Vrije Universiteit Amsterdam

# Performance consistency of NEAT vs fixed-topology Neuroevolution EA in multi-objective optimization

Evolutionary Computing - Standard Assignment - Task 2

Group 108:
Rick Geling (2804908)
Matúš Halák (2724858)
Rens Koppelman (2750795)
Isis van Loenen (2676369)

## Abstract

Consistency in performance is relevant in evaluating the reliability of evolutionary algorithms (EAs) when solving optimisation problems. This paper investigates the performance consistency of a NeuroEvolution of Augmenting Topologies (NEAT) model, which is known to outperform fixed-topology Neuroevolution EAs on several tasks. The increased flexibility in potential solutions of NEAT was expected to lower its performance consistency. This was tested by comparing NEAT to a baseline fixed-topology Neuroevolution EA when dealing with a multi-objective optimisation task within the EvoMan framework. The EAs were trained upon two different enemy sets to compare the generalisability when testing on all enemies. Results unexpectedly showed that the baseline EA outperformed NEAT, but with similar consistency in the testing phase. The suboptimal performance of NEAT could be improved by extending the number of generations and adding NEAT-specific multi-objective operators. This paper highlights performance consistency as an important algorithm characteristic and future research is needed to investigate the performance-consistency trade-off in the context of NEAT and multi-objective optimisation tasks.

**Keywords:** Evolutionary Algorithm, NEAT, Performance consistency, EvoMan

# 1 Introduction

Evolutionary Computation (EC) is known to generate robust problem solvers applicable over a wide-range of problems and objectives. Analogous to natural evolution, the algorithmic evolution of problem solutions uses variation and selection operators stochastically, which affects the consistency in generated final solutions. While EA-design is typically focused on optimising the performance, consistency in producing solutions with given performance is essential for applications where EAs repeatedly need to generate good results, in contrast with design applications focused on finding one optimal solution [5]. Thus, comparison of performance consistency between different EAs is relevant and the main focus of this paper.

A popular EA approach for many optimisation problems are Neuroevolution (NE) algorithms, in which weights and biases within artificial neural networks (ANNs) undergo evolution to optimize the ANN's task performance [6]. Standard NE-EAs use fixed-topology ANNs, unlike the NeuroEvoluton of Augmenting Topologies (NEAT) method [11] in which an optimal topology evolves as well. NEAT uses historical markings of genes (enabling efficient crossover between different topologies), maintains innovation and diversity by speciation, and grows in complexity incrementally by adding new nodes while evolving [12].

Earlier research has shown that NEAT outperforms the highest-performing fixed-topology NE-algorithms across a range of tasks [11] [12]. Additionally, NEAT can deal with platformer-game environments, outperforming state-of-the-art reinforcement learning algorithms [2]. This warrants choosing NEAT over fixed-topology NE-EA for complex optimisation problems when aiming to increase the best solutions' quality. However, less is known about the quality consistency of NEAT. Here, we aim to investigate the quality consistency of the generated best solutions using NEAT compared to a fixed-topology NE-EA. Our hypothesis is that the increased search space of NEAT has the trade-off of lower consistency compared to fixed-topology EAs.

We investigate this research question using the electronic game framework EvoMan, which presents 8 different enemy environments which need to be defeated using unique strategies [9][10]. Our EAs will evolve individual ANN controllers able to beat multiple enemies (representing multiple simultaneous objectives). Both EAs will be trained upon 2 subsets of enemies, but later tested on all enemies. Comparing the quality consistency in both training and testing will reveal insights about the performance and quality consistency of NEAT compared to a fixed-topology EA in multi-objective optimization.

# 2 Methods

## 2.1 Baseline EA

For our baseline EA, we used an EA from our Task 1 and changed its operators to match performance demands of the multi-objective optimization task. Individual solutions of the baseline EA were vectors with 265 elements corresponding to weights and biases of a fixed-topology ANN controller with 20 inputs, 10 hidden nodes and 5 output nodes. To limit demands on computational resources, we set the population size to 150 individuals. To ensure that our initial solutions can explore diverse regions of the complex search space, we used Latin Hypercube Sampling for population initialization [8]. Other operators chosen for their diversity-promoting properties were the BLX-$\alpha$ recombination (allowing offspring genes to lie outside the range of the parents' genotypes) and fitness sharing (we discovered in Task 1 that low sharing distance ensured not punishing similar good solutions and not losing them from the population) [5]. We also implemented a number of operators to promote performance and keep best solutions around, such as elitist survivor selection, tournament-based parent selection, Hall of Fame, Uncorrelated Mutation with $n$ step sizes [5], and finally, a specialist-injection procedure described in detail in section 2.3. The hyperparameters for above operators were optimized as described in section 2.4. The baseline EA is described in Algorithm 1.

