Evolutionary Computing - Task II: Generalist Agent
Group 65

18/10/2021

Bruna Aguiar Guedes
Student Number: 2698211

Enrico Calleris
Student Number: 2692023

Caterina Fregonese
Student Number: 2737116

Romy Vos
Student Number: 2595704

1 INTRODUCTION

Evolutionary algorithm is an umbrella term used to describe population based stochastic direct search algorithms that mimic natural evolution [3]. The idea behind EAs lies in the problem-solving nature of evolution, whose principles can be better defined taking this perspective into account. In this way, adaptation can be described as the solving of a particular (set of) problem(s) in a given environment. Similarly, survival of the fittest can be explained as having better chances of being rewarded with a higher level of contentment, faster growth and reproductive advantages for those individuals who are the best problem solvers [8]. In this report, we will take a closer look at these concepts in the context of EAs, where natural selection is replaced with artificial selection via the fine-tuning of input parameters to optimize the algorithms' performance. This will be implemented in the context of EvoMan, a python framework for testing optimization algorithms with artificial agent controllers in electronic games [4]. One of the common artificial intelligence (AI) applications in electronic games consists in fact in making an artificial agent learn how to execute some determined task successfully in a game environment, which can be configured to run in different experiment modes. After implementing two instances of the same EA that use the simulation mode "individual evolution" of the framework to train a generalist agent, we used two groups of enemies to experiment with them and, for each group, we made an independent experiment evolving a generalist agent. We then tested, analyzed and compared their behaviour. Our aim was that of selecting high performing individuals that are capable of winning against the highest number of enemies. More specifically, we implemented the classical fitness function in EA1 and a function with penalty in EA2. Our expectations were that, by applying a penalty to the fitness function in EA2, its average performance would be negatively impacted (e.g., fitness values would decrease and variance would increase). However, this would be a valuable trade-off for finding a new maximum regarding individual performance.

2 METHODS

In this section, we explain in detail the methods used in the framework for testing optimization algorithms with artificial agent controllers in EvoMan.

Evolutionary Algorithm 1

General Settings. The two instances of the same EA we used to test a generalist agent against two groups of enemies (group 1: enemies 2, 5, 7 and group 2: enemies 4 and 8) consist of a population of neural networks with a single hidden layer of 10 nodes, with sigmoid activation function for the hidden and output layer. The input vector consists of 20 sensors which, according to how the values are modified, yield different output results.

Genotype. The genome of each individual in a population consists of 265 (= (20 + 1) • 10 + (10 + 1) • 5) weights: these are continuous values that can be thought of as alleles in a chromosome.

Population Size and Number of Generations. To find well performing individuals we take as our starting point a population of 100 individuals with random initialization, which populates the initial population with completely random solutions. Once the sensors' parameter are adjusted the population can evolve until it reaches a termination point which, in our case, is set at 20 generations.

Fitness Function and Individual Gain. The performance of each individual was evaluated using the following fitness function:

$$f = 0.9 \cdot (100 - e) + 0.1 \cdot p - logt, \qquad (1)$$

where p refers to the player's energy level after the end of the fight, e to enemy's energy level, and t to the number of timesteps needed to reach the end of the fight. The function aims at maximizing the energy of the player by the end of the level, while minimizing the energy of the enemy. The last term penalizes fights that take too long to be finished. We then evaluated the best individuals to save for each run based on the individual gain:

$$IG = p - e \qquad (2)$$

where IG < 0 indicates a loss, while IG > 0 indicates a player win. In our experiment, the fitness was calculated as a consolidated mean of fitnesses for the various enemies in the group (mean-std). Following the same approach, we calculated the consolidated gain.

Mutation Operator. The type of mutation we used is called non-uniform. This idea stems from the concept of self-adaptation, where the algorithm controls the setting of these parameters itself embedding them into an individual's genome and evolving them [9]. The essential feature is that the step sizes are also included in the chromosomes and they themselves undergo variation and selection [6]. We have decided to apply uncorrelated mutation with one step size to test both instances of our EA. In this type of mutation, the same distribution is used to mutate all x_i, therefore we only have one universal strategy parameter σ embedded in the individual, which is mutated each time step by multiplying it by a term e^{Γ}, with Γ a random variable drawn each time from a normal distribution with mean 0 and standard deviation τ that must be implemented before the solution vector values x_i.

Crossover Operator. As for the other operator used to combine genetic information and to pass it from the parents to the offspring, the crossover, we decided to use discrete recombination.

