# 1 INTRODUCTION

Selection is one of the main stages of evolutionary algorithms. The choice on which individuals should be selected, based on their performances, and how this is done influences the subsequent breeding (crossover operation). We will investigate how altering the selection operator before running equal evolutionary algorithms will affect fitness and individual gain for evolving agents from the framework EvoMan (see below).

The selection operators taken into consideration are the "Roulette selection" (also known as Fitness Proportionate Selection) and "Tournament selection". These functions are explained in detail in the "methods" section.

For carrying out the project we used the EA framework DEAP in which all the methods and algorithms necessary for the complete realization of our research are developed.[1] Evoman is a game playing framework, inspired by the game Mega Man, for testing game-playing agents. In this framework the player (prey) should beat at most 8enemies (predators). Both agents' behavior may be controlled by an Artificial Intelligence algorithm. The behavior of the player (move left/right, jump, release jump, shoot) is optimized by training a neural network. The number of neurons to activate in the net's hidden layer is decided before the run, while the training is performed evolving the network weights during the run.[2]

# 2 METHOD

To compare the two selection methods for each the same algorithm is run 10 times over 3 enemies and the results are compared.

The algorithm used is based on the "simpleEA" algorithm within the DEAP framework, but tweaked to fit the needs for storing the statistics. A schematic overview of how the algorithm works is displayed in figure 1.[1]

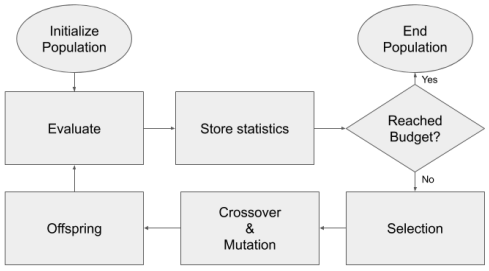


**Figure 1: Schematic overview of the evolutionary algorithm**

The initialization of the population is done by randomly generating weights for the neural network with the given hidden layers. Here each set of weights for the neural network is considered an individual that is to be evaluated. The standard fitness returned by the simulation of the EvoMan Framework is used as the basis of the evaluation. The fitness returned by the simulation is calculated with the function:

$$\text{fitness} = 0.9 \cdot (100 - e) + 0.1 \cdot p - log\ t$$

with $p$, $e$ the energy of the player and enemy after simulation respectively, ranging from 0 to 100 and $t$ the time steps of the simulation.[2] Since the roulette selection method expects a fitness larger than 0 and this standard fitness has the potential to go below 0, there was a fixed term added of 10. After evaluation there are several statistics computed. Here the mean and max of the fitnesses of the generation are computed as well as the best individual is updated. This updating is done by comparing the fitness of the current best individual and the individual of the generation with max fitness and setting the max fitness of those as the new best individual. Then based on whether the budget has been reached, either the roulette or tournament selection is used to select the populations for the crossover and mutation. The first assigns a probability of selection to each individual chromosome of a generation and, like in a casino roulette wheel, the parents of the offspring are chosen. The tournament selection, instead, selects a given number of random individuals from the population performing arbitrary many tournaments where the fittest individual is the winner.

After selecting the candidates with the set probability the crossover is done by "TwoPoint", which selects an interval for two individuals and swaps their weights within this interval. Following with a different set probability to mutate an individual by "ShuffleIndexes", which selects with another set probability within the to be mutated individual which weights are to be swapped with each other. This then generates a new population that is fed back to the cycle again starting with the evaluation step.[1]

The initial parameter values for the population was set small, for crossover high and for mutation low as these were found appropriate in another research on parameter selection.[3] After some test iterations it showed that the generations were converting to the same individual without changing, so the value for mutation and independent mutate probability were increased. The number of generations was set to a fixed amount to keep the total running time of the program to max 5 hours. For the hidden neurons it was set to be equal to the number of possible actions. The used parameters can be found in table 1.

