UNIVERSITY OF OSLO

Master's thesis

MEAT: Morphological Evolution of Augmenting Topologies

Tobias Remman Paulsen

Informatics: Robotics and Intelligent Systems
60 ECTS study points

Department of Informatics
Faculty of Mathematics and Natural Sciences

Spring 2023

Tobias Remman Paulsen

MEAT: Morphological Evolution of Augmenting Topologies

Supervisors:
Frank Veenstra
Kyrre Glette

Abstract

Evolutionary Algorithms (EAs) are effective tools for solving various optimization problems. One of these problems is the co-optimization of robot morphology and control. In modular robotics, this challenge lies in the reconfiguration of the shape of a robot composed of various modular components. With EAs, these configurations can be evolved to create robots able to accomplish a task. Because of the wide range of different configurations, finding the optimal arrangement simultaneously with the control parameters can be very challenging. Even though contemporary approaches are promising, evolutionary runs often lead to premature convergence to suboptimal configurations.

To investigate the challenge of premature convergence, this thesis evaluates the performance of evolving modular robots through three different experiments. Experiment 1 investigates how a simple initialization approach compares to a random one, looking at the effects on performance, morphological convergence, morphology, complexity, and diversity. Experiment 2 analyzes the effects of morphological protection, thereby also the performance of combining the simple initialization with protection to create *Morphological Evolution of Augmenting Topologies* (MEAT). Lastly, Experiment 3 examines how evolving in a challenging environment impacts evolution.

The results demonstrate that starting with an initial population without diversity quickly improves and matches methods with higher initial diversity. MEAT performs significantly better in both environments while leading to less complex morphologies. Despite this lower complexity, there are no indications of less exploration of the search space. Maintaining the morphological diversity in the population through morphological protection enables more exploration, and is thereby an effective tool in delaying morphological convergence. These results highlight the potential of MEAT, as augmenting morphological topologies is beneficial when co-optimizing the morphology and control of robots.

Contents

List of Figures

List of Tables

Acronyms

AFPO Age-Fitness Pareto Optimization. 20, 36, 37, 91

ALPS Age-Layered Population Structure. 20, 37, 91

ANN Artificial Neural Network. 14

CO Coupled Oscillator. 28–30, 67, 83, 89

CPG Central Pattern Generator. 17, 29

CTRNN Continuous-Time Recurrent Neural Network. 17

EA Evolutionary Algorithm. 1, 3, 4, 7–14, 16, 18, 20–22, 25, 26, 39, 40, 43, 56, 82, 86, 87

EC Evolutionary Computation. 7

GED Graph Edit Distance. 42

GHS Graph Heurisic Search. 16, 22

MEAT Morphological Evolution of Augmenting Topologies. 3–5, 25, 36, 37, 39, 45, 46, 59–64, 66–69, 71–73, 75, 76, 79–84, 86–88, 90, 91, 93, 94

MPC Model Predictive Control. 16, 22

NEAT NeuroEvolution of Augmented Topologies. 3, 14, 20, 36, 37, 91

RNP Random Initialization No Protection. 39, 40, 45, 46, 48–51, 53–56, 60–64, 67, 68, 72, 73, 75, 76, 80, 81

RP Random Initialization Protection. 40, 60–64, 67, 68, 72, 73, 75, 76, 80, 81

SNP Simple Initialization No Protection. 39, 40, 45–50, 52–56, 60–64, 67, 68, 71–73, 75, 76, 80, 81, 86, 87, 93

Acknowledgements

I would like to thank my supervisor, Frank Veenstra, for his invaluable guidance and encouragement. This thesis would have been possible without inspiring discussions and feedback.

Additionally, I would like to thank my friends and co-students for motivating and supporting me.

The work for this thesis was performed on the Fox high-performance computing cluster, owned by the University of Oslo Center for Information Technology.

Chapter 1

Introduction

1.1 Motivation

All living organisms have gradually evolved throughout millions of years to thrive in their respective environments. Despite the vast diversity of animals that can be found in nature today, they have all evolved from one simple common ancestor that lived a few billion years ago. Charles Darwin's theory of evolution introduced the concept *survival of the fittest* [1]. Natural selection favors individuals that are best adapted to their environment and can outcompete others for limited resources. Thus, these individuals will have a greater chance of survival and reproduction. New species emerge through the interplay of random genetic variations caused by mutations and the process of natural selection. Over many generations, these small mutations can accumulate, leading to significant differences between descendants and their ancestors. At a certain point, the individuals have diverged so much that they are categorized as a new species. Distance and isolation is furthermore a catalyst for speciation, where populations adapt to their environments through millions of years of evolution.