Survivor Selection Method. We opted for a fitness based survival selection method, where the children tend to replace the least fit individuals in the population. We used (μ , λ) with elitism as selection operator (where μ indicates the size of the parent population, while λ is the size of the offspring), which combines the population of parents with their offspring and sorts them all based on fitness (from high to low). The genome with the highest fitness is then selected and put in the new population. In our case every new generation was made up of both parents (20%) and their offspring (80%). Following this mechanisms, genomes are picked until the desired population size is reached. This concept is known as elitism and guarantees that the solution quality obtained by the EAs will not decrease from one generation to the next [1].

Evolutionary Algorithm 2

All parameters and evolutionary mechanisms used, except for the fitness function, are the same as EA1.

Fitness Function. The performance of each individual was evaluated using the following fitness function:

$$eval(\overline{x}) = f(\overline{x}) + W \cdot penalty(\overline{x}), \tag{3}$$

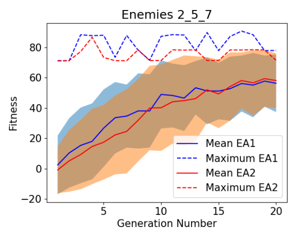
where f is the objective function, *penalty(x)* is zero if no violation occurs and is *1* otherwise, and W is a user-defined weight prescribing how severely constraint violations are weighted. To adjust the evaluation function over time, the static parameter W can be replaced by a function *W(t)* using the following formula:

$$W(t) = (C \cdot t)^{\alpha}, \tag{4}$$

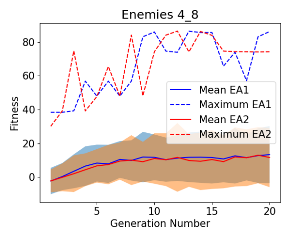
where C and *alpha* are constants. We set C at 0.1 (this value is meant to add a small penalty to the offspring that is less fit than the parents, without giving more weight to this fact than to the fitness itself) and alpha at 1.5 based on [4], as it was found that for best performance $\alpha \in \{1, 2\}$. Penalty functions, whose aim is that of converting constrained problems into unconstrained problems by introducing an artificial penalty for violating a certain constraint, are perhaps the most common approach for tackling constrained optimization problems using evolutionary algorithms [11]. For EA2 we used penalties, which are dependent on the current generation number by way of increasing as the generations evolve. More specifically, we used an offline policy to establish penalties' values, where the rules to tune the feedback parameters are calculated in advance before the estimation process [10].

Experimental Setup

We tested EA1 and EA2 against two groups of enemies to investigate whether changing the fitness function would yield different output results. For the first group, we selected three enemies with increasing level of difficulty: enemy 2 (Airman), 5 (Metalman) and 7 (Bubbleman). For the second group, we selected two enemies: enemy 4 (Heatman) and enemy 8 (Quickman). We decided to use this strategy because, as described in [4], evolving a general agent capable of beating the complete set of enemies by training on a subset of the enemies set has proved to be an efficient and successful strategy (which optimizes running time and produces generalizable results). We picked enemies that were not tested in the baseline paper [4], as we were interested in analyzing how our evolutionary algorithms would perform against different enemies, with the hope and intent of bringing new insights to the understanding of the relation between parameters tuning and an EA's behaviour change. In the training phase, the EAs were tested ten times independently with each group of training enemies. We kept track of the EAs' behaviour by analyzing their output values. Using these values, we computed the maximum fitness, the consolidated mean, the SD and the consolidated gain for every generation. In the testing phase, we compared EA1 and EA2 behaviour for the two groups of enemies we selected, testing each enemy five times with our final best solution for each of the ten independent runs. The resulting (consolidated) gains and their spread for the ten best individuals are presented in box-plots in the result's section (Figure 2). We then picked the BIs among all individuals with the mean of the consolidated gain as reference measure. Our main aim was finding the best possible individual, the one who could beat the highest number of enemies



(a)



(b)

Figure 1: Mean and Maximum Fitness for (a) Enemies [2,5,7], (b) Enemies [4,8], with *SD* of the mean as shaded region.

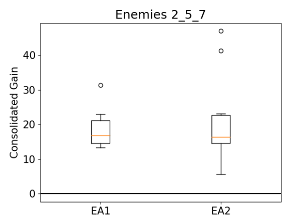
(whose life point's average is higher than the enemies'). We did this by testing each of the best individuals resulting from our previous calculations five times against all eight enemies. After each run, we saved both player and enemy's life.

