Standard Assignment Task I: Specialist Agent

Evolutionary Computing- Group 21

Wafaa Aljbawi
ID: 2750123

Victor Retamal Guiberteau
ID: 2741820

Janneke van Baden
ID: 2741570

Félix Nastar
ID: 2744996

1 INTRODUCTION

Evolutionary algorithms (EAs) are widely used to produce a population of individuals (Artificial Agents) that develop over time to the optimum solution by using biologically inspired operators such as cross-over, mutation and selection.

The main objectives of this report is to showcase an implementation of two evolutionary algorithms that are able to play a video game using a python framework called EvoMan.

1.1 Research question

The scope of this paper is to find the answer to the following research question:

- *What is the effect, in terms of performance, of using a fitness-based crossover, in combination with a heavily fitness-based parent selection mechanism in comparison to using non-fitness-based crossover and roulette wheel selection for evolutionary algorithms?*

Experiments are carried out to find an answer to this question, and the findings are presented in section 3. Performance is defined as the average and best fitness values of the population.

2 METHODOLOGY

2.1 Experimental setting

The two Evolutionary Algorithms (EAs) are trained within the Evoman framework. The environment provides 8 different enemies to challenge the EAs. We will train two specialist EA, which means, We will generate 3 versions of every EA. Each version will be specialized in one enemy. The EAs are trained against enemies 1,2, and 8 separately. The two EAs follow a general Genetic Algorithm design. Real value representation is selected in this research; the genes of every individual will represent the weight to feed to a neural network with ten hidden layers. In the environment, random initialization is set to "yes", the enemy mode is set to "static", and the player controller is handled by a neural network with 10 hidden layers.

2.2 Training and testing phase

For the training phase, we report the average and standard deviation of the mean and maximum fitness of the agents averaged over 10 runs with 100 individuals per generation with a total of 20 generations. The testing phase was done with the individuals of these runs with the best fitness, in five individual runs, reporting the fitness and the best mean player energy against a specific enemy.

2.3 The Design of Evolutionary Algorithms

In order to achieve the main objective of this paper, two evolutionary algorithms were implemented. The general setup for both algorithms can be found in Table 1 and Table 2. These tables correspond to the symbolic and numeric parameters of the used algorithms, respectively. Symbolic parameters refer to the operators employed in each EA to guide the algorithm towards a solution to the given issue [9]. These operators are namely mutation, crossover and selection, and they are essential to the efficiency and capability of EA optimisation in obtaining the best solution [1]. The numeric parameters, on the other hand, relate to the particular values of parameters that are associated with a certain symbolic parameter, such as the probability of mutation, population size and the number of generations, and so on.

	Algorithm 1	Algorithm 2
Representation	Weights of a neural network	Weights of a neural network
Crossover	Blend Alpha Crossover (BLX-α)	Blend Alpha Beta Crossover (BLX-$\alpha - \beta$)
Mutation	Uniform Mutation	Uniform Mutation
Parent Selection	Roulette Wheel Selection	Elitist Selection
Survival Selection	$\mu + \lambda$	$\mu + \lambda$

Table 1: Symbolic Parameters

	Algorithm 1	Algorithm 2
Hidden neurons in neural network	10	10
Population size	100	100
Blend crossover parameters	α = 0.5	α = 0.55, β = 0.45
Mutation probability	0.1	0.1
Offspring per Recombination	2	2
Number of generations	20	20

Table 2: Numeric Parameters

2.3.1 *Fitness Function.*

The fitness function evaluates an individual based on how well it satisfies the criteria that the algorithm is optimizing for [7]. The fitness criteria are applied to the population at each stage to determine which parts of the population should be chosen for subsequent computations [7]. The fitness function is application-specific and contains various constraints based on the application's requirements to produce the optimum results [7]. In [2], the proposed fitness function tries to balance the minimization in the energy of the adversary, the maximization in the energy of the current player and the minimization in the time taken to end the game. The fitness was modeled as described by the following equation:

$$fitness = \gamma * (100 - e_e) + \alpha * e_p - logt$$

where

- e_e *and* e_p *are the energy measures of enemy and player, respectively, varying from 0 to 100*
- t *is the number of time steps of game run*
- γ *and* α *are constants assuming values 0.9 and 0.1, respectively*

