Using a genetic algorithm as a gameplay mechanic to play Chrome Dino game

Prakhar Srivastava, Mohga Emam, Aleksandr Krylov

University of Tartu
Institute of Computer Science

https://github.com/prakharsdev/ChromeDino

Introduction

In the game, we implemented our own genetic algorithm which mimics the process of natural evolution that is selection, mutation and crossover. In the framework, the population of Dino evolves iteratively in each generation to increase the fitness of Dino to jump over all the cacti.

Idea

The idea behind the project is our hunger to score more points in Chrome Dino game as well as to expand our knowledge of how genetic algorithm works. So that in future it could be utilised in the betterment of game development.

Game Framework

The GUI for the game framework is designed using python pygame library. The game is a cloned version is google chrome dinosaur. The objective of the game is the same as the original game: To score more and more point without colliding with the cactus.

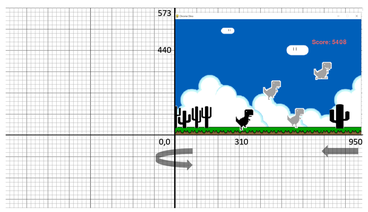

Fig. 1: genetic algorithm ERD diagram

About Genetic Algorithm

Genetic algorithm uses the heuristic approach to generates the solution to the optimization problems using the theories of evolution that happens in the real world. Hence the genetic algorithm is often described as an evolutionary algorithm. The genetic algorithm works on evolving the population of individuals to a population of high-quality individuals. The high-quality individual here is the individuals that have high fitness value as compared to the initial population. In brief, the algorithm works on two processes:

1. First, the individual is selected among the population of an individual based on the fitness perspective.
2. Second, the fittest individuals are then selected for producing the offsprings using mutation and crossover.

This workflow of Genetic algorithm that is selection, mutation and crossover works iteratively.

Using a genetic algorithm as a gameplay mechanic

As the Chrome Dino game starts the process of evolution take place. In the initial stage, the population is randomly generated which is the set of jump attempts. We consider all jump attempts as parents. We then evaluate the fitness of these parents based on the successful Dino's jump over the cactus. After that, fittest parents are selected. This stage is called the selection stage. The next stage is the mating stage. Also known as a genetic stage. These are:

1. First is the crossover stage in which we have applied k- point crossover. Crossover is the same as mating. So basically we consider parents as genomes and each genome has chromosomes. In the game, we have taken k equal to 2 which means chromosome are interchanged at two random points. In the k-point crossover, the crossover is applied in random chromosomes between two fittest parents. These parents produce children of high fitness value as compared to their parents.
2. After the cross over stage comes the mutation stage. Mutation functions the same as its biological name(i.e., to alter the chromosomes). In our approach mutation is perform very rare or in other words, it is performed only when the children produced from the k-point crossover are not fit as expected.

The children produced from the above stages are then considered as a parent for the next generation. These new parents are mate to produce the next generation of high fitness value. This process keeps on iterating throughout the game. For the simplification, we have taken the number of generation equal to 10 and the size of the chromosome equal to 6.

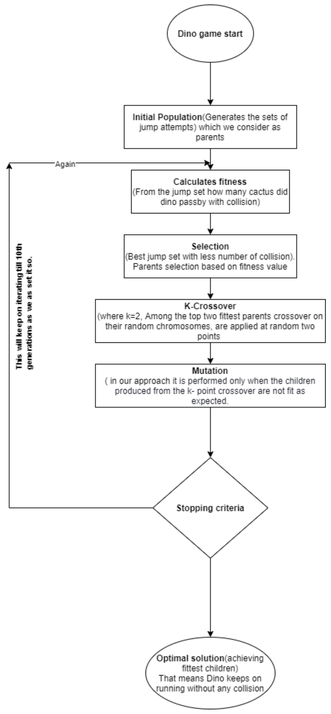

Fig. 2: genetic algorithm ERD diagram

NEAT Algorithm

To bring more intelligence into the game, NeuroEvolution of Augmenting Topologies [1], or shortly NEAT, algorithm has been used. The main idea behind the algorithm is the structural evolution of Artificial Neural Networks by using genetic algorithms. The genetic information that the algorithm operates with is represented by nodes of the Neural Network and the connections between them. Initially, NEAT algorithm starts with a population (of a fixed size) of Neural Networks having minimal structure with no hidden nodes. It incrementally progresses further through next generations, with each generation being produced by reproduction (involving crossover) and mutation of most fit individuals of previous generations [2]. As a result, new nodes and connections may occur in future generations making the structure of Neural networks more and more complex. The crossover between two networks of different topologies is maintained through historical marking which makes it possible to differentiate between individuals of the population and group them into species based on topological similarity [1]. Hence only homologous individuals are matched up for crossover.

Neural Network structure for Dinosaur Game

Applying NEAT algorithm to the Dinosaur Game, we start with a population of 50 dinosaurs, with each individual having a Neural Network (see Figure 2) that controls its behavior. The network takes the current y-coordinate of a dinosaur and the distance to the closest cactus as inputs. The output node of the network defines if a jump state should be initiated, i.e. if the output value exceeds 0.5-threshold, a dinosaur should jump, otherwise keep running.

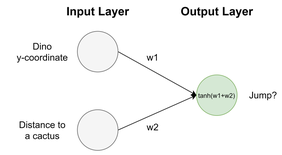

Fig. 3: Game Implementation.

Fitness Function

The fitness of each dinosaur refers to how far they have progressed in the game and is evaluated by the number of collisions with cacti. Dinosaurs having less collisions from the current population are chosen to mutate and breed them to create a new better generation getting rid of those individuals that performed poorly.

Termination criteria

The algorithm stops when one of the dinosaurs reaches the score of 10,000 points in the game.

References

[1] Kachmar Terletskyy Thesis
[2] Pygame Documentation