A Comparative Analysis of Evolutionary Algorithms for Game-Playing Agent Evolution:

CMA-ES vs. a Custom EA in the EVOMAN Framework

Joshua Brook
MSc Artificial Intelligence
2719998
j.w.brook@student.vu.nl

Máté Nagy
MSc Artificial Intelligence
2697010
nagymatepeter@student.vu.nl

Aisha Malik
MSc Econometrics
2624228
a.a.malik@student.vu.nl

Marin Marian
MSc Artificial Intelligence
2698703
m.marian@student.vu.nl

Julien Testu
MSc Artificial Intelligence
2695398
j.p.testu@student.vu.nl

Vrije Universiteit Amsterdam

X_400111: Evolutionary Computing

Task 1: Specialist Agent

Team 13

01/10/2023

1 INTRODUCTION

Evolutionary computing is a powerful approach in Artificial Intelligence for optimisation problems, inspired by the process of biological evolution. It includes a wide range of algorithms and techniques that aim to solve complex problems by iteratively evolving populations of candidate solutions. Within this context, the Evoman game-playing framework provides a platform for experimenting with evolutionary algorithms, particularly in the domain of game-playing agents [1]. In this framework, we have access to a set of tools and an environment where specialized agents can be evolved to defeat enemies in virtual game worlds.

This report aims to investigate and compare the performance of two Evolutionary Algorithms (EA) methods, namely our own EA implementation and the Covariance Matrix Adaptation Evolution Strategy (CMA-ES), which is one of the most promising derivative-free optimisation methods [2]. The research question that guides our work is:

Is our EA capable of rivaling the CMA-ES and which algorithm provides a higher fitness in terms of the specialist agent's overall performance?

We evaluate both CMA and our custom EA implementation within the framework's individual evolution mode with the single objective of evolving a specialist game-playing agent. Meaning that we focus on optimizing the agent to defeat a specific enemy. However, we also experiment with two other distinct enemies, keeping the same parameters, which allows us to check the generalization capabilities and adaptability of our specialist agent. With the help of boxplots and other statistical graphs we can assess the overall performance and draw a conclusion.

2 METHODOLOGY

In order to answer our research question, our methodology compares two EAs, one built from the ground up by our team, and one based on CMA-ES. In this section, we define and explore the architecture of both EAs and lay out the framework for analyzing their comparable fitness. We conduct ten runs each on three different randomly-chosen enemies to assess the quality and consistency of the EA's performance across different challenges. We record various metrics, including the best fitness achieved, the mean fitness of the population, and the progress of the evolution process over generations in order to know if our EA can outperform the CMA-ES.

2.1 Own EA

The EA we have created for this experiment is made of four parts: *parent selection*, *mutation*, *crossover*, and *survivor selection*. A population of weights used in the neural network is iteratively passed through those four operations, leading to a convergence toward optimal performance in the Evoman simulation. We start by initializing a population of 40 candidate solutions, where the genetic representation of an agent is encoded as a vector of real-valued parameters, which determine its behavior in the game. These parameters correspond to the weights of the neural network that controls the agent's actions.

$$f = 0.9 \cdot (100 - e_e) + 0.1 \cdot e_p - log(t) \quad (1)$$

Each generation, the population's performance in the game is determined by a fitness function (1), based on the energy measures of the enemy (e_e) and player's (e), number of time steps (t) while 0.9 and 0.1 are assumed values of constants.

The energy measures (e) have a range of 0 to 100, and thereby the fitness function for a given candidate can vary from -100 (really bad) to 100 (really good).

I - Parent Selection

In each generation we select a subset of the top 10 individuals from the current population to act as parents for producing the next generation. These individuals with the highest fitness are mutated and recombined to create children in the next generation.

II - Mutation

Mutation introduces genetic diversity into the population by randomly altering the parameter values of selected individuals [6]. We employ a Gaussian noise mutation strategy, where each parameter of the selected individuals is perturbed by a small random value drawn from a Gaussian distribution.

The mutation rate (20% in this case) determines the probability of each parameter being mutated. If a parameter is chosen for mutation, then random Gaussian noise between 0 and 1 is added. There is also a parameter in the mutation function to check where no evolutionary progress has been made in the last 10 generations, and, if so, noise is added between 0 and 10, in order to try to find an exit to a potential local optimum. This mutation strategy is performed on all chosen parents, which can now be combined into new children via crossover.