**Table 1: Parameters**

| | | | |
|---|---|---|---|
| Population size | 5 | Hidden neurons | 5 |
| Crossover rate | 1.0 | Tournament size | 2 |
| Mutation rate | 0.2 | Independent mutate probability | 0.5 |
| Generations | 120 | | |

## 3 RESULTS AND DISCUSSION

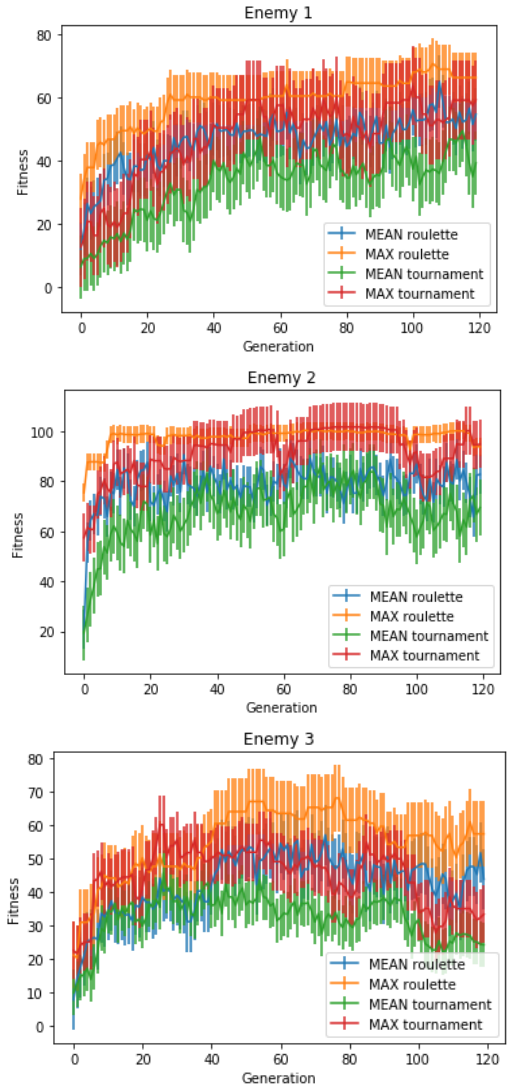


**Figure 2: Fitness Enemy Comparison Results**

When running the algorithm on three distinct enemies, the selection operator roulette has a larger impact than the tournament operator on increasing fitness across generations. In addition, there are some instances when the tournament operator has higher maximum values compared to the roulette despite having lower mean values, indicating a higher variability in fitness. This is also supported by the tournament method showing a larger standard deviation. Therefore, our data argue that a roulette mechanism is most likely to produce higher quality offspring most of the time, but there are a few cases where the tournament operator can outcompete it when measuring highest-fitness populations, this effect is most notable on graph Enemy 2 across 40-60 generations.

A second way to measure the performance of the two different methods is the individual gain, which is computed as *individual gain = player energy - enemy energy*. The individual gain of the best solutions out of the 10 runs are compared with boxplots and a t-test.

