Evaluating the crossover recombination method for higher average Individual Gains in a population of Generalist Agents when using 2-point Crossover compared to 1-point Crossover

Evolutionary Computing Assignment II: Generalist Agent

Team 21

October 12, 2020

Wouter Kok
Vrije Universiteit Amsterdam
The Netherlands
2639768
w.j.kok@student.vu.nl

Israa Ahmed
Vrije Universiteit Amsterdam
The Netherlands
2639584
i2.ahmed@student.vu.nl

SarahLynne Palomo
Vrije Universiteit Amsterdam
The Netherlands
2636769
s.l.palomo@student.vu.nl

Fay Beening
Vrije Universiteit Amsterdam
The Netherlands
2680969
r.f.l.beening@student.vu.nl

1 INTRODUCTION

In recent years video games have become increasingly popular as an AI benchmark [1]. Consequently, game frameworks can be an ideal environment for testing the main objectives for AI, in particular the ultimate goal of creating general intelligence [2]. The challenge now is to create a general game playing agent that is not over-fitted on one particular task[3].

In our previous paper [4] we discussed the use of Evolutionary Algorithms (EA's) to train a population of Specialist agents, and we compared two different EA's on performance levels based on the resulting mean and maximum values. These EA's used variations of the recombination operators: 1-point versus 2-point crossover. For the experiment, the Distributed Evolutionary Algorithms in Python framework (DEAP) was implemented to apply these functions [5].

For our current study, we again examine the differences between the 1-point versus 2-point crossover but in the context of training a *Generalist* agent. And instead of the previous fitness equation we use another fitness value calculation that does not include a time value. Note that a Specialist agent fights against a single enemy during a game while a Generalist agent fights *multiple enemies* simultaneously during a game.

Thus the main goal of this study is to determine whether the recombination method 2-point crossover results in higher mean values of Individual Gain than 1-point crossover, across a population of Generalist agents, over a set number of generations. We will also attempt to evolve a Generalist agent that can beat at least 5 of the 8 enemies in the EvoMan game framework.

1.1 Experimental Hypotheses

To achieve the research goal, two hypotheses are tested for different enemy groups where the mean of the samples are compared. Two groups are chosen to improve validity of the experiment and are explained in the implementation section. In the hypotheses, *1* is 1-point crossover and *2* is 2-point crossover. A general significance level α of 0.01 is used.

The first null hypothesis is tested for enemy group *A*. Alternatively we check whether they are not the same.

$$H_0^1 : \mu_{1_A} = \mu_{2_A} \tag{1}$$

$$H_a^1 : \mu_{1_A} < \mu_{2_A} \tag{2}$$

The second null hypothesis is tested for enemy group *B*. Alternatively we check whether they are not the same.

$$H_0^2 : \mu_{1_B} = \mu_{2_B} \tag{3}$$

$$H_a^2 : \mu_{1_B} < \mu_{2_B} \tag{4}$$

2 METHODS

To set up the study, Python v.3.6.0 and DEAP framework v.1.3.1 [5] are used in order to implement Miras' EvoMan game framework [3], using neural networks with 10 hidden neurons.

2.1 Evolutionary Algorithm design

Starting with Holland's Simple Genetic Algorithm (SGA) [6] as a basis for the EA design, we then modified the parameter values that were necessary for our investigation into the n-crossover question, as seen in Table 1.

In this case, the individual is represented as a list of 265 float values. DEAP 1-point and 2-point crossover are used for recombination of the parent individuals and for mutation DEAP Index Shuffle function is used. For parent selection we used DEAP Tournament selection with a tournament size of 2 and for Survivor selection: DEAP Random selection function.

Table 1: EA technical tableau

Parameter	Value
Representation:	List of floats
Recombination:	n-point crossover (n={1|2})
Mutation:	Index shuffle
Parent selection:	Fitness proportional (Tournament)
Survivor selection:	Random

For both EA's, the control parameter values for our research goal were kept consistent, as shown in Table 2.

