Evolutionary Computing

Task 2 - Generalist Agent , Group 10

Shwetambari Tiwari (2643671)
Konstantinos Sakellariou (2672808)
Apala Saha(2671291)

1 INTRODUCTION

The assignment is based on experimenting and implementing evolutionary algorithms. In this problem space, the goal of our experiments is to implement an evolutionary algorithm and a variation of it to perform against groups of enemies of the Evoman framework. Our main research question is to understand how we can find the optimal solution based on the evolution algorithm and the optimal parameter setting to implement a generalist agent to beat selected groups of enemies from the Evoman framework. Furthermore, we investigate the possibilities to increase diversity in our population. To answer this question, optimal parameters along with their values were identified and tested for both algorithms. We expect a dominant performance of Genetic Algorithm 2 as in [1]. Finally, we check the statistics of how both the algorithms performed on each group of enemies.

2 RELATED WORK

With the Evoman framework, both prey (player) and predator (enemy) may be controlled by an Artificial Intelligence algorithm (agent). Similar work is identified in [2]. The performance of two evolutionary algorithms was tested against all enemies. Following the same way we implement two evolutionary algorithms that perform against groups of enemies in the training phase, so we also have a generalist agent and a multiobjective purpose.

3 GENETIC ALGORITHM DESCRIPTION

In this section we describe the two algorithms we used to perform our experiment and achieve our goal. We borrow and apply the techniques developed by [3]. We refer to the algorithms as 'Genetic Algorithm 1', which is our primary algorithm and as 'Genetic Algorithm 2' its variation.

Genetic Algorithm 1 consists of the basic steps of an evolutionary algorithm as described in [4]. These steps are crossover, mutation and selection. We initialize our population randomly according to a uniform distribution function. The fitness function described in [2] is our criterion to evaluate our current population to select the best individuals as parents. Next we apply the crossover and mutation operators to produce the new offspring of the next generation. After the evaluation of the population the best individuals are selected for mating and then crossover is applied to the center of two parents, that means that the new offspring will have its first half of its genes taken from the first parent and its second half of its genes taken from the second parent. The mutation operator changes randomly a number of genes based on a uniform distribution function. Finally, the new population consists of both the parents and the offspring and the same steps are performed for a number of generations.

Genetic Algorithm 2 is a slightly different variation of Genetic Algorithm 1. It is important to mention the differences between them, which are located in the crossover and mutation operator. Crossover takes place in the first third of the parents and for the mutation operator we change randomly 3 genes.

The pseudo-code of the Genetic Algorithms is presented below:

Algorithm 1 ***Genetic Algorithm***.

(1) Initialize the population
(2) Evaluate the population with a fitness function
(3) Repeat until final generation
  (a) Select the best parents
  (b) Apply crossover operator
  (c) Apply the mutation operator
  (d) Create the offspring and evaluate it

4 EXPERIMENTAL SETUP

In this section we describe the experiment set up. We use the Evoman Framework to perform our experiments. Our goal is to evolve the weights of the neural network, which is the 'player_controller' that makes the decisions within the game. The neural network consists of 10 hidden neurons and the structure is described in [2]. Enemies were distributed into 2 groups. Group 1 consists of enemies [1,2,3,4,6] and group 2 consists of enemies [2,3,4,7,8]. In Table 1 we present the final parameters we chose for Genetic Algorithm 1 (GA 1) and Genetic Algorithm 2 (GA 2). Performing this experiment we tried multiple values until we end up with the final values. In parenthesis are the values we rejected.

Table 1: Parameters and Values of Genetic Algorithms

Parameters	GA 1	GA 2
Generations	10	10
Population	100 (50)	100 (50)
Number of weights	265	265
Number of parents mating	2 (1)	2 (1)
Location of Crossover	1/2	1/3
Number of random mutations	1 (2)	3 (4)
Crossover Prob.	100%	100%
Mutation Prob.	100%	100%

5 EXPERIMENTAL RESULTS

In this section we present the results we obtained from our experiments. In Table 2 we present the average energy points of the player and the enemy for our best solution for 5 repetitions, that came from Genetic Algorithm 1 against all enemies.