Evolutionary Robotics takes inspiration from the natural evolutionary process and tries replicating it to create well-performing robots for various tasks. Both the control systems and the body plans of robots can be optimized with the help of Evolutionary Algorithms (EAs), but evolving them both simultaneously is a complex task. Instead of manually designing the robot and tuning the controls to a specific need, this process is accomplished through simulated evolution. By modeling a real-life environment accurately, robots can be designed and

optimized automatically through simulations for work in the real world as well. This ambitious goal can lower the cost of designing and producing working robots, as there is no need for human interaction. The solutions found can be very different from the human-made designs but might end up performing better by using unconventional designs. Instead of trying to model robots after animals, evolutionary robots are designed through random mutations.

In 1994, Karl Sims evolved virtual creatures in an artificial world with simulated physics [2]. Both the body plans and the behavior of these creatures were optimized simultaneously. His work demonstrated that these creatures managed to complete various tasks such as swimming, walking, jumping, and following a light source. Sims' research has inspired many researchers since its publication, including [3–9]. As a result, the field of co-optimization of morphology and control has emerged.

A few years prior to Sims' groundbreaking paper, the work on modular robots had already started [10]. These robots consist of separable modules or units that isolate a specific function of the robotic system. Using a finite set of modules, a large diversity of robots can be created by connecting the modules in various ways. Identifying optimal control parameters for a particular modular robot can be a challenging task in itself. By searching for good combinations of both morphology and control, the search space will be orders of magnitudes larger. As a result, finding the best solutions within such a vast search space is a very challenging task. The simultaneous evolution of robot morphologies and control can be achieved through the incorporation of modules as building blocks.

Although a lot of work has been done with co-optimizing morphology and control, understanding how to improve it takes much time and research. Despite the increased computational power since Sims' work with virtual creatures, there has been stagnation in the field with less progress than many had hoped for. One of the main challenges with co-optimization is that the body plans in a population of evolving robots converge earlier than the control systems. This leads to the fact that in the later generations of evolution, only the controls are optimized. Cheney et al. [11] theorized that the reason for this is rooted in *embodied cognition*. When a robot's morphology is changed, there is an added effect that the controller becomes "scrambled" since the commands sent by that controller will be interpreted differently. The controller then has to re-adapt to this new body to be as effective as it was before.

Several approaches have been presented to avoid or reduce the problem of embodied cognition. One is the morphological protection technique presented by Cheney et al. [7], where novel morphologies introduced in the population are protected. This has the effect of protecting newly mutated individuals, allowing them time to adapt their control system to their new body. Additionally, this helps to keep the population morphologically diverse. Other approaches to avoid premature morphological convergence also preserve the diversity in the population through speciation [12], evolving for novelty [4], MAP-Elites [13], and evolution in changing environments [14].

There has not been much research on the effect of the initial population in Evolutionary Robotics, and most of the studies in this field rely on some sort of randomly initialized population. For EAs in general, research has shown varying importance of a diverse initial population [15–18]. Stanley and Miikkulainen introduced the algorithm NeuroEvolution of Augmented Topologies (NEAT) [19], showing that starting with minimal solutions and gradually growing them is desirable when doing Neuroevolution, as there are fewer parameters to be searched initially. When using a random population, the networks start off with many unnecessary nodes and connections already present, and this can make them more complex than necessary.

This thesis will explore a novel approach to co-optimize modular robots' morphology and control, called Morphological Evolution of Augmenting Topologies (MEAT). Inspired by the fact that all living organisms have evolved from a simple ancestor, MEAT employs a minimal initialization method and gradually augments the morphology of the individuals. The evolved robots are also inspired by biology and will exhibit bilateral symmetry, which is found in the vast majority of animals. This trait is very useful for targeted locomotion, as it offers the ability to move in all directions and the ability to turn efficiently. The aim of the experiments is to explore whether starting the evolution with only very minimal body plans can result in diverse and high-performing individuals or if a starting population with higher diversity is necessary. NEAT is another source of inspiration, raising the question of whether the advantages demonstrated here will transfer to the field of Evolutionary Robotics. Specifically, this thesis focuses on whether this can help create high-performing yet less complex morphologies compared to a randomly initialized population. To reduce the problem of premature morphological convergence, MEAT will use a morphological protection method based on age,

inspired by Cheney et al. [7].

