AINT255 Option 1 Coursework

Lukas Ngo
Computing & Games Development
Plymouth University

Abstract

This paper examines the use of genetic algorithms to evolve parameters of a car in TORCS racing simulator. The goal is to find best values for the car parameters to achieve meaningful racing behaviour. This paper also discusses the technology behind evolution computation, how the genetic algorithms are implemented and presents experimental results obtained from using different approaches to genetic algorithms in the evolution process.

Related works section of this paper shows application of evolutionary algorithms in many different games, how they were used and the reason behind the usage of such technologies.

Results obtained from these experiments demonstrated the suitability of using evolutionary computation in parameter optimisation. Many different behaviours can be evolved to provide more diverse and appropriate car parameters for different tracks and settings in short amount of time.

Two experiments presented in this paper were designed to find out whether increasing the number of generation at the cost of lowering the population size can have a positive impact on the evolution. The final experiment was designed to find out what happens when best individuals in the population are prevented from mutation and the impact of this on the evolution.

1 Introduction

1.1 Technology Overview

Artificial intelligence is an important part of modern video games. This section focuses on the techniques behind evolution computation, genetic algorithms and neuroevolution in games. Many AI problems can be viewed as function optimisation problems that can be solved using an evolutionary approach. Neuroevolution only requires a measure of performance which could be a score of a computer game to train ANNs [4]

The common idea behind genetic algorithms is that computational problems can be encoded in such a way that evolutionary process can be applied to find an optimal solution [2]. Given a population of individuals, the environmental pressure causes natural selection and the fitness of the population is growing [5].

Necessary steps of the evolutionary process are understanding how to encode a problem, so it can be solved using genetic algorithms.

First step is to initialise a population of individuals. Each individual in the population is different to each other with various traits assigned to them either by random or some more informed process, however this more informed approach could create a population that lacks genetic diversity to find the best solution [2]. Each individual has potential to find a solution to the problem.

Next step is the evaluation process. The population is evaluated using a fitness function to determine which individual is more successful. The fitness function represents the requirements to adapt to and defines what improvement means. [3] Individuals closer to find the optimal solution have a higher fitness score. The fitness function must be designed to be very specific to the problem at hand and the overall success of the genetic algorithm is heavily dependant on the design of the fitness function [2]. In terms of games development, fitness function can be the score achieved in a game, distance travelled in a car simulator, kill count in a shooter game or some other measure of fitness.

Next step is selection. To ensure that evolution is taking place in a genetic algorithm, some of the better individuals from the population are selected to seed the next generation of individuals. One of the selection methods is stochastic roulette, where each potential parent is allocated a portion of the roulette wheel depending on their fitness. Fitter individuals have a bigger portion of the roulette wheel, therefore a bigger chance to be selected. [2] Some other selection methods are selecting parents based on their fitness ratio and depending on that fitness they're assigned a number which determines how many times they can produce offspring. Individuals with higher fitness can mate more than individuals with lower fitness. Another method is selecting a cut-off point based on fitness and individuals below that point are not allowed to mate. New individuals are created by applying crossover to two or more individuals (parents) to create new individuals (children) by exchanging the information between parents [5]. Weaker parents also have a chance to produce children but much lower than fitter individuals, otherwise the population could get stuck in a local optimum. Crossover using more that two parents is also possible and could have a positive effect on the evolution but is not commonly used [3].

Mutation is a process to insert random variation to new children in order to ensure the diversity of the population. It prevents stagnation and premature convergence in a population [2]. Premature convergence is a phenomenon where the population starts to converge in a very narrow region of the search space [5]. This can be prevented by using a higher level of mutation to introduce new and diverse individuals. Mutation is supposed to introduce a random unbiased change [3].

When new individuals are introduced into the population, depending on the genetic algorithms, they usually replace the weaker individuals from previous generation. Unlike the stochastic selection of parents, the individuals to be replaced are determined by their fitness and individuals with lowest fitness are the ones to be replaced. Usually the population size is constant and is not changing during the evolutionary process [3]. Once the new population is created, the process repeats until a termination condition is met.

