Task 1: Specialist Agent - Role of Recombination

Influence of Parent Size on Evolutionary Algorithms Within a Gaming Framework

Evolutionary Computing

Team 39 - 1/10/2023

Vrije Universiteit van Amsterdam

Ilinca Dumitras
2812183

Joshua Haas
2825076

Anna Nijmeijers
2618830

Doruk Tarhan
2814491

Lukas Unruh
2772548

1 INTRODUCTION

Innovative solutions are needed to meet demands for new gameplay experiences, streamlined design, and increased realism in the fast evolving video game business [1]. Game difficulty is crucial for player immersion [2]. Recent research has shown that AI players can estimate game difficulty and immersion [3].

Neuroevolution, a computational method that applies the idea of natural selection to artificial neural networks [4], is a novel AI player construction method. Neuroevolution uses evolutionary algorithms to create artificial neural networks (ANNs) with modified connection weights and structures. Thus, allowing them to adapt and develop over time, making them useful for video games and other applications. Neural network weights can learn and dynamically respond to game changes, thus improving their capacity to construct AI players [5].

Neuroevolution relies on evolutionary algorithms (EA) and, in some cases, evolution strategies (ES). Beyer et al. define "EA" as all probabilistic approximation and optimization methods based on Darwinian evolution [6]. The variation-selection paradigm approximates optimal states with succession-passed enhancements. Variation operators provide genetic diversity and selection guides evolution [6]. ES is an EA version that uses the natural problem representation without a genotype-phenotype mapping for object parameters [6]. These methods optimize neural network evolution for greater performance [5]. In this paradigm, EAs and ES fine-tune neural networks to fit the game environment, improving the gaming experience. ES are established algorithms for optimizing complex real-valued functions [5]. The use of ES is well justified, particularly with recent advancements such as the Covariance Matrix Adaptation (CMA) [7]. Neuroevolution has great potential, but optimising evolutionary algorithms is difficult, especially in complicated games [4]. EA efficiency and AI player sophistication are among these challenges.

EA optimization can be achieved by manipulating the recombination. Recombination creates new genetic material from two or more parents [8]. This fundamental evolutionary mechanism creates genetic variety and allows adaptive evolution [9].

Recombination operators can be modified by increasing the parent size, which generates the next generation's offspring. Researchers have found that increasing parent size improves the EA algorithm [10]. However, minimal studies in this sector [11] highlight the need for greater research on how to improve AI players. EvoMan is used to test AI players, i.e., the modified neural network. EvoMan is a game-playing competition framework that tests competitive game-playing agents in various challenges, including learning how to win a match against a single opponent, generalizing the agent to win against all opponents, and co-evolving the agent and opponents to produce intelligent opponents of increasing difficulty [12].

Our current research examines how increasing parent size affects ES performance and compares it to a classic ES utilizing the Evoframework to evaluate AI player performance. Using neuroevolution, the research facilitates the continuous development and advancement of the video game industry. This paper provides a comprehensive and well-referenced assessment of neuroevolution's use and promise in video games.

2 METHOD

2.1 Evolution Strategies

ES were invented in the early 1960s by Rechenberg and Schwefel [5]. Parent selection in ES determines how candidates are chosen to serve as parents for the next generation. It is a critical component in shaping the genetic diversity of a population. Survivor selection in ES involves deciding which candidate solutions from the current generation will be transferred to the next generation. This selection process is critical for shaping the characteristics and evolutionary trajectory of the population. ES typically operates with a relatively high selection pressure, i.e., it maintains a relatively low takeover time. The takeover time is defined as the number of generations required to fill the population with copies of the best individual, starting from a single copy [4]. High selection pressure intensifies competition between individuals and promotes rapid convergence toward optimal solutions.

2. 2 Experimental setup

Two algorithms are tested against three of the eight Evoman enemies. Following one-run tests against all enemies. Because of their best performance, enemies 1 (FlashMan), 7 (Bubbleman) and 8 (Quickman) were chosen. The final test ran each algorithm 10 times for each enemy resulting in a total of 20 runs. The mean and maximum fitness of the entire population, including all solutions, are documented for each generation.

To analyze the outcomes, we modified the weights of an artificial neural network (ANN) using the ES algorithm. The ANN has 20 input neurons for EvoMan sensors. These neurons' input values are normalized from -1 to 1. The ANN also has 10 hidden neurons and 5 output neurons for player actions. ANNs use sigmoid activation. To evaluate ANN efficiency, we use the following fitness function:

$$fitness = \gamma \cdot (100 - e_e) + \alpha \cdot e_p - log(t)$$

This function reflects the enemy's energy level, e_e, the player's energy level e_p with e_e, $e_p, \in [0, 100]$. Furthermore t denotes the game's total time steps. For γ and α values of $\gamma = 0.9$ and $\alpha = 0.1$ are used, based on their superior performance in previous studies [10]. The fitness function reduces opponent energy, increases player energy, and considers game time.

