A comparison of Evolutionary algorithm operators using a Genetic Cars Simulation

By Jon

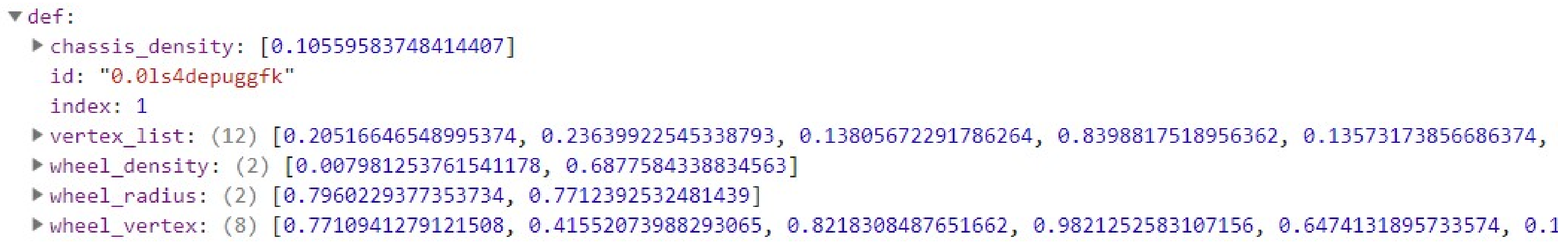
My Project

- My project is to create a Evolutionary Algorithm for a pre-existing framework.
- The pre-existing framework is Javescript simulation run on a internet browser. Example: https://rednuht.org/genetic_cars_2/

Car Data example

- A cars data consist of 5 Arrays with varying lengths. I treat the 5 arrays as the main datapoints.

- Car Data Example



```
▼def:
  ▶chassis_density: [0.10559583748414407]
   id: "0.0ls4depuggfk"
   index: 1
  ▶vertex_list: (12) [0.20516646548995374, 0.23639922545338793, 0.13805672291786264, 0.8398817518956362, 0.13573173856686374,
  ▶wheel_density: (2) [0.007981253761541178, 0.6877584338834563]
  ▶wheel_radius: (2) [0.7960229377353734, 0.7712392532481439]
  ▶wheel_vertex: (8) [0.7710941279121508, 0.41552073988293065, 0.8218308487651662, 0.9821252583107156, 0.6474131895733574, 0.1
```

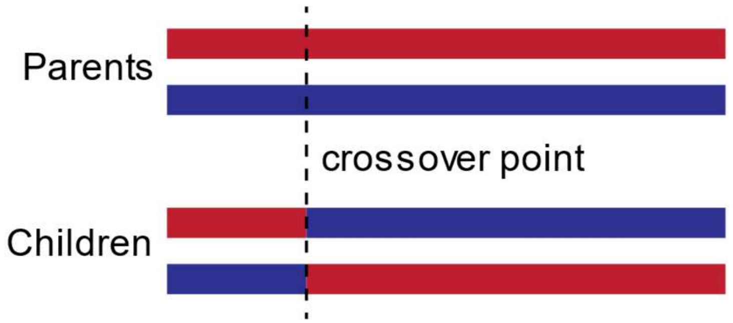
Selection operators Implementation

- Roulette-wheel selection (fitness proportionate), A cars fitness determines the chance of a car being chosen such as the fittest car having the highest percentage chance of being chosen compared to the weakest car.
- Tournament selection – A set of cars are randomly chosen from the population and put into another array, the fittest car in this new array is chosen as a parent.
- Uniform random selection – All cars in the population has a equal chance of being chosen.
- Tournament selection getting the strongest and weakest – Like tournament but 1 parent is the weakest car in the population and the other is the strongest.

Crossover operators implementation

- One-point crossover – only 1 point is chosen in the data
- Two-point crossover

Crossover example



Ref: R0oland (2013). OnePointCrossover. [image] Available at: https://commons.wikimedia.org/wiki/File:OnePointCrossover.svg [Accessed 6 Mar. 2019].

Mutation operators implementation

- 1 Mutation – only 1 mutation happens
- Multi-mutation – Multiple mutations effect a car depending on the criteria, the criteria what I’m using is that the bottom half of the population has 3 mutations instead of the normal 1.

