Evolutionary Computing - Task 1: Specialist Agent

Group 44

Jelle Steenbeek
Vrije Universiteit,
2592245

Jorrim Prins
Universiteit van Amsterdam,
11038934

Karin Lintzen
Vrije Universiteit,
2630038

Xingkai Wang
Vrije Universiteit,
2668688

ABSTRACT

CCS CONCEPTS

• **Computing methodologies** → *Intelligent agents*;

KEYWORDS

Genetic Algorithm, Evolutionary Computing, Intelligent agents, Specialist Agents, Artificial Intelligence, Individual evolution

1 INTRODUCTION

The current biological world does not know any animal species that reproduce with more than two biological parents without human intervention, since this would be needlessly complicated. However in certain specific cases, like rare mitochondrial mutations in the mother, a third biological parent is introduced to fix the genetic defect in the offspring. Therefore there is a president for beneficial effects on offspring created from more than 2 biological parents. The field of Evolutionary Computing (EC) has been experimenting with this process, as it is not dependent on a species to present multiple-parent reproduction. Evolutionary Algorithms (EA) can be designed to create offspring by multiple-parent crossover and this research will explore two similar algorithms that change in the number of parents used for offspring creation, looking at differences in performance and computational time. The two algorithms in this research consist of offspring creation by one *regular* two-parent process and a four-parent process.

This main principle of EC is to apply Darwinian principles to automate problem-solving. The fundamental concept behind EC is that there is a population of individuals (solutions to a problem), that there is an environmental pressure, and that this causes 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 [1]. The population in this research consists of single-layer neural networks that aim to win a videogame called *EvoMan* [2].

EvoMan is a 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. The player has the ability to perform several actions, including taking the action to walk, shoot, jump and interrupt the jump. These actions are the output of the individual neural networks. The input is a set of 20 sensors that includes: the absolute horizontal and vertical distance between the player and the enemy, the face direction of both player and enemy and the absolute horizontal and vertical distance between the player and an enemy's projectile. The game ends when either of the controller's life points are down to zero and the fitness of an individual can be calculated by the specified fitness function.

Two algorithms competing against a static enemy in the *EvoMan* game, differing in the offspring creation process, are evaluated and compared by multiple measures. Results will help answer the research question: Is a four-parent offspring creation process, compared to a two-parent process, beneficiary for the effectiveness of an EA trying to win the EvoMan game?

2 METHODS

The EA in this research consists of a population of neural networks with a single hidden layer of 10 nodes and sigmoid activation functions for the hidden and output layer. The input vector consists of values for 20 different sensors and an individual in the algorithm therefore has 265 (= $(20+1)\cdot 10 + (10+1)\cdot 5$) weights, continuous values that can be viewed as the alleles in a chromosome. A population of 50 individuals is randomly initialized with weight values between -1 and 1 and evolves, by application of crossover, mutation and survivor selection, to find well-performing individuals. The evaluation of performance is done by the following fitness function, taking both quality and speed into account:

$$0.9 \cdot (100 - life_{enemy}) + 0.1 \cdot life_{player} - log(t) \qquad (1)$$

The initial population is evaluated and parents are selected based on linear-ranking probabilities, the number of offspring creations is equal to $\frac{50}{n_{parents}}$ and rounded down if it is not integer. Every offspring creation needs either 2 or 4 parents, dependent on the EA under examination, and produces a random number of offspring between 1 and 5. For every weight in an individual offspring, a mutation taken from a random standard normal distribution is added to the weight with a probability of 20 percent. All individual weights in the offspring are thereafter limited between -1 and 1 to speed up convergence. The complete population of parents and offspring compete in a Round Robin tournament to cut the total population back to 50 individuals, ensuring stability of the population size. In this tournament, every individual competes against 10 other randomly chosen individuals and collects "wins" based on the comparison of fitness values. The 50 individuals with the most wins survive and become the next generation, note that this mechanism will ensure survival of the best individual as it will beat every opponent in the tournament (and it is unlikely that more than 49 other individuals achieve 10 wins as well).