Evolutionary algorithms are stochastic and there is no guarantee to find an optimal solution and therefore another condition is required to stop the evolution process, so it doesn't run forever. Common termination conditions are reaching the maximum allowed CPU elapses, reaching the maximum number of generations, fitness not improving in a given period of time, or the population diversity dropping below a given threshold [3]

Choosing the right input representations for ANN in games development can significantly influence the ability of the evolutionary process to learn autonomously and bias the evolutionary search. Choosing wrong or irrelevant inputs can harm the performance. [1]. Choosing different types of input can determine what strategies and behaviours will be evolved using ANN.

Neuroevolution has many attractive properties, however there are also reasons why it can cause problems to games developers. One of the problems is determining how the learned behaviour will work inside a game environment as it can be very difficult to debug learned behaviour [1].

1.2 Related work

Artificial intelligence in games is often evolved using evolutionary computation. The behaviours of computer-controlled characters in games are often guided by a large collection of parameters and it's often difficult to achieve desired behaviour [2]. Using evolutionary algorithms to evolve character behaviour has been used in many games including TORCS, Creatures, GAR or NERO, Return Fire II, Sigma, Spore [1,2].

Togelius and Lucas [6] conducted a series of experiments evolving controllers for a racing simulator using a number of inputs and outputs for steering, acceleration and braking. Using evolution, a number of different driving styles could be developed. Fitness function used in their experiment was determined by how far along the track the car has managed to drive in a given time. It was shown that these controllers can be evolved to drive on a variety of different tracks. Process of evolving controllers for different tracks involved incremental evolution approach, where each car was first evolved on one track and when the fitness of the population was high enough, more tracks were added to the evolution and fitness averaged [1]. It was shown that this incremental approach to evolution could develop more general driving skills and no such development was observed when evolving on all tracks at the same time [1].

Bullen and Katchabaw [2] developed a mutator module for Epic's Unreal Engine that allowed them to apply genetic algorithms to in-game bots and then use them in Unreal Tournament 2004 game. In their experiments they used bots to form their population and their various characteristics such as accuracy, alertness, aggression, jumpiness, strafe ability, combat style, reaction time, favourite weapon, retreat threshold, pickup threshold, and stakeout threshold. These characteristics have direct influence on decision making process of Unreal bots. A number of fitness functions was defined to evaluate bots' success in playing the game. Gross kills, deaths, net kills, and kill/death ratio are the fitness functions they developed but from their experiments it was unclear which of the fitness functions provided the best results.

The use of AI agents in games controlled using an evolved neural network can be financially and time less expensive than scripting the behaviour. Behaviour that is evolved rather than scripted by humans means that the evolved agent may create new gameplay tactics and strategies or find loopholes and bugs in games that a scripted agent would not find [7].

Stanley, Bryant and Miikkulainen [8] developed their video game NERO which evolves increasingly complex artificial neural networks in real time. NERO lets the player train a team of robots for combat. The evolution method used in their game allows individuals to evolve and improve during the game. They developed and introduced the real-time Neuroevolution of Augmenting Topologies which allows users to interact with evolving individuals. Using this method, they created a new game where the individuals adapt in real-time to player's actions. Each character in the game has a neural network that learns through experience and individuals who make more progress during the game have more children than the individuals who perform worse and eventually better individuals are evolved [7].

Video games have been used as a test bed for artificial intelligence for a long time and Evolution computing is being used in many games as evidenced by many researchers. Video games offer significant challenges to AI researchers and evolving character behaviour and strategies for video games is an important part of nearly every modern video game as it provides a more rewarding experience for the player [2,7].

2 Experiments and Results

For the purposes of this research several experiments ware conducted using genetic algorithms to optimise a set of parameters of a car in a simulated racing game TORCS.

For evaluation purposes, TORCS offers four performance measures, these include distance raced, top speed, damage and best lap. In the experiments described in this paper, it was decided to use a fitness function that evaluates distance raced – 2 * damage. The same fitness function was used across all experiments for results consistency.

Population consists of individuals with 22 parameters to evolve, with values assigned randomly to provide better diversity across the population. Some parameters can be hard coded but for this experiment it was decided to evolve all parameters.

Population size and number of generations are different across experiments to determine whether lower population size and higher number of generation yields better results.

