EVOLUTIONARY COMPUTING: STANDARD ASSIGNMENT TASK I: Specialist Agent - Team 101

Date: 01/10/2021

Christodoulou Alexandros, ID: 2739584
Vrije Universiteit Amsterdam, Faculty Science,
Artificial Intelligence
Amsterdam, Netherlands
a2.christodoulou@student.vu.nl

Christofi Konstantinos, ID: 2742715
Vrije Universiteit Amsterdam, Faculty Science,
Artificial Intelligence
Amsterdam, Netherlands
k.christofi@student.vu.nl

Hadjigeorghiou Christos, ID: 2751135
Universiteit van Amsterdam, Faculty Science,
Forensic Science
Amsterdam, Netherlands
c.hadjigeorghiou@student.vu.nl

Lamkaraf Redouan, ID: 2672658
Vrije Universiteit Amsterdam, Faculty Science,
Artificial Intelligence
Amsterdam, Netherlands
r.lamkaraf@student.vu.nl

ACM Reference Format:
Christodoulou Alexandros, ID: 2739584, Christofi Konstantinos, ID: 2742715, Hadjigeorghiou Christos, ID: 2751135, and Lamkaraf Redouan, ID: 2672658. 2019. EVOLUTIONARY COMPUTING: STANDARD ASSIGNMENT TASK I: Specialist Agent - Team 101: Date: 01/10/2021. In *Proceedings of the Genetic and Evolutionary Computation Conference 2019 (GECCO '19).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The purpose of the research is to evaluate the effect of two different mutation operators on the objective function which is the default fitness function of the Evoman framework [4] [6] and subsequently on the performance of an agent-based player who is evolved with increasing generations in order to beat a specific enemy player. A standard pipeline of EAs has been created which highlights different mutation strategies with all the other sections being the same. To further analyze the effects of the mutation operators, the following research question has been posed:

Research Question - "How are the results of Evoman framework specialized agent influenced by using uncorrelated mutation of one sigma compared to non-uniformed mutation with decreasing step size?"

To answer the question, three measurable success criteria have been considered: the average mean of the fitness of population of individuals, the maximum fitness of individual per population and the Wilcoxon test [8] applied on the mean of 5 test of the best individual per run for each algorithm. The first two are presented in line plots while the last one is applied on the arrays of the box plots as shown in the results, see Section 4.

2 RELATED WORK

Evoman framework has been considered to test the player evolution using a Genetic Algorithm, the LinkedOpt Algorithm and the NEAT Algorithm and their results are compared using an objective function [2, 3]. Based on mentioned work, this paper focuses on two GAs where the influence of different mutation operators is to be explored with respect to the objective function and compare the results produced with the mentioned papers.

3 METHODS

This section relates to the methods considered and applied in the experimentation phase, the motivation behind them and the reasoning for certain parameter values. The experimental setup and the fitness function considered will enable the readers to reproduce the results provided as well as gain an insight on the budget.

3.1 Budget

In order to run an algorithm, a population of 100 individual is to be evolved with 30 generations for 10 runs per enemy.The second algorithm needed about 20 hours to be finished while the second need about 30. An important factor that influences the performance of the second algorithm is that it has different functions ... . Parameter tuning was performed in both algorithms but as the results show, it could have been better. In addition, there are more optimal solutions that could have been tested if more time was available to get more experience on tuning and optimizing the search space available. A framework for evolutionary computing algorithm could have been used but for clarity sake, it is opted to manually code everything so that explanations regarding the conclusion can be more easily drawn while considering the inherent variability of the EAs.

3.2 Algorithm Setup

The standardized method of an EA algorithm refers to 5 phases: representation stage, initialization phase, mating population and survivor selection phase, the crossover and the mutation operators and the termination phase [5, 9**?**].

