# The Fittest Mario: An Evolutionary Computation Approach to Super Mario Brothers

**Pedro Carrion Castagnola, Luke Guerdan, Marshall Lindsay, Jeffrey Ruffolo and Pooneh Safayenikoo,**

**Department of Electrical Engineering and Computer Science**
**University of Missouri, Columbia, U.S.A**

## Abstract

Super Mario Bros is a classic platformer video game in which the player tries to navigate Mario to the flag at the end of the stage. During this time, the player must successfully overcome obstacles and terrains in order to progress. Our project implements an evolutionary computation approach to finding a sequence of player actions to beat the game. We experiment with several operators for parent selection, crossover, mutation, and fitness evaluation. Results show successful convergence towards winning action sequences, with convergence rate responding to changes in strategies. We also observed different gameplay behaviors for chromosomes evaluated using different fitness functions.

## Chromosome representation

The chromosome is represented as a sequence of game actions taken by the player (such as up, left, or right), which when emulated result in a fitness score for the given chromosome. We also used the index of game over for crossover and mutation methods because after this index the actions do not influence on the fitness. The chromosome in our implementation is an object that contains: a list of action sequences (integers), fitness score and index of game over. Figure 1 shows a representation of the chromosome.

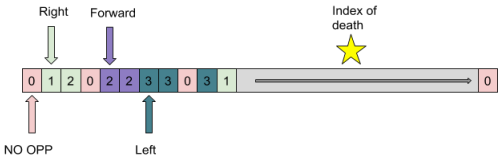


Figure 1

## Simulation environment

We used gym-super-mario-bros, an OpenAI Gym environment for Super Mario Bros on The Nintendo Entertainment System (NES) using the nes-py emulator [1]. This environment provides an easy-to-use, high-level API for making commands in Super Mario Bros. Figure 2 shows the general pipeline of our experiments.

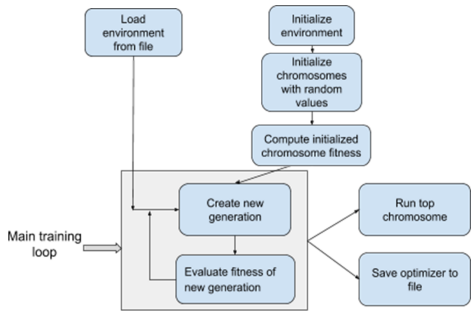


Figure 2

On all the experiment we used the following parameters as a baseline:3000 actions per chromosome (maximum number of actions), initial population of 40 chromosomes, 20 parents selection, 20 offspring created each generation, fitness equal to the position on "x coordinate", and 100 generations to run in total.

## Fitness function

Six different rewards were used; x position, coins, score, and speed. These rewards were modeled using Equations 1 – 6. Speed was modeled three different ways (Equations 3 – 6) in an attempt to find the fastest path to completion. Each of the different fitness functions resolved into different "player" behaviors.

(1) $F = FinalXPosition$

(2) $F = FinalCoins$

(3) $F = FinalScore$

(4) $F = \frac{FinalDistance}{ElapsedTime}$

(5) $F = \frac{FinalDistance}{ElapsedTime} \times \frac{FinalDistance}{TotalDistance}$

(6) $F = \frac{FinalDistance}{ElapsedTime} + (C) \times \frac{FinalDistance}{TotalDistance}$

Figure 3 and 4 show the maximum and average finesses of equations 1 and 6 respectively.