The algorithm calculates the population's best, mean, and standard deviation fitness scores for each generation. Furthermore the individual gain is calculated:

$$individual\ gain = e_e - e_p$$

The individual gain gauges performance against one opponent. t-test is performed to determine if the two algorithms' average individual gains for each opponent differ significantly.

Several components are crucial to ES algorithm optimisation. These components can be categorized for clarity:

Representation. Solutions are real-valued vectors with network weights in the ES framework.

Generation. The generation process rounds the results after a set number of repetitions, in this case 40. This stage affects algorithm progression.

Population. Each ES algorithm is initialized with 25 candidate solutions. Thus, the ES algorithms have a population size of 25. The population size greatly affects algorithm exploration and exploitation.

Offspring. Each generation produces 100 offspring in the population. These offspring are crucial to the evolution of the solutions and algorithm search.

Parent selection. The ES algorithm selects parents using uniform random sampling, ensuring a balanced mating pool. Therefore, the probability of becoming a parent is given by the formula: $P_{uniform}(i) = \frac{1}{\mu}$ for $1 \leq i \leq \mu$ [8]. Traditional ES methods inspired this approach [8].

Recombination in classical ES. Recombination in classical ES is discrete, with each offspring allele inherited from a randomly selected parent [8]. This recombination produces 100 offspring.

Recombination in multiparent ES. Recombination in ES with many parents involves crossover mating. The offspring inherits each gene from one parent, with equal chances of inheriting each parent's genes [8]. This is repeated 100 times to produce 100 offspring.

Mutation. Mutations happen when gaussian noise is added to the individual after being scaled by the sigmas, aiding the exploratory purposes of the solution domain.

Mutation-Stepsize. The mutation stepsize is self-adjusted to explore and exploit the search space for an optimal solution. Gaussian noise and learning rate τ govern adaptation rate by multiplying it to sigma which affects step size [8, p.57]. This formula greatly affects weight changes throughout the mutation.

$$p(\Delta x_i) = \frac{1}{(\sigma\sqrt{2\pi})} \cdot e^{-\frac{(\Delta x_i - \varepsilon)^2}{2\sigma^2}}.$$

Survivor selection. Survivor selection uses the $(\mu + \lambda)$ selection system, with μ representing parents and λ representing progeny. This selection approach was chosen over the (μ, λ) scheme to preserve valuable candidate solutions, as the latter would delete all parents and thus the valuable solutions [12]. Maintaining these valuable answers is the key to the algorithm's success.

3 RESULTS

Figures 1, 2, and 3 display the best and mean fitness values for each algorithm observed in each generation for enemies 1, 7 and 8. Moreover, the plots illustrate the average and standard deviation calculated over 10 runs for both the mean and maximum fitness.

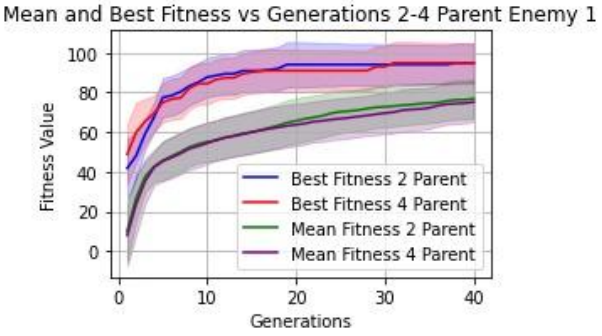


Figure 1 shows how the classic two-parent ES model and four-parent method perform versus enemy 1. Both algorithms' best fitness values stagnate around 94, indicating that their performance is nearly identical. The mean fitness is slightly greater for the classical ES technique with 2 parents for recombination than the 4 parent ES (2 parent = 76.8; 4 parent = 75.13) . The 4 parent strategy also has a little greater initial best fitness. Both algorithms begin with a maximum fitness of 50 and a mean fitness near 20.

