Can a two-point crossover operator outperform the uniform crossover operator from DEAP?

Evolutionary computing (X_400111), prof. dr. A.E. Eiben
Task 1: specialist agent

Team 1 - Popy:
Rumyana Krumova (2718449),
Melis Nur Verir (27260079),
Alicja Dobrzeniecka (2737091),
Dominique Woof (2634493)

01.10.2021

1. INTRODUCTION

Crossover is an essential component of evolutionary algorithms as it largely determines the genotype of new offspring. Therefore, the choice of crossover method is expected to have a great impact on achieving the desired results. This research focuses on the difference in performance of two evolutionary algorithm which differ only in choice of crossover operator. The first algorithm will make use of a self-written two-point crossover operator, while the second algorithm will contain the uniform crossover operator from the DEAP framework [1]. The research question of this study is as follows: Can a two-point crossover operator outperform the uniform crossover operator from DEAP? The hypothesis is that it is possible to outperform DEAP uniform crossover operator with the two-point crossover approach.

2. EXPERIMENTAL AND COMPUTATIONAL DETAILS

2.1 The Evolutionary Algorithms and their Operators

2.1.1 Parent selection

The choice of parents for future offspring is made by a variation on no-replacement tournament selection [2]. The implemented operator randomly chooses four potential parents from the current population, sorts them according to their fitness, and then returns the individuals with the first and third highest fitness to be the actual parents. The choice of the first and third best was made based on the intention to minimize the chance of being stuck in a local maximum, and on the assumption that the two best individuals do not always result in the best offspring [3].

2.1.2 Crossover

Evolutionary Algorithm 1: Two-point crossover

Two-point crossover is a technique where two new individuals are created by interchanging the genetic information in between two points in the genotypes of the parents. A crossover operator of such type is implemented in the first EA1 [7].

Table 1: Two-points crossover settings

PARAMETER	*VALUE*
FIRST AND SECOND POINTS	random value from uniform distribution

This operator chooses a pair of crossover points by randomly sampling them from a uniform distribution, after which it interchanges the genetic information by performing one-point crossover twice [4].

Evolutionary Algorithm 2: DEAP uniform crossover

Uniform crossover is a technique where the genotype of new offspring is created by a stochastic process. Each bit of genetic information, which in this research would be a single weight, is assigned a certain probability of being chosen from either parent. The second EA uses such uniform crossover operator taken from the DEAP framework [6]. The parameters used for this operator are described in Table 3.

Table 2: Uniform crossover settings

PARAMETER	*VALUE*
MATE INDPB	0.1
CROSSOVER THRESHOLD	0.5

The implemented operator takes both parents as starting points after which it swaps their weights according to a given probability. Within this operator, the rules for crossover are strict. In contract to the two-point crossover operator, crossover does not always take place. Additionally, even if crossover does take place, the chances of a weight being swapped is only 10%. This which means that the uniform crossover makes it harder to get diversity in the offspring.

The motivation behind the choices of crossover operators is the interest in examining the difference between a crossover method that interchanges single weights with respect to one interchanging entire sequences of weights. The latter is expected to provide more diversity and thus a stronger population.

2.1.3 Mutation

In both algorithms, after crossover has taken place, mutation is performed over each new individual with a chance of 10%. The type of mutation operator used is a gaussian operator provided by the DEAP framework [8]. This operator mutation is a technique, where to certain weights a value from Gaussian distribution is added on the condition that the current randomly chosen number is smaller than the mutation threshold. Regarding the parameters for the Gaussian distribution, the mean is set to 0 and sigma to 0.1 in order to make sure that the changes resulting from the mutation are not too large. With minimal changes the chances of leaving the maximum because of mutation are minimized.

2.1.4 Survivor Selection

The survivor selection operator determines which individuals are to be eliminated and which are to be kept in the next generation, while at the same time maintaining the population diversity and size [5].

