Evolutionary Computing - Assignment 2

Generalist agent

Group 52

Alexis Mourier *2696039*
Max Zwager *2709802*
Mickey van Immerseel *2693643*

Artificial Intelligence
Vrije Universiteit Amsterdam
Netherlands
18/10/2021

1 INTRODUCTION

Evolutionary computing has been around since 1963 when it was first put forward by Fogel et al. [5]. It was later made more accessible by Holland and Rechenberg in the 70's, who initially implemented it as an optimization system for irregularly shaped objects [4]. This is still a common use of evolutionary computing today. Evolutionary Algorithms (EA's) are problem solving methods that take inspiration from natural evolution to create the most efficient solution to a given problem. This is done by taking a randomly generated population (each individual represents a solution) and applying similar principles of natural evolution such as mutation, parent selection and recombination of genes. Each individual is measured through the use of a fitness function. Through iterative generational mutation and elimination of the weakest/selection of the strongest, an optimal solution can be found. EA's are effective at finding approximate solutions to problems where an exact solution is not required but a near optimal one is sufficient. This is due to the random nature of evolution where the exact optimal cannot always be promised.

1.0.1 Research question. This research paper will be looking at two generalised EA's each of which will use two different groups of enemies for training. The agent will be playing against certain enemies from the game EvoMan. The player agents moves are determined by the algorithms. This research paper will determine which EA performs best in defeating the enemies using a generalist approach. The algorithms will be utilizing the DEAP framework which offers a pre-existing structure [6, 13]. The algorithms are compared using fitness.

2 METHODS

2.1 EvoMan

The EvoMan framework is a game platform for the development and testing of optimization algorithms. The game consists of a main playable character (played by the created algorithms) that must defeat one or more of a total of eight unique bosses, often equipped with multiple attacks. The EvoMan framework provides a number of sensors (n=20) which can be used as an input for a model to generate the players' actions (walk left, walk right, jump, shoot, and release of the jump). The framework also provides a simulation function which returns the fitness of an individual using a predefined function.

This research made use of a player controller which is a single hidden layer neural network with 10 hidden neurons, which is provided in *demo_controller.py*. The neural network is chosen as it provides an evolutionary optimization problem of which the weights of the neural network (n=265) form the genotype. In this manner the genotype can be seen as a list of floating point numbers with a length of 265, which itself maps to a certain neural network instance that defines the players' behaviour.

2.2 Algorithms

Both evolutionary algorithms share a similar body, but with different operators and parameters. The procedure for one generation can be divided into three parts: selection, crossover and mutation. The role these parts play in both algorithms is discussed below. However, all parts are heavily influenced by specified operators and parameters. Firstly, the initial population is subject to selection based on the individual fitness value. To do this, the individual will play against all enemies in the group in a simulation. After this, the fitness will be calculated as follows:

$$F = 0.9 * (100 - h_e^C) + 0.1 * h_p^C - log(t^C) \quad (1)$$

Here, h_e^C is the combined enemy health, which is comprised of the mean - standard deviation of the enemy health for every enemy. The h_p^C and t^C are calculated likewise.

It is important to note that the selection operator for both algorithms selects a number of individuals equal to the population size. Because of the nature of some operators, the selection may contain duplicates of individuals in the original generation. After the selection process the selected individuals are subjected to crossover. During crossover the genetic properties of the subset are pairwise combined into two new individuals. After selection and crossover the whole original population will be replaced by the offspring, resulting in a mix of selected individuals and their offspring. Finally this new generation is subjected to mutation.

2.2.1 Algorithm 1. For the first algorithm certain characteristics from the first assignment were re-used, alongside some novel operators that had been previously considered but not implemented. They may be more efficient for a generalized agent compared to a specialized agent.

For the selection operator roulette is chosen, which is a stochastic selection method in which the probability of selection is comparative to its fitness. A selection is then randomly made where a higher fitness leads to a greater likelihood of getting chosen. An advantage of this method is that it gives greater importance to higher fitness methods but still allows all individuals a chance to get selected. This could be beneficial in creating a generalized agent as it would allow a greater variety of input. This operator has been extensively used in previous research [9, 11, 12].

For the crossover operator a uniform crossover is selected. Each gene is chosen from either parent with an equal chance. This once again has been used extensively in previous research [10] and has the benefit of not exhibiting positional or distributional bias. This bias has been shown, if not dealt with properly, can have detrimental effects on output [2]. It has even been claimed to outperform both one and two point crossover [15].

