Genetic Programming in Swift for Human-competitive Evolution

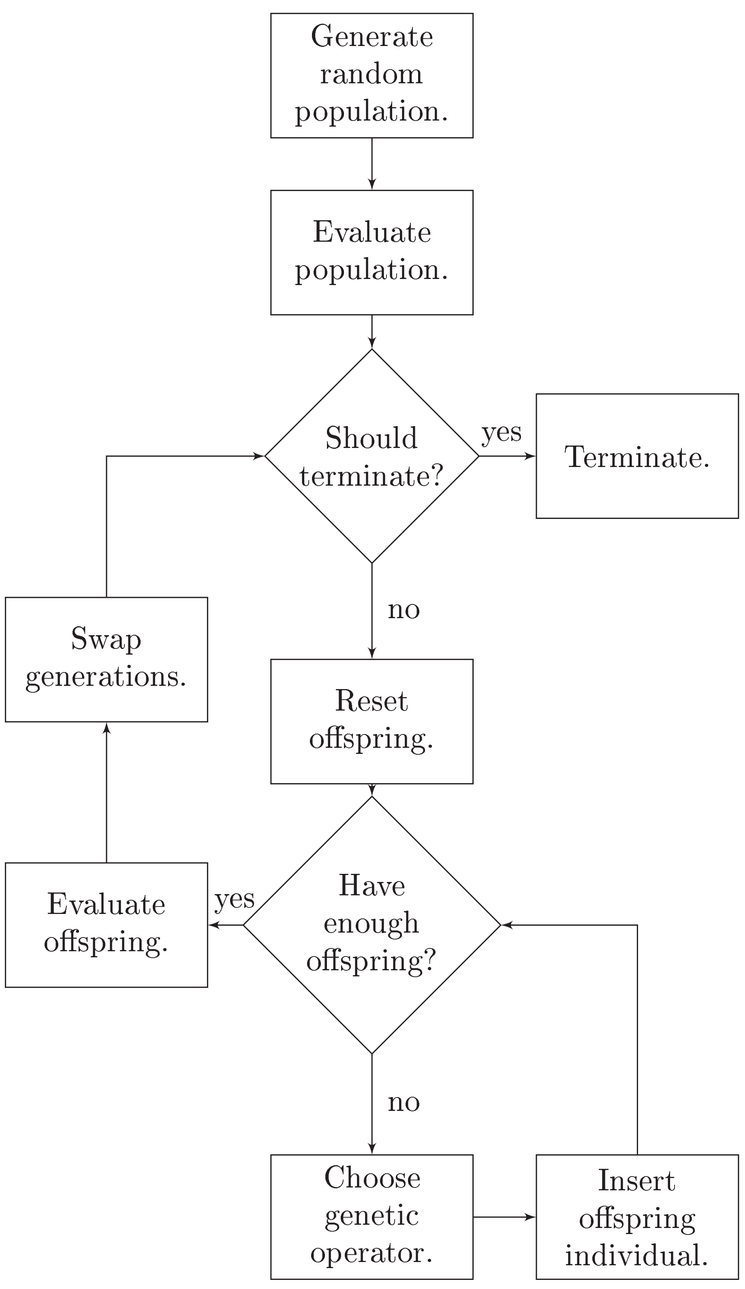
Petr Mánek, supervisor: František Mráz
Faculty of Mathematics and Physics, Charles University in Prague

Objectives

1. Implement a genetic programming library in the Swift programming language.
2. Demonstrate the usage of the library by applying it to sample problems.

Genetic Algorithms

Genetic algorithms (GA) are non-deterministic machine learning techniques inspired by natural processes. They have proven to be efficient solutions to some optimization and search problems in poorly-understood spaces. GA represent points of the domain space by individuals, which iteratively compete for their right to reproduce maximizing a set fitness function.



Swift Programming Language

The Swift programming language has been unveiled in 2014 by the Apple Corporation. Since then, it has been widely adopted by software developers and computer engineers, succeeding Objective-C as the main programming language used for application development on the Apple mobile device platform. Building on proven coding paradigms, such as generics and strongly-typed objects, Swift strives to be a modern, concise and safe alternative to popular languages like Python or C++ while attempting to maintain comparable performance in terms of computational speed and memory management.

