Vrije Universiteit Amsterdam

Bachelor Thesis

Co-Evolution of Generalist Morphology and Control for Diverse Environments

---

Author: Kubilay Tarhan (2721178)

1st supervisor: Anil Yaman
2nd reader: supervisor name

A thesis submitted in fulfillment of the requirements for the VU Bachelor of Science degree in Computer Science

July 9, 2024

Abstract

This section should contain a brief summary of the key points of your paper, including the problem statement, methodology, results, and conclusion.
Keywords: keyword1, keyword2, keyword3

1 Introduction

Since Karl Sims' publication in 1994, 'Evolving Virtual Creatures' [1], demonstrated virtual creatures interacting within a simulated three-dimensional physical environment, numerous researchers have pursued simultaneous co-evolution of both morphology and control [2–5]. By co-evolving both morphology and control, researchers can develop more effective AI agent systems for various tasks. This approach leverages the principle of embodied cognition, which suggests that intelligence arises not solely from the brain or an agent's control system, but from the dynamic interaction between the brain, body, and environment [6].

Moreover, co-evolving a morphology-controller pair (MC-pair) provides major benefits in robotic development, such as the ability to avoid the conventional physical design stage. Co-evolution enables to search for a potential morphological structures by evolving the MC-pair in a simulated environment, instead of wasting time iteratively redesigning and testing the robot in the physical world. Using an approach like this can furthermore speed up the development process and encourage the creation of more innovative morphologies that might not have been thought of in more traditional designs or deemed plausible.

Unfortunately, an evolved MC-pair will still specialize on the environment it has been trained in. This specialization occurs because the evolutionary process fine-tunes both the morphology and controller to the specific conditions of the training environment, resulting in high task performance in only the training environments but reduced task performance to new or marginally different environments. In the real-world, environments are inherently dynamic and unpredictable. Factors such as changing weather conditions, varying terrains, and unforeseen obstacles can significantly alter the operational context of an agent. This variability poses a substantial challenge to the robustness and generalizability of evolved MC-pairs. For example, an agent accustomed to smooth interior surfaces would find it difficult to maneuver over muddy, gravelly, or steeply inclined outdoor terrains. Agents created for static surroundings may also not be able to adjust to environments with changing lighting or moving impediments. Therefore, research into evolving a generalist MC-pair that can operate effectively across a wide range of environments is critically important.

To overcome this limitation, it is crucial to focus on developing robustness and generalizability in evolved agents. Robustness refers to the ability of an agent to maintain desirable behavior despite variations or perturbations in its input and output data [7–9]. Generalizability, on the other hand, refers to the ability of an agent to maintain desirable behavior under different conditions to those encountered during training [8, 9].

In this paper, we investigate the co-evolution of a generalist MC-pair for a wide range of environments, utilizing the Farama Gymnasium's Ant-v4 task [10] as our testing framework. We modified it to allow for the evolution of both the leg lengths and widths of the Ant, thereby adapting its morphology to suit various environments. Our training methodology is heavily based on the training algorithm specified by Triebold et al. [11] with minor modification tailored to our objective of evolving a robust and general MC-pair for wide range of environments. **Our results show...**

2 Background Information

2.1 Co-optimizing of morphology and control

2.1.1 Embodied cognition

Despite the increase of computing power and knowledge, significant progress in the co-evolution of MC-pairs has remained stagnant. Researchers have proposed various hypotheses causing this, ranging from deficiencies in optimization algorithms to the notion that the training environments are not complex enough to facilitate morphological evolution, as outlined by Cheney et al. [12]. However, they suggest a new hypotheses to explain the difficulty in co-evolving MC-pairs, which is the theory of embodied cognition.

Embodied cognition suggests that the cognitive processes arise not solely from the brain, but are a product of the dynamic interaction between the brain and morphology. This interplay means that

even minor changes in morphology can disrupt the connection between the controller and morphology, requiring the controller to adapt itself again to the new morphology. This is also the reason why in [12] the morphologies generally converge before the 100th generation out of a total of 5000 generations, because at that point, further morphological evolution becomes less beneficial as it will only result in lower fitness scores. This phenomenon of premature convergence poses a challenge in co-evolving MC-pairs.

