

# Report of FFR105 HP2

Siyu Hu, gushusii@student.gu.se, ID: 19950910-3702

2024-10-14

## 1 HP2.1 TSP

The city index order of best path is contained in the **BestResultFound.m** and the path length is **99.79161**. Figure 1 shows the path:

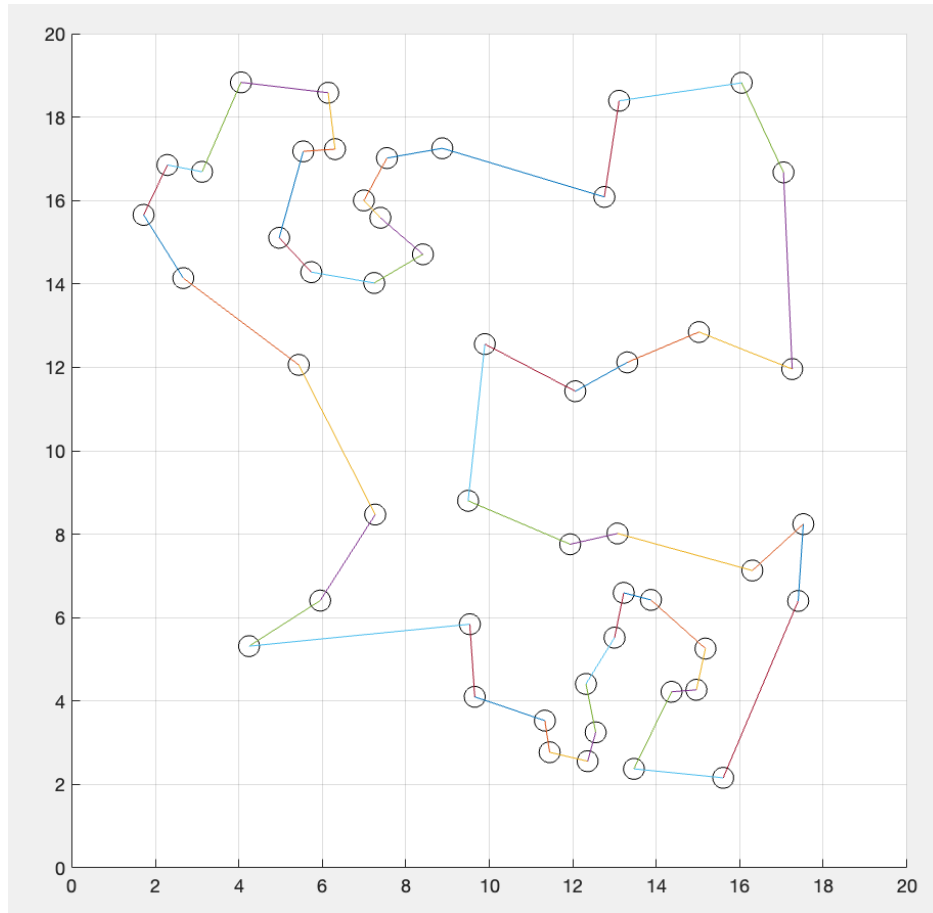


Figure 1: The best path of TSP

## 2 HP2.2 PSO

The position and value of minima shows in the Table 1 and Figure 2 is the objective function log contour with minima (red dot).

From the contour plot, it can be observed that the deeper the blue color, the smaller the value, indicating proximity to a local minimum. Table 1 lists four extremum points extracted from the program's output after removing duplicates. When their positions( $x_1, x_2$ ) are plotted on the contour, they fall in the areas where the blue is the deepest. This shows that the four extremum points found by the program are correct.

$x_1$	$x_2$	function value
-3.7793	-3.2832	$7.8886e-31$
3	2	0
3.5844	-1.8481	0
-2.8051	3.1313	$7.8886e-31$