Within this selection, randomness is introduced with respect to how many of the individuals with the highest fitness are guaranteed to survive. The random variable used to determine this takes on a value between 0.02 and 0.05. In the case of 100 individuals, this entails that between 2% and 5% of the fittest individuals will always survive. Then, the rest of missing individuals are taken with the use of probability, giving a higher chance to those individuals that obtained higher fitness, but not guaranteeing them that they will be chosen. As a result, diversity will be preserved and the assumption that the best individuals not always result in the best offspring is still met. Selecting and mating individuals with high fitness causes the offspring to just have the characteristics of its parents which causes the fall in the diversity.

2.1.5 Replacement

If during training, it appears that the current best fitness does not increase over the last three generations replacement is forced upon a subset of the current population. This operation aims at preventing the individuals with the worst fitness values from participating in the parent selection and pushing the population in wrong direction. This forced replacement removes the 25% with the worst fitness, after which it creates new individuals with random weights to reset the chances of creating better offspring.

2.2 Parameter Settings and Experimental setup

2.2.1 Initial setup and additional functions

Each experiment is initialized by using the default Evoman parameters, and by selecting the enemy necessary for the current experiment. The initial weights for the 100 individuals that created at the beginning of each experiment, are given a random value between -1 and 1. During evolution, there is also the use of a limitation method, which keeps the new weights in the range of -1 and 1. For the fitness function, the default fitness function provided by the Evoman framework is used. What is included in this fitness function corresponds to what is considered important for a successful individual (enemy life, player life, and time). Finally, a normalization function is used, that is necessary for the process of converting the fitness values to probabilities in survival selection.

2.2.2 Parameter settings

During this research the following parameters were used for both evolutionary algorithms.

Table 3: General settings

PARAMETER	*VALUE*
POPULATION	100
GENERATIONS	30
WEIGHT LIMITS	<-1,1>
HIDDEN LAYERS	10
ENEMIES	[2.5.8]
MUTATION SIGMA	0.1
MUTATION MU	0
MUTATION INDPB	0.2

The predetermined budget assumed roughly 40 minutes per experiment. Eventually, it took around 13 hours to run 20 experiments on a Ryzen 7-4800H/32GB/960 GTX1660Ti 144Hz.

3. RESULTS AND DISCUSSION

3.1 Training results

Figure 1 shows that the value of best fitness obtained by each evolutionary algorithm is very similar. It can also be seen that the training accelerates between generation 0 and 5 but rapidly decreases in pace after. The same situation applies for the mean and its standard deviation of the two evolutionary algorithms; however, the standard deviation of the evolutionary algorithm with the two-point crossover operator has a larger fluctuation than the evolutionary algorithm with the uniform crossover operator. After generation 10 the pattern for fitness for both algorithms show an incremental behavior.

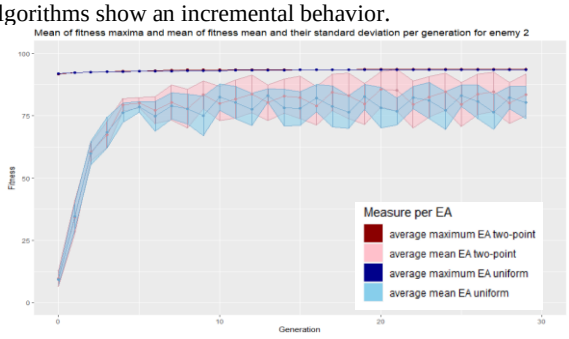

Figure 1: Line plot for enemy 2

The lines in *Figure 2*, show that during the training of both algorithms the behavior of the best result is indistinguishable; however, the mean fitness obtained with two-point crossover fluctuates a little more than the mean of uniform crossover. Overall, both algorithms represent very alike behavior for enemy 5.

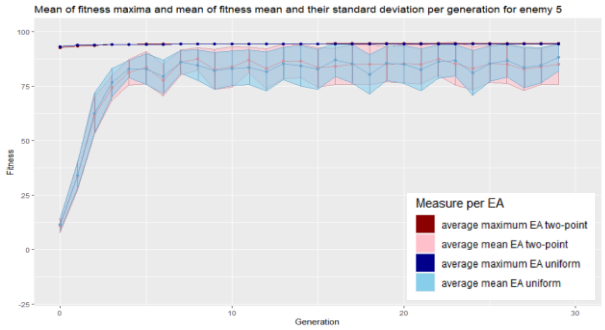

Figure 2: Line plot for enemy 5

The lines in *Figure 3*, show that there is an insignificant difference between the uniform crossover and two-point crossover during the training stage. Nevertheless, after generation 5 the standard deviation of the algorithms shows noticeable contrast. In more detail, the maximum and minimum points of the standard deviation values of both algorithms fluctuate in opposite directions.