For the mutation operator a Gaussian mutation is used with individual mutation probabilities. In this case if an individual is chosen to be mutated every gene has a probability of 0.2 of actually being mutated. The mutation itself is an application on the current value of the gene of a random value taken from a normal distribution with $\mu = 0$ and $\sigma = 1$. After the mutation the weights of the individuals are then normalized

2.2.2 Algorithm 2. For the generation of the second algorithm, a combination of the findings of the first assignment were used. In the first assignment it is hypothesized that an amalgamation of

the two specialist algorithms would create the most optimized EA. This is put to the test in this generalist task. The operators from the second specialist EA were used and from the first specialist EA the dynamic Decreasing of High Mutation ratio/dynamic Increasing of Low Crossover ratio (DHM/ILC) is re-used.

The combination of two selection operators, namely roulette-tournament is used. Roulette and tournament are the most commonly used selection operators. As it turns out, according to Rahman et al., the combination of the two selection operators performed better than either separately [14]. The combination is implemented by first performing the roulette selection, after which the tournament selection is executed over the previously selected data.

For crossover the algorithm uses two-point. Two points are selected on the parent chromosomes and exchanged, which results in two offspring. This enables both exploration and exploitation by exchanging genes of individuals with high fitness (selected in the tournament) and exchanging genes with other different individuals [1].

The mutation operator shuffle indexes is used. According to Hassanat et al. the exchange type mutation operators perform well when the algorithm makes use of a single mutation [8]. Out of the mutation operators provided by the DEAP framework, the shuffle indexes seems to be the closest to what is described in that paper.

To balance exploitation and exploration, both algorithms make use of DHM/ILC. Previous research has suggested this dynamical balance can increase performance when dealing with small populations [7]. Starting the experiment, the individual chance of crossover is set to 0 and the chance of mutation to 1. This results in maximum exploration. During the run, the chance of crossover is increased to almost 1 and the chance of mutation is decreased to almost 0, both with step-size 1/n with n being the number of generations.

2.3 Hypothesis

In the first task it is hypothesized that a combination of the two specialist algorithms would create an optimized algorithm for this Evoman problem. As both specialist algorithms were based on literary research on what would work best for this particular type of problem, it is hypothesized that the second EA will outperform the first EA. The second part of the hypothesis is whether there is also a difference in effectiveness between the training groups that were used.

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

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

2.4 Experimental setup

The experiment is run with a population size of 100 and a generation size of 50. The population will be randomly initialized, with a random seed set for reproducibility. Both algorithms will be subject to both groups of enemies, and for every algorithm, for every group, the evolutionary process is repeated 10 times. This results in a total of 2*2*10 = 40 runs. During every run the best, mean and standard deviation of the population fitness is saved for every generation. Both algorithms will be run on separate hardware.

Table 1: EA comparison

EA steps	EA 1	EA 2
Population	Size=100 Generations=50	Size=100 Generations=50
Hyper-parameters	DHM/ILC	DHM/ILC
Selection	Roulette	Tournament-Roulette (size=8)
Crossover	Uniform Crossover	Two-point
Mutation	Gaussian (μ=0, σ=1)	Shuffle indexes $prob_{gene}$=.03

After running, the logs are parsed and for all 10 runs per algorithm and group of enemies the best, mean and standard deviation of the population fitness over the 50 generations are averaged over these 10 runs. Then, for every algorithm and group of enemies this fitness over generations will be plotted, resulting in 4 plots. These 4 plots visualize the performance of the algorithms for both groups of enemies and will be visually compared.

Then, the best individuals after 50 generations for all 40 runs are tested against each enemy 5 times. After this, the mean of the 5 times for each solution of the algorithms is calculated, and these means will be plotted in a boxplot. This will result in 2 pairs of boxplot (one pair per training group), each containing 10 data points being the average of 5 runs. These averages are finally analysed with a statistical test to see if there is a significant difference between training groups, algorithms or any combination between them.

Finally, a statistical analysis is performed on the gathered data to make a statement about a possible significant difference in performance between the two EA's and training groups. To compare the results of the different algorithms a Kruskal Wallis test is used on the mean fitness of the generations of each run, per group. The Kruskal Wallis test gives answer to whether the two algorithms and training groups significantly differ in their mean fitness per enemy. If it is significant, post-hoc comparisons between groups are required to determine which groups are different.

2.4.1 Enemies. Two groups of enemies for training are defined. Group one consists of enemies 2, 5 and 6. Group two consists of enemies 7 and 8. These enemies were chosen based on the findings of the base article and deductive reasoning on the workings of the enemies (and whether the tactics could be generalized over all enemies)[3].