2.3.2 *Crossover.*

In algorithm 1, Blend crossover-α was used as recombination technique with α = 0.5, finding a balance between exploitation and exploration. Blend crossover is a recombination technique that allows to generate offspring in an n-dimensional area bigger than the rectangle spanned by its parents. In algorithm 2, Blend crossover-$\alpha\beta$ was used as recombination technique with α = 0.55 and β = 0.45 [5]. This version of blend crossover is tweaked to give more importance to exploitation since the fitness is taken into account.

2.3.3 *Mutation.*

In Algorithm 1, the mutation phase involves changing the allele into a uniform randomly selected real value between the user-specified upper and lower bounds. Every mutation is performed with probability $0 < pr_m \leq 1$ after the crossover phase, whether or not crossover was performed.

Similarly, for Algorithm 2, perform mutation at random with probability pr_m, such that $0 < pr_m \leq 1$. If mutation is used, apply the same uniform mutation approach as in Algorithm 1.

This operator was used to promote diversity within the population and avoid premature convergence [4]. Following these operations, the algorithm will have produced a set of new chromosomes, which will be assessed and combined with the present population [2]. Following that, a selection method is used to keep the population at a constant size [2].

2.3.4 *Parent Selection.*

To investigate the effect of giving each individual in the population a chance to be chosen, the roulette wheel selection was used in Algorithm 1. This method ensures that all individuals have a weighted chance of being selected, where the probability of selecting an individual is proportional to its fitness [6]. The fitness function used by the roulette wheel is described in section 2.3.1.

The second selection algorithm is based on a combination of elitism, over-selection and uniform mutation. The top 10% of solutions (based on fitness) is always chosen. Only the best performing solutions are chosen if the number of parents is smaller than 10% of the population size. If the number of parents to be selected is bigger than 10% of the population size, the remaining parents consist of 75% of uniformly chosen parents in the top 50% of the population (based on fitness) and for 25% of uniformly chosen parents of the bottom 50% of the population (based on fitness).

2.3.5 *Survival Selection.*

For both algorithm, $\mu + \lambda$ survival selection was performed. After every generation, the set of offspring and parents are merged and rated based on (estimated) fitness, with the best being retained to produce the next generation [3].

This method was chosen because it guarantees that the fitter individuals are not forced out of the population while also maintaining population diversity [8].

2.3.6 *Parameter Settings.*

The generation size was fixed at 20 in a weighted decision due to the lack of computational power and the fact that no increase in performance was found in parameter tuning in comparison with 25 and 30.

The α parameter of the Blend crossover α was set to 0.5, giving a balance between exploration and exploitation.

The α parameter of Blend crossover α-β was set to 0.55 to give more weight in the recombination to the fittest parent.

The β parameter of Blend crossover α-β was set to 0.45 to give less weight in recombination to the less fittest parent.

Mutation rate was set to 0.1 shows a range of optimal values for this numeric parameter, where we picked the values for our EA's from [3].

3 RESULTS AND DISCUSSION

Figures 1, 2 and 3 are line plots of the mean of the mean fitness values and best fitness values of ten independent runs of the two EAs, for enemy 1, 2 and 8 respectively. The standard deviations of these means are also shown.

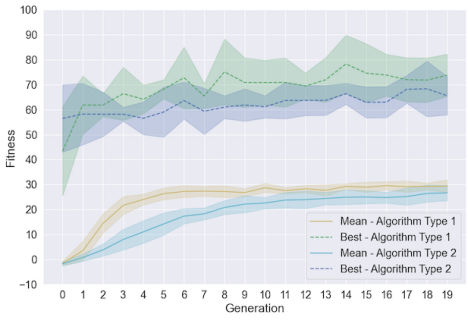

Figure 1: Performance of EA by Type, for Enemy 1

Interesting to note in Figure 1 is that even though the best solution is preserved every generation, due to the type of survivor selection, the fitness of a later generation might be lower than the current one. This occurs because the evaluations are performed every generation and, as the enemy is randomly initialized, the fitness might change.

