Evolutionary Computing - Task 2: Generalist Agent

Group 44 - 12/10/2020

Karin Lintzen
Vrije Universiteit
2630038

Jorrim Prins
Universiteit van Amsterdam
11038934

Jelle Steenbeek
Vrije Universiteit
2592245

Xingkai Wang
Vrije Universiteit
2668688

1 INTRODUCTION

The main principle of Evolutionary Computing (EC) is to apply Darwinian principles to automate problem-solving. The fundamental concept behind EC is that there is a population of individuals representing solutions to a problem and environmental pressure, causing natural selection. Various processes exist for population initialization, parent selection, offspring creation by crossover, mutation and survivor selection and these processes basically mimic natural selection as we know it from Darwin's principles. Performance of the population, in this research measured by a specified fitness measure, increases by the process of natural selection [2]. The population in this research consists of single-layer neural networks that aim to win an electronic game called *EvoMan* [5]. EvoMan is game including a *player* and an *enemy*, both starting with 100 life points and aiming to defeat the other by taking away all of its life points. More detailed description of EvoMan's specifics can be found in the work by Miras & de França [4].

The process behind Evolutionary Algorithms (EA's) automatically leads to a trade-off between complexity of the evolutionary framework and individuals in the populations. This directly follows from the well-known trade-off between performance and computational time, as one can not choose both identities to be complex without adding computational time. Two EA's will be compared in this research, one relatively simple algorithm with a more complex neural network structure (the 40-node algorithm from now on) and one more complicated algorithm with a basic neural network (the 10-node algorithm from now on). To investigate the trade-off, the research question will be: Can a 40-node algorithm find an optimal individual for a generalist agent problem that performs better than the optimal individual of a 10-node algorithm? Performance will be evaluated on two components, the speed of convergence and the quality of the final solution, with the latter given priority.

2 THEORETICAL BACKGROUND

The two major differences between the tested EA's can be found in the settings for mutation, the survivor selection mechanism and of course the number of hidden nodes in the neural network. Plenty of research has been performed on these processes and will be discussed below.

The research of Cazacu [1] compares three different mutation methods in uniform, polynomial and Gaussian mutation. Evaluating models by the average fitness value of the highest 20 percent of the scores clearly shows a preference for Gaussian mutation. The process of Gaussian mutation as described in the work by Cazacu [1] is however computationally expensive as every value in an individual is mutated (or not) separately. The mutation probability decides per value if it is mutated or not, this can be simplified by deciding for a complete individual if it will be mutated. The complex neural network in this research will therefore contain mutation that is performed over a complete individual, adding a Gaussian mutation to every value in an individual simultaneously if it is chosen to be mutated.

Another way to adjust the complexity of an EA is by altering the selection methods for parent and survivor selection. The least complex method for parent selection would be the linear or exponential ranking method which pick individuals based on ranking their fitness values, where individuals with higher fitness values have a higher chance of being picked as a parent. More complex selection methods use tournament structures, with a defined number of opponents in the tournament. Ranking and tournament methods are compared in the paper of Razali & Geraghty [6]. They conclude that these selection methods have equal performing results for the best solution for most tested instances, the difference is small for the remaining instances. The biggest difference between these two selection methods is that the ranking method converges a lot slower to the best result than the tournament approach. For some instances, convergence with ranking selection needed more than ten times as many iterations. Selection for the EA's in this research is done by exponential ranking selection, $(\mu + \lambda)$ selection and round robin tournaments.

The complexity trade-off under examination in this research is extended with the number of hidden nodes in the underlying neural networks, as Zou et al. [7] come to the logical conclusion that having more hidden nodes results in a higher computational time. They also conclude that, for their problem, the optimal trade-off between the classification rate and computational time occurred at 17 nodes. However, to amplify the influence of this parameter setting, this research will use a higher number of hidden nodes. Other extensions could be made based on the results by Hayashi et al. [3]: they find that EA's with single-layer neural networks can only be used to solve linear separable problems. The Generalist Agent problem in this research might be too hard to solve with a single-layer network and a multi-layer network might be a helpful adaptation.