Table 2: Player and enemy energy points

Enemy	Player energy points	Enemy energy points
1	0	80
2	68	0
3	0	70
4	0	60
5	41.8	0
6	0	60
7	0	10
8	56.2	0

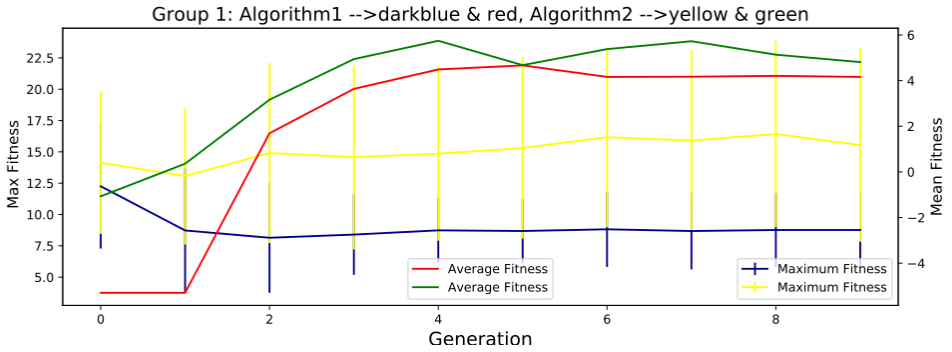

Figure 1: Means of Maximum fitness and average fitness of group 1.

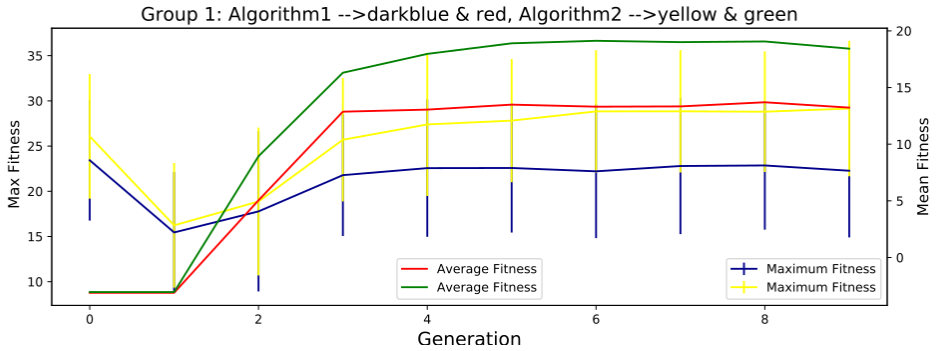

Figure 2: Means of Maximum fitness and average fitness of group 2.

The following results are presented in form of figures and reported by means of the average of the obtained values. Both genetic algorithms evolve their solutions for 10 generations and the experiments were repeated 10 independent times. Figures 1 and 2 describe the means of the average of maximum and average fitness over 10 generations for the first and second group of enemies of the Evoman framework respectively. The x-axis represent the generations and y-axis represent the fitness in two different scales, the left scale is for the maximum fitness and the right scale is for the mean fitness of the genetic algorithms. Max fitness is represented in all figures including the mean of the standard deviation of each generation. In Figure 1 and 2 we observe the results of the experiment for the 2 group of enemies respectively. Furthermore, in Figure 3 we present 2 boxplots obtained from the best solutions found across the 10 independent runs of our genetic algorithms against all enemies. These best solutions were tested five times and the mean individual gain was calculated. The boxplots show us on the x-axis the name of the algorithms and on y-axis the individual gain, which is calculated as $playerlife - enemylife$. Finally, we performed the Welch two sample t-statistical test to verify if the differences in the average of these means are significant between the groups of best solutions to compare the algorithms. The p-value for group 1 was 0.5336, for group 2 was 0.7017. Because these p-values are greater than the significance level $\alpha = 0.05$ we can conclude that