As the concept of the evolution is seen by the passing of generations, the mean population fitness which is the encapsulation of all individuals should become fitter. More concepts considered when developing EAs are exploration and exploitation[**?**] and how they need to be balanced as the former relates to the exploration of the search space and the discovery of promising regions while exploitation relates to focusing resources on making the best use of such areas to find a better solution. With this in mind, certain considerations have been made: the standard deviation of steps taken per generation should be reduced as time passes which will lead to smaller step sizes thus prevent omitting the top of the promising regions and the selective pressure should be increased prioritizing more fit individuals [9].

3.3 General Algorithm Considerations

In the representation stage, the chosen representation of the individuals dictates how good the solution of the algorithm is. In this paper, the representation is the one provided which is a one-to-one mapping of the phenotype to genotype where an individual is a 1D-array of 256 elements-neurons that is fed to the neural network which controls the player agent [3]. Each individual corresponds to a single solution of the algorithm where a fitness function is defined as a metric to evaluate how good an individual is in which case the following objective function to be maximized:

$$f = 0.9 * (100 - e) + 0.1 * p - \log(t) \tag{1}$$

as defined in the original paper [3].

The general parameters utilized in the experiments are a population of 100 individuals where each individual is an array of 256 entries or 257 depending on the mutation operator. The experiments have been repeated for 3 enemies where for each one, 10 runs of 30 generations for each algorithm.

3.4 Initialization Phase

In this section, the initialization of the population is described. For the one algorithm the population of individuals is randomly drawn from uniform distribution of values between -1 and 1. The population is then evaluated to get the mean fitness as well as the maximum individual fitness of the current generation. Then the iterative process of selection of mating population and of survivor population as well as the operators of recombination and mutation begins up until the termination condition is met.

3.5 Subsequent Generations

3.5.1 Mating / Parental Selection Phase.

In the parental selection phase, the Fitness Proportional Selection (FPS) mechanism is selected to pick the parents which will be part of the mating population by considering the fitness value of the individuals as indicated by Eq. (2).

$$P_{FPS}(i) = f_i / \sum_{j=1}^{\mu} f_j \tag{2}$$

It is incorporated with sigma scaling as it enhances standard FPS with information about the average and the standard deviation of the array of individuals for better representative selection [9]. More specifically, all zero and negative fitness values are assigned a minimum number followed by the sigma scaling equation as indicated below:

$$f'(x) = max(f(x) - (\bar{f} - c * \sigma_f), 0) \tag{3}$$

with c being 2, see page 81 [9].

Afterwards, the fitness values are normalized using the function 'MinMaxScaler' of the python library 'sklearn.preprocessing' [7]. As EA is a stochastic algorithm [1], values at 0 are replaced with a minimum value of 0.0001 so that it can still be selected even with a minimal probability in an attempt to discover new regions.

3.5.2 Survivor Selection Phase.
In this section, sigma scaling FPS is utilized again to stochastic select individuals but this time to prioritize the worse individuals in terms of fitness in order to be replaced. The process is identical to the one in the mating populations but first the range of the array is computed computed by subtracting the lowest fitness value from the highest. Then, each fitness value is reduced by the range making the negative values even lower, the originally high values to be closer to zero and by taking the absolute value on the array, originally small values become large. This along with a division by the sum of the fitness of the array will prioritize them when the individuals are sampled.

3.6 Crossover/Recombination Operators

For the crossover/recombination step, the whole arithmetic recombination is implemented which takes two parents and produces two children with the α parameter differentiating the two offspring. This parameter is drawn randomly from uniform distribution ranging from 0 to 1 which implies that the offspring is located in the search space bounded by the parents [9]. The mathematical equation for this recombination is as follows:

$$z_i = \alpha * x_i + (1 - \alpha) * y_i \tag{4}$$

with i being the element in the array, z referring to the offspring, x referring to the first parent and y to the second parent. Similarly, for the second offspring, the two parents are reversed so that the second parent is multiplied by α and the first by $(1-\alpha)$ as indicated by page 66 [9].