1.2 Research Questions

Co-optimization of robot morphology and control tends to result in the morphology converging prematurely. When starting with a population of only minimal robots, the hypothesis is that the morphology will converge even earlier than a random initialization approach. By using morphological protection, the hypothesis is that this early convergence will be avoided, and thus the fitness will increase. Therefore, it is believed that MEAT will perform better or equal to the random initialization while also reducing the complexity of the solutions found.

From these hypotheses, the main research question is created:

- What are the effects of gradually augmenting morphologies from a minimal body plan?

Interesting avenues are initialization, morphological protection, and the environment. The thesis contains three main experiments, each focusing on one avenue and one research sub-question. The three sub-questions are derived as a way to help answer the main question in different contexts:

1. How does the initial population affect the search process?
2. What are the effects of using morphological protection based on age?
3. How does the environment impact the evolution of modular robots?

1.3 Contributions

The main contribution of this thesis is to shed light on how minimal initialization, morphological protection, and environment affect the evolution of tree-based modular robots.

Another contribution is in giving insights into the effects of a multi-objective selection operator on fitness and morphological age. This selection operator maintains higher population diversity, enabling more exploration and a later convergence compared to single objective EA measuring only fitness.

The results show that MEAT leads to better performance for both environments

used in the experiments, at the same time as it leads to less complex morphologies. Thus, MEAT is a promising new method for co-optimizing modular robots.

1.4 Thesis Outline

This thesis is structured into eight chapters: Introduction, Background, Implementation, Experiment 1, Experiment 2, Experiment 3, Discussion, and Conclusion.

Chapter 2 gives an overview of background theory and previous work done that is relevant to this thesis. The topics focused on are Evolutionary Algorithms, Modular Robotics, and Evolutionary Robotics. The following chapter, Chapter 3: Implementation, provides an overview of the methods used and presents the experiment setup. The implementation and design of the modules are discussed, and how the controller and algorithms work are described.

The three experiment chapters each focus on one of the sub-questions as well as the main research question. Experiment 1 focuses on initialization, Experiment 2 on morphological protection, and Experiment 3 looks into the impact of the environment. Each of these chapters includes an analysis of the respective results before a more general discussion of the results and work is provided in Chapter 7: Discussion. Here the limitations and future work are also presented.

Lastly, Chapter 8: Conclusion highlights the main findings and their significance.

Chapter 2

Background

This chapter contains an overview of background theory and related work relevant to this thesis. First, Evolutionary Algorithms (EAs) will be introduced, including a basic overview of how they function. This is followed up by an introduction to Modular Robotics, focusing on the benefit of using these robots and how they can be controlled. The final section of this chapter is about Evolutionary Robotics, with a primary focus on how EAs are used to develop virtual creatures and modular robots.

2.1 Evolutionary Algorithms

EAs are metaheuristic optimization algorithms and are a subset of Evolutionary Computation (EC) [20]. EAs leverage the principles of evolution to perform optimization or learning tasks. Due to their ability to efficiently explore vast and complex search spaces, EAs are particularly useful for solving problems that are difficult or even impossible to solve in polynomial time using traditional algorithms. Examples of these problems are NP-Hard problems and problems with multiple objectives [21]. By introducing variation from random mutation, EAs do not require a gradient in the search space as supervised approaches do. This also makes the algorithms useful in deceptive and discrete search spaces where no gradients can be calculated.

EAs take inspiration from natural evolution and use the concept of natural selection to find the best solutions [22]. Successful or *fit* solutions will have a greater chance of survival and therefore are more likely to reproduce. Introducing variation to the offspring is crucial to improve the average fitness

of the whole population over time. Both in nature and EAs, this variation is produced by random mutations and recombination of parents. While there is no guarantee that an EA will find a globally optimal solution, the stochastic nature of the evolutionary process enables the algorithm to move against a gradient or move when there is no gradient. This randomness can lead to a lot of variation in results.

The basic premise of an EA (Figure 2.1) is to begin by initializing a population of individuals. Each of these is then evaluated and assigned a fitness value. Based on their fitness, a selection of individuals are chosen that are allowed to reproduce. These selected parents are recombined to form new individuals. The offspring are then mutated to introduce variation. After this, the individuals allowed to survive to the next generation are selected. The process of parent selection, recombination, mutation, evaluation, and survivor selection is repeated until a termination condition is satisfied [22].

