Evolutionary Computing
Task 2: Generalist Agent

Group 23

Kleio Fragkedaki
Student number: 2729842

Matilda Knierim
Student number: 2700374

Carol Rameder
Student number: 2747982

Robin Stöhr
Student number: 2750340

1 INTRODUCTION

Evolutionary algorithms (EAs) are optimization algorithms inspired by the biological evolution. With the concept of the biological processes of selection, recombination and mutation of a population over generations, EAs are able to evolve in a complex environment based on a fitness function.

In our last paper, we investigated whether a dynamically adaptive diversity for individuals, implemented with the self-adaptive mutation method, is a better optimization method for training an agent in the EvoMan framework [3] than an EA with a standard Gaussian mutation method. Our results did not indicate that the self-adaptive mutation is superior. As a possible reason, we suggested that a different survivor selection strategy might enhance the positive implications the self-adaptive mutation method has to offer. Therefore, in the current paper, we aimed to take up our previous research and compare two EAs with the self-adaptive mutation method and different survivor selection methods, namely the comma strategy survivor selection and the age-based survivor selection. With that, we aim to gain insight into the effect of the balance between diversity and quality.

The two forces of diversity and quality are crucial for the performance of a high quality EA. Population diversity is accomplished due to the methods of recombination and mutation. It ensures novelty of the population over the time of generations and therefore avoids local optima and premature convergence. Quality is ensured by parent and survivor selection, which decreases diversity.

In order to balance the population diversity initiated by the self-adaptive mutation method and enhance population quality, we chose the "comma strategy"[1]. For comparison, we tested the stated EA against a second algorithm in which an age-based survivor selection method was implemented. Therefore, the second algorithm should have more diverse offspring and hence less quality. This experiment aims to answer the research question whether an EA with a comma survivor selection strategy, which displays a more balanced approach between diversity and quality, can be seen as superior as an EA with an age-based replacement strategy, and thus higher diversity and lower quality. We investigate the research question by comparing two EAs with a different survivor selection strategy in the EvoMan environment [3] using the evolutionary computation framework DEAP [2].

In the gaming framework EvoMan [3], an agent fights against enemies equipped with different weapons and movements. Movements of the agent include moving left or right, shoot, jumping, and interrupt the jump. The agent and enemies have 100 energy points, which decrease once they are hit by each other's projectiles. The game stops if one of the energy levels turn zero. For our experiments, the two algorithms are tested against two sets of enemies.

2 RELATED WORK

The self-adaptive mutation method was chosen due to previous research emphasizing its superiority compared to other mutation methods [5]. Self-adaption was defined as the dynamic changes of parameters. These changes happen during the evolution, based on the inclusion of parameters within a genetic encoding that is exposed itself to evolutionary process and pressures [1]. Moreover, the comma strategy was emphasized as a high quality survivor selection method [1]. This is due to the ability of the comma strategy to escape local optima by discarding all parents but keeping quality by the possibility to choose the highest quality individuals for the next generation.

3 ALGORITHM DESCRIPTION

In order to test our research question, two different EAs were used to train the a generalist AI agent in the EvoMan framework. The EAs were used to optimize the single-layer neural network of the agent, where the weights represent the genotype of each individual. The different genotypes are different configurations of the model and can be evaluated based on the fitness performance of the agent during the game. Both EAs use all standard methods of evolution. The different genotypes, or individuals, are evaluated on *fitness* to measure their quality. The fitness function used in this experiment was provided by the framework and can be described as:

$$fitness = 0.9 * (100 - e_e) + 0.1 * e_p - \log(t) \quad (1)$$

In the stated above equation, e_e represents the enemy's energy and e_p the player's energy, with values in the interval $e_e, e_p \in [[0, 100]]$. The parameter t denotes the number of time steps until the game is finished.

The *gain*, used for selecting the best individuals after creating all generations, is a simplified fitness function, not taking into account any weights or the time. The equation goes as follows:

$$gain = e_p - e_e \quad (2)$$

Based on *fitness*, parents are selected to create the next generation. Therefore, the optimal features are propagated to the next generation, most likely leading to an improvement.

Crossover represents the reproduction process, in which the genetic information of two or more individuals is combined and the resulted offspring represent the population of the next generation.

Furthermore, mutation stands for random changes in the genetic material of the offspring. In our experiment, we use self-adaptive mutation as a mutation method. The key concept is that the mutation step sizes are not set by the user; rather the parameters co-evolve with the solutions. The crossover and mutation operators are used to increase the diversity of a population. By applying these methods, the search space of our problem is well overall explored.

Two survivor selection methods are applied in our EAs for improving the elitism of our solutions, those being comma strategy survivor selection and age-based selection. In the following of the paper, we refer to the EA using comma strategy survivor selection as EA1 and the EA using age-based replacement survivor selection as EA2.

For the parent selection, a fitness-based method is used in order to find high-quality parents for improving offspring.

In the main loop of the algorithm, for each new generation, we select the parents for breeding. Crossover produces the offspring that are further selected or not for the next generation depending on the used method. Mutation diversifies the offspring, which are further evaluated according to their fitness and become the population of the next generation.