3.7 Uncorrelated mutation with one sigma

According to the book, one normal distribution is considered for each individual where each element in the individual is mutated considering one σ that is incorporated in each individual making the size of the individual one more than the other algorithm. This mutation step size is influenced by the passing of each generation where it is multiplied by an exponential term that is drawn randomly from normal distribution with parameters of mean 0 and standard deviation 1 multiplied by t which is inversely proportional to the square root of the size of the individual as indicated by the Eq. (4.2) and (4.3) of page 58 [9]. To prevent standard deviation of having a negligible effect also a threshold boundary 0.001 is indicated restrict the minimum value of the step size again as described in the same book:

$$\sigma < \varepsilon_0 \longrightarrow \sigma = \varepsilon_0 \tag{5}$$

New individuals created are assigned a σ value of 1. Just as for next mutation process, a random probability is drawn from uniform distribution of limits between 0 and 1 and if it is below 0.25 the individual will undergo mutation process.

3.7.1 Non-Uniform mutation with reducing step size. Under a non-uniform distribution, mutation was applied with a high initial value and gets lower as generations proceed. As indicated below, σ takes the value of

$$\sigma = 1 - (currentgeneration/maxgenerations) \tag{6}$$

so that selection pressure increase as time passes and less step-size is taken. In the case of mutation, noise drawn from normal distribution with parameters mean of 0 and standard deviation of σ is added to the value of the individual. Then limits are imposed to ensure the value of the individual remains between -1 and 1 which is also considered in the above.

3.8 Termination Phase

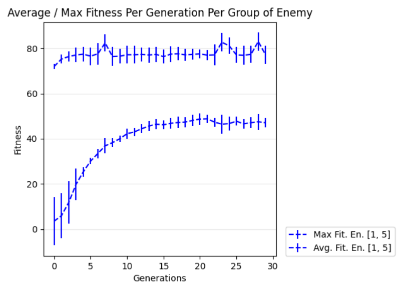
Although there are different conditions in order to terminate an algorithm [9], the one that was chosen for this report is the total number of generations which is 30. Once the 30 generations of 100 individuals are finished, the algorithm terminates. Another approach that is considered but has not be implemented is checking for stagnation of the average fitness of the population where if it does not increase after a number of iterations, there is no more diversity. In such case, a possible solution would be to randomly create new individuals to replace a proportion of the population and continue for the remaining generations or decrease the selection pressure so that less fit individuals can be included.

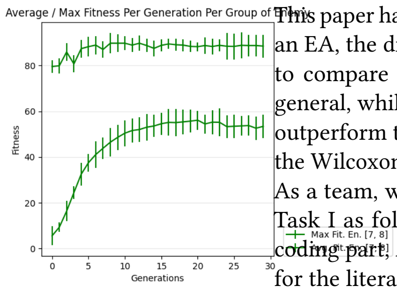
4 RESULTS AND DISCUSSION

As it can be observed at the line plots and the box plots, the results obtained by this paper are comparable with the baseline paper [3]. More specifically, as enemy 2, 3 and 5 are tested in this report, the results obtained regarding enemy 2 and 5 match the ones mentioned about their maximum fitness value, but the maximum value of enemy 3 is significant below the one reported in the baseline paper. This difference can be explained due to the limited budget and experimentation like the lack of parameter tuning as well as the use of whole arithmetic recombination where the search space is bounded by the two parents in contrast with the blend crossover where the distance difference enables search space exploration outside of the region bounded by the parents. Enemy 8 was also tested but because of the restriction of time it could not have been tested in the same run as 2 and 5 but the ideal choice of 3 opponents would be 2, 5 and 8. This and additional results regarding enemy 8 are provided in the Appendix B.

4.1 Line Plots

The line plots below, see Fig. **??**, **??**, illustrate the average and max fitness of all 10 runs for each of the 30 generations and of for both group of enemies for the two algorithms respectively.Both groups





of enemies seem to have the same numbers regarding the maximum and the mean fitness values.