Elitism implementation

- The car/s with the highest fitness in the population are passed through to the next generation without any mutation altering the cars data, It is also possible for Elitism to use other criteria to pass a car to the next generation.

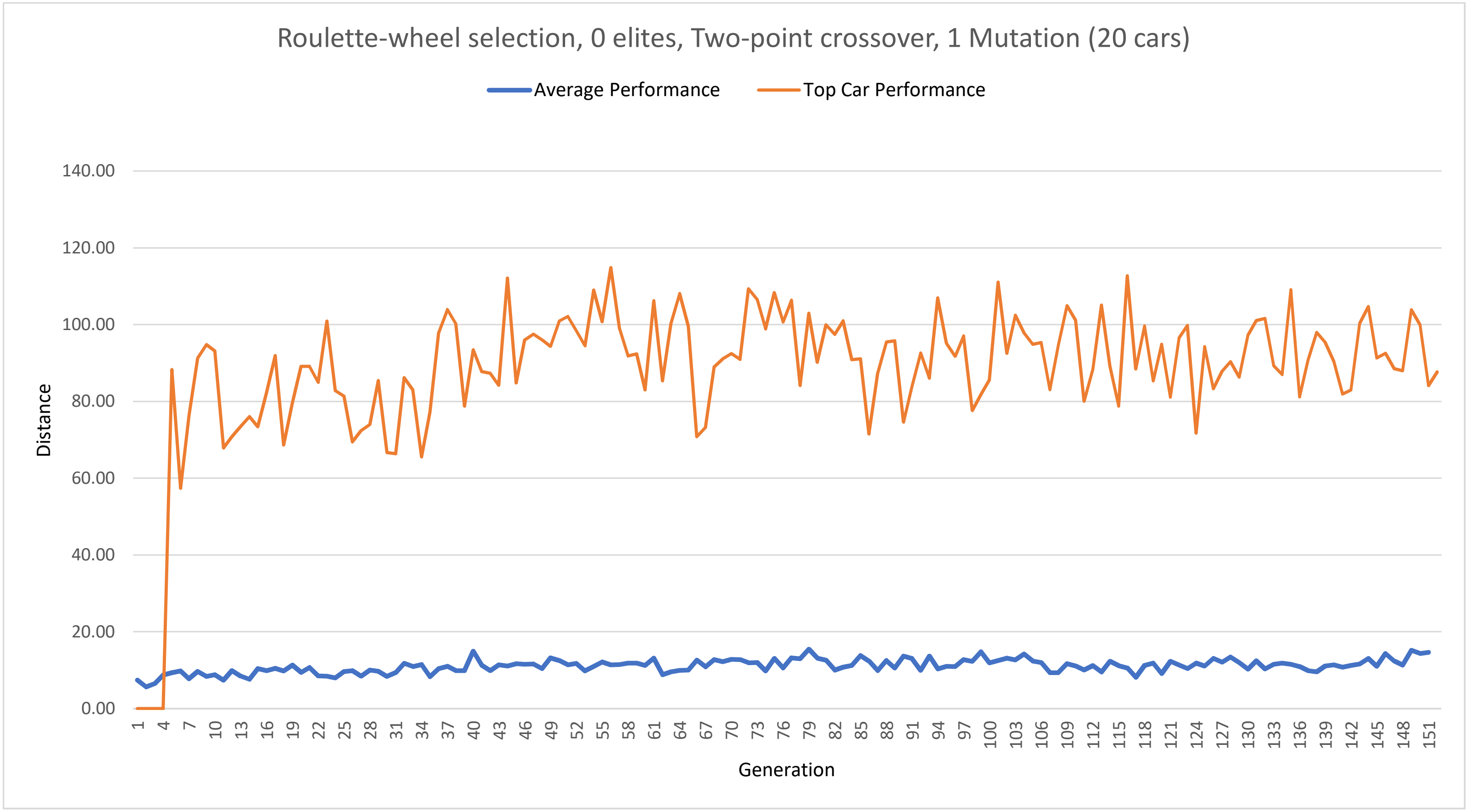
Cluster Re-scoring

- There are 5 clusters for a cars 5 data points. A cars data point is put into the clusters and the cluster is ordered highest to lowest so a cars data point will be surrounded by similar numbers.
- K-nearest neighbour is used to get the surrounding cars scores in a cluster, and the scores are averaged and added to the cars original score.