2.1.2 Premature convergence

Premature convergence occurs when the population quickly converges to a local optimum, resulting in a lack of genetic diversity and suboptimal solutions. In the co-evolution of MC-pairs, this leads to early stagnation of morphology evolution, as morphological changes disrupt optimized controllers and are being discarded by selection pressure, preventing the benefits of co-evolution [5].

For the co-evolution of MC-pairs, Lehman et al. [13] propose the Novelty Search with Local Competition (NSLC) algorithm, which utilizes a multi-objective search to optimize both diverse morphologies and fitness in conjunction with local competition, rewarding agents that outperform others with similar morphologies. While this approach did show an increase in the novelty of morphology, it did not yield higher fitness scores compared to using only a fitness objective function. Another method to address premature convergence is fitness sharing, which modifies the fitness function so that agents that are similar get penalized, thereby preserving and encouraging more morphological diversity [14]. Subsequent studies by Cheney et al. [2] propose explicitly protecting agents that have undergone a recent morphological mutation, which reduces the selection pressure and gives the controller more time to adapt to the new morphology. Using the open-source soft-body simulator VoxCad as the physics engine, their experiments showed that this method produced significantly higher fitness scores and increased morphological diversity. Additionally, it delayed premature convergence, as the best-performing agents emerged in later generations, who initially would have been discarded otherwise. Stensby et al. [3] extended upon this work by utilizing a more indirect approach to mitigate premature convergence. They discuss that increasing the agents' exploration of new morphologies can be facilitated by training the agents in a diverse range of environments. These environments were made incrementally more challenging using the Paired Open-Ended Trailblazer (POET) algorithm. Their findings demonstrated that environments generated by POET increased morphology diversity, indicating that POET, or other forms of curriculum training, could be effective in delaying convergence.

2.2 Evolutionary strategies

2.2.1 Neuroevolution

Neuroevolution, the process of evolving artificial neural networks using evolutionary algorithms, has been proven to be a great alternative to traditional reinforcement learning algorithms, particularly in continuous and high-dimensional input spaces [15]. **Give some general knowledge on Neuroevolution and why it is so cool I guess**

2.2.2 CPPN-NEAT

Conventionally, the topology of the neural network was established and fixed prior to training, which is a huge drawback, because the network's topology significantly influences the agent's performance. This drawback led to the development of Topology and Weight Evolving Artificial Neural Networks (TWEANNs), with the most prominent being the NeuroEvolution of Augmenting Topologies (NEAT) algorithm [16]. This algorithm evolves both the weights and the topology of neural networks. It initializes a population of minimal networks with only input and output nodes and incrementally adds nodes and connections through mutations while utilizing a genetic algorithm for optimization. An important technique NEAT uses is speciation, which protects mutated networks, ensuring that new structures are not discarded and have the chance to evolve.

A notable advantage of NEAT is its use in conjunction with Compositional Pattern-Producing Networks (CPPNs). Unlike conventional neural networks, CPPNs utilize a variety of activation functions to generate complex and regular patterns [17]. This can be employed for encoding morphology parameters by mapping morphology features to specific values. In a study by Cheney et al. [18], CPPN-NEAT was used to encode the morphology of a soft robot, demonstrating that generative encoding using CPPNs assigned tissue cells to the voxels in a logical way, ensuring global coordination, resulting in improved locomotion. In contrast,

the direct encoding assigned the voxels more independently of their neighboring voxels, resulting in less coordination and worsening locomotion.

2.2.3 Constraint-handling

Many optimization situations are subject to inherent constraints, such as the angular limits of a robotic arm, the allowed range of values for a mathematical function or simply in real life to use as less of a resource as possible. To effectively adhere to these constraints, evolutionary algorithms can employ several methods that direct the search away from prohibited areas of the solution space. For instance, penalty functions, which reduce the fitness score of solutions that violate constraints, therefore discouraging their selection. Additionally, repair algorithms actively modify solutions to meet the constraints that are in place. Another method is the multiobjective approach, which treats each constraint as a separate objective to be minimized alongside the main objective [19].