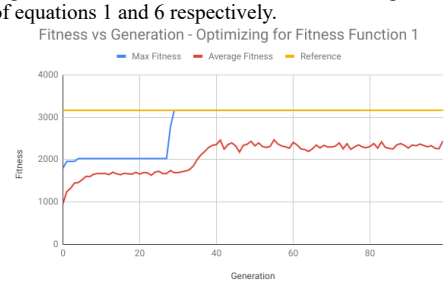


Figure 3

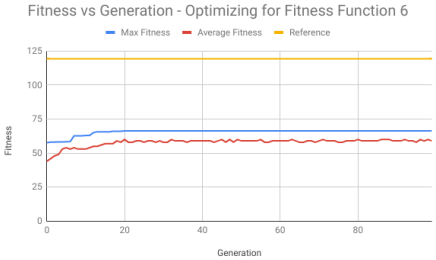


Figure 4

## Parent selection

For parent selection, we compared strict elitism, proportional selection and linear ranking (Figure 6). In both (μ, λ) and (μ + λ) selection, λ offspring are generated from μ parents by mutation method applied to the parent pool. In (μ, λ), the μ individuals from only offspring pool are selected based on a selection method as the parents for next generation. On the other hand, in (μ + λ), the μ individual from the μ parents and λ offspring are selected based on a selection method for next generation. These mechanisms are shown in Fig. 5.

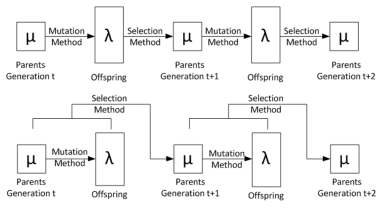


Figure 5

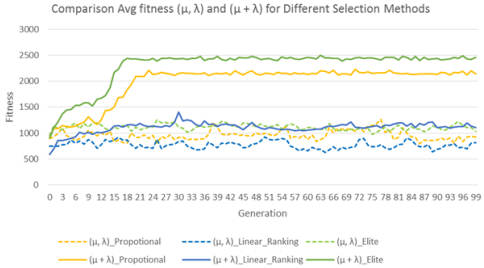


Figure 6

## Crossover methods

We explored three variations on the process of selecting the crossover points. The first was sampling two points from the chromosome using a uniform distribution. The second technique limited the distribution to only the genes that Mario had used before dying. The third technique limited sampling to the active genes, but changed the sampling distribution to a normal, centered on the point of death. Figure 7 demonstrate the second and third strategies. Figure 8 shows the effects on best and average chromosome fitness per generation for each of the crossover strategies we tested.

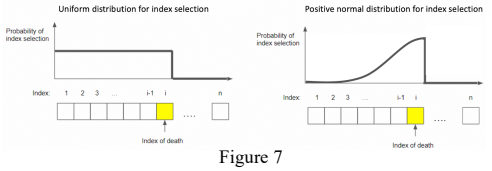


Figure 7

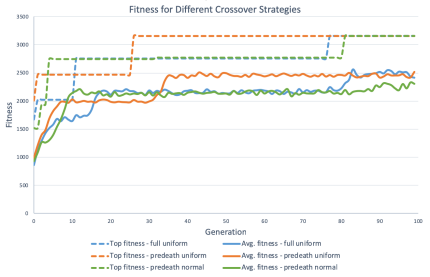


Figure 8

## Mutation methods

We made experiments changing the number of mutations (mutating 5%, 10% and 20% of the number of genes), selection of index for mutation and selection of new action. Figure 9 shows the two probability distribution functions used for the selection of index for mutation. Figure 10 shows the results of mutating different number of times using triangular distribution and mutation of 10% of the genes using uniform distribution.

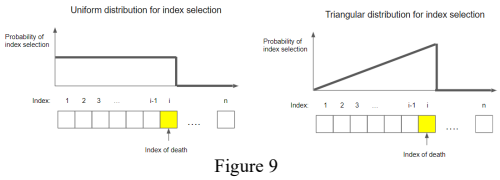


Figure 9

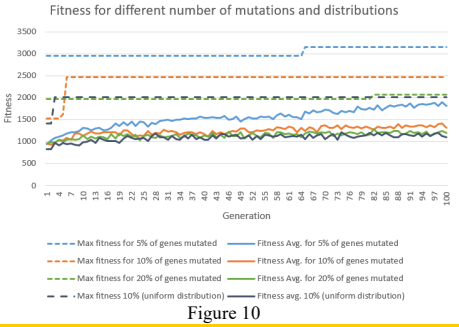


Figure 10

## References

1. Kautenja/gym-super-mario-bros. Github project https://github.com/Kautenja/gym-super-mario-bros
2. Chakraborty, U. K., Deb, K., & Chakraborty, M. (1996). Analysis of selection algorithms: A Markov chain approach. Evolutionary Computation, 4(2), 133-167.