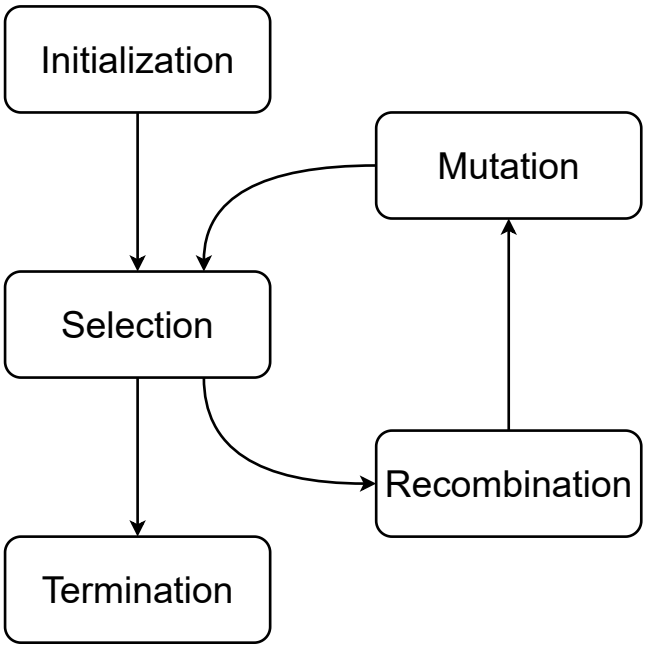


Figure 2.1: Basic scheme of EAs.

In nature, all living organisms carry a set of genes known as their *genotype*, and the set of physical and observable traits are referred to as their *phenotype*. In EAs, these same terms are used to represent the solutions. Each solution is encoded as a set of parameters, the genotype, and the actual solution is referred to as the phenotype. The phenotype is what is evaluated during the evolution process. Therefore, the genotype of an individual has to be mapped to the corresponding phenotype to evaluate the solution. Choosing a fitting way to

represent and map a solution is one of the challenges when using EAs [23].

Initialization

Usually, the first generation of individuals is generated randomly. However, problem-specific heuristics can also be utilized during this stage to increase the initial fitness values of the individuals [23]. Starting with poor-quality individuals can increase the time it takes to find good-quality solutions or possibly lead to premature convergence. Therefore techniques to increase the diversity of the initialized population exist [17, 24].

Fitness Function

The fitness function assigns each individual a fitness value based on how good the solution is or how well it performs. This function has to be designed or adjusted for the specific problem being solved. The measured fitness allows for easy comparisons of the different solutions. This value is traditionally the basis for selection. Therefore, designing a suitable fitness function representing the whole problem is essential. Choosing different fitness functions can yield very diverse solutions.

If the goal of the EA is to optimize for one objective, e.g. maximizing structural strength or minimizing costs, it is very straightforward to choose the fitness function. Here the desired property is what determines the fitness, and the EA will optimize for this value directly. The same can be done when optimizing for robot locomotion, where a simple fitness function measures the distance a robot moves in a particular direction. Often, a single objective is insufficient to describe the whole problem, and several objectives must be included. This is called *Multi-Objective Optimization*. In the context of robot locomotion, this can entail optimizing for both stability and distance traveled. To address the multiple objectives, one approach is to have all objectives in a vector and use the concept of Pareto optimality to make the selection [25]. If one solution is to dominate another, it has to have a higher or equal value at all the objectives. Scalarization [26] is another strategy where a weighted sum of the desired objectives is calculated and used as fitness:

$$fitness(x) = w_1 \cdot f_1(x) + w_2 \cdot f_2(x) + ... + w_k \cdot f_k(x)$$

where x is the individual, w_n is the weight of a certain objective, and f_n is the

objective. One issue here is that the weights have to be set and tuned manually based on how important the different objectives are.

The fitness landscape can often be quite deceptive, so going in a direction with gradual improvement of fitness is not necessarily in the direction of the global optima [27]. Another related problem is the rugged fitness landscape, with many peaks and valleys. This is a problem because it is very easy to be trapped in local optima with suboptimal solutions. Although Multi-Objective EAs can be useful in tackling these problems, other solutions to these problems are often based on preserving the diversity in the population and are called *Quality Diversity methods* [28]. Examples are to optimize for novelty alone [27] and MAP-Elites [29].

Selection Operators

The selection operators of EAs are *parent selection* and *survivor selection*. These operators work on the population level and are responsible for the selection pressure in the algorithm.