**Algorithm 1** Baseline EA

```
Initialize a population of individuals using Latin Hypercube Sampling.
Evaluate the fitness of each individual.
Set the best solution to the current best individual.
Initialize and fill the Hall of Fame with best solutions from the initial population.
Set stagnation to 0.
for each generation until the maximum is reached do
    Encourage diversity through fitness sharing.
    Select parents using a tournament-selection.
    if generation number  mod  injection interval = 0 then
        Replace a random subset of parents by specialists
    end if
    Create offspring by BLX-α recombination of parents.
    Apply Uncorrelated Mutation with n step sizes to offspring.
    Select the next generation based on fitness using elitism.
    Update Hall of Fame if better solutions found.
    Evaluate the fitness of the new population.
    Record fitness and diversity statistics.
    if Generational best fitness < best solution fitness then
        Increase stagnation
        if Stagnation ≥ stagnation threshold then
            Replace worst solutions by Hall of Fame members.
        end if
    end if
    Update the best solution if a fitter individual is found.
end for
return  The best solution found.
```

## 2.2 NEAT

To implement NEAT we used the Neat Python library [7]. Individual solutions of our NEAT algorithm were complete ANN controllers with distinct topologies (recurrent connections allowed) and connection weights, each with 20 inputs and 5 outputs. Population size was also limited to 150 individuals. Initial networks in the NEAT population traditionally have minimal complexity and via evolution networks with beneficial hidden nodes and connections emerge [12]. To speed up the evolution, we optimized the number of starting hidden nodes, alongside other influential parameters as described in section 2.4. For most parameters we used the default values from Neat Python which are based on general best-practices [7]. Our NEAT algorithm is described in Algorithm 2.

**Algorithm 2** NEAT Algorithm

Initialize a population of neural networks with minimal structure (inputs, optimized number of hidden nodes, outputs).
Evaluate the fitness of each neural networks.
Set the best solution to the current best neural network.
**for** each generation until the maximum is reached **do**
    Divide the population into species based on network topology compatibility distance.
    Encourage diversity by speciation and adjusted fitness.
    Select parents from each species.
    Create offspring by crossover.
    Mutate offspring with one of the following mutations:
        - Add new **or** remove existing connection.
        - Add new **or** remove existing node.
        - Alter weights of existing connections.
    Select the next generation based on fitness using elitism.
    Evaluate the fitness of the new population.
    Record fitness and speciation statistics.
    Update the best solution if a fitter network is found.
**end for**
**return** The best neural network found.

## 2.3 Training procedure

Both EAs were trained upon 2 enemy sets in attempt to produce generalisable ANNs performing well against multiple enemies. Enemy set 1 consisted of enemies 2 ('Airman'), 5 ('Metalman'), 7 ('Bubbleman') and 8 ('Quickman'). These enemies were selected based on experimentation showing that they have similar strategies and are the easiest to beat when designing a specialist. Thus enemy set 1 is believed to be an easy 4-objective task. Enemy set 2 additionally included enemies 1 ('Flashman') and 3 ('Woodman') which used different strategies and were harder to beat, yielding a harder 6-objective task. The addition of the two enemies was expected to make the trained ANNs generalisable to more enemies.

For baseline EA training we devised a training procedure to enhance performance. It involved designing a "reserve pool" of three specialist agents for each enemy in the enemy set (for diversity in effective solutions) in advance (for computational efficiency). To bypass specialist elimination during parent selection, we replaced randomly chosen parent solutions with specialists enabling recombination of specialist and generalist genomes, encouraging higher performance on the specific hard-to-generalize enemies. Each specialist was evolved using the same hyperparameters. We simultaneously injected the entire reserve pool into the parent population at tuned injection intervals.