2.3 Robustness and generalizability

Robustness refers to the ability of an agent to maintain desirable behavior despite variations or perturbations in its input and output data [7–9]. Consequently, a robust agent is less susceptible to input and output perturbations. This makes robustness a critical attribute, as inputs and outputs from the training environment can differ significantly from those in the testing environment, especially in real-world scenarios. For instance, an input perturbation can be caused by a sensor defect, resulting in slight measurement errors. In reinforcement learning, this means that we need to build robustness against the uncertainty of state observations and the actual state. Similarly, an output perturbation can result from a motory issue of the agent. In reinforcement learning, this means that we need to build robustness against uncertain actions between the actions generated by the agent and the conducted actions [9].

On the other hand, generalizability refers to the ability of an agent to maintain desirable behavior under different conditions to those encountered during training. It assumes correct input and output data and is concerned on the actual differences between the training environment and the testing or production environment. Testing generalization is divided into two distinctive parts. The first part is called interpolation, which requires the agent to perform well in environments similar to those of training, with both the testing and training environment parameters drawn from the same distribution. The second part is called extrapolation, which requires the agent to perform well in environments different from those of training, with the testing and training environment parameters drawn from separate distributions [8, 9].

There are some ways to achieve more robust agents. In this example [20], they compared the robustness of the models DENSER and NSGA-Net on the CIFAR-10 image classification task. It was found that the DENSER model exhibited an higher robustness, which concludes that certain architectures inherently have a higher degree of exhibiting robustness. Furthermore, one of the most popular method is adverserial training, where the agent is trained on adverserially perturbated training data. One way of doing this is finding the worst case perturbation at each training episode and training the model using the dataset with this perturbation [21].

The two main approaches to achieving generalizable agents are either training a generalist agent, which involves a trade-off in performance under specific conditions compared to a specialist agent, as shown by Triebold et al. [11], or developing agents that can explicitly adapt to certain conditions [8]. In this paper, we will focus on the first approach. Recently, there has been a growing recognition of the importance of using variability to train better and more generalized agents. Raviv et al. [22] discusses the relationship between variability and learning outcomes, highlighting a universal principle that variability enhances learning. This principle also holds true for machine learning, where employing a more variable learning schedule can improve an agent's generalizability across a wide range of morphologies. Similarly, Stensby et al. [3], applied the same principle of variability by training agents in a curriculum of environments generated by the Paired Open-Ended Trailblazer (POET) algorithm, where the generated environments were incrementally more difficult, enabling the agent to learn more efficient and complex behaviors. These studies demonstrate that using a variable learning schedule increases both robustness and generalizability.

3 Method

For the evolution of our MC-pair, we adopted the same algorithm initially proposed by Triebold et al. [11] with some minor modifications. Consider the set of training environments $E = \{e_1, e_2, \ldots, e_n\}$ and the validation set $V = \{v_1, v_2, \ldots, v_n\}$, where in our study $E = V$. Both E and V were established prior to training. Triebold et al. explored different training schedules, which determined in what order the training set was used during the evolutionary process. They found that using an incremental training schedule, which means that the environment will be modified incrementally after every generation, yields the best results. Therefore, we will only consider the incremental training schedule in our analysis. Lastly, we initialize an empty generalist MC-pair set $G = \{\}$, which will store the evolved generalist MC-pairs throughout the evolutionary process, and an empty environment partition set $P = \{\}$, which will store partitions of the set V that corresponds the a generalist MC-pair in G. The algorithm will partition the environments and train a generalist MC-pair for that partition. This happens on instances where the environments greatly differ from one another and a generalist MC-pair is simply not possible.