Selection process is a different approach to elite selection. It takes 1/10 of the best individuals in the population for crossover with crossover probability of 0.7, then it replaces the worst individuals in the population with the children of the best individuals, then mutate the whole population with a chance of 1 / number of parameters, 22 in this case. Mutation magnitude is set to 0.1 for all experiments.

Shared parameters across all experiments:

- Fitness function = distance raced – (2 * damage)
- Parameters to evolve = 22 (all)
- Crossover probability = 0.7
- Mutation magnitude = 0.1
- Simulation time = 500

Experiment 0:

A control experiment was designed to be compared with other experiments. Settings of the genetic algorithm for this experiment:

- Population size = 100
- Number of generations = 20
- Evolution process = select 1/10 best individuals in the population for crossover, children replace the worst individuals, then mutate the population

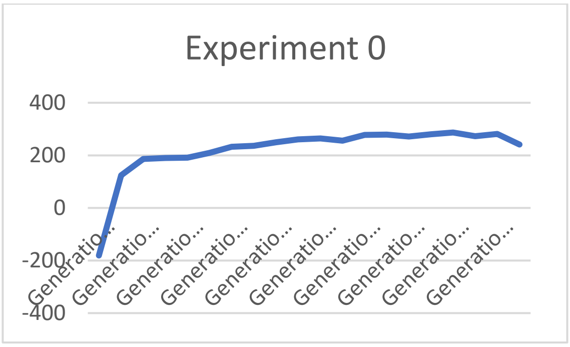

FIGURE 1

Observed characteristics of the evolved car: Car is bouncing off the road after any small jump and keeps locking wheels while bouncing. If the road remains flat without any jumps, eventually the car will drive well. Acceleration is poor.

- Oval Track: E-Speedway:
  - Fitness is: 3509.87
  - Top speed: 203.86
  - Distance raced: 3563.87
  - Damage: 27.00

Very chaotic car behaviour at the beginning of the race up until 20 second. After 20 seconds the car behaves well.

- Road Track: Alpine 1:
  - Fitness: 3097.07
  - Top speed: 140.80
  - Distance raced: 3097.07
  - Damage: 0

Car behaviour is better on this track than the oval track, car manages to drive without taking any damage and is driving well. This is most likely because the evolved car is driving and accelerating slowly which works well on this track.

- Dirt Track: Dirt 2:
    - Fitness: 1165.85
    - Top speed: 110.84
    - Distance raced: 1919.85
    - Damage: 377.00

Car behaviour on this track was very poor. Car kept bouncing of the track and locking wheels and took a substantial damage.

Experiment 1:

Compared to the control experiment, population size was increased from 100 to 140 and number of generations decreased from 20 to 14. Purpose of this experiment is to determine whether increasing the population size at the cost of lowering the number of generation can yield better results.

- Population size = 140
- Number of generations = 14
- Evolution process = same as control experiment

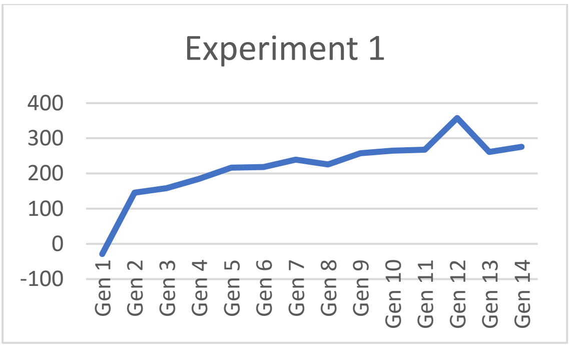

FIGURE 2

Observer characteristics of the evolved car: Acceleration is very high but upon reaching the top speed, the car slows down to about third of its maximum speed.

- Oval Track: E-Speedway:
    - Fitness is: 2410.27
    - Top speed: 208.75
    - Distance raced: 2410.27
    - Damage: 0

Started off well, then after a few seconds slowed down considerably for the remainder of the race.

- Road Track: Alpine 1:
    - Fitness: 3916.2
    - Top speed: 218.53
    - Distance raced: 3924.20
    - Damage: 4

Compared to the results of control experiment, this evolved car is doing much better on this track. It took some minor damage but overall is faster and raced a bigger distance.