III - Crossover

Crossover, also known as recombination, generates new candidate solutions (children) by combining the genetic material of two parent individuals [5]. In our implementation, we use a simple arithmetic crossover method, where two parents are randomly chosen from the mutated individuals, and a random index determines the crossover point. A child is created by combining the genetic material before and after this index. Lastly, these new children can be passed to the survivor selection algorithm.

IV - Survivor Selection

Survivor selection determines which individuals from the current generation will be part of the next generation. We use a modified elitism strategy that provides a means for reducing genetic drift [4] that keeps the 10 top-performing individuals (based on fitness) from the current generation. Additionally, we introduce offspring generated through crossover into the next generation. A crucial aspect of our survivor selection strategy involves comparing the fitness of offspring to that of randomly selected individuals from the previous generation. If an offspring outperforms the selected individual, it replaces the selected individual in the next generation.

2.2 CMA Algorithm

The Covariance Matrix Adaptation (CMA) Evolution Strategy (ES) is a highly regarded evolutionary algorithm, well-suited for the Evoman optimization task. It excels in handling high-dimensional search spaces, making it ideal for evolving a winning strategy in the complex Evoman environment. CMA-ES can dynamically adapt its search strategy based on the problem structure, using a covariance matrix to model the search distribution and updating it iteratively to focus the search on promising regions. This adaptability makes it robust to noisy fitness evaluations, allowing it to converge to good solutions, using a probabilistic model and iteratively updating the search distribution based on observed fitness values [3]. Lastly, CMA-ES strikes a balance between exploration and exploitation by combining global exploration and local exploitation through covariance matrix adaptation, efficiently exploring the search space while converging towards optimal solutions.

In our experiment, one individual was randomly generated and passed to the CMA algorithm as an initial solution. It then proceeds to generate multiple candidate solutions and evaluate them to build a performing population. In the case of the CMA algorithm, the evaluation function is transformed into a minimization problem: The fitness value is normalized between 0 and 100, and the absolute value of f - 100 is returned. This means that the maximum fitness value returns a 0 and the lowest returns 100.

3 RESULTS AND DISCUSSION

Here we present the results of our experiments, which reveal the comparative performance of the two EAs. To compare the performance of each algorithm against each enemy, we generated line plots representing the best average (*Avg*) fitness values across 50 generations. The standard deviation for the average mean fitness across the 10 runs is also shown, appearing to decrease as expected as an optimal solution is found.

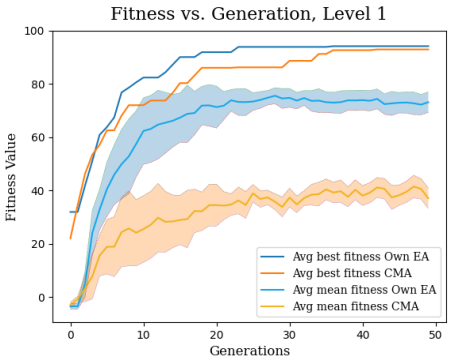

Figure 1. Graph level 1

Figure 1 suggests that our EA has a better overall performance for defeating enemy 1. Our custom EA

outperforms CMA-ES, achieving the best fitness on a high average and a higher mean fitness compared to the CMA algorithm much faster.

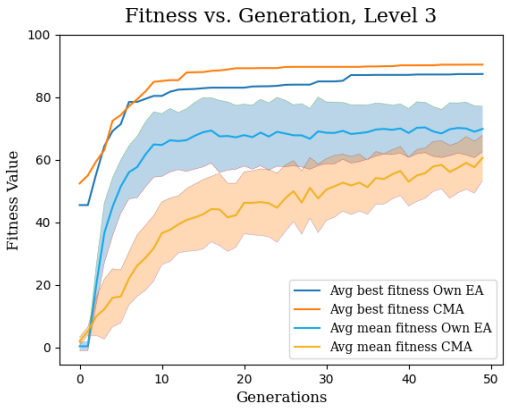

Figure 2. Graph level 3

Figure 2 for enemy 3, CMA achieves a better average best fitness, indicating that it can find high-quality solutions faster. However, our EA maintains a higher average mean fitness, suggesting better overall consistency in performance.