Table 2: Parameter values applying to both EA's

Parameter	Value
Population Size	100
Generation	15
Survival	50%
Crossover Prob	100%
Mutation Prob	20%

2.2 EvoMan Enemy Group Selection

Within the EvoMan game framework [3] there is a selection of enemies from 1 to 8, each with their own set of pre-determined fighting behaviours. In order to determine which groups of enemies we would use to train the Generalist agent, we reviewed the preliminary tournament results as outlined in Miras' paper [3] of the one-on-one and grouped combat games to see what combination of enemies could potentially result in the highest individual gains for the agent.

Initially, we were interested in using two sets of triples: One with enemy group (2,5,6) to be in line with the enemy group used in Miras' genetic algorithm with 10 hidden layers (GA10). And another with enemy group (4,6,7) since it contained a different set of enemies to the previous group; these also showed the best performance in Miras' preliminary results. However, after testing these combinations, they showed poor performance (only two enemies were defeated in the final rounds) so the enemy groups were modified to two sets of pairs instead:

With enemy group (7,8) since it had good results for GA10 in table VI and for all EA's in table VII of Miras' paper [3] And with enemy group (1,3) since it was never beaten by any of the training sets as outlined in Miras' table VII.

2.3 Implementation

2.3.1 Initialisation: Each algorithm initialises the first population using the same seed, which is then incremented per run iteration to generate the following populations. For the experiment we wanted to work with a population of 100 to be in line with Miras' setup [3], while using a population of 50 for preliminary testing.

2.3.2 Evaluation: Individuals (agents) in the Generalist population are evaluated using the EvoMan game framework [3] by playing the agent against the predetermined enemy groups, and assigning the fitness value as the individual gain over multiple enemies, as shown by formula (5):

$$IndividualGain = (\mu_A - \sigma_A) - (\mu_E - \sigma_E) \quad (5)$$

Where A represents the list containing the agent's life from the different games played and E represents the list containing the enemy's life from the different games played.

Once all individuals in the population had been assigned a fitness value, this then forms the mating pool.

2.3.3 Parent Selection - Tournament: Pairs of individuals are selected sequentially from the mating pool and compared by fitness value. The individual with the lower value then becomes a copy of the individual with the higher value. This process is repeated with the next pair until the entire population has been updated. As a measure to ensure random recombination in the following step, the population is shuffled.

2.3.4 Recombination: Pairs of individuals are selected sequentially from the shuffled population to be parents. During the crossover step two parents create two offspring. For EA1, the float lists of the parents are both split at the same randomly selected point, after which the tails of each parent are exchanged [7]. For EA2, the float lists are split at the same two randomly selected points where the alternating segments of the parents are then exchanged [7]. In the case of both EA's, the two resulting configurations become the children - that may or may not have mutation applied - and these are then added to the population. Note that the original parents remain unchanged in the population.

2.3.5 Mutation: The probability of an individual offspring to be mutated is set to $P_m = 0.2$ to be in line with Miras' paper for comparability.

Once the individual is selected for mutation, DEAP's Shuffle Index is applied to the float list representations; the float value positions are swapped within the individual's list with a probability of $P_i = 0.2$.

2.3.6 Survival selection: The enlarged population (parents + children) is reduced back to its original population size via Random selection. We did not use elitist survival in this case since previous tests showed that elitist survival in combination with tournament parent selection would result in early convergence.

2.3.7 Iterations: We ran 15 generations of a population of 100 and repeated this step 10 times. From the subsequent results we calculated the mean of the means and the mean of the maxima. The best performing individual during the training phase overall is 'remembered' and is later employed to play against all 8 enemies to see how many it can defeat.

2.4 Parameter Tuning

During parameter tuning, our early singular testing with population=100 and generations=10 resulted in *three* defeated enemies in the final game. When increasing the generation to 15, it resulted in *four* defeated enemies. For population=150 and generations=10 it remained at four defeated enemies. And for population=150 with generations=15, and population=100 with generations=20 the number of defeated enemies no longer increased above four.

Additional tests were conducted with the mutation probability set to 40% then 60%, but this mostly influenced the variance and not the trend of the results. Where an individual had been selected for mutation, the probability of changing one float value in its list representation was also tested with probabilities of 10%, 20%, 50% and even 100%(!), however a probability of 20% had more consistent results over multiple runs.