4 EXPERIMENTAL SETUP

In this selection, mutation, crossover and selection algorithms used will be explained in more detail. The DEAP framework [2] was used to embed the EAs and provided the parent selection method. The EvoMan framework provided us with a controller class called *demo_controller*, which uses a neural network to control the agent. The weights of this neural network are the chromosomes of the individuals. Both of our algorithms use a population size of $p = 100$ and a number of generations of $g = 25$.

4.1 Survivor Selection

In our experiment, we aim to evaluate the importance of the balance between diversity and quality by comparing two EAs with different survivor selection methods.

Algorithm 1: Comma Strategy

The survivor selection in our first algorithm is the *comma strategy*, or (μ, λ) -strategy. In this strategy, μ individuals create λ offspring. In our experiments we used $\mu = p = 100$ and $\lambda = 200$. That way we had twice the amount of offspring as the number of our population. For selecting the 100 offspring that were to replace our old population, we used a simple *select best* mechanism, where we took the fittest half of the offspring.

Algorithm 2: Age-based selection

For our second algorithm, we used the *age-based* selection. In it, we created the same amount of offspring as we had in our initial population, and replaced the whole population with all newly created offspring.

4.2 Parent Selection

For the parent selection, the *Tournament Selection* method from DEAP was used with a tournsize of $t = 20$. Hence, 20 random individuals from the current population are chosen to compete in a tournament against each other. The best individual with the highest fitness value out of the 20, wins the competition and is selected to be one of the parents to create offspring for the next generation. A high tournament size potentially leads to a decline of diversity, since the best performing individuals are selected too often. However, a low tournament size could harm the quality and diversity of the next population, since it increases the randomness in which individuals will be selected. For the experiment, different test runs indicated that a tournament size of 20 is a fitting middle ground for a population of 100 individuals.

4.3 Initialization

For the initialization of the population, the high performing individuals, classified as beating five or more enemies in the previous runs, were seeded in the initial population. With this method, we aimed to give our population a "head-start" and further improve the quality of genotypes in following runs. Moreover, every gene in the genotype of the other individuals consists of a random float number in the range $x_i \in [-1, 1]$.

4.4 Crossover

In both EAs, the method *Multi-parent uniform crossover* was used. Each chromosome is copied from one of the parents, with an equal probability for each child. We used the number of four parents for the crossover to keep balance between the integrity and the diversity of the chromosomes.

4.5 Mutation Algorithm

As a mutation method, self-adaptive mutation with n σ's was used for both EAs. Every individual has n additional parameters $\sigma_1 \ldots \sigma_n$, one for each chromosome $x_1 \ldots x_n$. The σ values are mutated first in each mutation step, as seen in equation (3). Then, the corresponding x values are changed according to their sigma values as shown in equation (4).

$$\sigma_i' = \sigma_i \cdot e^{\tau' \cdot N(0,1) + \tau \cdot N_i(0,1)} \tag{3}$$

$$x_i' = x_i + \sigma_i' \cdot N_i(0, 1) \tag{4}$$

In eq. (3), the constant parameters τ and τ' represent the learning rates, with τ being a coordinate wise learning rate and τ' a general one. The values are set to:

$$\tau' = 1/\sqrt{2n}, \qquad \tau = 1/\sqrt{2\sqrt{n}}$$

$N(0, 1)$ represents a draw from the normal distribution around the center 0 and a standard deviation of 1. Moreover, we aimed to set the minimum *expected* change of the old chromosome to at least 0.05 in either direction and, therefore, every sigma value has a lower bound of $\epsilon_0 = 0.05$. These bound values are additionally used for the random sigma initialisation, with each value being chosen from an interval out of $\sigma_i \in [\epsilon_0, 2 \cdot \epsilon_0]$. The sigma values are given to the offspring in the same way as the chromosomes are.

5 EXPERIMENTAL RESULTS

To test our research question, an experiment was held using two EAs with different survivor selection methods, while the same initialization, mutation, crossover and parent selection operators were applied. The algorithms were tested in the EvoMan framework [3] against all eight enemies. Each EA was trained against two sets of enemies. The first set contains the enemies *Flashman, Airman, Woodman,* and *Heatman* and the second one the enemies *Metalman, Crashman, Bubbleman,* and *Quickman.*

In order to evaluate the two EAs, two types of plots were created. First, the line plots, displayed in Figure 1, depict the average of mean and maximum fitness of each generation and their standard deviation. Secondly, the box plots, shown in Figure 2, represent the individual gain of the best individual of each run tested 5 times against all 8 enemies for both algorithms split in the two training data sets. Moreover, we compared the performance difference of the algorithms with a t-test.

The line plots show an interesting picture of the two EAs for the different enemy groups. In the first line plot, it can be seen that EA1 has a lower maximum fitness, but a higher average fitness than EA2 for the first enemy group. Second, EA2 shows a higher variance than EA1. The second plot, with both algorithms being trained against enemy group 2, displays that EA1 shows higher fitness maximum and average than EA2. As in the first line plot, the standard deviation of EA2 is higher than for EA1. It has to be noted that the fitness already starts at a high level. This is due to the seeded initialization.

