1 INTRODUCTION

There are different algorithms used for optimizing solutions to problems. Two popular algorithms are the genetic algorithm (GA) and the particle swarm optimization algorithm (PSO). In this paper we aim to answer what the difference is in results between a GA and PSO when training a generalist agent for an action-platform video game framework with similar parameters? 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] For the game framework we used the EvoMan framework and the built in neural network of the framework.[2] For the generalist approach we trained the neural network with two different groups of enemies and then used the set of weights with the best results to record their performances within all the enemies.

2 METHOD

The GA used is an adapted version of the one in our previous work. The overall GA did not change with its schematic overview in Figure 1.

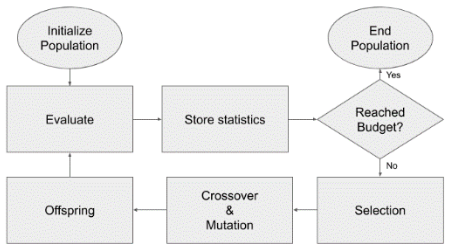

Figure 1: Schematic overview of the GA

The main change made was to the mutation method as in our previous work concluded this was a constraining factor. The mutation is now done by randomly generating a number between -1 and 1 and substituting a weight of an individual by this newly generated value. The mutation probability for each weight to be mutated was set to 0.1 as advised in R. Cazacu.[3] The mutation probability for an individual to be mutated was chosen based on parameter tuning. Further we went with the roulette selection method as this seemed to yield overall better results and consequently we kept an added fixed term of 10 to the fitness returned by the simulation to avoid fitness going below 0.[4] A schematic overview of the PSO can be found in Figure 2.

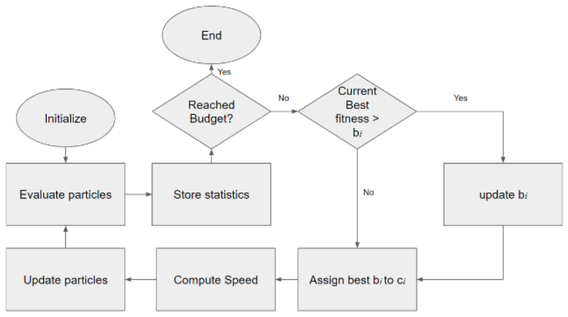

Figure 2: Schematic overview of the PSO

Initialization is similar to the GA, where particles are randomly generated to be a set of weights for the neural network. Each particle is updated according to the following:

$$u_{1,2_i} = U(0, \varphi_{1,2}) \qquad s_i{}' = s_i + (v_{u_1} + v_{u_2})$$

$$v_{u_{2i}} = u_{2_i}(c_i - x_i) \qquad v_{u_{1i}} = u_{1_i}(b_i - x_i)$$

$$x_i{}' = x_i + s_i{}'$$

Where x_i is the particle's position vector, s_i represents the speed vector and b_i is the best position vector of the particle so far, while c_i denotes the population global best. Furthermore, u_{1_i} and uu_{2_i} are the vectors of random values generated with an uniform distribution between the values 0 and $\varphi_{1,2}$. The population size remains the same as in our previous work, as it was suggested in Alajmi, A. Et al.[5] In table 1 the parameter values of the evolutionary algorithms are given. φ_1, φ_2, smax and smin are parameter values used in the PSO and the crossover rate, mutation rate and independent mutate probability are used in the GA.

Table 1: Parameters

Population size	5	Hidden neurons	10
Crossover rate	1.0	Independent mutate probability	0.1
Mutation rate	0.9	Generations	80
smax,smin	± 0.1	φ_1, φ_2	0.5