4.2 Box Plots

The box plot below, Fig. 1, is generated per enemy per algorithm where the mean gain of 5 independent test runs performed by the best individuals from each group of enemies for each of the 10 runs throughout all 30 generations. This is done as there is inherent randomness in the nature of EAs as well as due to the random initialization position by the "randomini" flag of the Evoman framework [6].

Keeping the research question in mind, the results from the plot in comparison to each algorithm per enemy depict a higher fitness in favour of algorithm two which is the one with the uncorrelated mutation with one sigma. More specifically, quartiles and the mean which is the red line of the box plot depict better results. Furthermore, conclusions are drawn using the statistical test of Wilcoxon which considers the difference between the arrays of the box plot per enemy of the two algorithms [8]. Two-tailed test with a 5% Confidence Interval has been applied to check whether there is no difference in the fitness values considered as well as to check the median of the differences being positive again with 5% CI. All the tests conducted have indicated a p-value of more than 5% which means that the null hypothesis cannot be rejected implying that the average performance of the two algorithms is not different. Specific details regarding the calculations of each test can be seen in Appendix A

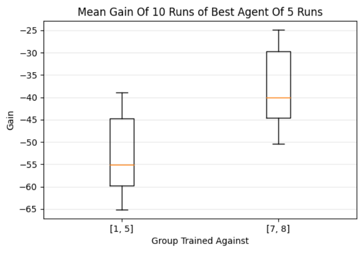


Figure 1: Box plot for average of best individual over 5 runs of best individuals per enemy .

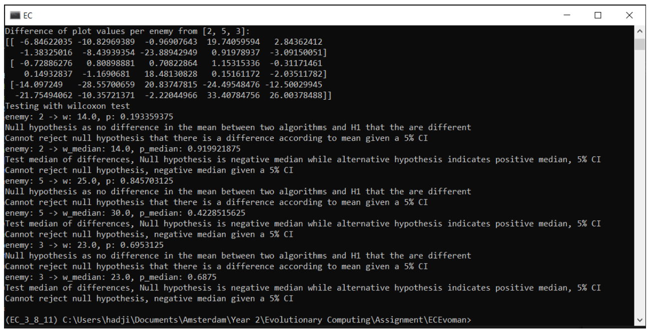
5 CONCLUSION

This paper has as a main goal to explain the procedure of developing an EA, the differences between two manually-developed EAs and to compare them with the results of the baseline paper [3]. In general, while the uncorrelated mutation with one sigma seems to outperform the non-uniform mutation with decreasing step sizes, the Wilcoxon test hints to no difference in the overall performance. As a team, we have divided the work that has to be done for the Task I as follows: Christos Hadjigeorhiou is responsible for the coding part, Alexandros Christodoulou and Konstantinos Christofi for the literature review and everyone contributing equally to the report.

A APPENDIX: WILCOXON TEST

This appendix provides further information about the exact computations performed on the difference of arrays per enemy for the box plot to draw statistical conclusions.

As it can be seen by Fig. 2, each array in the list corresponds to the difference of arrays between the two algorithms, first of enemy 2, then of enemy 5 and finally of enemy 3.



```
Difference of plot values per enemy from [2, 5, 3]:
[[ -6.84622035 -10.82969389  -0.96907643  19.74059594   2.84362412
   -1.38325016  -8.43939354 -23.88942949   0.91978937  -3.09150051]
 [ -0.72886276   0.80898881   0.70822864   1.15315336  -0.31171461
    0.14932837  -1.1690681   18.48130828   0.15161172  -2.03511782]
 [-14.097249   -28.55700659  20.83747815 -24.49548476 -12.50029945
  -21.75494062 -10.35721371  -2.22044966  33.40784756  26.00378488]]
Testing with wilcoxon test
enemy: 2 -> w: 14.0, p: 0.193359375
Null hypothesis as no difference in the mean between two algorithms and H1 that the are different
Cannot reject null hypothesis that there is a difference according to mean given a 5% CI
enemy: 2 -> w_median: 14.0, p_median: 0.919921875
Test median of differences, Null hypothesis is negative median while alternative hypothesis indicates positive median, 5% CI
Cannot reject null hypothesis, negative median given a 5% CI
enemy: 5 -> w: 25.0, p: 0.845703125
Null hypothesis as no difference in the mean between two algorithms and H1 that the are different
Cannot reject null hypothesis that there is a difference according to mean given a 5% CI
enemy: 5 -> w_median: 30.0, p_median: 0.4228515625
Test median of differences, Null hypothesis is negative median while alternative hypothesis indicates positive median, 5% CI
Cannot reject null hypothesis, negative median given a 5% CI
enemy: 3 -> w: 23.0, p: 0.6953125
Null hypothesis as no difference in the mean between two algorithms and H1 that the are different
Cannot reject null hypothesis that there is a difference according to mean given a 5% CI
enemy: 3 -> w_median: 23.0, p_median: 0.6875
Test median of differences, Null hypothesis is negative median while alternative hypothesis indicates positive median, 5% CI
Cannot reject null hypothesis, negative median given a 5% CI

(EC_3_8_11) C:\Users\hadji\Documents\Amsterdam\Year 2\Evolutionary Computing\Assignment\ECEvoman>
```