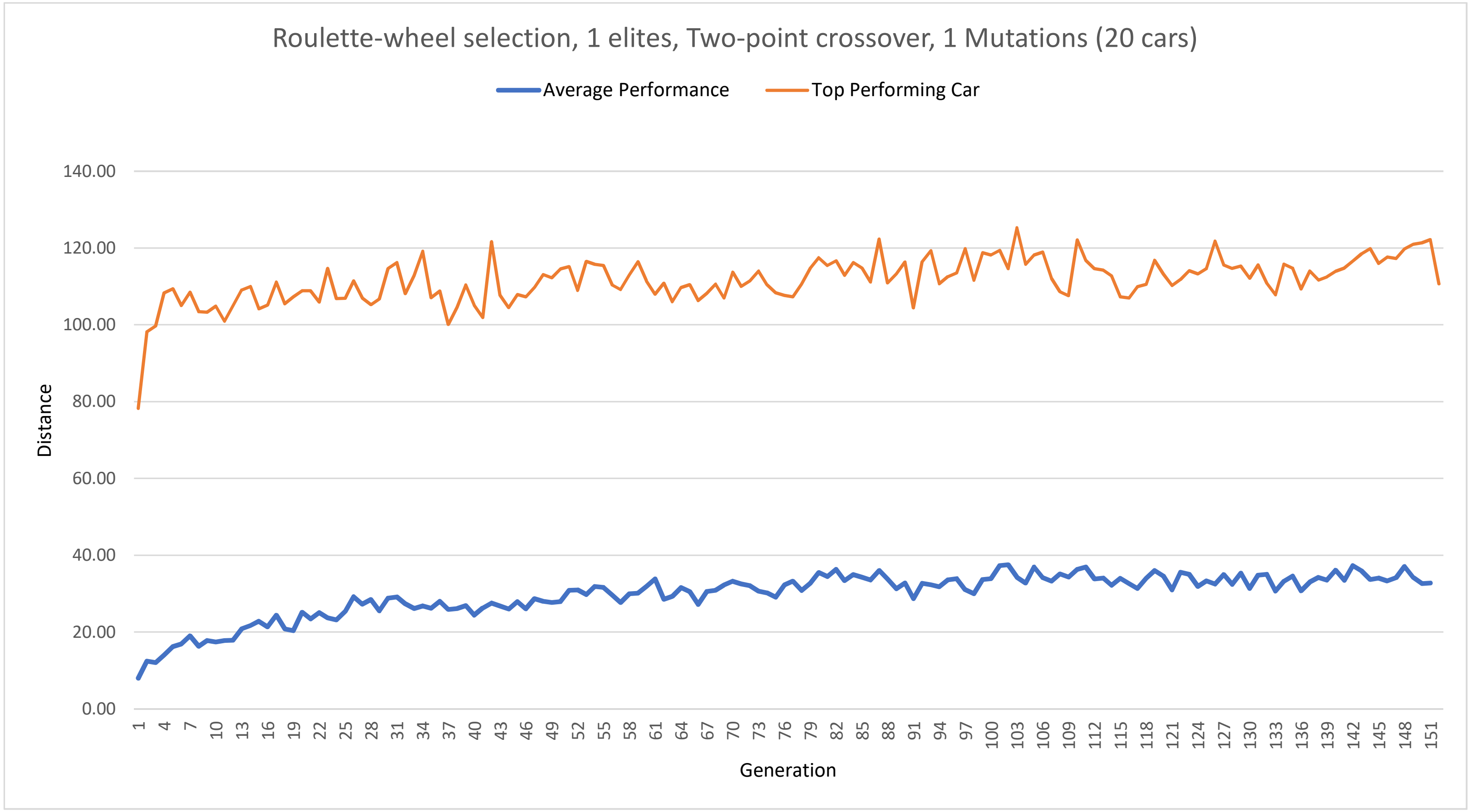
What I’m currently doing?