3 METHODS

The initial approach of this research was examination of the interaction between organization of the EA and the structuring of the individuals in it. Building on the neural network used in the specialist assignment, which uses a single hidden layer of 10 nodes and sigmoid activation functions for the hidden and output layer, alternative networks were created and their performance was tested. Networks were extended or adjusted by using different activation functions, different numbers of nodes and even additional hidden layers. The finalized two EA's in this research consist of a population of neural networks with 10 and 40 nodes in a single hidden layer, with sigmoid activation functions. The 40-node algorithm uses some simplified evolutionary components to compensate for the increase in computational time compared to the 10-node algorithm. As in the previous research, the input vector consists of 20 values and individuals in the algorithms therefore have 1045 $(= (20 + 1) \cdot 40 + (40 + 1) \cdot 5)$ and 265 $(= (20 + 1) \cdot 10 + (10 + 1) \cdot 5)$ weights respectively.

The generalist task of finding a solution that performs well against multiple enemies in the EvoMan game asks for various differences compared to the specialist objective: the EA has to be trained on multiple enemies so more generations are needed for convergence. To make up for the additional generations and the additional training instances (caused by the use of multiple enemies simultaneously) the population size is kept relatively small to 30 individuals. Evaluation metrics also have to be adjusted to the generalist setting, starting with the fitness function: the fitness function from the specialist assignment is used for every individual enemy and the final fitness is equal to the average minus the standard deviation over the trained enemies. Whereas selection in the evolutionary process is based on the adjusted fitness measure, final performance of the individuals is measured by the total *Gain*, summing the Individual Gain (IG) over all available enemies in the EvoMan framework:

$$Gain = \sum_{n=1}^{8} \left(life_{player} - life_{enemy} \right) \tag{1}$$

The initial population of the EA is randomly initialized with weights between $[-1, 1]$ and is exposed to two different evolutionary processes: one relatively expensive (in a computational sense) process to be used with the 10-node individuals and one simple process to compensate for the extra computational costs of using 40-nodes per individual. The simple process selects parents by exponential ranking and creates 1 to 5 offspring by arithmetic combination with a randomly chosen *alpha*. It subsequently mutates these offspring by mutating all weights of an individual with a specified mutation probability that decreases linearly over time. The Gaussian mutation in this process also has a linearly decreasing standard deviation over time. Survivors are selected by the $(\mu + \lambda)$ method. This entails that the parent population consists of μ individuals and the offspring consists of λ individuals. Of the combined parent and offspring population, the best μ individuals are picked as the next generation's population.

The alternative evolutionary process mutates offspring with equal (decreasing) probability and (decreasing) Gaussian standard deviation, but decides on every weight of an individual separately. It also limits offspring creation to 3 per parent couple. Survivor selection in the alternative case is done by a Round Robin tournament against 10 randomly chosen individuals, choosing 29 survivors and always adding the best individual from the previous population. As in the specialist case, the most underperforming individuals (30 percent) are swept after 5 generations with a non-improved optimal solution and replaced by randomly initialized individuals.

Two groups of two enemies ($[1, 6, 7]$ and $[2, 4, 8]$) are used to train both EA's. These groups are organised based on small initial experiments determining the similarities between enemies, the EA's tend to learn their generalist application best when an enemy group consist of enemies with a lot of variation between them. The complete evolutionary process is repeated 10 times for each training group and each algorithm to account for the randomness of the process and the game. The best individual of every replication plays against all enemies 5 times, this helps present the final results of the converged evolutionary processes.

4 RESULTS

Figure 1 presents the mean and maximum fitness values over generations, for 10-node and 40-node algorithms and for both training groups. These values are averages over 10 replications and bandwidths around these lines represent the standard deviation over the 10 replications. A first very remarkable result can be observed: both mean and maximum fitness values of the 10-node algorithm are higher than the values of the 40-node algorithm in the final generation for training group 1, while the opposite can be observed for group 2. For group 1, mean fitness values for both algorithms behave similarly, the 10-node algorithm steadily outperforming the 40-node algorithm by a little bit. The maximum values however go side by side for the first generations, after which the 10-node algorithm starts outperforming the alternative. A similar result can be observed for training group 2, however the conclusions about mean and maximum values and 10- and 40-node algorithms are opposite: maximum values behave similarly but with a steady advantage for the 40-node algorithm, mean values are comparable for the first generations and are thereafter better for the 40-node algorithm.