The parent selection is done after each individual has been evaluated and assigned a fitness value. This selection is the basis for which individuals are allowed to reproduce. There are several ways to do the parent selection, but normally it is done randomly with either uniform probabilities or fitness-proportionate probabilities. Probabilistic parent selection (fitness-proportionate selection) means that high-fitness individuals have a greater chance of reproduction than low-fitness individuals. The low-quality individuals are often given a small chance as well, and this randomness is introduced to reduce the selection pressure and introduce some noise. This can help combat premature convergence because it preserves more diversity. If only the high-quality individuals were able to reproduce, the search algorithm would be too greedy, and the chances of getting stuck in local optima would be higher. Tournament selection [30] is a different approach and works by randomly selecting (uniformly) k individuals for a tournament, where the individual with the highest fitness is chosen to reproduce.

Survivor selection chooses which of the individuals within a population survives to the next generation. This selection is done following the variation operators and is the last stage before the next generation. Like parent selection, the individuals are chosen based on their fitness value, and higher quality

individuals will have a better chance of survival. While parent selection normally has some randomness, survivor selection is often deterministic [23]. Ways of choosing survivors can be, for example, only the fittest individuals (elitism), only the offspring (generational replacement), or a combination of the two.

Variation Operators

Recombination and *mutation* are the two operators used to induce variation in the algorithm. These variation operators work on the individual level, and their role is to create new individuals with new features or new combinations of features.

Recombination is the process of creating offspring by combining the selected parents. The primary objective is to create offspring that inherit different desirable traits of both parents. There are different recombination operations based on how the individuals are represented. Normally the recombination is stochastic, where the parts of each parent are chosen and combined randomly [23]. After the parents are recombined into offspring, the children are modified through mutation. This facilitates the creation of new values and features that did not exist within the parents. Examples of mutations can be changing or swapping random values in the genome.

Termination Condition

The termination condition determines when the algorithm stops looking for new solutions. This can be when individuals with a desired fitness level have been found, a maximum allowed generations/time has been reached, the population fitness has stagnated, or the population diversity fell under a certain threshold [23].

If the global maximum is known, it might be desired to run the algorithm until a solution close enough to the global optima is found. Because of the stochastic nature of EAs, there is no guarantee that these solutions will be found, so other termination conditions should be used simultaneously to prevent the algorithm from running forever.

2.1.1 Exploration and Exploitation

Exploration and exploitation are two very important concepts of any search algorithm, including EAs. A balance between them is crucial for achieving great results [31]. Too much exploration of the search space will lead to slow or no convergence, and much time and computational power will be wasted on low-potential solutions. However, if the algorithm is too exploitative, the search will quickly lead to a local optimum and premature convergence. Especially if the search landscape is complex, the population will be very concentrated around this local optimum, and there will be only a tiny chance of escaping to better areas. Therefore, it is important to have an algorithm with a good enough balance between exploration and exploitation to avoid low-fitness local optima and to have enough exploitation to actually improve the population and find the peaks in the search space. There are a lot of variables that will impact the exploration-exploitation balance. Thus, finding the right balance can be difficult yet important.

Exploitation is typically done by the selection, while the search space is explored using variation operators like mutation and recombination [32]. An elitist selection and low mutation rate will give a lot of exploitation. In contrast, a high mutation rate and generational replacement (choosing only the offspring to survive) will lead to more exploration. The choices of selection operators and types of mutations will have a big impact. A larger mutation spread will cause bigger steps in the search space, thereby exploring more of the search space. The larger the steps, the more difficult it is for an EA to get stuck in a small area of the search space. The tradeoff here is that if the steps are too large, the algorithm might be unable to exploit the solutions found to find the higher peaks in the search space. A small mutation spread will be more exploitative, but the probability of getting stuck in local optima is greater.

When tournament selection is chosen, the tournament size is also important. A small tournament size gives low-quality individuals a greater chance of selection, while a very large tournament size has an incredibly low probability of selecting low-quality solutions. A very large tournament size will therefore be more exploitative, and the best individuals will be chosen repeatedly.

Other techniques for increasing the exploration of the search space have also been researched. These approaches are often based on maintaining the populations' diversity to avoid the whole population converging towards one

local optimum. Hutter [33] proposed a new selection scheme to increase diversity, "fitness uniform selection". The background for this was that standard selection schemes aim to increase the average fitness of a population. If the selection pressure then is too high, the EA can easily get stuck in local optima since diversity decreases too rapidly. Fitness uniform selection has the objective of finding a single individual of maximum fitness, and it is not primarily interested in converging the population to maximum fitness. It was shown that this selection scheme creates suitable selection pressure and preserves genetic diversity better than standard selection schemes.