Table 1: all minima and its function value

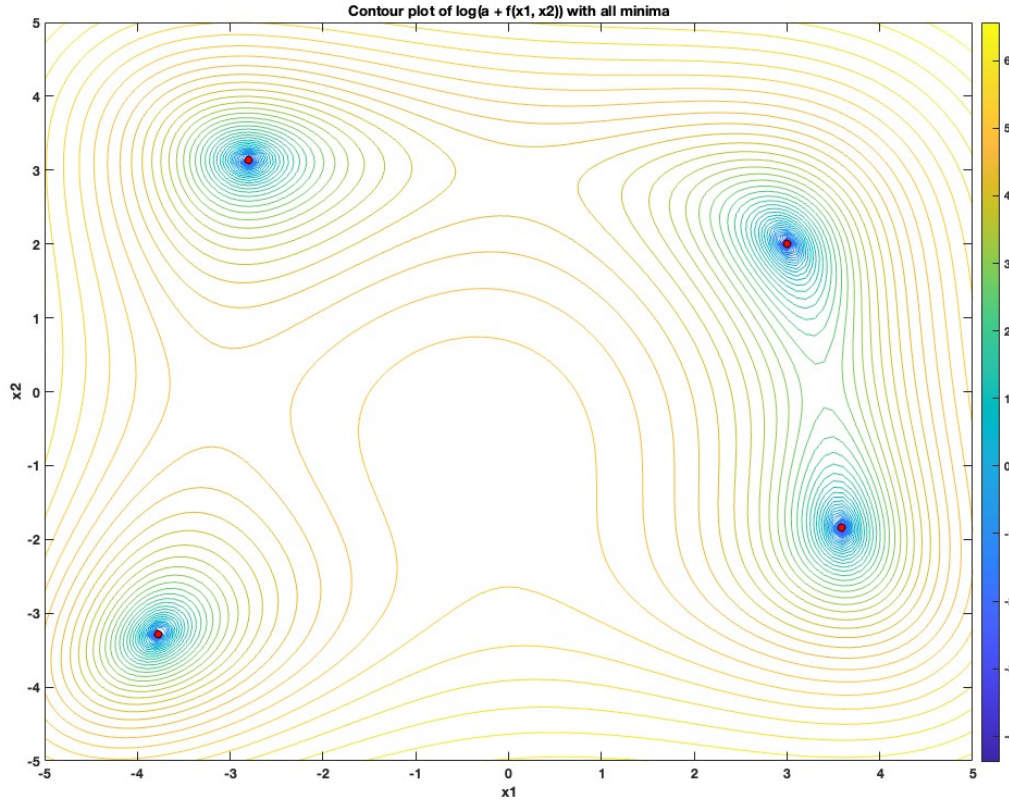


Figure 2: objective function log contour with minima(red dot)

### 3 HP2.3 Optimization of braking system

For HP2.3, this is a problem using a GA (Genetic Algorithm) to optimize an FFNN (Feedforward Neural Network) for truck brake system variables. In addition to the standard GA algorithm, I have constructed an FFNN model for the truck brake system. The key variables (constraints) are speed and brake pad temperature. I have built three datasets (training, validation, and test sets) to evaluate the performance of the optimization algorithm. *Due to the time constraints of the report, I will provide a brief exploration in the following section without deep verification and discussion.*

#### 3.1 Key implementations

The key settings I implemented are as follows:

1. In training phase, the *EvaluateIndividual.m* function of the GA algorithm returns **fitness value which is the average fitness of a single network (truck) over 10 training slopes**. The fitness calculation formula is:  $F_i = v_i d_i$ , where  $v_i$  is the average speed during the total running time of the truck, and  $d_i$  is the horizontal distance traveled during the total running time of the truck.
2. In each training generation  $i$ , I input the best chromosome of  $i$  th generation (has highest fitness value) into the validation , and calculate its **validation fitness value**, then store the training fitness and validation fitness of this chromosome in two separate arrays. When training stops, **plot this 2 arrays in Figure.5**.
3. For **early stopping strategy**, although I set the maximum number of generations to 500, considering the potential for overfitting during training, automatically stops training when the absolute difference between the validation fitness values of the **current generation** and the **previous generation** is less than 0.1 for 30 consecutive generations:

```

if generation >= 2
    if abs(validationFitnessHistory(generation) -
        ... validationFitnessHistory(generation-1)) <= 0.1
        stagnationCount = stagnationCount + 1;
    else
        stagnationCount = 0;
    end
end
...
if stagnationCount >= 30
    break;
end

```

4. As training stops, the best chromosome array is saved ( "number of generations" rows and "length of chromosomes" columns) .

### 3.2 Analysis

As seen in Figure 3, during the **first 38 generations** of training, the validation fitness shows significant fluctuations, while the training fitness maintains an upward trend. However, after around the 80th generation, the validation values **barely fluctuate**. It implies that the performance of the best chromosome obtained during training has stalled on the validation set, and further training will not improve the performance of the FFNN network on the validation set, i.e., the training has reached an overfitting state. Therefore, to achieve both high training fitness and validation fitness, I selected the **best chromosome from the 80th generation(its training fitness is 21170.6)** as the chromosome for testing.