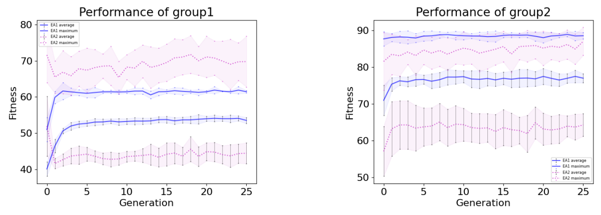

Figure 1: Line plots

The box-plots represent the best individual of each run tested 5 times against all 8 enemies. In relevance with the line plots, the box-plots of the first training group show that EA2 has a higher average gain and variance compared to EA1. Moreover, for the second training group, the variance is lower and EA1 has higher gain and mean than EA1. The t-test emphasized that the EAs do not significantly differ from each other, with a *p*-value of 0.97.

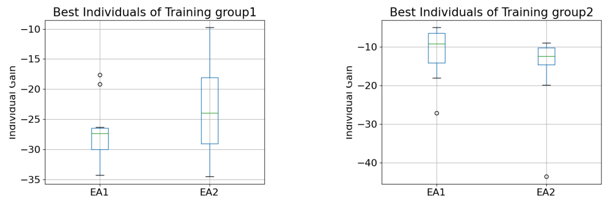

Figure 2: Box plots comparing the two groups

The Table 1 shows the average results of our best individual. The results show that the individual consistently wins against *Airman, Metalman, Crashman, Quickman*, and inconsistently against *Bubbleman*.

enemy	enemy life average	player life average
Flashman	78	0
Airman	0	37.2
Woodman	58	0
Heatman	74	0
Metalman	0	42.16
Crashman	0	5.8
Bubbleman	20	25.6
Quickman	0	47.2

Table 1: Average result of our best individual, over five runs.

6 ANALYSIS AND DISCUSSION

Our results emphasize that EA2 is more diverse than EA1. Moreover, EA2 performs better when the algorithms were trained on the first enemy group, while EA1 performs better when being trained on the second enemy group. The t-test did not show that the two EAs significantly differ from each other in terms of performance. In comparison to the baseline paper [4], Table 1 highlights that our algorithm is performing worse than the ones tested by [4]. While our algorithm only consistently beats 4 enemies with an average maximum player life of 47.2, the proposed NEAT algorithm beats all enemies consistently with a maximum player life of 94.

One possible reason for EA2 having higher values for the first enemy group could be the higher diversity of EA2, which allows it to explore the search space more generally than EA1, which has a high selection pressure. Even though EA2 has lower average fitness values due to its higher diversity acceptance, it performs better than EA1 at a generalist task when facing the group of enemies with diverse behaviour. Based on the gain values, it can be deducted that the first enemy group is harder to beat than the second one and hence might give an advantage to an EA with higher diversity. Potentially, a diverse training set might be favourable to train a diverse EA.

For the second group of enemies, EA1 has better values than EA2 for both maximum and average values. Therefore, we can observe that the EA1 performs better at training for a specific task, where the environment and the behaviour of the enemies do not differ as much as for enemy group 1. Finally, because of its higher diversity, EA2 has a larger variance than EA1, which make the population converge in a small interval of fitness values.

Taken together, our results shed light on how a training and testing set could be crucial for picking the diversity of an EA and emphasise that yet again, an EA has to be tailored individually to a problem. We reason that our training sets might have been too different in diversity and therefore lead to inconsistent results when being tested against the same enemies. Therefore, for our optimization problem, an EA using the comma survivor selection strategy, with a balanced approach between diversity and quality, cannot be seen as superior to an EA using an age-based survivor selection strategy with higher diversity.

7 CONCLUSION

To conclude, for our optimization problem, the two EAs do not show a significant difference. The interesting finding that each EA performs better when being trained on a different training set sheds light on a greater complexity of a diversity vs. quality balance than we assumed. We therefore encourage future research to continue investigating the balance between diversity and quality, especially with having the effects of training and testing data sets in mind.

8 GROUP WORK SET-UP

We decided on the research question as a group. Each group member then tried to experiment with different algorithms that explore the effect of higher diversity and a more balanced approach due to higher quality. We wrote the report together as a group.

REFERENCES

[1] A. E. Eiben and James E. Smith. *Introduction to Evolutionary Computing*. Natural Computing Series. Springer, 2003, pp. 1–300. ISBN: 978-3-662-05094-1.

[2] Félix-Antoine Fortin et al. "DEAP: Evolutionary Algorithms Made Easy". In: *Journal of Machine Learning Research* 13 (July 2012), pp. 2171–2175.

[3] Fabricio Olivetti de Franca et al. *EvoMan: Game-playing Competition*. 2020. arXiv: 1912.10445 [cs.AI].

[4] Karine da Silva Miras de Araújo and F. O. D. França. "Evolving a generalized strategy for an action-platformer video game framework". In: *CEC*. 2016.

[5] J. Teo. "Self-adaptive mutation for enhancing evolutionary search in real-coded genetic algorithms". In: *2006 International Conference on Computing & Informatics* (2006), pp. 1–6.