Several other studies have also explored how diversity preservation can be used to combat premature convergence. Some of these include crowding [34], fitness sharing [35], and local mating [36], all of which are based on the concept of niching found in nature. Niching involves dividing the population into species based on different traits or properties. Age has also been used for grouping the individuals [7, 37, 38]. Methods using some sort of niching or grouping enables the individuals within the same species or group to compete primarily among themselves. This creates local competition between "similar" individuals and reduces the selection pressure between very different solutions. By doing this, a wider range of solutions is preserved, thus increasing the population diversity.

2.1.2 Effect of Initial Population

There have been some studies on the effect of the initial population in EAs, but not a lot. Normally, the initial population of an EA is composed of individuals with a genome containing a set of random values. If there is some prior information, the population can be seeded with more promising individuals [39]. This has been shown to lower the time it takes to find high-performing individuals.

Maaranen et al. [15] showed that it can be more important to start off with individuals evenly distributed throughout the search space instead of completely random ones. To do this, they used quasi-random sequences. Maaranen et al. later did more research into the effect of the initial population on single objective EAs [16]. In this study, they looked at different ways to initialize the population and compared methods that gave a more uniformly distributed initial population with random initialization. The results showed that the random initialization performed as well as the more uniform ones.

Some more promising results were shown by Rahnamayan et al. [40]. They introduced a way to initialize the population using opposition-based learning and thereby starting the evolution with a fitter population. They showed that by starting with a fitter population, high-performing solutions were found 10% quicker.

2.1.3 Neuroevolution

One application of EAs is neuroevolution, where EAs are used to construct and optimize Artificial Neural Networks (ANNs). Traditionally ANNs are trained using backpropagation. This requires gradients to be calculated based on target values from a dataset, the gradients are then used to minimize the error by adjusting the weights of the ANN. Neuroevolution does not rely on a dataset, making it an attractive option where labeled data is impossible, like robot control. Neuroevolution works well for these cases, and because it does not require a dataset, it is much more general than backpropagation. The weights are instead optimized by using random mutations, allowing movement against the gradient as well. This will allow the search to escape local minima more easily than backpropagation.

Early work on neuroevolution only searched for the weights, while the network's topology was fixed [41], but neuroevolution can also be used to find the topology automatically. The creation of the network topology is a discrete optimization task, and this is, as previously stated, something EAs performs well at. This approach has shown a lot of promise in complex reinforcement learning tasks [42].

One method within neuroevolution is NeuroEvolution of Augmented Topologies (NEAT) [19]. Mutations can here change both the weights of the connections and the network structure itself. The population initially starts with very minimal networks, but over time the networks in the population get more and more complex. In the original NEAT algorithm, both connections and nodes could be added but not removed. In later implementations, deleting nodes and connections has also been made possible [43]. The smaller the network structure is, the faster it optimizes. Because of this, there is usually a decrease in fitness when adding nodes and connections. This leads to mutated individuals having a low chance of surviving to be further optimized. The solution to this problem was to protect innovation. This was done by dividing the population into species and allowing the individuals to compete within

their own niche. The speciation preserves diversity by creating small sub-populations and gives mutated individuals more time to optimize their structure through competition within their niche. The reason behind starting with simple structures was that starting with random, more complex topologies does not lead to minimal solutions. The individuals then start off with unnecessary nodes and connections already present. The excessive nodes and connections, therefore, need to be removed to reach more minimal solutions. The result of the study showed that evolving minimal solutions are desirable because it initially reduces the number of parameters that have to be searched.

2.2 Modular Robotics

Modular robots [44] are robots built of several units/modules. A module is a physical component that isolates a specific function of the entire robot. Each module can contain sensors, actuators, and computational- and communicational capabilities. The goal of modular robotics is to connect these modules effectively to create a robot capable of accomplishing some task.

A wide range of different modules are used in the research on modular robots. The modules used can be either all the same (homogeneous) or different (heterogeneous). Homogeneous modules are the most common and have quite a lot of redundancy. This is because all of the modules need to have the same components, like sensors, batteries, and actuators. The heterogeneous modules have the advantage of being more versatile as the modules can be specialized for specific tasks.

Modular robots offer several benefits, especially when it comes to the evaluation of the robotic system in the real world [45]. Using modules that have physical prototypes in the real world allows for easy and quick assessments of the system's performance. Additionally, modular robots are highly versatile because the modules can be connected in many different ways, resulting in the creation of robots with a wide range of different properties. This versatility enables rapid exploration of different morphologies and quick configuration of the robots to adapt to changing needs.

Another quality of modular robots is robustness. If one of the modules fails, the broken module can easily be replaced to keep the system working. Because all modules in a robot are the same or at least one of a limited selection of

modules, modular robots can also be produced at a low cost. This is because of reduced production costs through batch fabrication.

