# Combining Genetic Algorithms with Physical Simulation

Guillaume Labranche

COMP-521 Final Project

School of Computer Science

McGill University

Montreal, Quebec

2015-04-14

TABLE OF CONTENTS

## 1 Introduction

The main objectives were to make a computer find the best configuration for a car over an accidented terrain without prior knowledge of design strategies. In order to achieve such a result, I used evolutionary algorithms based on genetics to find the best configuration by trial and error. [1]

## 2 Background

Evolutionary algorithms have many advantages: [3]

- They are modelled after nature – and nature is ahead of us in terms of engineering.
- It allows for entertaining visualizations. Seeing the algorithm do its work, and vehicles improve as it works through the generations is nice.
- Not much engineering of our own is needed. We just have to program the algorithm properly, and it will work through any challenge thrown at it. [2]

A nice property is their modularity, similar to compilers that are divided into distinct phases.

## 3 Methodology

### 3.1 Fitness Measurement

Measuring the fitness of an individual was made through simulating a physics-based environment in which certain capabilities would need to be optimized in order to success. I decided to represent my individuals as vehicles with an arbitrary shape and placement of wheels. The goal for them was to travel over a rough terrain (to add complexity, since simply a vehicle going over flat land is easy to create). The fitness measure was then how far they were able to travel before stalling.

### 3.1.1 Vehicles

The vehicles consist of a set of 8 vertices all randomly distributed around a center point. This defined the vehicle's chassis (frame) on which to attach wheels. Each vertex has a probability (20% in practice) of having a wheel at its tip. The vehicle's weight is also calculated from the triangles making up its `Mesh`. It is then set as a property of the `RigidBody2D` to create more realistic simulations (heavy vehicles have a hard time going uphill but are more stable). Each wheel also has a weight dependent on its radius ($\pi r^2$).

### 3.1.2 Terrain

I initially tried to use a well-made Unity plug-in but it had not been updated in a while and Unity itself had upgraded two versions since. I therefore wrote my own simple implementation. The process is quite simple: each new segment is based on the previous, and subject to some constraints. The constraints are namely:

- An absolute minimum/maximum slope throughout all the terrain.
- A maximum change of angle between connected segments.
- An easiness factor that affects the first 40 segments, where the above two constraints are multiplied by $i/40$ where $i$ is the index of the segment.

The third factor makes the terrain rather flat at the beginning, so that the evolutionary algorithms can optimize for basic movement before getting into more challenging terrain. Otherwise it would be very possible to have a very rough edge right at the start, and our first generation of vehicles would not be able to get over it. We would then not get any information out of this since everybody got a close fitness score.

The new segment's length is fixed for all. Its angle is then chosen randomly through a uniform distribution that is the maximum window satisfying the previously-mentioned constraints. It leads to a terrain that is quite natural-looking.

## 3.2 Evolution

### 3.2.1 Selection

This is the part that deals with selecting the best individuals in order to improve the gene pool. In essence, this algorithm uses the fitness factors computed with our physical simulation to take a subset of the population with a higher score average than that of the entire population. It then uses the selected set of individuals for "mating" or "reproducing". There are many ways of doing this, such as always simply selecting the best individuals, or choosing randomly a fixed number of individuals with a weighted probability distribution using the fitness scores. I chose to implement the technique called "tournament" [3] which goes like this: we select $k$ individuals from the population, and only keep the highest individual (some versions keep the top $t$ highest, where $t <= k$). We proceed until there are less than $k$ individuals in the population. This leads to $n/k$ individuals (where $n$ is the size of the gene pool) to be passed onto the crossover phase. An advantage of this technique is that there is a significant amount of randomness, which assures that we do not over over-select some properties at the beginning and end up in a local maxima.

### 3.2.2 Crossover

The crossover phase deals with mating two individuals and to create two off-spring who keep some properties of their parents. Basically, this takes care of keeping the good traits that the top-scoring selected individuals have, and even improve on

it. Again, many techniques have been developed for this. The main ones treat the chromosome (set of values that define an individual's traits) as a sequence of numeric values and split it in different ways, such that some parts are given to the first children and other parts to the other. In my implementation, I used the two-point crossover technique, which yielded good results right away. In my particular physical model, this proved to be efficient since the vertices defining the chassis of the vehicle are stored in counter-clockwise order. Therefore, if an individual has some part of his body that benefits him greatly (such as a concave section between two wheels), it is more likely to be part of a single segment and to be untouched and reproduced in one children.

### 3.2.3 Mutation

Mutation is the last step in the process of evolution. It deals with introducing random changes throughout the genome in order to provoke improvements (or handicaps) within all individuals. If it was not present, the algorithm would certainly optimize the initial configuration but the best performing individuals might not be exposed to unplanned changes which could benefit their performance. In the real evolution of species, it is the result of drastic changes in populations. In practice, this is done by setting a mutation probability and for every chromosome, assign a new random value to it if a random variable is under the probability.