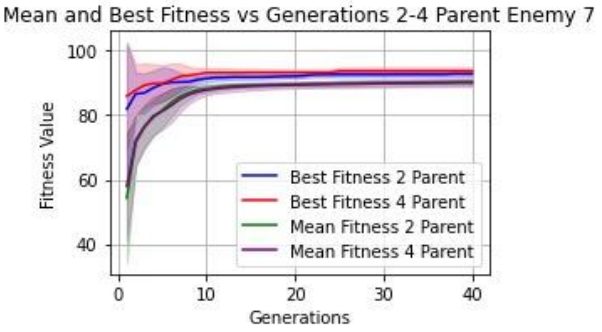


Figure 2 compares the classic ES model and four parents approach against enemy 7. In all cases, the best starting fitness is 85, but the four parents model starts somewhat higher.The best fitness is ultimately close to 93 for both algorithms. Still, 4 parents seem to produce a little greater best fitness value (4 parents = 93.6; 2 parents = 92.8) . Both algorithms average fitness starts about 60. Nearly identical trends lead to a mean fitness of 90 for both algorithms. The variance in each case approaches zero after 15 rounds, indicating that the algorithms converge.

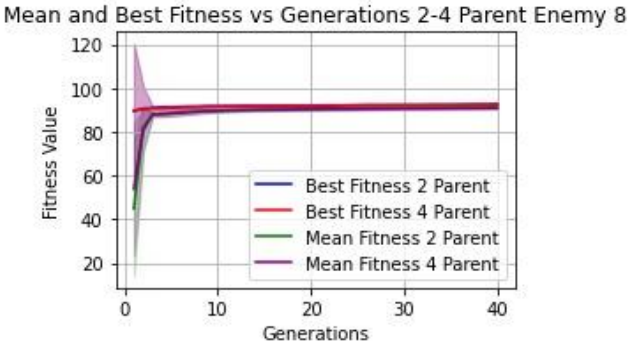
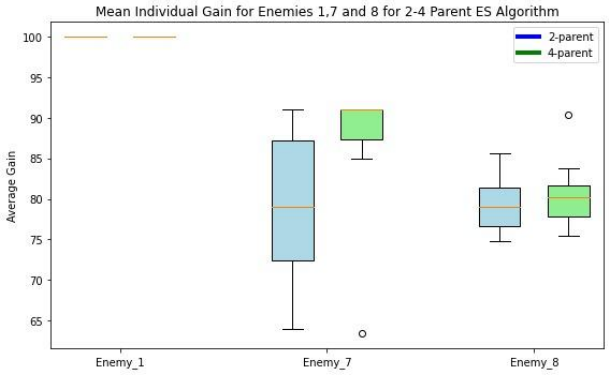


Figure 3 shows how the classical ES model and four parents approach performed versus enemy 8. The best fitness for both

algorithms are around 92. The average fitness for both algorithms are around 91. The models appear to perform similarly. In this case, the starting best fitness is 90 and the mean fitness is 50. Both methods appear to converge after 5 iterations.



Lastly, the boxplots in *Figure 4* show the individual gain for each algorithm for each enemy. The individual gain for Enemy 1 remained at 100 for both the classic ES and the 4 parent ES. Therefore, there was no distinction between the performance of the algorithms for Enemy 1. Additionally, the Wilcoxon t-test is utilized to assess any statistically significant difference between the two algorithms' mean individual gains. The Wilcoxon t-test was chosen due to small sample size. The Wilcoxon t-test result on Enemy 7 showed that the means do not significantly differ from each other (p-value = 0.069, t = -0.787). Similar results are yielded for Enemy 8, the means between the two algorithms do not differ significantly (p-value = 0.432, t = -0.787).

4 DISCUSSION AND CONCLUSIONS

The data displayed in Figures 1, 2, and 3 reveals a variety of observations regarding the performance of the ES algorithms when using different parent sizes for recombination against different enemies within the EvoMan framework. Looking at the performances, the results highlight a strong similarity between the classical two parent ES model and the four parents approach. Furthermore, the individual gain of the algorithms did not differ significantly from each other, which is another indication that both algorithms the classical ES and the 4 parent ES perform similar. This consistency could indicate that the performance of ES algorithms might rely more strongly on other components rather than the recombination one. The lack of change in the results as the parent configurations were modified may highlight the robustness of the ES algorithms since they are able to maintain their performance regardless.

Nonetheless, some differences were observed between the two approaches and among the different enemies. For instance, the starting fitness values varied as the enemies changed. In particular, the starting best fitness of enemy 8 was around 90, which was higher compared to the value of around 80 observed in enemy 7 and 50 seen in enemy 1. Furthermore, the starting best fitness was in general slightly greater for the four parents ES compared to the two parents one. This may suggest that the larger parent size leads to quicker progress toward the optimal fitness. Lastly, the algorithms converge after a different number of iterations depending on the enemy. For instance, when playing against enemy 8 both algorithms converge after approximately 5 iterations instead of 15 iterations as observed against enemy 7.