For enemy 1, both the mean and the best fitness values are lower than the fitness values for enemy 2 and enemy 8. Moreover, the best fitness is a lot more changeable. This might be dependent of the behaviour of the enemy, and how easy it is for the player to find a good strategy to defeat it.

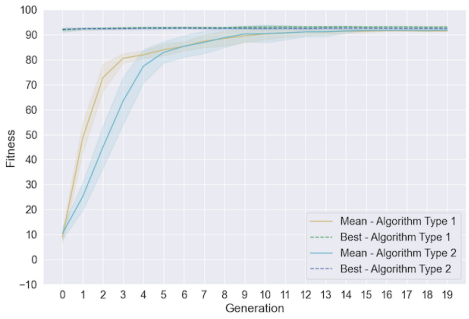

Figure 2: Performance of EA by Type, for Enemy 2

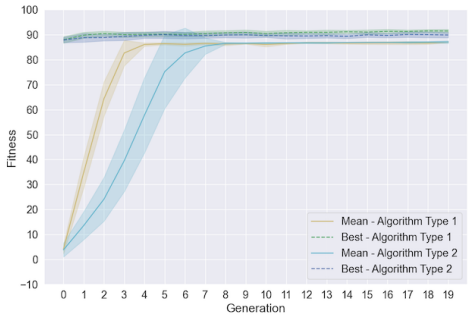

Figure 3: Performance of EA by Type, for Enemy 8

Aside from this, it is noticeable that the algorithm of the first type gets to the maximum earlier and faster than the algorithm of the second type for all enemy types that were tested. For enemy 1, this algorithm type also results in a higher mean and best fitness in general.

Figures 4 and 5 show box-plots of which the data points are means of the fitness values of 5 evaluations of the best solution of every ten independent runs, for enemy 1, 2 and 8 respectively.

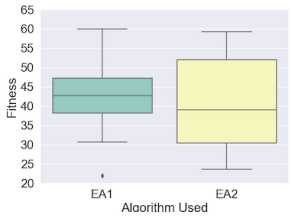

Figure 4: Performance of Best Solutions for Enemy 1

It can be seen that the box plots of the two types of evolutionary algorithms are close. Both of them perform better against enemy 2 and 8 than against enemy 1, which might be because of the difficulty level of said enemy.

In Figure 5 of enemy 8, there are two outliers. A possible explanation of this might be that there were multiple unfortunate random placements of the enemy during the runs that resulted in this data point, which affected the mean negatively.

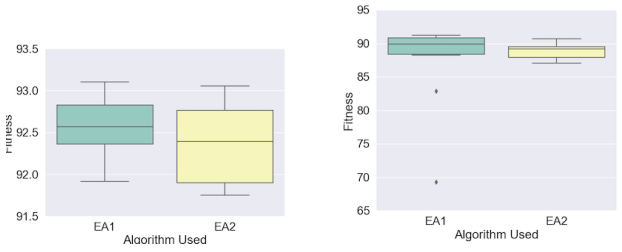

Figure 5: From left to right: Performance of Best Solutions for Enemy 2 and 8, respectively

Checking if the performance of the best solutions created by one algorithm outperform the best solutions by the other algorithm, is done by a statistical test. Using the Shapiro-Wilk test, it is determined whether the data is normally distributed or not. If the p-value is below 0.05, the data is not normally distributed. If it is normally distributed, the unpaired t-test is used, otherwise, the Mann-Whitney U test is used. The results for the Shapiro-Wilk and the significancy tests are shown in Table 3.