The evolutionary process starts by initializing the first generation of MC-pairs, comprising both the ANN weights $\overrightarrow{W} = \{w_1, w_2, \ldots, w_n\}$ and the morphology parameters $\overrightarrow{M} = \{m_1, m_2, \ldots, m_n\}$. For the optimization process, $\overrightarrow{W}$ and $\overrightarrow{M}$ are concatenated into a singular vector $\overrightarrow{I} = \{w_1, w_2, \ldots, w_n, m_1, m_2 \ldots, m_m\}$ and fed to the XNES optimizer. Because the order of magnitude of the ANN and morphology parameters are very different we encode the morphology parameters to the same order of magnitude as the ANN parameters.

After every generation i, each MC-pair of the population is evaluated on the current training environment e_i and the MC-pair with the highest fitness score, denoted as $I_{\text{i, best}}$, undergoes further evaluation on the entire validation set V, producing a generalist score g_{best}. If this generalist score is an improvement than MC_{best}, then this is stored in the variable MC_{best}. After h number of generation, when no improvement is found, the evolutionary process is stagnated and the current MC_{best} is evaluated on the validation set, giving a $MC_{\text{best, mean}}$ and $MC_{\text{best, std}}$. Each environment in V, where MC_{best} scored higher than $MC_{\text{best, mean}} - MC_{\text{best, std}}$ will be added as a partition to P and removed from E and V. From here, this process will start again until E and V are empty and thus every partition corresponds to a generalist MC-pair.

We integrated a penalty function within the fitness evaluation to ensure adherence to the constraints set for the morphological parameters. This function considers the decoded morphological parameters $\overrightarrow{M}$, alongside the lower and upper bound of the constraints, C_{lb} C_{ub} respectively, a scalar α, and a growth rate r^i, where i is the number of generation. The penalty function is defined as:

$$Penalty = \alpha r^i \cdot \sum (\max(0, C_{\text{lb}} - \overrightarrow{M}) + \max(0, \overrightarrow{M} - C_{\text{ub}})) \tag{1}$$

Thus the generalist fitness function then becomes:

$$g_{\text{best}} = \frac{1}{|V|} \sum_{i=1}^{|V|} (evaluate(MC_{\text{best}}, v_i) - Penalty) \tag{2}$$

Here $evaluate(MC_{\text{best}}, v_i)$ represents the evaluation function, returning the score of the best MC-pair of that generation on the validation environment v_i.

4 Experimental Setup

4.1 Environment

We use the Farama Gymnasium's Ant-v4 environment [10] to evolve generalist MC-pairs. This environment consists of a flat plane for the ant to walk on with the objective to traverse a long distance. The ant's morphology comprises of four legs, each composed of an upper and a lower leg, with the upper leg attached to the torso. Figure 1 shows how the ant looks like. To evolve a generalist MC-pair, we have to enable the ant to modify its morphology. Each leg part is parameterized into two atttributes: length and width, resulting in a total of 16 distinct morphological parameters. The leg lengths are constrained between 0.1 and 1.5, while the widths are limited from 0.08 to 0.2.

To ensure a fast and succesfull experiment, it was necessary to implement additional modifications to the ant's environment. We set the *termin ate_when_unhealthy* flag to false, because it would terminate the run when the ant's torso of the ant ascends to a specific height. Disabling this allowed for the ant to become larger due to its capability to grow long leg lengths. Furthermore, we introduced additional settings to terminate the run

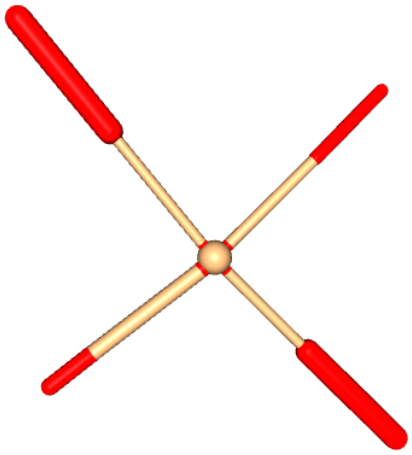

Figure 1: Example of an ant in the environment. The red legs depict the lower leg part and the light beige the upper leg part.

whenever no significant movement forward was detected, suggesting that the ant froze into place. Another custom rule was implemented to detect if the ant flipped upsidedown by monitoring its z-vector. These two additional implementations reduced evaluation times significantly lower, especially at the start of the evolutionary process.