3 RESULTS AND DISCUSSION

3.1 Results

Both EA's were run on different devices overnight, taking approximately 10 to 15 hours to complete the output for each group. After receiving the raw data, the evaluation script is run for each EA which took a few minutes. Finally, plots were made using the output from the evaluation using the plotting script and the statistical test is run using the generalist_stat_test.py script.

3.1.1 Plots. Figure 1 shows the fitness statistics of the best individuals over all enemies per group for the first EA, Figure 2 shows the same statistics only these are for the second EA. Note that in

the plots the first and second EA are referred to as *gen_1* and *gen_2* respectively. In both Figures 1 and 2 there is a clear difference visible between the two training groups. However, when comparing the boxplots per algorithm they look visually very similar. Just looking at the boxplots it seems like the second algorithm has a higher mean for the enemies, compared to the first algorithm. The second EA has more outliers than the first EA.

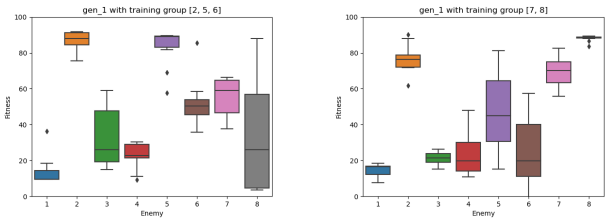

Figure 1: Fitness of all enemies using the best individuals from each training group for EA 1

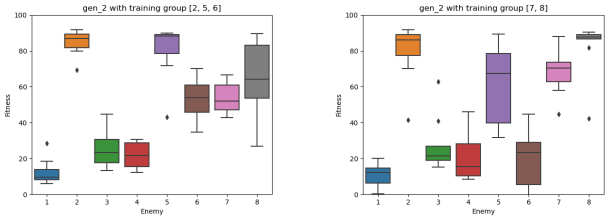

Figure 2: Fitness of all enemies using the best individuals from each training group for EA 2

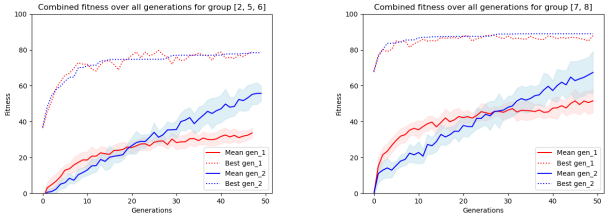

Figure 3: Population statistics over generations per algorithm

Figure 3 shows the population statistics over all generations, containing the best, mean and standard deviation of the population fitness. Both algorithms are visualized and are separated per training group. Table 2 shows the score per enemy of the overall best solution.

Table 2: Best results

Enemy	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
Score	12.9	91.8	62.7	44.3	89.4	2.4	70.6	81.8

Table 3: Kruskal Wallis test

Kruskal Wallis	**p-value**
45.969	*p=5.759e-10*

3.1.2 Statistical analysis. Table 3 shows that the *p-value* for the Kruskal Wallis test is significant (*p-value<0.05*) This means the null hypothesis may be rejected and post-hoc comparison have to be performed to determine which groups are different. Conover's test is used for this, with the Holm procedure due to the multiple comparisons. The results of the post-hoc comparison can be found in Table 4. All of the groups are significant in the post-hoc test, except for EA2 gr1 and EA2 gr2.

Table 4: Conover's test

	EA1 gr1	**EA1 gr2**	**EA2 gr1**	**EA2 gr2**
EA1 gr1		0.013	9.748e-10	2.809e-08
EA1 gr2	0.013		2.952e-04	2.539e-03
EA2 gr1	9.748e-10	0.000		0.5096
EA2 gr2	2.809e-08	0.003	0.5096	

3.2 Discussion

According to the Kruskal Wallis test and the post-hoc all groups differ significantly from each other, except for EA2 gr1 and EA2 gr2. This means that the null hypothesis is rejected. There is a significant difference between the means of the different EA's and their training groups. Only in the second algorithm the difference between training groups is not significant. Based on Figure 3 it can be concluded that the second EA outperforms the first EA. This was what was expected beforehand, based on the literary research. For the first algorithm the training groups significantly differ in the fitness by generalisation over the enemies. Figure 3 shows that the mean of the second algorithm was still ascending as the generations increased. This means that the second algorithm might actually be even better when using more than 50 generations. The EvoMan framework is provided with a base article describing the effectiveness of their demo-algorithms [3]. The article unfortunately uses individual gain instead of fitness to compare their algorithms, removing the possibility of comparing it with their paper. Finally, we see that the best player can only beat enemy 2, 5 and 8 (maybe 7 sometimes). We can therefore conclude that the algorithm of the base article outperforms our algorithm.