Possibly the most interesting observation on these plots is the absolute values of the mean and maximum fitness, as the scale of the y-axis on both graphs is equal it clearly shows an advantage for the second training group. All 4 lines in the plot end up above a value of 60 in the final generation, whereas the lines for the first group end between a value of 30 and 50. To assess the ultimate performance of the EA's, the individual of maximum fitness in the final generation is of most importance as this individual solution will perform the task at hand. These individual solutions will vary over

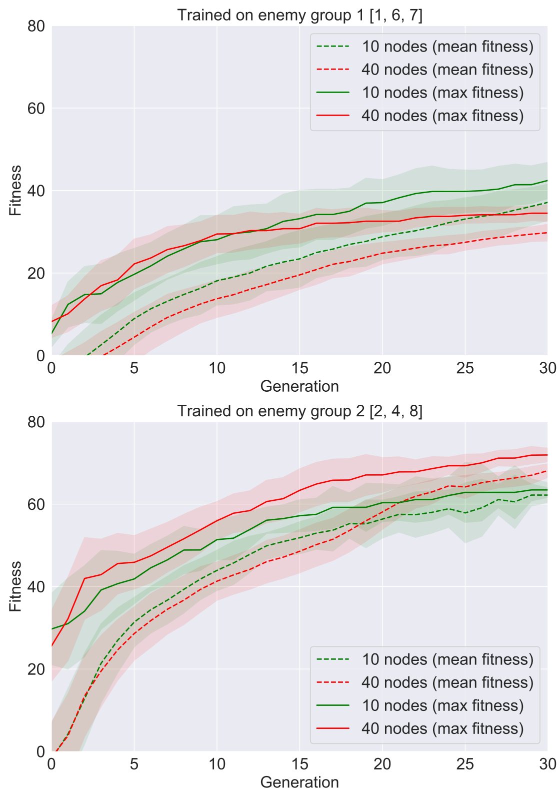

Figure 1: Fitness evolution over generations for training group 1 and 2 and for both algorithms

the 10 replications in the experiment and the average over these 10 replications can be compared to the average of the best individuals of another EA. A t-test is performed with H_0 defined as the mean fitness (over 10 replications) of the maximal-performing individual of an algorithm being equal to this mean fitness of another algorithm. All 6 possible combinations of algorithms to compare are evaluated and Table 1 provides results of these t-tests. H_0 for the across-group comparisons can be rejected for every comparison at the 1 percent level, H_0 for comparing algorithms that are both trained on group 1 can be rejected at the 5 percent level. No significant difference between the algorithms both trained on group 2 can be shown.

Table 1: T-test on maximum fitness after 30 generations

Model 1	Model 2	t-value	p-value
10 nodes, group 1	10 nodes, group 2	-4.37	<0.005
10 nodes, group 1	40 nodes, group 1	2.86	0.011
10 nodes, group 1	40 nodes, group 2	-4.38	<0.005
10 nodes, group 2	40 nodes, group 1	6.60	<0.005
10 nodes, group 2	40 nodes, group 2	-1.05	0.306
40 nodes, group 1	40 nodes, group 2	-5.93	<0.005