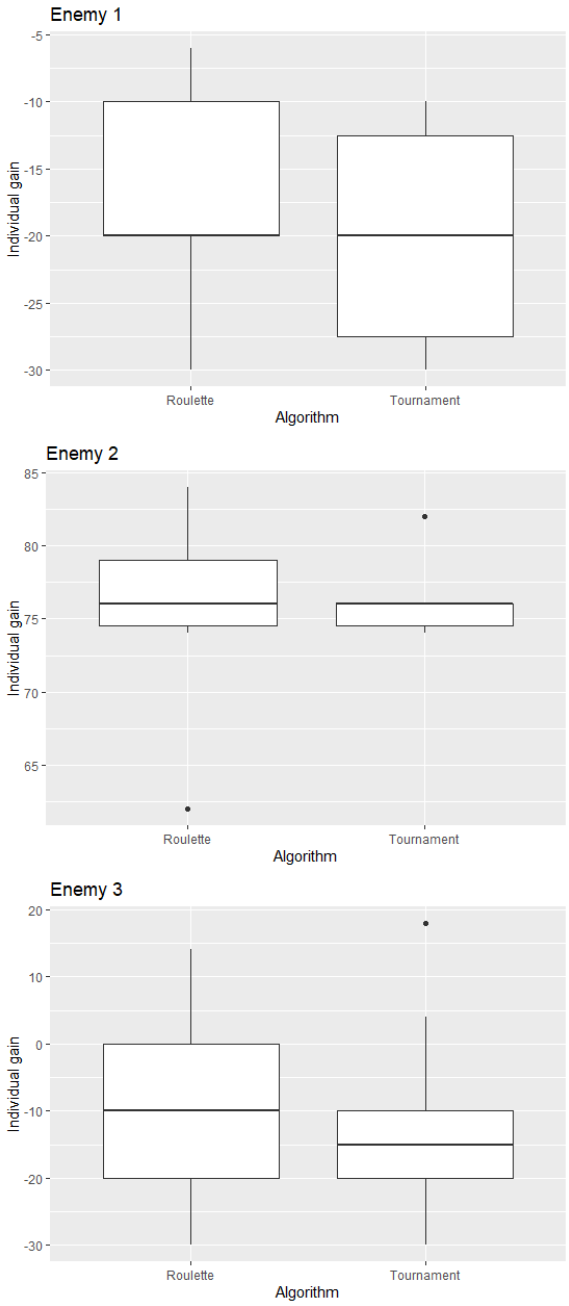


**Figure 3: Individual Gain Enemy Comparison Results**

The boxplots in figure 3 indicate that the roulette selection method tends to have a higher individual gain than the tournament selection method. However, the differences across the selection methods are not significant. Results of the t-test show p-value of 0.052 for enemy 1, 0.99 for enemy 2 and 0.53 for enemy 3 respectively.

Given the results for the fitness and gain, the performance of both methods is similar, with a slightly better performance for the roulette method with regards to finding an individual that yields high individual gain. A possible explanation why the performance of the selection methods is not significantly different could be that

both methods suffer from the same problem, namely that the outcomes can show a high variance from the theoretical probability distribution.[4]

A comparison is made between the two algorithms that are described in this paper and the Genetic Algorithm in the baseline paper [5]. In table 2 the average player energy of the best solution in all runs is given.

**Table 2: Average Player Energy of Best Solution**

| Enemy | Roulette Wheel selection | Tournament selection | GA baseline paper |
|---|---|---|---|
| 1: Flashman | 2 | 0 | 90 |
| 2: Airman | 76 | 78 | 90 |
| 3: Woodman | 3 | 2 | 58 |

The algorithms with the roulette selection and tournament selection have overall a lower performance compared to the GA of the baseline paper. Some elements of the three algorithms are similar, they all use the same fitness function, number of neurons and the selection method of the GA baseline is exactly the same as the method used in the tournament selection algorithm. The big differences between these algorithms are adding back the population before the selection method, the mutation operator, crossover operator and the population size. A possible explanation why the GA algorithm of the baseline paper is performing better is that the mutation and crossover operators used by the GA algorithm are able to create more variation, since the mutation and crossover operators used in the tournament EA and roulette EA are only shuffling and exchanging weights, but not creating new weights. Another reason for the lower performance could be that a population size of 5 is quite low.

## 4 CONCLUSIONS

Overall, the roulette selection method seems to yield better results with regards to the individual gain, however this is not significantly higher compared to the tournament selection method. When compared to the GA baseline paper the performance of both selection methods using the algorithm described in this paper is overall lower, which is most likely caused by the inability of introducing new weights through the mutation method.

## REFERENCES


[1] *DEAP documentation — DEAP 1.3.1 documentation*. Deap.readthedocs.io. (2020).

[2] *Miras K. EvoMan - Framework 1.0* Documentation. (2019)

[3] *Alajmi, A., Wright, J., Selecting the most efficient genetic algorithm sets in solving unconstrained building optimization problem (2014)*

[4] *Eiben E.A., Smith J.E., Introduction to Evolutionary Computing (2003)*

[5] *Da Silva Miras de Araujo, K., Olivetti de Franca, F., Evolving a Generalized Strategy for an Action-Platform Video Game Framework. (2016)*