For NEAT we could not implement 'specialist-injection' because of incompatibility with the innovation numbers NEAT uses to identify past solutions within a given NEAT run. However, as NEAT can form a recurrent deep neural network with any topology it has other degrees of freedom to compensate for no specialist injection.

## 2.4 Hyper-parameter optimization

For hyper-parameter optimization of both our EAs, we employed the Tree-structured Parzen Estimator (TPE) algorithm [3]. TPE was chosen for its ability to focus on promising regions of high-dimensional hyperparameter spaces by using Bayesian inference.

In our implementation, we conducted 50 TPE trials per EA on the smaller enemy set 1 (we lacked the resources to optimize each EA on both enemy sets). The explored hyperparameter ranges were based on careful consideration of edge-cases. Each hyperparameter set was evaluated over three independent runs and the Mean Best Fitness was used as the utility function. This tackled the EAs' stochasticity and provided a more reliable measure of a hyperparameter set's performance. The optimized parameters for both EAs can be found in Tables 2 and 3 in the Appendix.

## 2.5 Evaluation

For performance evaluation of each individual during the evolutionary process we used the following fitness function which both EAs aimed to maximise:

$$fitness = \sum_{i=1}^{n} p_i - \sum_{i=1}^{n} e_i \quad (1)$$

Where $p_i$ is the remaining energy of the player after battling enemy i and $e_i$ is the energy of the enemy i after the battle. We chose this averaged gain fitness function because it allowed the EAs to find solutions which beat more enemies compared to the default fitness function. Per generation the population's best, mean and standard deviation of fitness are tracked. Both EAs are trained for 100 generations on enemy set 1, and for 200 generations on enemy set 2 to match the greater task complexity. We performed 50 training runs for each EAs and each enemy set. Single all-time best solutions from each run and their fitnesses (best solution groups) were analyzed for performance and consistency between EAs for both enemy sets. Comparisons between fitnesses are statistically evaluated using the Wilcoxon signed-rank tests and variances of distributions are evaluated using the Levene's test.

# 3 Results and discussion

## 3.1 Comparison of Fitness

First, we saw that higher fitness values are reached when training on enemy set 1 for both algorithms (for both $p<0.001$), confirming that enemy set 2 represented a harder multi-objective optimisation task, see Figure 1A and Figure 1B. Moreover, comparing the algorithms within the same enemy set showed that on average there were no significant differences in best fitness between the two algorithms for enemy set 1 ($p=0.79$), see Figure 1A. This could be because NEAT eventually outperformed baseline EA, supporting earlier research showing NEAT's superior performance (although this difference was not statistically significant in the last generation ($p>0.05$)). For enemy set 2 the average best fitness of the baseline EA was significantly higher than of NEAT ($p<0.001$), see Figure 1B. However, the best fitness of NEAT seemed to continue increasing in the last generations, suggesting that NEAT takes longer to converge in harder multi-objective tasks. Follow-up studies could investigate this with runs using more generations. Note that for both enemy sets the fitness of solutions within NEAT only started increasing after 20 generations. This could be because NEAT works with a much bigger search space and starts with overly simple solutions, therefore after initialization it requires more time to compose useful ANNs. Our use of specialist injection can be seen in the mean fitness curve of baseline EA in both enemy sets by the small dips every 15 generations. The benefit of using specialist injection should be studied in more detail across other multi-objective EAs and tasks.

## 3.2 Comparison of consistency

To investigate EAs' performance consistency, we compared the distributions of the best solutions fitnesses, see Figure 1C and D. For enemy set 1 the variances of baseline EA and NEAT of the best solutions fitnesses did not differ significantly (p=0.41) indicating comparable consistency, although the best solutions of baseline EA outperformed NEAT on average (p<0.001). For enemy set 2 the best solutions fitnesses had significantly higher variance for baseline EA ($\sigma^2$=110) than for NEAT ($\sigma^2$=34) (p<0.01), while also outperforming NEAT (p<0.001). The combination of higher performance and higher variance in baseline EA represents a trade-off between performance and consistency. Surprisingly these results contradict our hypothesis. The lower performance of NEAT can be explained by not using advanced operators tailored for multi-objective tasks. Research has shown that adjustments can be made to increase the multi-objective performance of NEAT [1][4].