4.2 Experimental parameters

For the controller, a fully connected feedforward ANN is evolved, based on the topology used by Triebold et al. [11], which consists of a single hidden layer with 20 neurons. The input layer corresponds to the ant's continuous observation space, which includes observations such as the angles between the torso and leg connections or velocity values, being in total 27 observations and thus 27 input nodes. The output layer corresponds to the ant's action space, where actions are values ranging from [-1, 1], representing the torque applied to the rotors, totaling 8 actions and thus 8 output nodes. Each layer also includes a bias node.

For the parameters used in the XNES algorithm we used the default settings provided by the algorithm. The initial standard deviation of the search was set to 0.01 and the initial parameter ranges for the controller and encoded morphology parameters where [-0.1, 0.1]. To decode the morphology parameters a linear transformation is used where the lower bound morphology constraint maps to -0.1 and the upper bound morphology constraint maps to 0.1. The lower and upper bound constraints used for the length and width of the legs are [0.1, 1.5] [0.05, 0.2] respectively. **Should I also insert these linear transformation formula's that I used specifically?**

The parameters used for the penalty function where decided based on initial experimental analysis we executed. For the growth rate r^i we settled on 1.03 in conjunction with two scale factors for α. This means we are using two different penalty functions. The reason being is that when the morphological parameters are decoded into negative values, the environment will crash due to not being able to have negative values for morphology. In these cases, α is set to 1000 and otherwise to 100.

For the evolutionary process, we decided to evaluate the generalist scores after 2500 generation, instead of for every generation from the start. The reason being, to decrease the computational cost, but also because at the start of the evolutionary process, it is hard to gain high fitness scores, due to the search for a morphology to exploit effectively. After 2500 generations, a generalist score will be computed from the most fit individual in each generation. If after 200 generation no improvement is found, the process is terminated or a partition will occur.

4.3 Training data

The MC-pairs are training consist of different simulated environments, where the terrain is algorithmically generated based on the inputs. In this experiment, we consider three different environments. The first environment, referred to as the default environment, is a single static environment where the surface is flat. The two remaining environments are dynamically generated, named the rough terrain environment and the hill terrain environment.

4.4 Testing and evaluation

5 Results

6 Discussion

7 Conclusion

References

[1] Karl Sims. "Evolving virtual creatures". In: *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '94. New York, NY, USA: Association for Computing Machinery, 1994, pp. 15–22. ISBN: 0897916670. DOI: 10.1145/192161.192167. URL: https://doi.org/10.1145/192161.192167.

[2] Nick Cheney et al. *Scalable Co-Optimization of Morphology and Control in Embodied Machines*. 2017. arXiv: 1706.06133 [cs.AI].

[3] Emma Hjellbrekke Stensby, Kai Olav Ellefsen, and Kyrre Glette. "Co-optimising robot morphology and controller in a simulated open-ended environment". In: *Applications of Evolutionary Computation* (2021), pp. 34–49. DOI: 10.1007/978-3-030-72699-7_3.

[4] Joshua E. Auerbach and Josh C. Bongard. "Environmental Influence on the Evolution of Morphological Complexity in Machines". In: *PLOS Computational Biology* 10 (2014), pp. 1–17. DOI: 10.1371/journal.pcbi.1003399. URL: https://doi.org/10.1371/journal.pcbi.1003399.

[5] Luis Eguiarte-Morett and Wendy Aguilar. "Premature convergence in morphology and control co-evolution: a study". In: *Adaptive Behavior* 32 (2024), pp. 137–165. DOI: 10.1177/10597123231198497. URL: https://doi.org/10.1177/10597123231198497.