- Dirt Track: Dirt 2:
    - Fitness: -14.26
    - Top speed: 103.23
    - Distance raced: 1213.74
    - Damage: 614

Car behaviour on this track was very poor is worse than car in control experiment. Car is not bouncing of the track anymore, but the acceleration is too high, causing the car to spin and crash several times.

Considering the performance on all 3 tracks, the car evolved in this experiment is worse than the one in the control experiment, this could be due to not having enough generations to evolve the car properly. From this observation it looks like that increased population size and decreased number of generations evolved a worse performing car, however without further experimentations on this configuration and deeper analysis to provide more evidence, this experiment alone cannot serve as a conclusive proof that more individuals and less generation produce worse results than less individuals and more generation albeit this experiment points in that direction.

Experiment 2:

Compared to the control experiment, population size was decreased from 100 to 60 and number of generations increased from 20 to 33. Purpose of this experiment is a continuation of experiment 1 to determine whether increasing the number of generations at the cost of lowering the population size can yield better results.

- Population size = 60
- Number of generations = 33
- Evolution process = same as control experiment

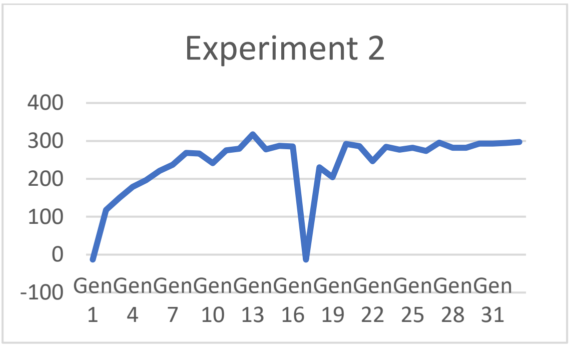

FIGURE 3

Observed characteristics of the evolved car: Acceleration is very high and so is the speed of the car, however it takes corners very sharply often causing it to spin and crash.

- Oval Track: E-Speedway:
    - Fitness is: 6073.07
    - Top speed: 265.30
    - Distance raced: 6073.07
    - Damage: 0

At the beginning of the race, the car started spinning for a few seconds but after that it performed much better than the control experiment and experiment 1. It managed to complete a lap which previous cars didn't.

- Road Track: Alpine 1:
    - Fitness: 3205.63
    - Top speed: 231.74
    - Distance raced: 3495.63
    - Damage: 145

Car is driving too fast for this track, not slowing down enough before taking sharp turns, resulting in very drifty behaviour in the corners and taking too much damage.

- Dirt Track: Dirt 2:
    - Fitness: 511.83
    - Top speed: 109.40
    - Distance raced: 1299.83
    - Damage: 394

Like experiment 1, car behaviour on this track was very poor and worse than car in control experiment. Acceleration and speed is too high, causing the car to spin and crash several times.

Considering the performance on all 3 tracks, the car evolved in this experiment is better than control experiment and experiment 1. This was probably achieved by letting the car evolve for more generations even at the cost of smaller population size. This experiment adds more evidence to the theory that smaller population size and more generations produce better results than bigger population size and smaller number of generations. However, as previously said, there is still very little evidence to produce a conclusive proof to this theory but evidence so far points in that direction.

Experiment 3:

Compared to control experiment, population size was decreased from 100 to 60 and number of generations increased from 20 to 33 as this was observed in experiment 2 to produce better results. Same settings as experiment 2 with only difference being in the mutation, where in the control experiment the whole population has a chance to be mutated, in this experiment, parents and children of the best individuals were excluded from mutation. Purpose of this experiment is to determine whether keeping the best individuals from mutation can produce better results.

- Population size = 60
- Number of generations = 33
- Evolution process = Adding parents and children to the population after the population was mutated to prevent the mutation of best individuals.

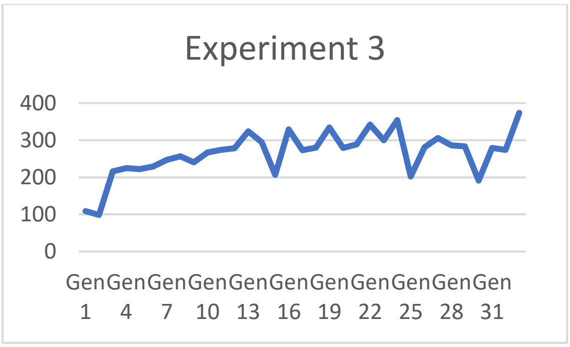

FIGURE 4

Observed characteristics of the evolved car: Front of the car is too low causing it to lose speed and spin often. Otherwise this could be a good candidate.