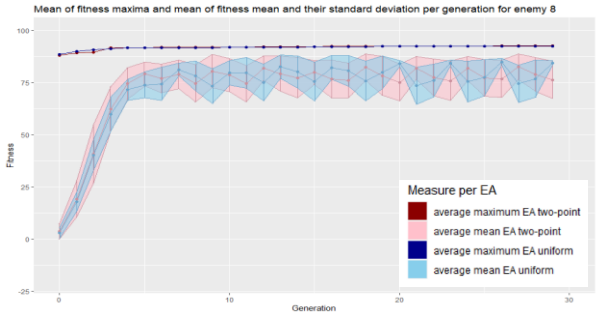

Figure 3: Line plot for enemy 8

3.2 Test results

Figure 4 indicates that for enemy 5, both the uniform crossover and two-point crossover operator perform similarly well. Additionally, it can be seen that the two-point crossover algorithm has a higher median than the other. However overall, both methods are centralized around a similar fitness value which is above 80.

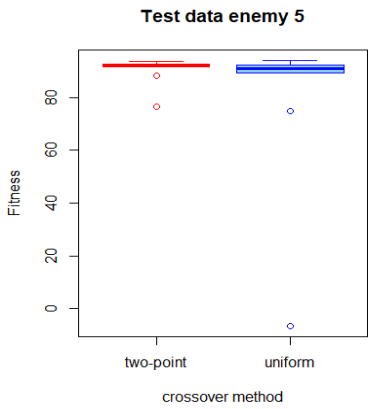

Figure 4: Boxplot for enemy 5

Figure 5 shows great difference in fitness between both crossover operators. While the uniform crossover method has a minimum value between 0 and 20, the two-point crossover method has a minimum value between 65 and 80. Also, the median value of the two-point method is noticeably higher than the uniform method. Overall, the fitness of the algorithm using the uniform operator has a higher variance compared to the two-point method; however, both methods have a skewed mean distribution.

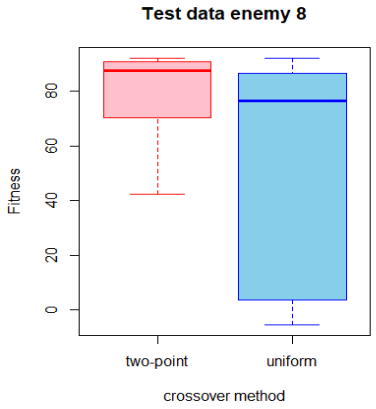

Figure 5: Boxplot for enemy 8

Figure 6 shows clear difference in the obtained fitness value by both crossover methods. While the median of the algorithm using the uniform operator is approximately 80, all data of the two-point operator lays above 80. It can be seen that the uniform crossover method has a lower whisker across the fitness axis. Additionally, the distribution of the two-point crossover method shows that the data is more condensed.

Figure 6: Boxplot for enemy 2

3.3 Comparison with Evoman framework initial experiments

In the paper *Evolving a generalized strategy for an action-platformer video game framework[11]*, which was included in the Evoman framework, experiments are with the same gam are performed, while using the NEAT algorithm. In the table below one can find the comparison between the NEAT results and the results of the two EAs from this research.

Table 4: Comparing fitness results

METHOD	ENEMY	FITNESS FOR BEST INDIVIDUAL
NEAT	2, 5, 8	94, 94, 86
TWO-POINT	2, 5, 8	94, 94 ,93
UNIFORM	2, 5, 8	94, 93, 87

3.4 T-test analysis

To determine if the observed differences in fitness mean during the testing of both algorithms is of statistical significance, a t-test was performed over the test data of each of the three enemies. For all tests the hypotheses were the following:

H_0: *the true difference in means is equal to* 0
H_1: *the true difference in means is not equal to* 0

The significance level used for each test is 0.05. The results obtained from the t-tests are denoted in *table 5*.