[6] Josh C. Bongard. "Evolutionary robotics". In: *Commun. ACM* (2013), pp. 74–83. ISSN: 0001-0782. DOI: 10.1145/2493883. URL: https://doi.org/10.1145/2493883.

[7] R. Mangal, A. V. Nori, and A. Orso. "Robustness of Neural Networks: A Probabilistic and Practical Approach". In: *2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*. 2019, pp. 93–96. DOI: 10.1109/ICSE-NIER.2019.00032. URL: https://doi.ieeecomputersociety.org/10.1109/ICSE-NIER.2019.00032.

[8] Charles Packer et al. *Assessing Generalization in Deep Reinforcement Learning*. 2019. arXiv: 1810.12282 [cs.LG].

[9] Mengdi Xu et al. *Trustworthy Reinforcement Learning Against Intrinsic Vulnerabilities: Robustness, Safety, and Generalizability*. 2022. arXiv: 2209.08025 [cs.LG].

[10] Mark Towers et al. *Gymnasium*. 2023. DOI: 10.5281/zenodo.8127025. URL: https://gymnasium.farama.org/.

[11] Corinna Triebold and Anil Yaman. *Evolving generalist controllers to handle a wide range of morphological variations*. Sept. 2023. DOI: 10.13140/RG.2.2.27217.71522.

[12] *On the Difficulty of Co-Optimizing Morphology and Control in Evolved Virtual Creatures*. Vol. ALIFE 2016, the Fifteenth International Conference on the Synthesis and Simulation of Living Systems. Artificial Life Conference Proceedings. 2016, pp. 226–233. DOI: 10.1162/978-0-262-33936-0-ch042. URL: https://doi.org/10.1162/978-0-262-33936-0-ch042.

[13] Joel Lehman and Kenneth Stanley. "Evolving a diversity of creatures through novelty search and local competition". In: 2011, pp. 211–218. DOI: 10.1145/2001576.2001606.

[14] Robert I McKay. "Fitness Sharing in Genetic Programming." In: *GECCO*. 2000, pp. 435–442.

[15] Paolo Pagliuca, Nicola Milano, and Stefano Nolfi. "Efficacy of Modern Neuro-Evolutionary Strategies for Continuous Control Optimization". In: *Frontiers in Robotics and AI* 7 (2020). ISSN: 2296-9144. DOI: 10.3389/frobt.2020.00098. URL: https://www.frontiersin.org/journals/robotics-and-ai/articles/10.3389/frobt.2020.00098.

[16] Kenneth O. Stanley and Risto Miikkulainen. "Evolving Neural Networks through Augmenting Topologies". In: *Evolutionary Computation* 10.2 (2002), pp. 99–127. DOI: 10.1162/106365602320169811.

[17] Kenneth O. Stanley. "Compositional pattern producing networks: A novel abstraction of development". In: *Genetic Programming and Evolvable Machines* 8.2 (2007), pp. 131–162. ISSN: 1389-2576. DOI: 10.1007/s10710-007-9028-8. URL: https://doi.org/10.1007/s10710-007-9028-8.

[18] Nick Cheney et al. "Unshackling evolution". In: *ACM SIGEVOlution* 7 (2014), pp. 11–23. DOI: 10.1145/2661735.2661737.

[19] Oliver Kramer. "A Review of Constraint-Handling Techniques for Evolution Strategies". In: *Applied Comp. Int. Soft Computing* 2010 (Jan. 2010). DOI: 10.1155/2010/185063.

[20] Inês Valentim, Nuno Lourenço, and Nuno Antunes. *Adversarial Robustness Assessment of NeuroEvolution Approaches*. 2022. arXiv: 2207.05451 [cs.NE].

[21] Kai Liang Tan et al. *Robustifying Reinforcement Learning Agents via Action Space Adversarial Training*. 2020. arXiv: 2007.07176 [cs.LG].

[22] Limor Raviv, Gary Lupyan, and Shawn Green. "How variability shapes learning and generalization". In: *Trends in Cognitive Sciences* 26 (May 2022). DOI: 10.1016/j.tics.2022.03.007.