3 RESULTS AND DISCUSSION

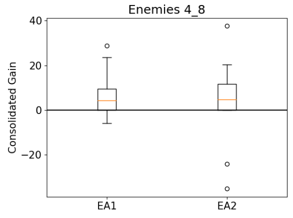
After implementing the methods previously described, we analyzed and plotted the results. We started by plotting the means and the maximum values for both EA1 and EA2's fitness which, in the following line plots (Figure 1), are compared per group of enemies. As it can be observed, there is quite a big difference between the two groups of enemies and the EAs' performance. Instead, there appears to be little to no variation in the EAs' performance against the same group of enemies. When comparing the mean fitness values, both EAs reach a plateau in later generations when tested with the first group of enemies [2, 5, 7], whereas we can observe an almost constant plateau phase for enemies [4, 8]. When comparing the maximum fitness values we can see that, especially for the EAs tested with enemies [4, 8], there is a significant difference at an individual level between EAs, with EA2 producing more high performing individuals. The stochastic distribution of the maximum fitness values supports our hypothesis that applying a penalty to the fitness function can yield better results at an individual but not at an average level, with EA2 producing the best performing individuals.

In the following box plots (Figure 2), we compared the mean of the consolidated gain of the BIs for each run. It can be observed that it is easier to beat the first group of enemies [2, 5, 7] (the gain is higher than 10) than it is to beat the second group of enemies [4, 8] (the gain is lower than 10). The average gain for BI never goes below 0, meaning that, on average, the group of enemies we chose to test our EAs are beatable. Moreover, the mean values we obtained for EA1 and EA2 are almost identical, meaning that there is no significant difference between the EAs at an average level. However, at an individual level, we can notice that EA2 can produce high performing individuals, with an individual gain of around 40. This finding supports our hypothesis that applying a penalty to the fitness function is a good strategy to generate better results at an individual rather than at an average level.

None of the samples was found to adhere to normality, so a Kolmogorov Smirnov test was used to assess significant differences



(a)



(b)

Figure 2: Boxplots for best individuals run of (a) Enemies [2,5,7], (b) Enemies [4,8]. Straight line shows a gain of 0.

Table 1: Statistical comparison for BIs of EA1 vs. EA2 for different groups of enemies. *KS = Kolmogorov-Smirnov Test.*

Enemy Group	Statistical Test	D-Statistic	p
[2, 5, 7]	KS	0.2	.99
[4,8]	KS	0.2	.99

Table 2: Best solution's mean player and enemy energy against all enemies. The averages were calculated on a total of 5 runs. *Best solution: run 5, EA2 enemy group [2, 5, 7].*

Enemy	Player Energy	Enemy Energy
1	0.0	80.0
2	74.8	0.0
3	0.0	36.0
4	0.0	60.0
5	49.1	0.0
6	0.0	90.0
7	43.0	0.0
8	46.6	0.0

between EA1 and EA2 BIs for our two groups of enemies. The results we obtained are non-significant.

The absolute best individual (based on consolidated gain) was found to be individual number five of EA2 trained against enemies [2, 5, 7]. The following table (Table 2) shows the average values of player and enemy life for the best solution over 5 runs against all eight enemies. By comparing our results with those presented in a pioneer paper [4], we noticed that the values we obtained are slightly better or similar to those obtained by [4]. EA2 was in fact able to beat four out of eight enemies, which is the same maximum number of enemies beaten in the pioneer paper. Interestingly, the means of the player's energy are well above zero, a very different results than what was observed in [4], where the final mean energy of the player who won against the maximum number of enemies (4) ranged from 1 to 9. Moreover, enemy five is beatable with a similar value of final mean energy by both EA2 (49.1) and the EA used in [4] (48).

To sum up, when analyzing the consolidated gain, EA1 and EA2 behaviour's is similar. However, we believe that this measure might not be descriptive with regards to our EAs' performance and to the two group of enemies we selected. Further measures might be used to analyze our data. Recognizing the need for better measures and better ways of analyzing data can indeed help researches better interpret data's quality and content. Choosing the most appropriate measure is often a trial-and-error (and inherently time-consuming) process. In our case, we observed good fitness values on average but, due to time constraints, we could not analyze our data using different (and perhaps more appropriate) statistics, which might have led to a different interpretation of our results. As for the two groups of enemies we picked, we wanted them to be representative of all eight enemies. For this reason, we decided to opt for enemies that have different difficulty levels, so that in the same group there are enemies that are easier and enemies that are harder to beat. However, due to limited time constraints, it is difficult to know whether our two groups of enemies and the results we obtained when testing the two EAs can be generalized or not. This could be an interesting topic to explore in future research.