While the best solutions groups of baseline EA were normally distributed, the best solutions groups of NEAT were not (both p<0.01). The NEAT distributions were skewed, tending towards bimodal, suggesting a higher-performing subgroup evolved. Future research could investigate whether the higher-performing subgroup would dominate the distribution of best solutions when extending the number of generations and adding multi-objective operators.

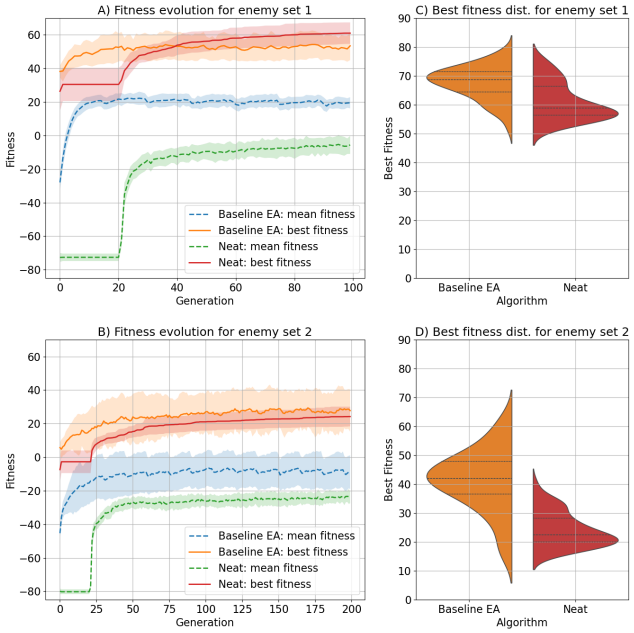


**Figure 1: Fitness evolution for baseline EA and NEAT: Mean fitness and best fitness with standard deviation ranges per generation is shown for enemy set 1 (A) and 2 (B) with 50 runs per EA. The distributions of the fitness of the best solutions groups (n=50) are displayed for both algorithms for enemy set 1 (C) and 2 (D).**

## 3.3 Comparison of generalizability

Lastly, we studied the generalisability of the best solutions by testing them against all enemies. For both enemy set 1 and 2 the baseline EA (median of 0.25 and 13.9 respectively)performed significantly better than NEAT (median of -4.9 and 1.6 respectively) (both p<0.001), see Figure 2. This suggests that baseline EA produces more generalisable solutions compared to NEAT. Future research could study whether NEAT with extended training and multi-objective operators produces more generalisable solutions, or instead, overfits to the training set.

The variances between the EA best solutions did not show significant differences for neither enemy set (for both p>0.05). This indicates that baseline EA's higher consistency in training does not carry over when generalising in the testing phase.

Interestingly both EAs performed better when trained upon enemy set 2 compared to enemy set 1 (for both p<0.001), confirming that using a harder multi-objective training set produces more generalisable solutions. The baseline EA trained upon enemy set 2 could beat 6/8 enemies and generated the ultimate best solution. Its results against all 8 enemies are shown in Table 1.

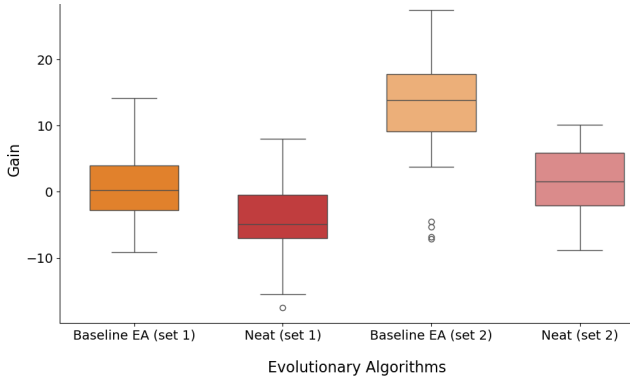


**Figure 2: Gains of best solutions in testing phase: for both enemy sets and for both baseline EA and NEAT the best solutions groups (n=50) are tested against all 8 enemies and their average gains are shown in boxplots.**

| Enemy | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| Player health | **100** | **72** | **42** | 0 | **62.2** | 0 | **79.6** | **22** |
| Enemy health | 0 | 0 | 0 | **70** | 0 | **100** | 0 | 0 |