The process from one generation to the next is repeated for 20 generations, as the algorithm has stabilized after approximately 15 generations. A sweep of underperforming individuals is executed after 5 generations with a non-improved best solution, this sweep deletes the worst 30 percent of the population and replaces these individuals with randomly initialized ones. Replacement of weakly performing individuals will help the evolutionary process escape local maxima and improve the final solution. The complete evolutionary process, that finds well-performing solutions starting from a randomly initiated population, is repeated 10 times to quantify randomness of the performance. Ultimately, the process described above is performed over 3 different enemies from the game, as the algorithm should be able to find adequate solutions for varying problem settings. Besides evaluation by the defined fitness function, the best solutions from all replications are selected based on their Individual Gain (IG):

$$IG = life_{player} - life_{enemy} \tag{2}$$

The best individual from every replication plays 5 games to map the stochasticity of the game-playing framework and the averages of these IG's are reviewed.

The specific crossover mechanism has not been defined above, as this is the process under examination in this research and needs some additional elaboration. Two different algorithms perform the evolutionary steps above, relying on a separate process for offspring creation. Arithmetic recombination of the two-parent algorithm originates from a linear combination where weights (that sum to one) are randomly picked for the influence of both parents. Four randomly initialized weights, that again sum to one, are chosen for the four-parent algorithm and a linear combination of the four parents results in offspring in this setting. These crossover alternatives will have an effect on the speed and performance of the algorithm and will be further investigated in the following sections.

3 RESULTS AND DISCUSSION

In Figure 2 the average and maximum fitness of the two- and four-parent algorithms are plotted over generations. The bandwidth around the lines represent one standard deviation above and below the average values. Table 1 shows significant statistical evidence for preference of the two-parent algorithm over the four-parent algorithm based on maximum fitness, only for enemy 1. The four-parent algorithm is not significantly better than the two-parent algorithm for any of the enemies. Figure 2 shows the maximum fitness of the two-parent algorithm for enemy 1 to be more than one standard deviation away from the four-parent algorithm. This is also the case for average fitness in the graphs for enemy 1 and enemy 3, again showing preference for the two-parent algorithm.

Results for enemy 5 show slight differences between the maximum and average fitness of both two- and four-parent algorithms after 20 generations. Maximum fitness for this enemy is around 90 from the start for both algorithms and this leads to fast convergence of the complete population to this value. There is barely any improvement from the initial best performing individuals and the only notable development is the two-parent algorithm converging its average fitness about three generations faster than the four-parent algorithm.

The boxplots in Figure 2 present the IG for the three enemies. As can be seen in these plots, the average IG for the two-parent algorithm is higher than for the four-parent algorithm, for every enemy. However, for enemy 3 and enemy 5 the average IG for the four-parent algorithm is within the confidence interval of the two-parent algorithm and vice versa. IG's for these enemies do not show a significant difference through these boxplots.

On the contrary, comparing the best IG for the algorithms per enemy paints a different picture. The two-parent algorithm produces a maximum IG for enemy 1 that is far outside the confidence interval of the four-parent algorithm. Surprisingly, the maximum IG for the four-parent algorithm against enemy 3 is located far above the confidence interval of the two-parent algorithm for this enemy. A significant difference between the two algorithms, both for the maximum and average IG, can only be found playing against enemy 1. The two-parent algorithm outperforms the four-parent algorithm evaluated by both measures.

In Table 2 the comparison between the IG of the two- and four-parent algorithm and the Genetic Algorithm (GA10) from the baseline paper[2] is made. Note that the IG in the baseline paper the formula for IG is

$$100.01 + life_{player} - life_{enemy} \tag{3}$$

Therefore the IG of GA10 is a lot higher than calculated for both EA's, of which the formula is given in the method section (2). When taking away the 100.01 that the IG of the baseline paper is higher and comparing them to the IG of both EA's it is clear that the maximum IG of both EA's is always just as good as the GA10 of the baseline paper. When only looking at the two-parent algorithm and the GA10, the difference is not that big for all enemies, but by adding the four-parent algorithm the difference for enemy 3 goes up to more than a 30% difference. Thus when taking both EA's into account the two- and four-parent algorithm are better together than the GA10.