- Currently I’m creating test data running the simulation 10 times for 1 iteration of the Evolutionary algorithm which will be put into graphs to show the performance of the different Evolutionary algorithms operators.
- Running the tests 10 times is only preliminary but I plan on adding an extra 20 runs of the simulation to add data to back up the existing data.
- I’ve currently got 26 sets of data which takes an hour each to run.

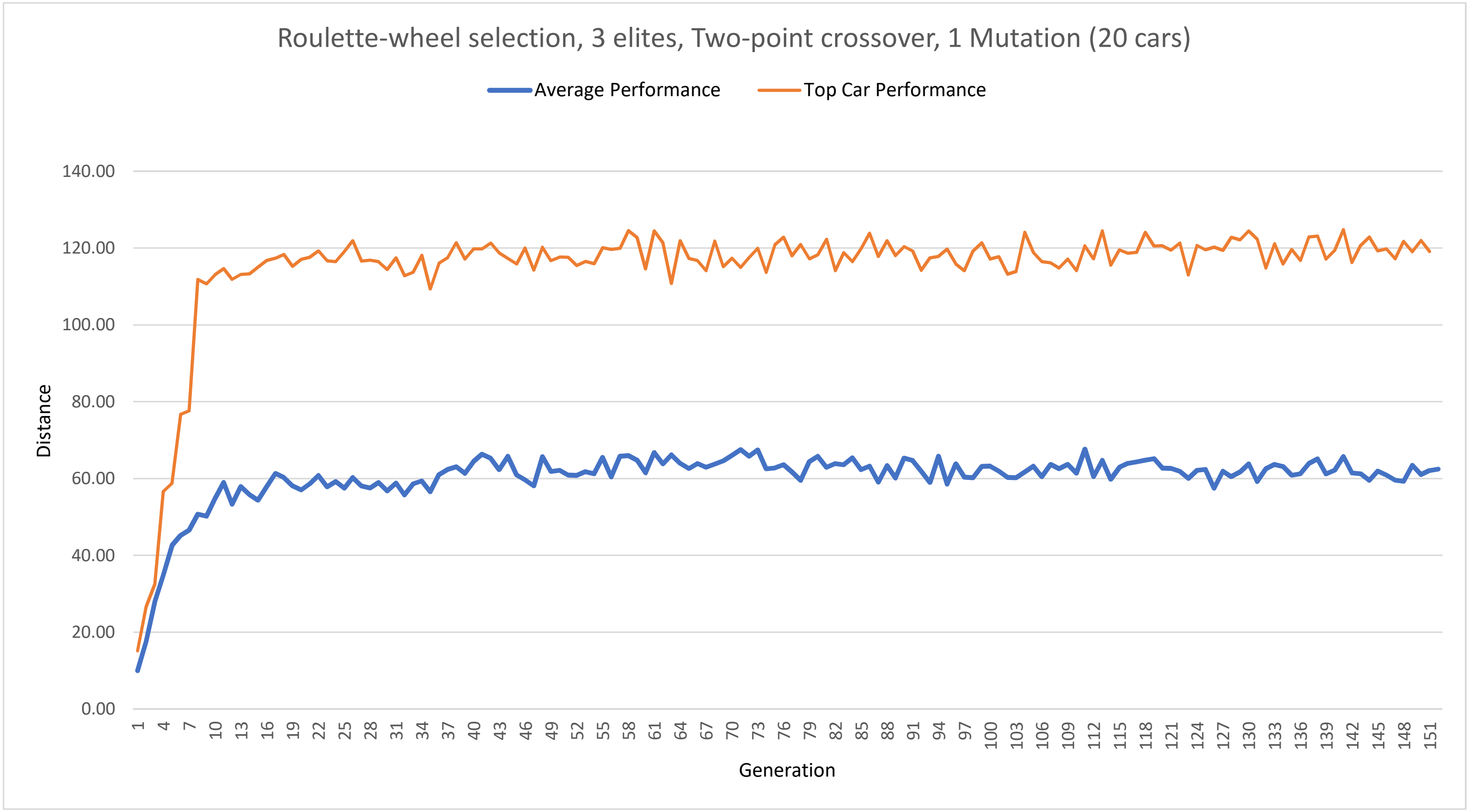
Baseline With no elites



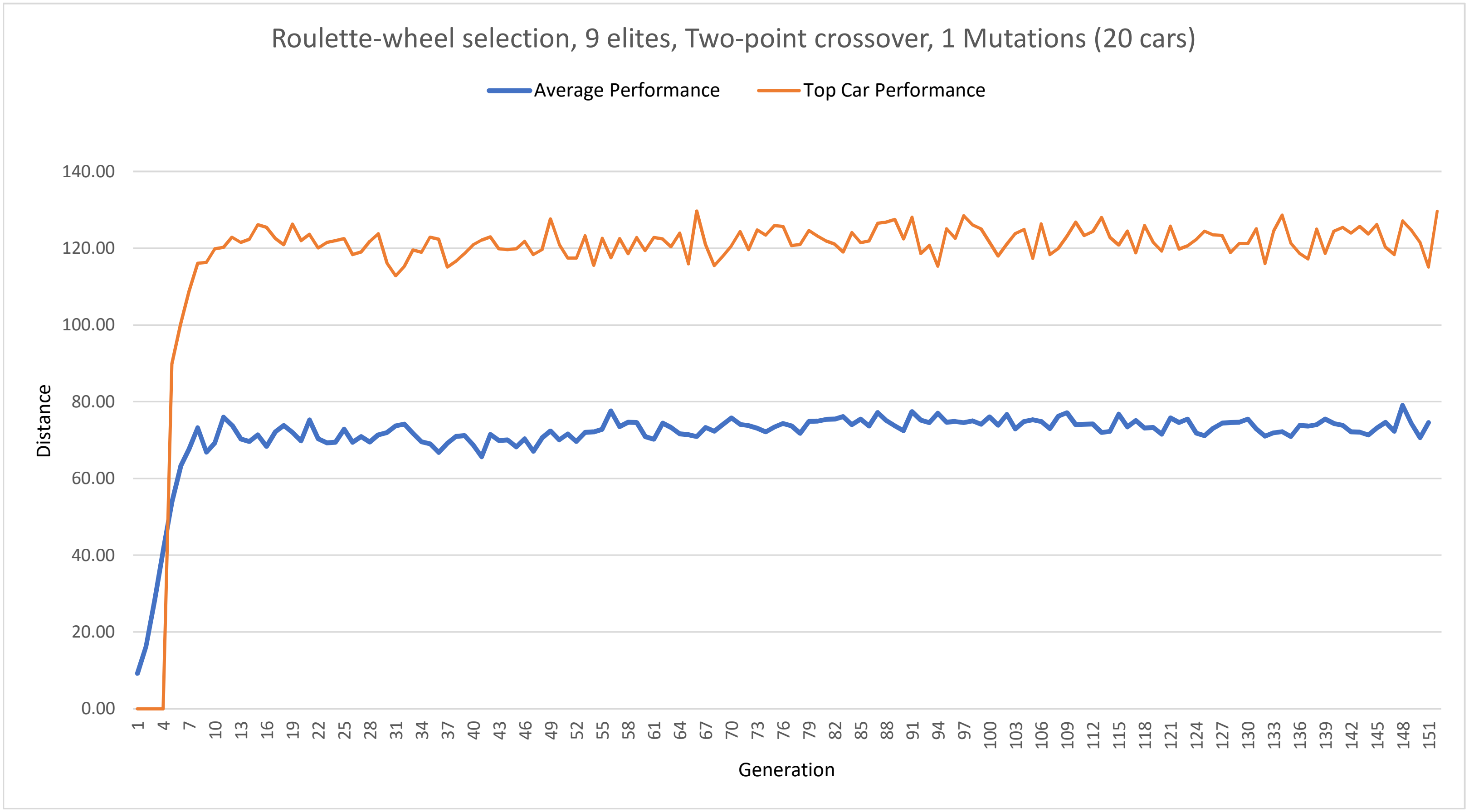
Examples Selection Pressure Increase



Examples Selection Pressure Increase



Examples Selection Pressure Increase


Roulette-wheel selection, 9 elites, Two-point crossover, 1 Mutations (20 cars)
Average Performance
Top Car Performance
Distance
Generation