5 Constraints and Fitness Function

The constraints ensured that the truck did not exceed the minimum or maximum velocity limits, nor overheat its braking system.

For the fitness function, I chose to calculate the product of the average velocity and the distance traveled along the x-axis. I also experimented with a fitness function that focused on the worst performance across all training slopes. The motivation behind this approach was to ensure that a chromosome would only be considered superior if it performed well on all the slopes. This contrasted with calculating the average fitness across all slopes, which could favor trucks that excel on easier slopes while neglecting the more challenging ones, leading to an inflated average fitness. However, the chromosomes derived from this alternative method did not perform as well on the test set slopes compared to those obtained from the original fitness function. As a result, I decided to retain the initial approach.

For each slope, the fitness was computed by multiplying the distance traveled before any constraints were violated by 10, and then multiplying the result by the average speed. This method prioritizes covering more distance over achieving higher speed. Thus, a truck that travels farther, even if at a slower pace, is considered more fit than a faster truck that covers less distance.

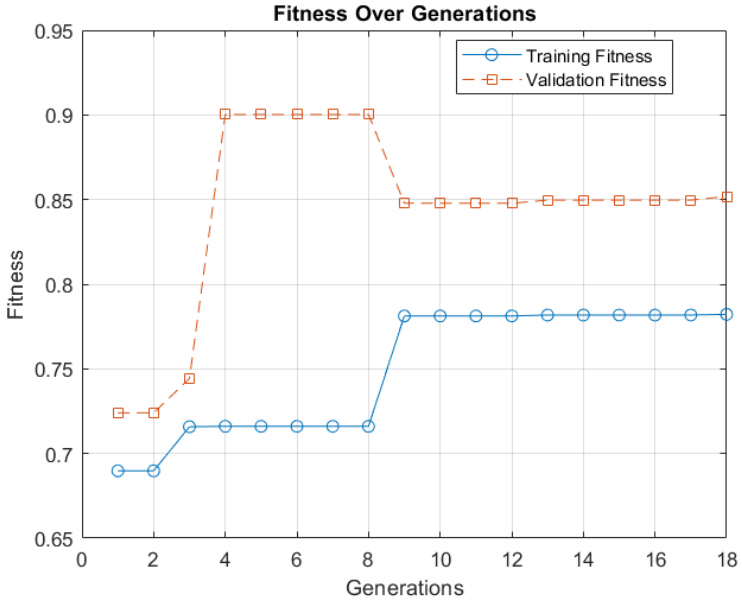

Figure 3: Fitness vs. generations: Best chromosome (blue) and best validation chromosome (orange).