The test results are shown in Figure 4.

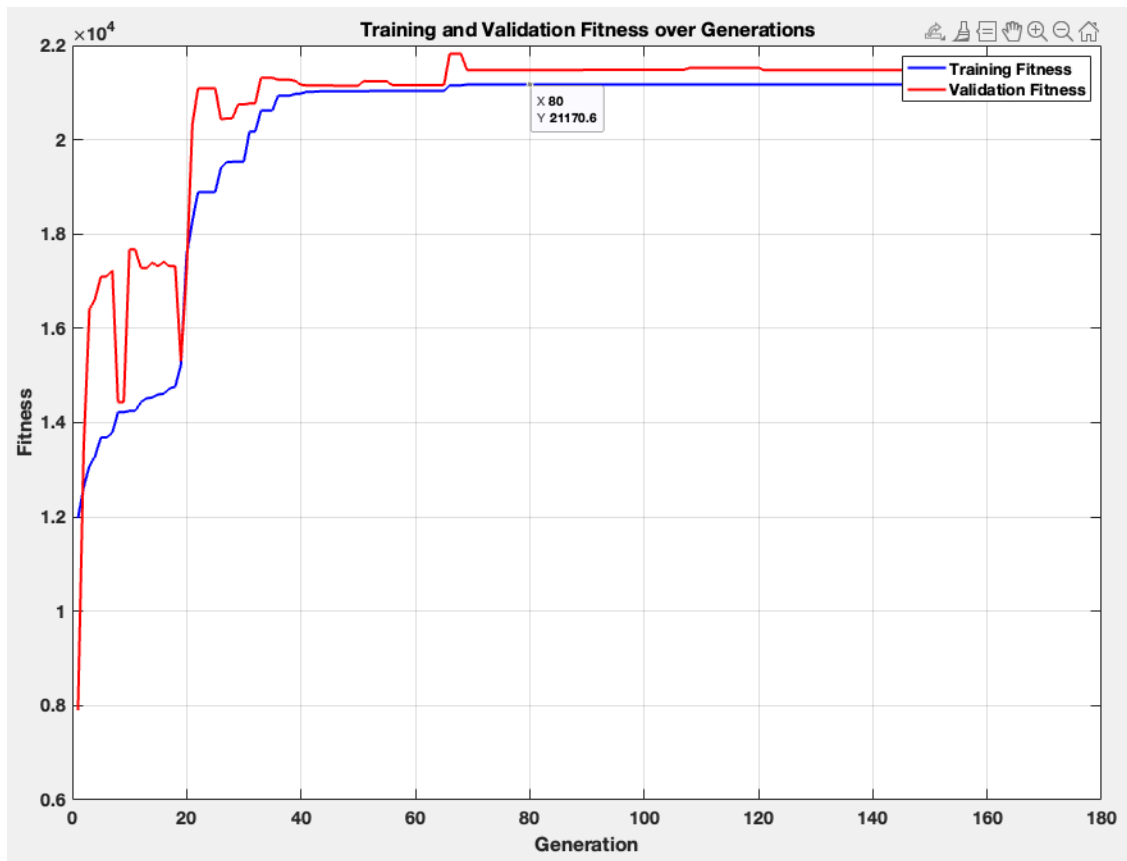


Figure 3: Training fitness(blue line) and validation fitness (red line) with generations

In Figure , it can be seen that in the **third slope of the test set**, the best chromosome (truck) did not complete the entire distance (max distance = 1000), but stopped at 250-300 meters due to velocity over 25 m/s. However, in the other four slopes, the truck successfully completed the entire distance without violating any constraint conditions.

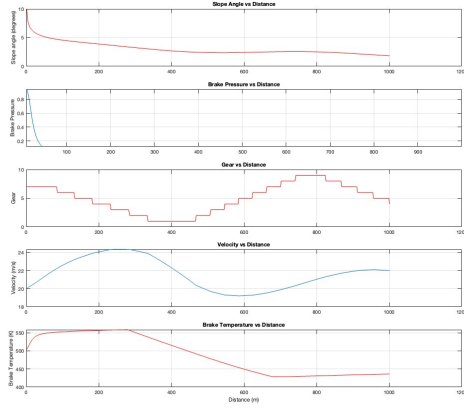
I adjusted the construction of the training set, validation set, and test set multiple times and in the final test set, the log and tanh functions, which did not appear in the training set, were added, but this phenomenon persisted—the best chromosome was not always able to complete the test set (within the speed and brake temperature constraints).