Table 5: P-values of t-tests

ENEMY	P-VALUE
2	0.02798
5	0.3204
8	0.08206

4. CONCLUSIONS

As this research showed, with the assumptions that were made and parameters settings that were chosen, it was not possible to prove that the implemented two-point crossover operator outperforms the DEAP uniform crossover operator in the Evoman game. The training of both algorithms looked very similar which may indicate that the chosen and tuned crossover operators are in essence not differing enough to make a large difference in the creation and the result of offspring. It is possible that other operators played a more important role, and that changing those would have resulted in more radical results. The boxplots showed us that although the training behavior may be similar in both algorithms, there can still be a more significant difference in the results when testing the best individuals. A possible explanation for this is that the uniform crossover operator led to overfitting of the weights and therefore resulting in worse testing results than the two-point crossover.

Considering the significance level of 0.05, the null hypothesis can only be rejected for enemy 2. This would suggest that the true difference in means is not equal to 0. However, because the null hypothesis cannot be rejected for enemy 5 and 8, thus suggesting that it is not trivial that the means are significantly different, it cannot be concluded that one of the two algorithms performs significantly better than the other.

From the data in *table 5* it can be concluded that the NEAT and two-point crossover algorithm behave a little better during training, than the uniform crossover, with the two-point algorithm having the best results.

However, to verify this and the previous statements, further research is advised, as it cannot be excluded that the results obtained are due to either chance, inaccuracies, amount of data, or other factors.

5. ACKNOWLEDGMENTS

Alicja Dobrzeniecka, Rumyana Krumova and Melis Nur Verir participated in coding and divided the code into parts, later training the model and testing it. Rumyana Krumova implemented the crossover and the mutation functions. Dominique Woof performed the coding for visualizing the data, worked on improving the replacement operator and fine-tuned the report. Alicja Dobrzeniecka and Melis Nur Verir implemented the normalization, tournament selection and doomsday methods. All members participated in preparing the report by dividing it into sections.

REFERENCES

[1] DEAP documentation — DEAP 1.3.1 documentation. (2009). DEAP. https://deap.readthedocs.io/en/master/index.html

[2] Fang, Yongsheng & li, Jun. (2010). A Review of Tournament Selection in Genetic Programming. 181-192. 10.1007/978-3-642-16493-4_19.

[3] Eiben, Agoston E., and James E. Smith. Introduction to evolutionary computing. Vol. 53. Heidelberg: springer, 2003.

[4] Jebari, Khalid. (2013). Parent Selection Operators for Genetic Algorithms. International Journal of Engineering Research & Technology. 12. 1141-1145.

[5] Fang, Y., & Li, J. (2010). A Review of Tournament Selection in Genetic Programming. Advances in Computation and Intelligence, 181–192. https://doi.org/10.1007/978-3-642-16493-4_19

[6] Evolutionary Tools — DEAP 1.3.1 documentation. (2009). DEAP. https://deap.readthedocs.io/en/master/api/tools.html#deap.tools.cxUniform

[7] Bera, S. (2021, 18 september). Crossover Operator — The Heart of Genetic Algorithm. Medium. https://medium.com/@samiranbera_66038/crossover-operator-the-heart-of-genetic-algorithm-6c0fdcb405c0

[8] Evolutionary Tools — DEAP 1.3.1 documentation. (2009b). DEAP. https://deap.readthedocs.io/en/master/api/tools.html#deap.tools.mutGaussian

[9] Kumar, Rakesh, Knowledge based operation and problems representation in genetic algorithms, 2012

[10] Choi, S., & Moon, B.R. (2003). Normalization in Genetic Algorithms. GECCO.

[11] Evolving a generalized strategy for an action-platformer video game framework, https://github.com/karinemiras/evoman_framework/blob/master/multi_evolution.pdf