4 CONCLUSION

The goal of this paper was to examine how an EA's ability to produce BIs is impacted by a fitness function with penalty in the context of EvoMan (a videogame framework). In the proposed approach, a penalty factor is gradually and systematically increased during training as the iteration episodes proceed one generation after the other, thus allowing a generalist agent to evolve with the aim of beating the highest possible number of enemies. This approach was found to be effective at an individual but not at an average level. The individual with the highest performance was generated by EA2, where the fitness function with penalty was implemented. Using a fitness function with penalty is a strategy that has already been adopted in the past and that has been found to be accurate, efficient, robust [2] and easy to implement [5]. Moreover, it has been shown to facilitate the location of the global optimum and the avoidance of local optima [7]. For this reason, we think this might be an interesting strategy to further explore in future works. More specifically, we suggest to train the EAs with different parameter tuning for C and α in order to find the best possible values to use in the fitness function with penalty. Results' analysis is in fact enhanced when using optimal parameters. Due to limited time constraints, we were not able to do stick to this strategy, as we had to consider the trade-off between computational power and time. We also suggest to train the algorithms with more than two groups of enemies so as to allow results' generalizability. Overall, we believe that our could be a useful model from which other researchers can build on. Parts of this study are of an explorative nature and our results suggest further analysis, as indicated in the discussion section. Just like science, data analysis is in fact an iterative process where we search for truth by successive mistakes, until we find good enough parameters that can correctly analyze and model data to discover and extract useful information. For this reason, we believe the strategy we adopted is promising and could be further explored in future research.

CONTRIBUTIONS

All members of the group equally contributed to the project.

REFERENCES

[1] Shumeet Baluja and Rich Caruana. 1995. Removing the Genetics from the Standard Genetic Algorithm. In *ICML*.

[2] Helio Barbosa and Afonso Lemonge. 2008. *An Adaptive Penalty Method for Genetic Algorithms in Constrained Optimization Problems*. https://doi.org/10.5772/5446

[3] Thomas Bartz-Beielstein, Juergen Branke, Jorn Mehnen, and Olaf Mersmann. 2014. Evolutionary Algorithms. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 4 (05 2014). https://doi.org/10.1002/widm.1124

[4] Karine da Silva Miras de Araujo and Fabricio Olivetti de Franca. 2016. Evolving a generalized strategy for an action-platformer video game framework. In *2016 IEEE Congress on Evolutionary Computation (CEC)* (Vancouver, BC, Canada). IEEE, 1303–1310. https://doi.org/10.1109/CEC.2016.7743938

[5] Elmer P Dadios and Jamshaid Ashraf. 2006. Genetic algorithm with adaptive and dynamic penalty functions for the selection of cleaner production measures: a constrained optimization problem. *Clean Technologies and Environmental Policy* 8, 2 (2006), 85–95.

[6] A.E. Eiben and J.E. Smith. 2015. *Introduction to Evolutionary Computing*. Springer Berlin Heidelberg, Berlin, Heidelberg. https://doi.org/10.1007/978-3-662-44874-8

[7] S Kazarlis and Vassilios Petridis. 1998. Varying fitness functions in genetic algorithms: Studying the rate of increase of the dynamic penalty terms. In *International conference on parallel problem solving from nature*. Springer, 211–220.

[8] Arnold Loof. 2004. Evolution: The problem solving strategy as the basic unit of adaptation? *Belg. J. Zool* 134 (08 2004).

[9] Silja Meyer-Nieberg and Hans-Georg Beyer. 2007. *Self-Adaptation in Evolutionary Algorithms*. Vol. 54. 47–75. https://doi.org/10.1007/978-3-540-69432-8_3

[10] Kartikeya Rambhatla, Simone Evaldo D'Aurelio, Mauro Valeri, Emanuele Polino, Nicolò Spagnolo, and Fabio Sciarrino. 2020. Adaptive phase estimation through a genetic algorithm. *Phys. Rev. Research* 2 (Jul 2020), 033078. Issue 3. https://doi.org/10.1103/PhysRevResearch.2.033078

[11] Özgür Yeniay. 2005. Penalty Function Methods for Constrained Optimization with Genetic Algorithms. *Mathematical and Computational Applications* 10 (04 2005), 45–56. https://doi.org/10.3390/mca10010045