- Oval Track: E-Speedway:

Fitness is:	3958.83
Top speed:	273.53
Distance raced:	3972.83
Damage:	7

Car started off well but after a few seconds it's behaviour became very unpredictable, car kept changing speed and crashed once.

- Road Track: Alpine 1:

Fitness:	2857.42
Top speed:	185.97
Distance raced:	2869.42
Damage:	6

Car was driving steady on this track, not too fast but also little slower than it should. It also took some minor damage.

- Dirt Track: Dirt 2:

Fitness:	58.55
Top speed:	172
Distance raced:	1714.55
Damage:	828

Overall performance of the car on this track was very poor. Car kept crashing and spinning too much.

Considering the performance on all 3 tracks, the car evolved in this experiment is worse than the car evolved in experiment 2. This shows that excluding parents and children from mutation yields worse results, however this could be just a coincidence and without further experiments it can't serve as a proof of this statement.

3 Discussion

Experiment observations:

After conducting experiments 1 and 2 it was observed that increasing the number of generations at the cost of lowering the population size can yield better results, however producing a conclusive proof to this theory is outside the scope of this paper. Car evolved in experiment 2 which was the best one across all experiments drove very well on the oval track, slightly worse on the road track and very bad on the dirt track. This was because the car was evolved only for the oval track so the worse performance on other tracks is not surprising. Car evolved in experiment 3 produced worse results than the car in experiment 2. The only difference between experiments 2 and 3 is the steps in the evolution process. Adding the parent and children to the population after it was mutated evolved a worse performing car and it looks like keeping the best individuals from mutation can have negative impact on the evolution, however without running the evolution several times and doing more tests this could very well be just a coincidence.

Concluding remarks:

This paper presents the results of evolving parameters of a car in TORCS racing simulator using genetic algorithms. Experiments conducted show the suitability of GA's to evolve non-player character behaviour in a racing game. Scripting a car behaviour to obtain good racing capabilities for every track would be very time consuming. Using GA's is much quicker and cheaper. Experiments in this paper show that it's a powerful problem solver for parameter optimisation achieving good results in short time and warrants further study of methods to improve game artificial intelligence.

There are several possible ways to improve the genetic algorithm used in this experiment. Using a more efficient selection algorithm like stochastic wheel roulette selection could yield different, possibly better results. There are also different approaches to crossover and selection that could be applied in this genetic algorithm and tested by new experiments to see if better results can be obtained. Additional experimentation would be clearly beneficial to confirm some theories presented in this paper. There are still many configurations that have potential to provide better results.

References

[1] Risi, S., & Togelius, J. (2014). Neuroevolution in Games: State of the Art and Open Challenges. IEEE Transactions on Computational Intelligence and AI in Games.

[2] Bullen T., and Katchabaw M. J., (2008) Using Genetic Algorithms to Evolve Character Behaviours in Modern Video Games. Proceedings of the 2008 GameOn North America Conference, Montreal, Canada

[3] Eiben. A. E and Smith. J. E. (2003) Introduction to Evolutionary Computation Springer-Verlag, Chapter 2: What is an Evolutionary Algorithm - sections 2.1 to 2.3

[4] Yannakakis G. N. and Togelius J. (2018) Artificial Intelligence and Games. Springer

[5] Eiben, A. E. and M. Schoenauer, M (2002) Evolutionary computing, Information Processing Letters, 82(1): 1-6

[6] Togelius, J and Lucas, S. M., (2005): Evolving Controllers for Simulated Car Racing. Proceedings of IEEE Congress on Evolutionary Computation, 1906-1913

[7] Lucas, S.M. and Kendall, G., (2006). Evolutionary computation and games. IEEE Computational Intelligence Magazine, 1(1), pp.10-18.

[8] K.O. Stanley, B.D. Bryant, and R. Miikkulainen, (2005) “Evolving neural network agents in the NERO video game,” Proceedings of IEEE Symposium on Computational Intelligence and Games, pp. 182–189, 2005.