The majority of work within modular robotics is done through simulation, even when there exist physical prototypes. Performing evolution with physical hardware has a higher cost because of power, communication, and other reality constraints [46]. When doing a lot of the work through simulation first, there will be some differences in the following physical experiments of the robot. This difference is called the *reality gap*, and it can be challenging to minimize this gap between simulation and physical performance.

Because there are so many possible configurations of the modules, evolutionary approaches are well-fitting methods to design modular robots (Section 2.3). EAs can also be a good tool to optimize both the morphology and control of modular robots simultaneously (Section 2.3.1).

An alternative way to optimize modular robots was done in "RoboGrammar: Graph Grammar for Terrain-Optimized Robot Design" [9]. Here Zhao et al. represented a robot's design as a graph and used a graph grammar to constrain the configuration possibilities. The grammar employed used simple and intuitive rules, allowing for the efficient creation of robots. These rules restricted the design space, ensuring that only robot designs feasible with the available components were possible. Using a set of input components, simulated robots were generated based on these robot graphs. The control input for each design was optimized using Model Predictive Control (MPC). These grammar-generated robots were then optimized with Graph Heurisic Search (GHS). GHS learns a function that maps incomplete designs to the best performance values that can be reached by expanding the incomplete designs. This is done simultaneously as it explores the design space and allows the exploration of the most promising branches of the space. Zhao et al. used this approach to optimize robots in different terrains and compared the results to a Monte Carlo tree search and a random search. The results showed that GHS performed quite a bit better in all the terrains and that it created fabricable designs.

2.2.1 Control

In order to control modular robots, a fitting control system has to be selected. One challenge with the control of modular robots is that the number of modules

in the robots can differ vastly. As the robot body is developed, modules are added and removed to optimize the morphology. Consequently, it can be difficult to optimize a controller for a robot with a changing number of inputs and outputs.

A *centralized* control system for modular robots needs to be able to deal with the varying number of actuators. One solution is to use a central control system that outputs a higher number of outputs than what is needed by the robot. Some of the outputs, therefore, will be unused unless another module is added.

An option not needing to deal with this issue is *decentralized/distributed* control systems. Here every module will have its own control system, but synchronization between modules will have to be dealt with. There are several ways to build both centralized and decentralized control systems. Centralized control for modular robots is rarely used, but its normally solved by using neural networks [47, 48], here every joint in the robot has a corresponding oscillator in the network. For decentralized control, Central Pattern Generators (CPGs) [49], open-loop wave generators [50, 51], and neural networks [52–54] all have been used.

Kvalsund et al. [54] compared the performance and morphologies of evolved modular robots with centralized and decentralized control. Both the centralized and decentralized controllers used Continuous-Time Recurrent Neural Networks (CTRNNs). The centralized approach worked by having one big CTRNN with a fixed number of inputs controlling all the modules. Because of the varying number of modules, parts of the inputs and outputs were unused. Another decentralized controller used in the study was called “the copy controller”. This controller functioned by having a list of two CTRNNs for one robot. The modules used one of these two networks as control, and mutations could make a module switch which of them is used. The results showed that the copy controller achieved the best overall fitness. This controller also worked well for a variety of morphology sizes. The centralized controller struggled the most with early convergence.

2.3 Evolutionary Robotics

Evolutionary robotics [55] is a biologically inspired approach to designing adaptive, autonomous robots. Evolutionary approaches, such as EAs, allow for the automatic optimization of both the parameters and architecture of robots' control systems without the need for human interaction. A population of robots is evolved in a simulated environment, and if the environment accurately models real-world conditions, robots can be optimized to work in the real world as well. EAs can also optimize the body plan of the robots by either optimizing an existing design or creating new designs from scratch. Optimization of robot morphologies can work by changing the properties of an existing robot, like the length of different body parts, or by adding/removing parts, as can be the case in modular robotics (Section 2.2).

By utilizing evolutionary robotics, non-traditional robot solutions or solutions that are not intuitive for humans might emerge. Instead of trying to mimic animals, the evolutionary robots are designed through trial and error. These solutions might be even better suited for the task at hand than what humans would design [56].

2.3.1 Co-Optimization of Morphology and Control

Since Karl Sims published the study "Evolving Virtual Creatures" in 1994 [2], many researchers have been inspired to simultaneously evolve both the body (morphology) and the brain (control) of virtual creatures. This has also led to the usage of virtual creatures within the field of evolutionary robotics to simulate and optimize the morphology and control of robots. Despite the huge increase in computational power since 1994, there has been less progress in the field than expected [57].