3 RESULTS AND DISCUSSION

We see in Figures 1 and 2 representations of the results of 10 runs of both EA's over 15 generations, where the generalist agent fights against enemy group 1&3 and group 7&8 respectively. In both cases, the differences between the 1-point crossover and 2-point crossover results are negligible.

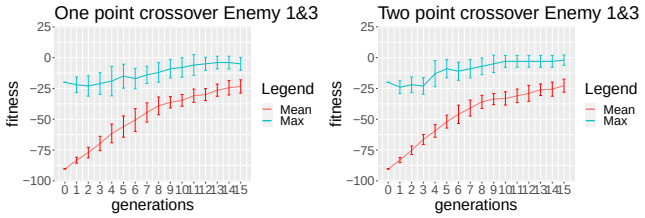

Figure 1: Evolution of the Generalist population against Enemy 1&3 where 1-point crossover (above) and 2-point crossover (bottom) have been used

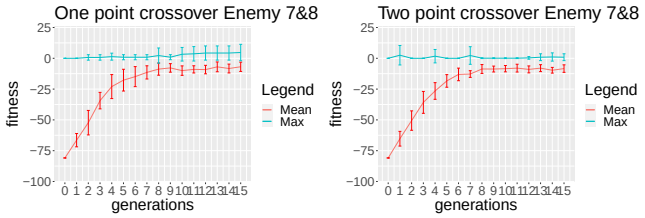

Figure 2: Evolution of the Population against Enemy 7&8 where 1-point crossover (above) and 2-point crossover (bottom) have been used

For Figure 1, the mean fitness values with standard deviations (red lines) for enemy 1&3 show near identical values between the 1-point and 2-point crossover experiments. The maximum values with respective standard deviations (blue lines) show correlating movements in fitness value between 1-point and 2-point crossover.

For Figure 2, the mean fitness values with standard deviations (red lines) for enemy 7&8 show similar values between the 1-point and 2-point crossover experiments, with slight variation in standard deviations. The maximum values (blue lines) show similar fitness levels however their respective standard deviations do show differences in variation between 1-point and 2-point crossover.

Although in this case the 2-point crossover introduces additional diversity to the population fitness, it is still averaged out to the mean,

which is on par with the 1-point crossover. This suggests again, that the number of crossover points used in recombination has little influence over the final outcome of fitness on the Generalist population.

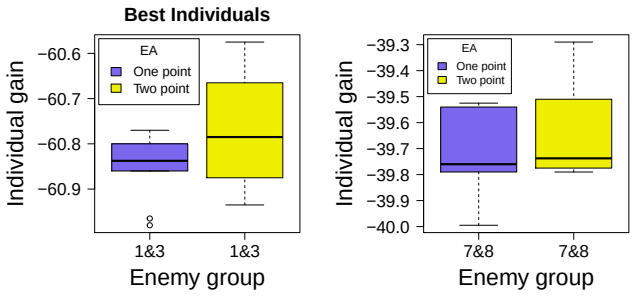

Figure 3: Average Individual Gain of the best individuals for 1-point and 2-point crossover from 10 runs, each repeated 5 times for the 8 enemies, with the enemy training group 1&3 (left) and 7&8 (right)

The best individuals from the 10 runs for both EA's and both enemy groups were then used to play against each of the eight enemies (5 times). The average individual gain was then plotted in Figure 3. It is interesting to note the very slight variance of Individual gain in the 1-point crossover compared to the relatively larger variance of Individual Gain in the 2-point crossover for enemy group 1&3 (left). Another interesting observation in the results is that the individual gain (y-axis) of enemy group 1&3 for the n-crossovers is nearly twice as small than that of enemy group 7&8 in Figure 3.

Two unpaired Two Sample t-tests were conducted on different enemy sets with an alternative hypothesis that 2-point crossover has a greater mean than 1-point crossover with a significance level α of 0.01. The Shapiro-Wilk test was done for the normality assumption and showed that the four samples were normally distributed.

The F-test was done under the assumption of homogeneity between the variances and showed that the variances of both sets 1&3 and 7&8 between 1-point and 2-point crossover were equal.