In my personally opinion, this issue may be related to the scale and quality of the dataset:

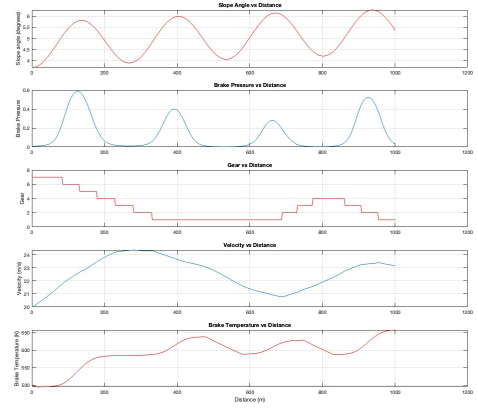
- If the scale of the training set is increased, it might allow the population to be more adequately trained (allowing the truck to train on more diverse slopes), which could enhance the generalization ability of the best chromosome.
- If the validation set and the test set have more similar distributions, it could help us better identify the critical point of over-fitting and prevent excessive training, which would lead to a loss in the generalization ability of the best chromosome, thereby improving its probability of completing the 1000-meter distance in the test set.

Besides, there is still room for improvement in the EA algorithm, such as parameter settings, crossover, and mutation strategies might further improve the training results.

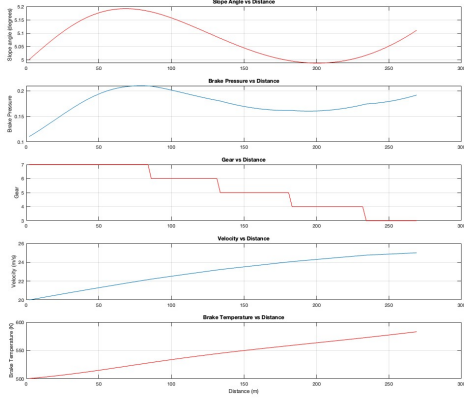
NOTE: These are hypotheses without definitive conclusions, and further exploration is needed. However, due to the time constraints of the assignment, further discussion is not possible at this moment.



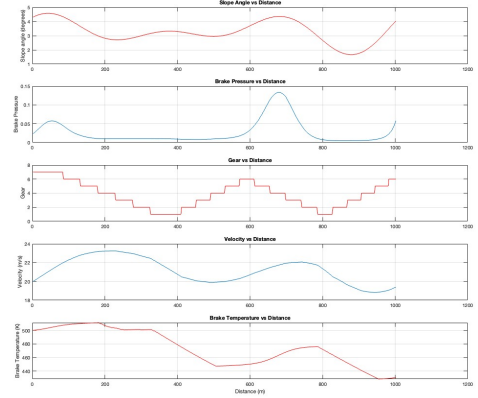
(a) Test set: slope 1



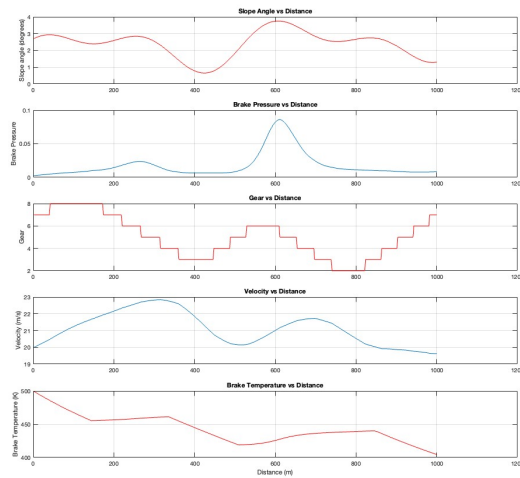
(b) Test set: slope 2



(c) Test set: slope 3



(d) Test set: slope 4



(e) Test set: slope 5

Figure 4: Run RunTest.m with BestChromosome.m, subplots is(up to down) Slope angle, Brake pressure(Pp), Gear, Velocity, Brake temperature(Tb) , and all the horizontal is travel distance.