	Enemy 1	Enemy 3	Enemy 5
T-value	3.826	0.678	0.161
p-value	0.001	0.507	0.873

Table 1: T-test on max. fitness after 20 gens (2-tailed)

Enemy	GA10	Two-parents	Four-parents
1	190.01	90	50
3	158.01	56	78
5	188.01	93,76	93,68

Table 2: Max. IG in baseline paper [2] compared to both EA's

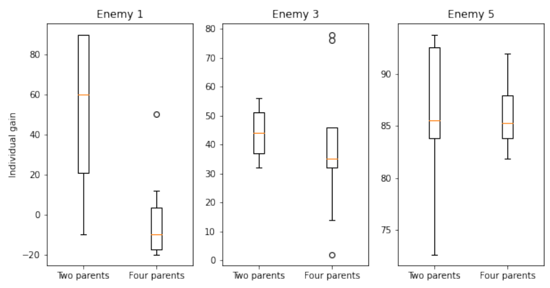

Figure 1: Boxplot of IG per enemy for both EA's

4 CONCLUSION

The goal of this paper was to find out if a four-parent EA is better at finding an optimal solution for a specialist problem than a two-parent EA. As shown in section 3, for some cases the two- and four-parent EA find the same solution. However the four-parent is never better than the two-parent algorithm.

This could be the result of having all the parents around different local optima and when averaging them for the offspring it will not be around any of those local optima. This results in non-improving offspring. If this predominantly happens within a generation, the result will be barely to not improved for that generation. Another reason could be that since the offspring is an average of the parents and when averaging over four parents a parent with a high fitness will contribute less to the offspring than in the two-parent case. This also has as result that there will be less outliers with a tendency to low fitness, but since low-fitness individuals get cut off; it has less influence on the final outcome of the algorithm. Future research might look into a more generalized intrinsic preference for two-parent algorithm in other or multiple experiments. Another interesting approach would be the evaluation of algorithms using other amounts of parents for offspring creation.

REFERENCES

[1] Eiben. A.E. and Schoenauer, M. (2002) Evolutionary computing. Inf.Process.Lett., vol. 82, no. 1, pp. 1-6
[2] Olivetti de Franca, F., Fantinato, D., Miras, K., Eiben, A. E., and Vargas, P. A., (2019) "EvoMan: Game-playing Competition", arXiv:1912.10445

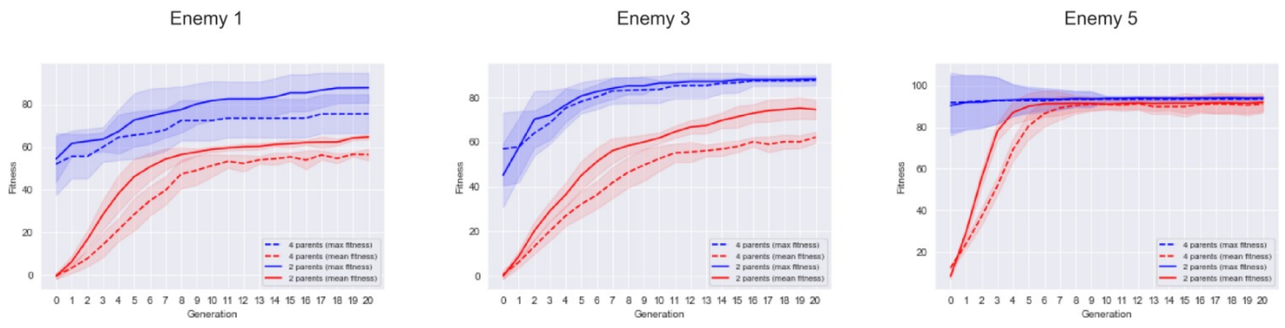

Figure 2: Avg. and max. fitness per generation for both EA's