Figure 2: Test performed for each enemy considering the arrays used to create the box plot.

As it can be seen by Fig. 2, each array in the list corresponds to the difference of arrays between the two algorithms, first of enemy 2, then of enemy 5 and finally of enemy 3.

B APPENDIX: ENEMY 8

This appendix provides information about enemy 8 who is not included in the chosen list of 2, 3 and 5 as there was not enough time to run 2, 5 and 8 together in a list of run.

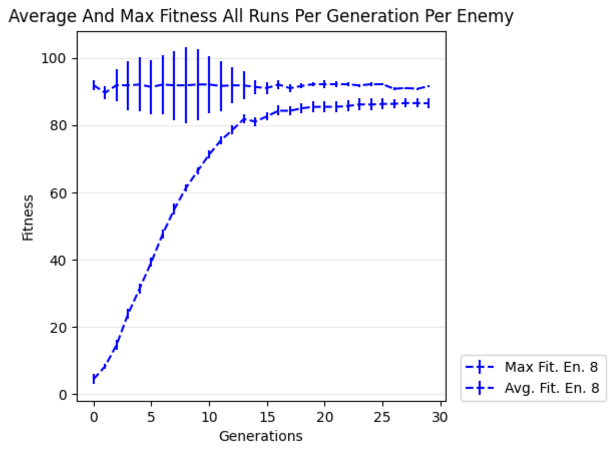


Figure 3: Enemy 8 algorithm a line plot

As indicated by Fig. ??, it would taken triple the time to add the enemy in the predefined list of enemies to beat so it is added in the appendix.

If however it is not allowed, please disregard this appendix.