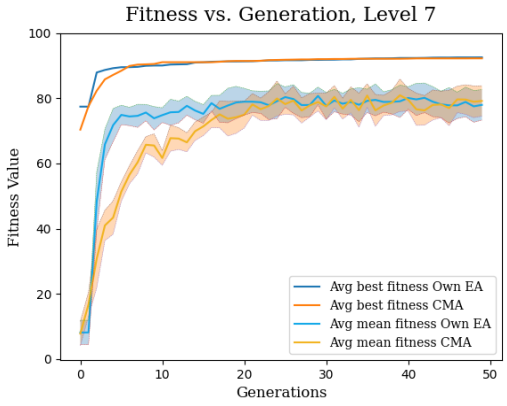

Figure 3. Graph level 7

Figures 3 and 4, both algorithms are close in terms of performance, although the CMA shows a more mellow evolution slope during the first 10 generations, meaning that it evolves at a slower pace. CMA shows a slower mean convergence while our implementation focuses from the beginning on the best 10 individuals and generating offspring from them. Thus, rapidly increasing the population's mean fitness by replacing the worst individuals with better offspring. As for the best fitness, both algorithms are quite similar and both exceed a fitness value of 80 around the 10th generation.

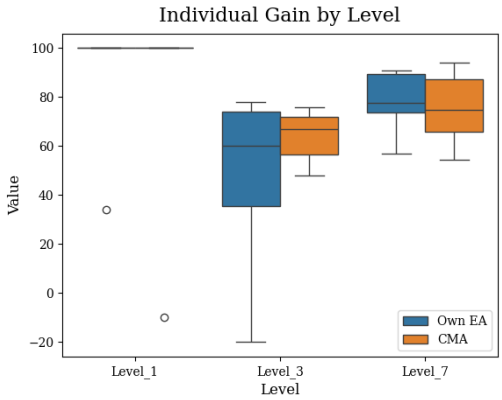

Figure 4. Boxplot Individual Gain by Level

Figure 4 represents the individual gain of the agents per level. Both algorithms obtained very similar results for level 1, except for 2 outliers at 39 for our EA and -10 for the CMA. For level 3, the boxes show a higher mean for the CMA algorithm but our EA shows a higher outlier, which means that CMA shows better consistency but our EA can get better results sometimes. For level 7, the roles are reversed which shows that the EA's performance is influenced partially by the current level although they stay quite similar in terms of performance.

4 CONCLUSIONS

In this task, we compared the performance of two evolutionary algorithms in evolving specialist game-playing agents within the Evoman framework. Our custom EA outperformed CMA-ES against enemy 1, showing its ability to make rapid progress. It also demonstrated overall more consistent results, while CMA-ES initially explored more, resulting in slower early-stage progress.

Future improvements involve adding methods from the CMA into our own EA such as the natural gradient descent which consists of updating the sampling distribution parameters according to an area giving promising fitness values [7] or implementing the maximum likelihood updates which would further help with convergence. In the next Task, involving training a generalist agent, we would like to research further into our chosen method of survivor selection, as well as implementing more parameter control.

REFERENCES

[1] K. Da, S. Miras De Araújo, and F. Olivetti De França, "An electronic-game framework for evaluating coevolutionary algorithms." https://arxiv.org/pdf/1604.00644.pdf

[2] M. C. Fontaine, J. Togelius, S. Nikolaidis, and A. K. Hoover, "Covariance Matrix Adaptation for the Rapid Illumination of Behavior Space," Proceedings of the 2020 Genetic and Evolutionary Computation Conference, pp. 94–102, Jun. 2020, https://doi.org/10.1145/3377930.3390232.

[3] Hansen, Nikolaus. "The CMA Evolution Strategy: A Comparing Review." Towards a New Evolutionary Computation, vol. 192, 2006, pp. 75–102, https://doi.org/10.1007/3-540-32494-1_4.

[4] H. Du, Z. Wang, W. Zhan and J. Guo, "Elitism and Distance Strategy for Selection of Evolutionary Algorithms," vol. 6, pp. 44531-44541, 2018, doi: 10.1109/ACCESS.2018.2861760.

[5] Jansen, Wegener The Analysis of Evolutionary Algorithms—A Proof That Crossover Really Can Help . Algorithmica 34, 47–66 (2002). https://doi.org/10.1007/s00453-002-0940-2

[6] Qingfu Zhang, Jianyong Sun and E. Tsang, "An evolutionary algorithm with guided mutation for the maximum clique problem," in IEEE Transactions on Evolutionary Computation, vol. 9, no. 2, pp. 192-200, April 2005, doi: 10.1109/TEVC.2004.840835.

[7] Akimoto, Y., Nagata, Y., Ono, I., & Kobayashi, S. (2010). Bidirectional relation between CMA evolution strategies and natural evolution strategies. In Parallel Problem Solving from Nature, PPSN XI: 11th International Conference, Kraków, Poland, September 11-15, 2010, Proceedings, Part I 11 (pp. 154-163). Springer Berlin Heidelberg.

Appendix

Our code is available in a Github repo at
https://github.com/joshbrook/evoman13