## 4 Results

As a result of this project, vehicles are found to travel through the accidented terrain within a few generations. As we can see in Figure 1, the average increases steadily.

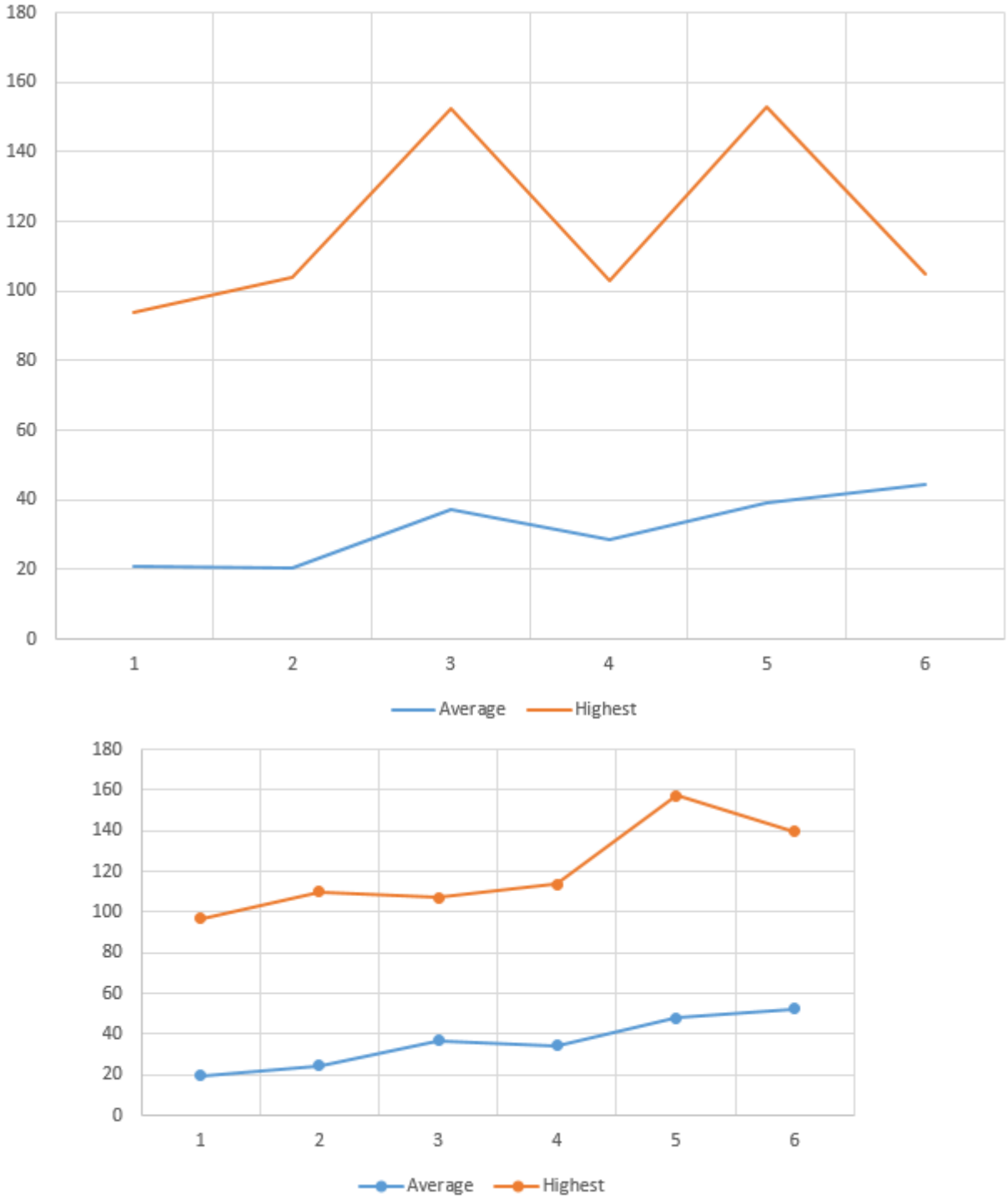


Figure 1: Scores per generation

## 5 Conclusion

In conclusion we see that evolutionary algorithms are both powerful and entertaining. They can adapt to a wide range of problems and require less implementation code than the custom algorithms adapted to the specific case.

We could extend to other forms of physical simulation. We could also add user interaction such a designing their own cars to try to compete against the algorithms.

References

[1] Thomas Geijtenbeek, Michiel van de Panne, and A. Frank van der Stappen. Flexible muscle-based locomotion for bipedal creatures. *ACM Transactions on Graphics*, 32(6), 2013.

[2] Insop Song Mehrdad Dianati and Mark Treiber. An introduction to genetic algorithms and evolution strategies.

[3] Brad L Miller and David E Goldberg. Genetic algorithms, tournament selection, and the effects of noise. *Complex Systems*, 9(3):193–212, 1995.