It is observed that there seem to be no significant differences regarding the results of the runs of the best solutions of the two EA types, for all enemies. The algorithms do not outperform each other.

Table 3: Results Shapiro-Wilk Test for Normality and Significancy Test

Enemy	p-value Shapiro	Test	p-value
1	EA1-0.98 | EA2-0.40	Unpaired t-test	0.79
2	EA1-0.88 | EA2-0.37	Unpaired t-test	0.27
8	EA1-0.00 | EA2-0.57	Mann-Whitney U	0.21

Table 4: Best Mean Player Energy For Our Algorithm

EA	Enemy 1	Enemy 2	Enemy 8
1	36	84.8	71.2
2	0	84.8	68.2
Baseline	96	94	86

Table 4 compares the best mean player energy of the best solutions runs and the best final mean player energy for every enemy in the baseline paper.

The algorithms of the baseline paper outperform the algorithms created in this paper. This can have multiple reasons, such as the fact that the baseline paper uses 100 generations, as well as the fact that the baseline paper uses multiple sophisticated methods.

4 CONCLUSIONS

It is shown that no significant difference between the two algorithms is found. The EA 1, which gives less weight to fitness in evolution and has a balance between exploration and exploitation, seems to reach the optimum faster than EA 2, where fitness significantly impacts the evolution against the three enemies. However, only a small sample size is used for these tests, and therefore further research could provide more insights into if one algorithm is better than the other.

5 CONTRIBUTIONS

Felix went over the report, helped with the design of Evolutionary Algorithms and added comments to the code. Wafaa implemented the roulette selection and crossover alpha-beta methods, performed experiments and wrote the design of evolutionary algorithms section. Victor implemented and optimized crossover operators, created file structure, wrote part of the report. Janneke implemented the elitist-based and survivor selection and mutation method, performed experiments, created the plots, performed statistical tests and wrote the results section.

REFERENCES

[1] Mohamad Faiz Ahmad, Nor Ashidi Mat Isa, Wei Hong Lim, and Koon Meng Ang. 2021. Differential evolution: A recent review based on state-of-the-art works. *Alexandria Engineering Journal* (2021).

[2] Karine da Silva Miras de Araújo and Fabrício Olivetti de França. 2016. An electronic-game framework for evaluating coevolutionary algorithms. *arXiv preprint arXiv:1604.00644* (2016).

[3] Agoston E Eiben, James E Smith, et al. 2003. *Introduction to evolutionary computing*. Vol. 53. Springer.

[4] Ahmad Hassanat, Esra' Alkafaween, Nedal A Al-Nawaiseh, Mohammad A Abbadi, Mouhammd Alkasassbeh, and Mahmoud B Alhasanat. 2016. Enhancing genetic algorithms using multi mutations. *arXiv preprint arXiv:1602.08313* (2016).

[5] Francisco Herrera, Manuel Lozano, and Ana M Sánchez. 2003. A taxonomy for the crossover operator for real-coded genetic algorithms: An experimental study. *International Journal of Intelligent Systems* 18, 3 (2003), 309–338.

[6] Sourabh Katoch, Sumit Singh Chauhan, and Vijay Kumar. 2021. A review on genetic algorithm: past, present, and future. *Multimedia Tools and Applications* 80, 5 (2021), 8091–8126.

[7] Haneet Kour, Parul Sharma, and Pawanesh Abrol. 2015. Analysis of fitness function in genetic algorithms. *International Journal of Scientific and Technical Advancements* 1, 3 (2015), 87–89.

[8] Jonathan E Rowe and Dirk Sudholt. 2014. The choice of the offspring population size in the $(1, \lambda)$ evolutionary algorithm. *Theoretical Computer Science* 545 (2014), 20–38.

[9] Mujahid Tabassum, Kuruvilla Mathew, et al. 2014. A genetic algorithm analysis towards optimization solutions. *International Journal of Digital Information and Wireless Communications (IJDIWC)* 4, 1 (2014), 124–142.