4 CONCLUSION

Overall, the performance of the second EA outshines that of the first algorithm. The performance of the EA's seems to be influenced by the amount of generations as the mean fitness increases as the generations increase. Both algorithms seem to find the best solutions at a comparable pace. For future research we propose more in depth work on the second algorithm and running it for more generations. Due to time constraints we were unable to fully test to see if the mean would converge around the best fitness for the second algorithm. The second algorithm has potential in becoming a great optimization solution to the EvoMan problem.

REFERENCES

[1] Pratibha Bajpai and M. Kumar. 2010. Genetic Algorithm - an Approach to Solve Global Optimization Problems. *Indian Journal of Computer Science and Engineering* 1 (10 2010), 199–206.

[2] Caruana, Eshelman, and Schaffer. 1989. Representation and hidden bias II: Eliminating defining length bias in genetic search via shuffle crossover. *11th international joint conference on Artificial intelligence* 1 (1989), 750–755.

[3] Karine da Silva Miras de Araujo and Fabrício Olivetti de Franca. 2016. Evolving a generalized strategy for an action-platformer video game framework. (2016).

[4] Fogel. [n. d.]. Evolutionary programming - Scholarpedia. ([n. d.]). http://www.scholarpedia.org/article/Evolutionary_programming

[5] David Bruce Fogel. 1993. Evolving artificial intelligence. (1993).

[6] Felix-Antoine Fortin, Marc Parizeau MARC Francois-Michel De Rainville, Marc-Andre Gardner, and Christian Gagn. 2012. GDEAP: Evolutionary algorithms made easy. *The Journal of Machine Learning Research* 13, 1 (2012), 2171–2175.

[7] Ahmad Hassanat, Khalid Almohammadi, Esra' Alkafaween, Eman Abunawas, Awni Hammouri, and VB Prasath. 2019. Choosing mutation and crossover ratios for genetic algorithms—a review with a new dynamic approach. *Information* 10, 12 (2019), 390.

[8] Ahmad Basheer Hassanat, Esra'a Alkafaween, Nedal A. Al-Nawaiseh, Mohammad Ali Abbadi, Mouhammd, Alkasassbeh, and Mahmoud Bashir Alhasanat. 2016. Enhancing Genetic Algorithms using Multi Mutations : Experimental Results on the Travelling Salesman Problem.

[9] Ho-Huu, Nguyen-Thoi, Truong-Khac Le-Anh, and Vo-Duy. 2018. An improved differential evolution based on roulette wheel selection for shape and size optimization of truss structures with frequency constraints. *Neural computing and applications* 29(1) (2018), 167–185.

[10] Hu and Di Paolo. 2009. An efficient genetic algorithm with uniform crossover for air traffic control. *Computers & Operations Research* 36(1) (2009), 245–259.

[11] Lipowski and Lipowska. 2012. Roulette-wheel selection via stochastic acceptance. *Physica A: Statistical Mechanics and its Applications* 391(6) (2012), 2193–2196.

[12] Pencheva, Atanassov, and Shannon. 2009. Modelling of a roulette wheel selection operator in genetic algorithms using generalized nets. *International Journal Bioautomation* 13(4) (2009), 10.

[13] DEAP project. [n. d.]. DEAP Documentation. ([n. d.]). https://deap.readthedocs.io/en/master/index.html

[14] Rosshairy Abd Rahman, Razamin Ramliand Zainoddin Jamari, and Ku Ruhana Ku-Mahamud. 2016. Evolutionary Algorithm with Roulette-Tournament Selection for Solving Aquaculture Diet Formulation. *Mathematical Problems in Engineering* 2016 (2016), 10. https://doi.org/10.1155/2016/3672758

[15] Syswerda. 2018. Uniform crossover in genetic algorithms. *In Proceedings of the third international conference on Genetic algorithms* 29(1) (2018), 2–9.

5 AUTHOR CONTRIBUTIONS

Max Zwager and Alexis Mourier have written the code for the first EA and the plotting. Mickey van Immerseel has written the code for the second EA and the statistical test. All three of us worked on writing the paper. We would like to thank Prof. Dr. A.E. Eiben and the course TA's for guiding the course.