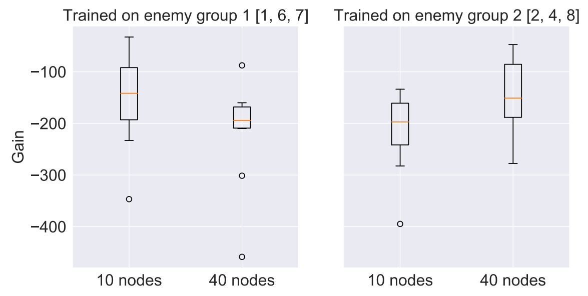

Figure 2: Boxplot of Gain for both EA's per training group

The boxplots in Figure 2 show the total Gain for the two GAs, presenting two plots that almost look mirrored. The averages and quartiles for the 10-node algorithm on group 1 are similar to the values for the 40-node algorithm on group 2, and the other way around. Some outliers can be found, especially when training on group 1. The very best solution based on total Gain comes from the 10-node algorithm trained on group 1, it is slightly better than the 40-node algorithm trained on group 2. Comparing the best performing solutions from this research to the results generated by the Genetic Algorithm specified as GA10 in the baseline paper [5] is done by the mean and standard deviation of the Gain in Table 2. All four models in this research show to severely underperform compared to the best GA10 model, but the 10-node algorithm trained on group 1 and the 40-node algorithm trained on group 2 outperform the worst GA10 model. Note the high standard deviations compared to the GA10 models, presented in the final column of the table.

The very best solution from this paper comes from training our 10-node algorithm on group 1, and evaluation of this generalist agent is done by running it against all enemies 5 times. Resulting

Table 2: Mean and standard deviation of Gain per agent, compared to GA10 agents from baseline research

Agent	Trained on	Mean	Std
10-node	[1,6,7]	-152	91
10-node	[2,4,8]	-216	79
40-node	[1,6,7]	-215	101
40-node	[2,4,8]	-144	74
Best GA10	[2,5,6]	-43	7
Worst GA10	[1,5,6]	-201	20

player and enemy life, averaged over the five runs, is presented in Table 3 and the optimal solution is able to beat half of the available enemies in the EvoMan framework.

Table 3: Average player and enemy life for optimal solution

Enemy	1	2	3	4	5	6	7	8
Player life	44.0	30.0	0.0	0.0	50.8	0.0	0.0	47.8
Enemy life	0.0	0.0	90.0	60.0	0.0	40.0	20.0	0.0

5 CONCLUSION

We define two EA's in this research to investigate the trade-off between individual complexity and overall algorithm complexity, these algorithms are trained on two groups of 3 enemies from the EvoMan framework. The first important conclusion can be drawn from the significant difference in performance between algorithms trained on both groups: specification of a set of enemies on which the algorithm is trained will have an important effect on performance of the final solution. The two algorithms we defined can not be found to perform significantly different when looking within a training group. Convergence of the two algorithms is also dependent on the training group, the 40-node algorithm seems to converge faster in one group and the 10-node algorithm in the other. However the final solution of our algorithm performs relatively well and is able to defeat 4 out of 8 enemies in EvoMan, comparison to the GA10 algorithm of de Franca et al. [5] shows that the standard deviation of the total Gain of all agents is very high. This indicates dependence of the agents on the stochasticity of the game framework. Further research would optimally use more replications to determine the optimal model, lowering the standard deviation of the fitness and Gain and hopefully find significant differences within training groups. Additionally, it is important to spend more time defining the optimal training groups, as it shows to heavily affect ultimate performance.

REFERENCES

[1] Cazacu, R. (2016) "Comparative Study between the Improved Implementation of 3 Classic Mutation Operators for Genetic Algorithms", Procedia Engineering, 1, 1-7

[2] Eiben. A.E. & Schoenauer, M. (2002) Evolutionary computing. Inf.Process.Lett., vol. 82, no. 1, pp. 1-6

[3] Hayashi, Y., Sakata, M. and Gallant, S.I. (1990) "Multi-Layer Versus Single-Layer Neural Networks and An Application to Reading Hand-stamped Characters", Proc. of Intl. Neural Network Conf., Vol.2, pp.781-784

[4] Miras, K. & Olivetti de França, F. (2016) "An electronic-game framework for evaluating coevolutionary algorithms", arXiv:1604.00644

[5] Olivetti de Franca, F., Fantinato, D., Miras, K., Eiben, A. E., & Vargas, P. A., (2019) “EvoMan: Game-playing Competition”, arXiv:1912.10445

[6] Razali, N.M. & John Geraghty, "Genetic algorithm performance with different selection strategies in solving TSP" (2011)

[7] Zou W., Li Y. & Tang A. (2009) "Effects of the number of hidden nodes used in a structure-based neural network on the reliability of image classification." Neural Comput Appl 18(3):249–260