## 4 HP2.4 Function fitting using LGP

### 4.1 Key implements

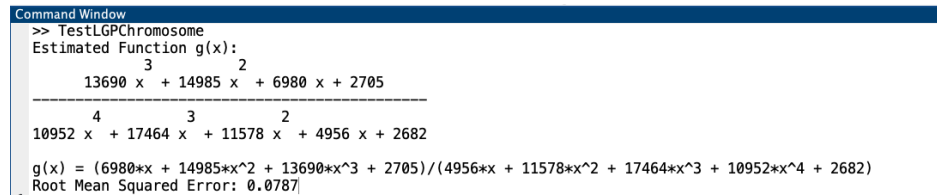
- Variable Registers: this program uses three variable registers (r1, r2, r3).
- Constant Registers: The following constants were used in the constant register pool: [ -10, -1, 0, 0.5, 1 ]. These constants were chosen to ensure that the evolved functions had a broad range of potential constant terms, strengthening the ability to fit both steep and flat regions of the target data.
- To prevent the Chromosome Length from becoming excessively long during iterations, which could affect computational efficiency and lead the model to premature convergence to local optima, a penalty coefficient *lengthPenalty* = 0.95 was introduced. After crossover, if the length exceeds 500 (maximum allowed length is 800) , the chromosome's fitness is adjusted as  $fitness = fitness * lengthPenalty$

### 4.2 Conclusion

Best chromosome saved in the **BestChromosome.m** file. Its fitness is **12.704374**, length is **156**.

Run the **TestLGPChromosome.m**, we can get **Root Mean Squared Error(RMSE)** which is **0.0787** and the best fit function  $g(x)$  :

$$g(x) = \frac{6980 * x + 14985 * x^2 + 13690 * x^3 + 2705}{4956 * x + 11578 * x^2 + 17464 * x^3 + 10952 * x^4 + 2682};$$



```

Command Window
>> TestLGPChromosome
Estimated Function g(x):
      3      2
13690 x  + 14985 x  + 6980 x + 2705
-----
      4      3      2
10952 x  + 17464 x  + 11578 x  + 4956 x + 2682
g(x) = (6980*x + 14985*x^2 + 13690*x^3 + 2705)/(4956*x + 11578*x^2 + 17464*x^3 + 10952*x^4 + 2682)
Root Mean Squared Error: 0.0787

```

Figure 5: Expression of best fit function  $g(x)$

**Figure 6** shows the curves of original data(blue) and the best fit function (red) :

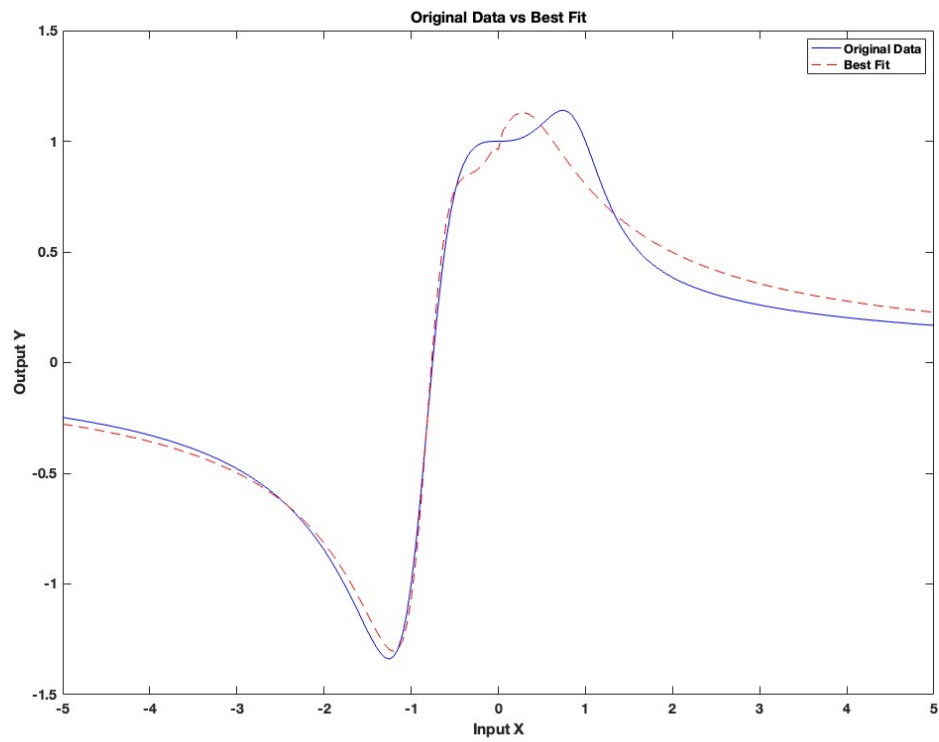


Figure 6: Original data (blue line) with best fit function  $g(x)$  (red line)