Despite the difference in evaluation criteria between the two publications, de Araujo and de France [10] published only the maximum fitness values for the enemies: 90 for enemy 1, 56 for enemy 7, and 83 for enemy 8. Instead, our methods, the classical ES and 4-parent ES, performed better, with average best fitness values of 94, 93, and 92 for enemies 1, 7, and 8. This indicates that our methods outperformed de Araujo and de France's Genetic algorithm (GA). The discrepancies between ES and the GA may be due to the components of the algorithms. The improved performance observed in the current research paper may be attributable to the distinct mutation and recombination strategies utilized. This is supported by prior research showing the efficacy of ES algorithms in adjusting network weights [7]. Another contributor to the disparate results may be the disparity in selected parameters. Moreover, during the current research the ES algorithms showed the best results when playing against enemies 1, 7 and 8 . However, in their case the authors found that the GA10 performed best against enemies 1, 2 (AirMan), and 8. The performance discrepancy highlights how unique the challenges presented by each enemy are and how each strategy can perform differently depending on the facing issue. This might suggest that instead of generalizing algorithms for all players, components such as mutation, recombination and survival should be tailored according to the enemy faced.

The lack of significant difference between the classical ES and the 4-parent ES could be due to various factors, such as the time constraints of the experiment. Due to time limitations, comprehensive parameter testing, including variations in population sizes and fitness function optimization, was not feasible in the current research paper. Thus, future research should focus on hypertuning the parameters, which could offer a broader overview of parent configurations and their performance impacts. At the same time, these results also question whether parent configuration truly impacts the performance of ES or whether other components could be more detrimental.

In conclusion, this study did not find that multiparent recombination improves the ES algorithm's performance significantly, but it did provide valuable insights on AI players within the gaming industry. The robustness of ES given different parent configurations, could support the development of more flexible players. Similar to Arajo and de Franca [9], the report showed disparities in enemy performance, emphasizing the need for further investigation into how evolutionary algorithm components interact with each enemy within the EvoMan environment. However, the current research showed that 2-parent and 4-parent ES outperformed the GA algorithm of de Araujo and de France [10], The results of the current research provide valuable insights into the effectiveness of ES algorithms in game development. These findings offer a deeper understanding and suggest potential adaptations in strategy development and enemy considerations, crucial for optimizing AI agents in the gaming industry.

REFERENCES

[1] F. Handrich, S. Heidenreich, and T. Kraemer, 'Innovate or game over? Examining effects of product innovativeness on video game success', Electron Markets, vol. 32, no. 2, pp. 987–1002, Jun. 2022, doi: 10.1007/s12525-022-00521-7.

[2] Wong, K.W., Fung, C.C., Depickere, A., Rai, S., 2006. Static and dynamic difficulty level design for edutainment game using artificial neural networks. Edutainment 2006 LNCS 3942, 463–472.

[3] Roohi, S., Guckelsberger, C., Relas, A., Heiskanen, H., Takatalo, J., & Hämäläinen, P. (2021). Predicting game difficulty and engagement using AI players. Proceedings of the ACM on Human-Computer Interaction, 5(CHI PLAY), 1-17.

[4] S. Risi and J. Togelius, 'Neuroevolution in Games: State of the Art and Open Challenges', IEEE Transactions on Computational Intelligence and AI in Games, vol. 9, Oct. 2014, doi: 10.1109/TCIAIG.2015.2494596.

[5] E. Galván and P. Mooney, Neuroevolution in Deep Neural Networks: Current Trends and Future Challenges. 2020.

[6] Beyer, H.-G., Brucherseifer, E., Jakob, W., Pohlheim, H., Sendhoff, B., &To, T. B. (2002). Evolutionary algorithms—terms and definitions. http://ls11-www.cs.uni-dortmund.de/people/beyer/EA-glossary/def-engl-html.html

[7] N. Hansen and A. Ostermeier, 'Completely Derandomized Self-Adaptation in Evolution Strategies', *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, Jun. 2001, doi: 10.1162/106365601750190398.

[8] A. E. Eiben and J. E. Smith, Introduction to Evolutionary Computing. in Natural Computing Series. Berlin, Heidelberg: Springer, 2003. doi: 10.1007/978-3-662-05094-1.

[9] Otto, S. P., & Lenormand, T. (2002). Resolving the paradox of sex and recombination. Nature Reviews Genetics, 3(4), 252-261.

[10] Eiben, A. E., & Smith, J. E. (2015). From evolutionary computation to the evolution of things. Nature, 521(7553), 476-482.

[11] K. Miras and F. De França, 'Evolving a generalized strategy for an action-platformer video game framework', Jul. 2016, pp. 1303–1310. doi: 10.1109/CEC.2016.7743938.

[12] Beyer, H. G., & Schwefel, H. P. (2002). Evolution strategies–a comprehensive introduction. *Natural computing*, *1*, 3-52.