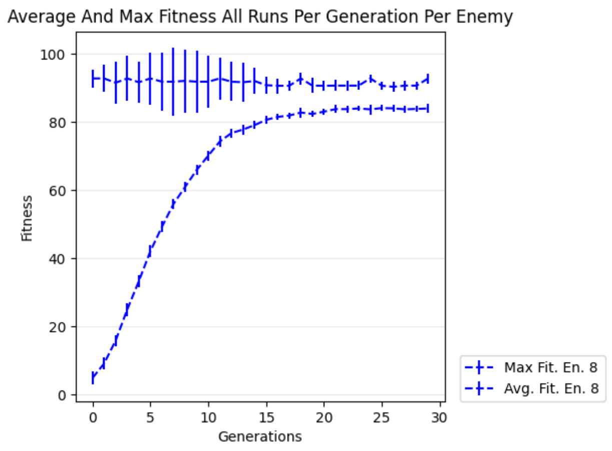


Figure 4: Enemy 8 algorithm b line plot

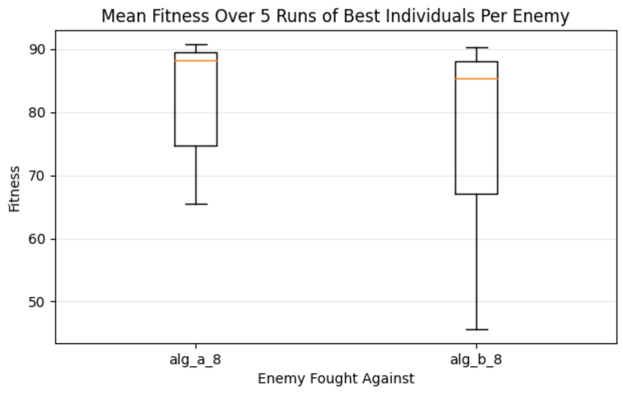
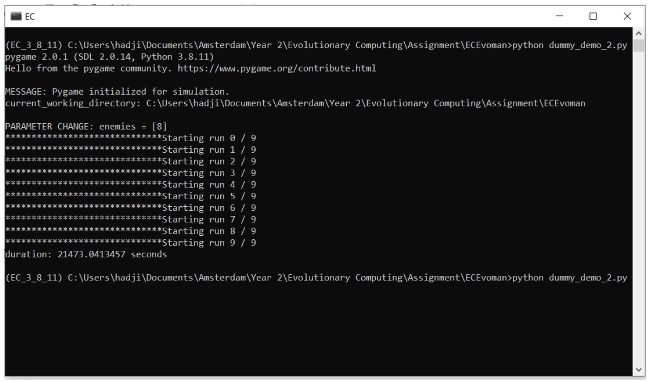


Figure 5: Enemy 8 box plot



```
(EC_3_8_11) C:\Users\hadji\Documents\Amsterdam\Year 2\Evolutionary Computing\Assignment\ECEvoman>python dummy_demo_2.py
pygame 2.0.1 (SDL 2.0.14, Python 3.8.11)
Hello from the pygame community. https://www.pygame.org/contribute.html

MESSAGE: Pygame initialized for simulation.
current_working_directory: C:\Users\hadji\Documents\Amsterdam\Year 2\Evolutionary Computing\Assignment\ECEvoman

PARAMETER CHANGE: enemies = [8]
******************************Starting run 0 / 9
******************************Starting run 1 / 9
******************************Starting run 2 / 9
******************************Starting run 3 / 9
******************************Starting run 4 / 9
******************************Starting run 5 / 9
******************************Starting run 6 / 9
******************************Starting run 7 / 9
******************************Starting run 8 / 9
******************************Starting run 9 / 9
duration: 21473.0413457 seconds

(EC_3_8_11) C:\Users\hadji\Documents\Amsterdam\Year 2\Evolutionary Computing\Assignment\ECEvoman>python dummy_demo_2.py
```

Figure 6: Time taken for a single enemy to finish, 21473 seconds which is 5.96 hours

REFERENCES

[1] James E. Baker. 1987. Reducing Bias and Inefficiency in the Selection Algorithm. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic Algorithms and Their Application*. L. Erlbaum Associates Inc., USA, 14–21. https://doi.org/10

[2] Fantinato D. Miras K. Eiben A. Vargas P. De França, F. 2019. EvoMan: Game-playing Competition.

[3] Miras K. Fabricio, D. 2016. Evolving a generalized strategy for an action-platformer video game framework.

[4] Miras K. Fantinato, F. 2016. An electronic-game framework for evaluating coevolutionary algorithms.

[5] John H. Holland. 1992. Genetic Algorithms. *Scientific American* 267, 1 (1992), 66–72. https://doi.org/10.1038/scientificamerican0792-66

[6] K. Miras. 2019. EvoMan - Framework 1.0.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[8] Frank Wilcoxon. 1945. Individual Comparisons by Ranking Methods. *Biometrics Bulletin* 1, 6 (1945), 80. https://doi.org/10.2307/3001968

[9] Ágoston E. Eiben and James E. Smith. 2015. *Introduction to evolutionary computing* (2nd. ed.). Springer-Verlag, Berlin.