Architecture

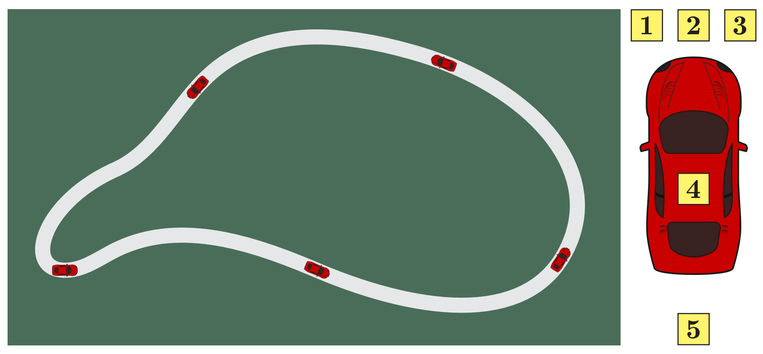
TODO

Properties

TODO

Self-driving Car Simulation

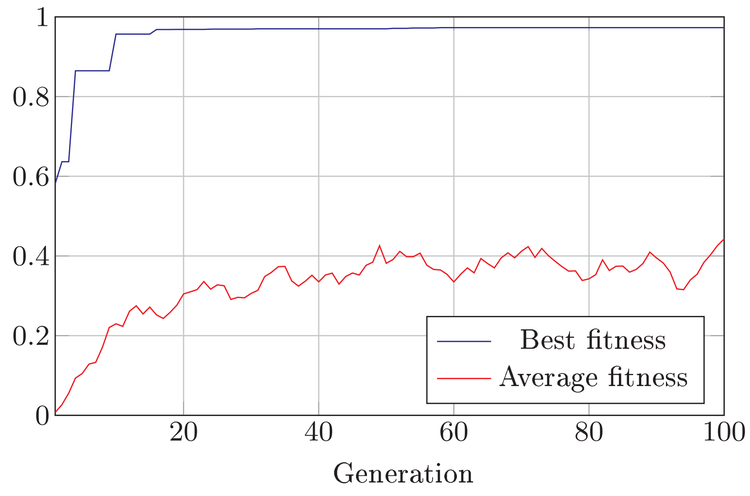
The presented library was used to evolve a control program for a self-driving car. The environment assumed Newtonian physics model (ignoring friction forces) and the parameters of the experimental car were modelled to match a real vehicle. The road was a randomly generated closed Bézier curve, which was detectable by 5 sensors positioned in the front, middle and the back of the car (see the picture below).



The car was controlled by a 3-layer feedforward neural network with 10 neurons in the hidden layer, whose connection weights were encoded in the genotype. Every program was evaluated in 5 independent 1-hour simulations, which tracked the total distance driven over the road. The fitness function was defined as

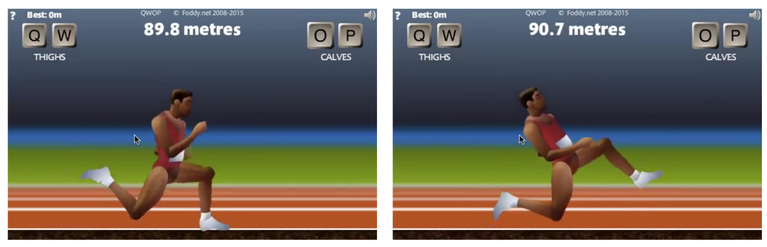
$$f(\hat{d}_1, \hat{d}_2, \ldots, \hat{d}_5) = \frac{1}{5d_{max}} \sum_{i=1}^{5} \hat{d}_i$$

where $\hat{d}_1, \hat{d}_2, \ldots, \hat{d}_5$ denote the total distances driven over the road in the individual simulations and d_{max} denotes the maximum achievable distance given the simulation parameters. The best control program after 100 generations was able to stay on the road in approximately 70% of simulations.



QWOP Player

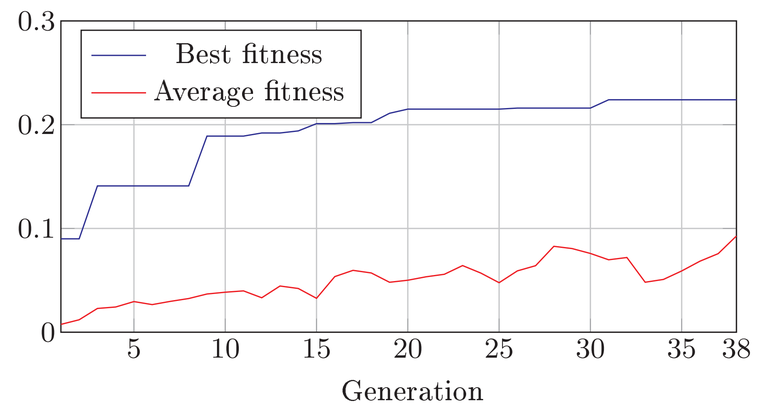
QWOP is a popular online game, in which the player drives an athlete to finish a 100-meter sprint race as fast as possible. QWOP's difficulty is caused by its control scheme, which only allows the player to move the athlete by contracting individual muscle groups within his body. The challenge of the game is in that sense comparable to the problem of evolving bipedal gaits in physical robots.



The presented library was used to evolve an artificial QWOP player and partially replicate human-competitive results achieved by **TODO**. In every generation, 80 game strategies were generated and encoded as simple programs (genotype strings), then evaluated by the fitness function

$$f(d_1, d_2, \ldots, d_n) = \frac{1}{100n} \sum_{i=1}^{n} d_i$$

where $d_1, d_2, \ldots, d_n$ denote the distances achieved in n trial 30-second runs. The best strategy after 38 generations was able to complete the race in approximately 152 seconds.



Conclusions

This thesis has managed to satisfy all of its objectives. The presented library is available online as an open-source project and has many potential applications.

References

TODO