**Table 1: Best controller against each enemy**

## 4 Conclusion

Our paper investigated the performance consistency of NEAT compared to a fixed-topology baseline EA in multi-objective optimisation tasks. The results showed that unexpectedly NEAT did not perform better, neither in training nor testing. Although the baseline EA showed lower consistency in the training phase for enemy set 2, the algorithms did not differ in consistency anymore in the testing phase. This implicates that the increased flexibility in possible ANN topologies produced by NEAT does not result in a trade-off between performance and consistency. NEAT's subpar performance could be caused by suboptimal design and training of the NEAT EA, which could be addressed by extending the number of generations and adjusting the algorithm for multi-objective tasks [1][4]. Research on how such improvements would affect the performance consistency would aid in further understanding whether a quality versus consistency trade-off exists between NEAT compared to fixed-topology EAs.

## References


[1] Omer Abramovich and Amiram Moshaiov. 2016. Multi-objective topology and weight evolution of neuro-controllers. (2016), 670–677. https://doi.org/10.1109/CEC.2016.7743857

[2] Mikael Andersson. 2022. How does the performance of NEAT compare to Reinforcement Learning? (2022). https://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-309729

[3] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for Hyper-Parameter Optimization. In *Advances in Neural Information Processing Systems*, Vol. 24. 2546–2554. https://proceedings.neurips.cc/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf

[4] Sharat Chidambaran, Amir Behjat, and Souma Chowdhury. 2018. Multi-criteria evolution of neural network topologies: Balancing experience and performance in autonomous systems. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Vol. 51760. American Society of Mechanical Engineers, V02BT03A039.

[5] A.E. Eiben and J.E. Smith. 2015. *Introduction to Evolutionary Computing* (2nd ed.). Springer. https://doi.org/10.1007/978-3-662-44874-8

[6] Dario Floreano and Claudio Mattiussi. 2008. Neuroevolution: From architectures to learning. *Evol Intell* 1 (03 2008). https://doi.org/10.1007/s12065-007-0002-4

[7] Alan McIntyre, Matt Kallada, Cesar G. Miguel, Carolina Feher de Silva, and Marcio Lobo Netto. [n. d.]. *neat-python*.

[8] M. D. McKay, R. J. Beckman, and W. J. Conover. 1979. A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code. *Technometrics* 21, 2 (May 1979), 239. https://doi.org/10.2307/1268522

[9] K. Miras and F. Olivetti. 2016. An electronic-game framework for evaluating coevolutionary algorithms. (2016). arXiv:1604.00644 [cs.NE] https://arxiv.org/abs/1604.00644

[10] K. Miras and F. Olivetti. 2016. Evolving a generalized strategy for an action-platformer video game framework. (07 2016), 1303–1310. https://doi.org/10.1109/CEC.2016.7743938

[11] Kenneth O. Stanley and Risto Miikkulainen. 2002. Evolving Neural Networks through Augmenting Topologies. *Evolutionary Computation* 10, 2 (2002), 99–127. https://doi.org/10.1162/106365602320169811

[12] Kenneth O. Stanley and Risto Miikkulainen. 2004. Efficient evolution of neural networks through complexification. (2004). https://api.semanticscholar.org/CorpusID:22423688


## Appendix

**Table 2: Optimized Parameters for EA1**

| Hyperparameter | Value |
|---|---|
| Fitness sharing distance ($\sigma_{share}$) | 0.05 |
| BLX-alpha ($\alpha$) | 0.49 |
| Tournament size (k) | 4 |
| Elitism fraction for survivor selection | 0.48 |
| Mutation rate | 0.22 |
| Crossover rate | 0.71 |
| Specialist injection frequency | 15 |

**Table 3: Optimized Parameters for NEAT**

| Hyperparameter | Value |
|---|---|
| $\mathbb{P}$(add connection) | 0.45 |
| $\mathbb{P}$(delete connection) | 0.59 |
| $\mathbb{P}$(add node) | 0.31 |
| $\mathbb{P}$(delete node) | 0.23 |
| Number of starting hidden nodes | 23 |