The results of the t-tests showed that there was no significant difference in the results between 1-point and 2-point crossover for enemy set 1&3 (M= -60.8495, -60.7685; p= 0.03606) and for enemy set 7&8 (M= -39.7015, -39.6440; p = 0.222), respectively. However, there was a significant difference between the different enemy sets for the same crossover method. 1-point crossover with enemy set 1&3 and 7&8 resulted in p-value < 2.2e-16 and 2-point crossover with enemy set 1&3 and 7&8 in p < 2.2e-16.

These values suggest that training with groups of different types of enemies within the same EA have an impact on the mean Individual Gain values of the Generalist population, and therefore changing the enemy groups would be more conducive to the agent defeating more enemies. On the other hand, it suggests that using 2-point crossover does *not* result in a significantly better mean individual gain of the Generalist population than 1-point crossover. Again, this is further evidence that the number of crossover points used during recombination has little influence on the overall results of this experiment.

Table 3: Final victory scores of the 2-point EA on the training set 7&8. The values represent the final energy of the player/enemy

Enemy	Player points	Enemy points
1	0.00	80
2	42.00	0
3	0.00	70
4	0.00	70
5	48.64	0
6	0.00	100
7	38.80	0
8	25.00	0

Table 4: Final victory scores of the GA10 from the baseline paper and the 2-point GA with training on enemies 7&8. The values represent the final energy of the player against the corresponding enemy, the blank spaces represent the enemies that the EA did not beat

Enemies	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
GA10 [3]		74		43	60			30
EA 2-point		42			49		39	25

Table 3 shows the results of the best individual, which is from the EA using 2-point crossover on the enemy group 7 8. In a comparison with the performance of the GA10 algorithm from Miras' baseline paper [3], we see that they share the same three out of four defeated enemies; 2, 5 and 8, as shown in Table 4.

4 CONCLUSIONS

The main purpose of this study was to investigate whether the use of 2-point crossover during recombination would result in higher mean and maximum values of Individual Gain over a population of Generalist agents, compared to 1-point crossover. What our results show, however, is that this was not the case, and that the number of crossover points used during recombination had *little influence on the overall performance of the Generalist agents*. This is in line with the results of our previous paper [4] where we evolved a Specialist agent and found that the use of 2-point crossover did not influence the maximum fitness values of the Specialist population. However, unlike the results of our previous paper, the *rate* of evolution of the Generalist population remained unaffected.

The secondary purpose of this study was to evolve a Generalist agent that would able to beat at least five of the eight enemies of the EvoMan framework by training the agent against arbitrary groups of enemies during the evolutionary process. Despite performing parameter tuning in order to maximise the number of defeated enemies, we were unable to achieve more than four wins. However, of these four defeated enemies, our Generalist agent was able to beat two enemies that it had previously not encountered.

Since our experiment trained the Generalist population against two groups of enemy pairs, further study can be conducted on alternative configurations of enemy groups in order to determine the optimal training sets for evolving the Generalist agent to be able to beat all eight enemies in the EvoMan game framework.

REFERENCES

[1] Philip Bontrager, A. Khalifa, A. Mendes, and J. Togelius. 2016. Matching games and algorithms for general video game playing. In *AIIDE*.

[2] 2016. General general game ai. English (US). In *2016 IEEE Conference on Computational Intelligence and Games, CIG 2016* (IEEE Conference on Computatonal Intelligence and Games, CIG). 2016 IEEE Conference on Computational Intelligence and Games, CIG 2016 ; Conference date: 20-09-2016 Through 23-09-2016. IEEE Computer Society, (July 2016). DOI: 10.1109/CIG.2016.7860385.

[3] Karine Miras and Fabricio De França. 2016. Evolving a generalized strategy for an action-platformer video game framework. In (July 2016), 1303–1310. DOI: 10.1109/CEC.2016.7743938.

[4] S. Palomo W. Kok I. Ahmed and F. Beening. 2020. Evaluating variation operators for higher fitness values in a population - recombination: 2-point crossover vs 1-point crossover. In.

[5] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. 2012. DEAP: evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13, 2171–2175.

[6] Holland John. 1992. Holland. genetic algorithms. *Scientific american*, 267, 1, 44–50.

[7] A.E. Eiben and J.E. Smith. 2015. *Introduction to Evolutionary Computing*. Volume 2, 49–54.