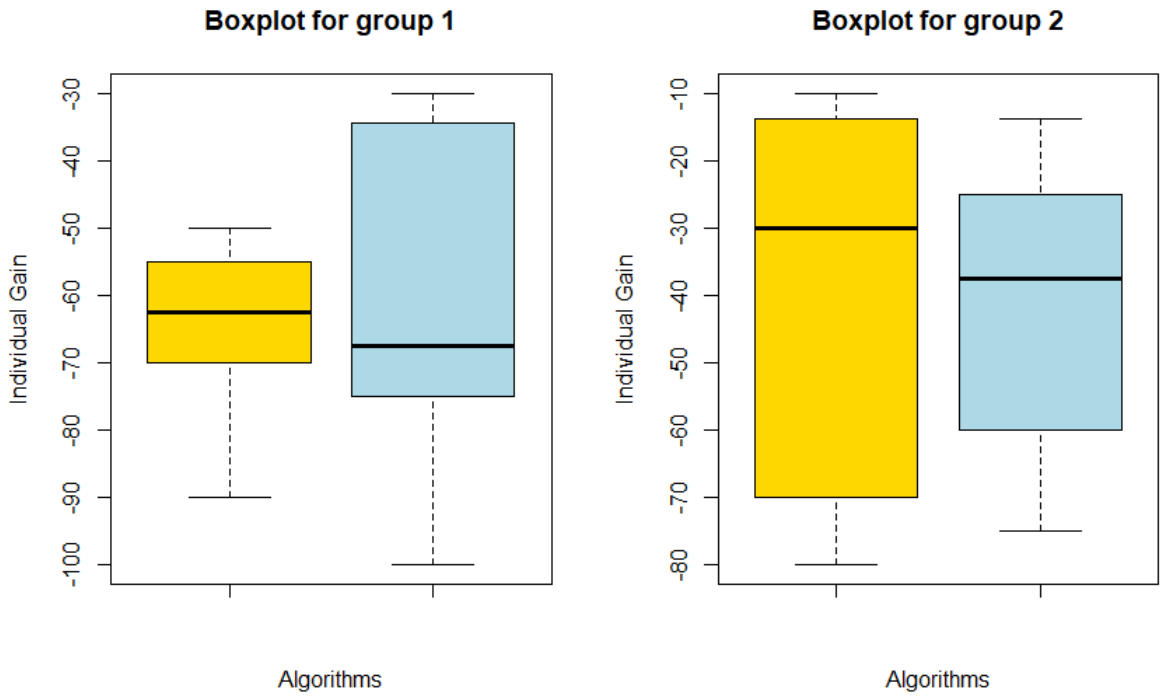

Figure 3: Boxplots for the 2 groups of enemies.

the individual gain of algorithm 1 is not significant different from the individual gain of algorithm 2 in both cases.

6 ANALYSIS AND DISCUSSION

In this section we aim to analyze our results and open a discussion of how the setup of the algorithms could be different. The first observation is that Genetic Algorithm 2, with the most "aggressive" operators, performed better than Genetic Algorithm 1 for the 2 groups of enemies in both fitness measures, max and mean. Especially, for group 2, the algorithms look very consistent in their performance having an upward slope towards the generation and peaking after generation 3. That leads us to conclude that this algorithm explored many different solutions, because of the operators used, which were leading the algorithm in diversity of the population. Second observation is the weakness of both algorithms to beat more than 3 enemies. The paradox of this situation is that our best solution comes from Genetic Algorithm 1, where we expected to be from Genetic Algorithm 2. Finally, the observation is coming from the boxplots, where the maximum gain was for group 2 at -10 and also we can observe that the medians for group 2 are higher than those of group 1.

Overall, we have a lot of data and statistics to make useful comparisons, but first it is important to discuss about the setup. The training phase consists of 10 generations, which is a small number comparing to [2]. Raising the number of generations may lead in better solutions. Also, the parameter settings could be different. Although we tried many possible values for our parameters, different values for the mutation and crossover operators and the probabilities of applying them would lead in different results. Our suggestion is to fine tune the parameters for both algorithms and compare these algorithms to a state of the art algorithm like NEAT.

7 CONCLUSIONS

The assignment was based on Evoman framework which is a Video Game Computational Intelligence framework. Task 2 comprised of a generalist task, in which the multi agent is evolved in order to defeat a group of enemies. A multi learning strategy was tested, wherein the agent learns to beat groups of enemies. In conclusion, the results looked very promising as we manage to beat 3 enemies, and we were close to beat more, but we can not be sure which algorithm is better than the other, due to the anomaly we detected were the overall results saw clearly an advantage for Genetic Algorithm 2, but the biggest gain came from Genetic Algorithm 2, so our first hypothesis that Genetic Algorithm 2 will be dominant is rejected. That answers our research question, where the goal was to identify the optimal parameter setting of an algorithm that could beat a group of enemies of the Evoman framework and understand the power of the diversity of the population.

REFERENCES

[1] S. K.Sakellariou, A.M.Saha, "Specialist agent, task 1," 2020.

[2] K. d. S. M. de Araujo and F. O. de Franca, "Evolving a generalized strategy for an action-platformer video game framework," in *2016 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1303–1310, IEEE, 2016.

[3] A. Gad, "Genetic algorithm implementation in python." https://github.com/ahmedfgad/GeneticAlgorithmPython/tree/master/Tutorial%20Project, 2020.

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