The performance between the GA and PSO will be examined using two different training groups. The first chosen training group contains enemies 7 and 8 and the second group contains enemies 2,5 and 6. These groups were chosen because a genetic algorithm with 10 neurons in Miras K. achieved high gain values with these two groups as well as high remaining energy values for the player.[2] The differences in the results of the GA and PSO algorithm for both training groups will be measured by the fitness function over generations and the gain for the best solutions. The individual fitness f is computed by:

$$f = 0.9(100 - e) + 0.1p - log\ t$$

With p, e the energy of the player and enemy after 10 simulation respectively, ranging from 0 to 100 and t the time steps of the simulation. Where the overall fitness over enemies is computed by:

$$F' = \bar{f} - \sigma + 10$$

With $\bar{f}$, σ the mean and standard deviation of the individual fitnesses respectively. Further the gain is computed as follows:

$$g = \sum_{1}^{8} p_i - e_i$$

Where p_i and e_i are respectively the remaining player energy and the remaining enemy energy at the end of the scenario, defined by enemy i,[2] For comparing the fitnesses of the algorithms for each generation the mean and max value is computed over 10 runs of which the average and standard deviations are determined as well as the individual with the highest fitness is stored. To compare these best individuals of the simulations, these are tested against all enemies for 5 times and their gain is computed.

3 RESULTS AND DISCUSSION

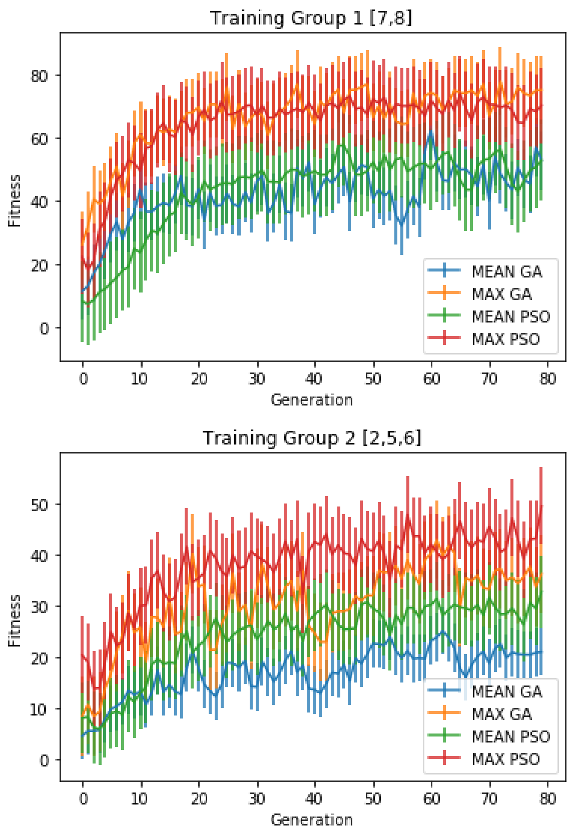

Figure 3: GA and PSO fitness performances during the 10 runs of the algorithms w.r.t. different groups of enemies

From Fig. 3 we can clearly notice that both algorithms perform more consistently and show higher fitness overall against enemies [7,8] compared to [2,5,6]. Examining more closely, the overall mean and max fitness are higher when using PSO for [2,5,6] compared to GA. However, overall fitness values remain very similar over generations against enemies [7,8] using both algorithms. The max fitness from PSO also tends to improve at a faster rate than the GA for both algorithms, notable from the high gradient of the red and green lines around generations 0 to 20. Whenever the mean fitness values decrease, it is likely due to our use of high mutation rates, which can produce a higher proportion of low-fitness individuals for the next generation compared to low mutation rates. The effect of GA's high mutation rate is also noticeable in the trend of the graphs, in fact the graphs generated by the PSO algorithm tend to show less fluctuation.

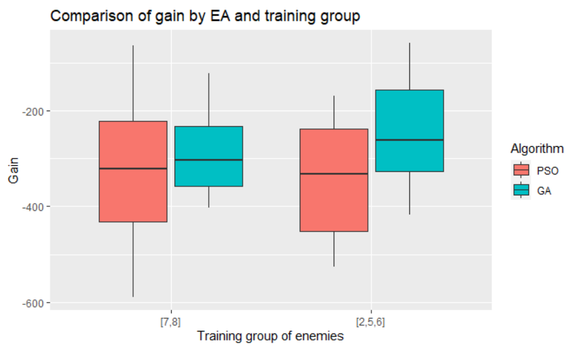

Figure 4: Gain Comparison Boxplots

The boxplots of Figure 4 show gain calculation for both algorithms against both training groups. They give an indication that for the training group of enemies [2,5,6], GA is performing better than PSO. For the other training group with enemies [7,8] they give similar results, since the median of the GA is higher, while the deviation from the PSO algorithm is higher, which results in some cases in a higher gain compared to the GA. To see whether there is a difference between the algorithms or groups the following hypothesis was tested:

H_0:There is no difference in mean gain between the training groups or algorithms

H_1:There is a difference in mean gain between the training groups or algorithms

The hypothesis was tested with α = 0.05 through a one-way ANOVA test and with (F(3,36)) = 1.21, p = 0.32 found to be not significant. A possible explanation why the differences between groups are not significant is that there is a high variability in the best solution gains of both algorithms. The big variability in gains could be caused by the low population number. Although the differences in gains between the EAs and training groups are not significant, it is interesting to mention there are differences in the way the search space is searched. This could be concluded from the big differences in mean and maximum fitness value for group [2,5,6] (see Fig. 3). It is also interesting to see that the strategy both algorithms develop is different. The GA is adopting a strategy that is moving forward (and sometimes backwards) while jumping and shooting. The PSO algorithm is adopting the same strategy as the GA in some cases, but a more static strategy in other cases, in which case the player remains in the same position while shooting. To compare the algorithms with the resulting means and standard deviation from table 6 in the baseline paper the average gain over 10 runs is computed and shown in Table 2.

Table 2: Overall gains obtained by GA and PSO for each of the training groups tested. The reported results are the average ± standard deviation, after 10 repetitions of each experiment

Training group	GA	PSO
(7,8)	-292 ± 87	-327 ± 157
(2,5,6)	-247 ± 113	-344 ± 126

The results of Table 2 are worse than almost all the results shown in the baseline paper regarding the overall gains obtained by the algorithms. The mean is lower and the standard deviation is higher. The high standard deviation could be due to the small population size (just 5) used by the two algorithms studied in this paper. The lower means could be caused by non-optimal parameter values. It could be interesting for further development to investigate more in parameter tuning and parameter control, to search for optimal parameter values, which will result in a higher individual gain and lower standard deviation.

The best solution of all runs is created with the GA with training group 1 [7,8]. The 'best' solution means, in this case, the highest number of defeated enemies, which is 4. The player's energy and enemy's energy are shown in Table 3, sorted according to the order in which enemies appear in the game. If we measure the 'best solution' by the highest obtained gain, then it is achieved by PSO with enemy training group 2 [2,5,6], with a gain of -59.

Table 3: Energy values for the best solution (highest number of defeated enemies)

Enemy	Player energy	Enemy energy
1 - FlashMan	0	80
2 - AirMan	**32**	**0**
3 - WoodMan	0	70
4 - HeatMan	0	70
5 - MetalMan	**60**	**0**
6 - CrashMan	0	100
7 - BubbleMan	**57**	**0**
8 - QuickMan	**48**	**0**

4 CONCLUSIONS

In summary, we have tested two distinct algorithms (PSO and GA) for the game framework EvoMan with newly adapted parameter tuning and optimized by training a neural network. We then tested them against two different sets of enemies and computed the fitness and gain values for each and assessed the differences. All in an effort to know the difference between a GA and PSO when training a generalist agent for an action-platform video game framework with similar parameters.

To answer our initial research question; examining fitness results suggests the PSO algorithm performs slightly better over enemies [2,5,6] only. In every other case the two algorithms perform very similarly in terms of fitness, though PSO improves at a faster rate. On the other hand, the gain comparison boxplots show that the GA algorithm is, on average, superior to PSO against enemies [2,5,6] with less variance and more consistency. That being said, the statistical significance of such differences is low. If a 'best solution' is sought after, no algorithm would be clearly superior over the other as and it would be dependent on whether we define 'best' in terms of defeated enemies or highest obtained gain.

5 REFERENCES

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

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

[3] *Cazacu R., Comparative Study between the Improved Implementation of 3 Classic Mutation Operators for Genetic Algorithms (2017)*

[4] *Ham N. van Kleij - Westveer C. Rossolini A. and Cantu A. Task 1: specialist agent.*

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