Evolving the morphology and the control simultaneously has been a large challenge within the field of evolutionary robotics [11]. The search space is often very large and complex, and much exploration is required to find good solutions. The body and brain of a robot also create a co-dependent system, where changes to the body plan often require a corresponding change in the control system to be able to utilize the modified body. E.g. if a limb grows, the old control system will likely be sub-optimal, and the individual's fitness will decrease with the mutation. Because of this, the morphology tends to converge early to

local optima in the search space. These designs can be suboptimal, but new morphological mutations are detrimental because the control already has had time to adapt to the body. This leads to the removal of newly morphologically mutated individuals from the population, even though they might have a lot more potential.

Much research has been done into how to avoid the premature morphological convergence, including protection (Section 2.3.2) and diversity preservation (Section 2.3.3). Another strategy that is shown to improve is a two-stage approach [58]. This approach involves co-optimizing morphologies and controllers in the first phase before locking the morphologies in the second phase to re-evolve the controllers. The results showed a $10 - 15\%$ improvement by re-evolving the controllers after convergence. This simulates the kind of lifetime learning we have in nature, and the individuals had a better chance of learning a good gait for their evolved body plan.

An alternative to using evolution to co-optimize morphology and control is reinforcement learning, as used by Schaff et al. [59] and David Ha [8]. In [59], they optimized the control through reinforcement learning to maximize the expected reward over the design distribution. During the training, they shifted the design distribution towards better-performing designs. Ha [8] showed that optimizing the agent's design at the same time as the control policy leads to solutions better suited for the task compared to optimizing only the control. The solutions found reached both a higher cumulative reward and also improved a lot faster than the sole control optimization.

2.3.2 Protection

Cheney et al. [7] introduced a technique for protecting morphological innovation when evolving voxel bots. The purpose of this morphological protection was to give the control system time to adapt to newly mutated morphologies. This means that after a mutation, the mutated lineage is protected, allowing the control system to undergo several generations of evolutionary change. They did this by tracking the time elapsed since the last morphological mutation for each individual, and used these times to compare them during the selection process. The selection was based on Pareto-optimality of age and fitness, leading to a multi-objective optimization of low age and high fitness. The results were very promising and showed a significant later convergence in morphology and fitness.

Age has also been used as an optimization parameter in research outside of robotics to avoid premature convergence. E.g. the Age-Layered Population Structure (ALPS) algorithm [37], where individuals are segregated into different layers based on their age. ALPS increases the age for all individuals every generation but also inserts new randomly generated individuals to avoid convergence. The results showed that ALPS helped avoid convergence to mediocre local optima, and thereby ended up with fitness values twice as good as their baseline EAs. ALPS inspired Schmidt and Lipson [38] to create a different approach, Age-Fitness Pareto Optimization (AFPO). Instead of dividing the population into layers of ages, they used a multi-objective optimization for both fitness and age. Similarly to ALPS, AFPO increases the age of all individuals and inserts randomly generated individuals every generation. AFPO performed even better than ALPS, resulting in both higher fitness and less computational effort.

A different protection method was introduced by De Carlo et al. [12]. Their method was inspired by the speciation from NEAT and used to evolve modular robots. The individuals in the population were divided into species based on how similar their morphologies were, and individuals only competed against other individuals in the same species. The similarities of individuals were measured by a weighted difference between a set of traits. If an individual was different enough from the other individuals, it was placed in its own species and protected for a few generations by artificially increasing the members of that species. Their results showed that speciation prevented loss of diversity compared to a standard EA and thereby avoided convergence towards one dominating morphology. The speciation led to a lot more exploration, and the solutions found were more interesting and different. The increase in exploration, in turn, led to a decrease in exploitation, but the maximum fitnesses were similar for both with and without speciation.

2.3.3 Diversity

Another way of dealing with the problem of premature morphological convergence can be to keep a higher diversity of individuals within the population (Section 2.1.1). This can give different solutions more time to adapt and improve before being completely removed from the population. The speciation method described in the previous section by De Carlo et al. [12] is one method that is shown to work well for modular robots.

It is a challenge to discover a wide diversity of high-fitness individuals within one single evolutionary run when the morphology tends to converge early. Too much exploration also leads to little exploitation, and the population will end up with a diverse group of low-fitness individuals. Lehman and Stanley [4] provided the evolution of virtual creatures with a multi-objective approach measuring both novelty and performance. They compared optimizing for fitness alone, novelty alone, both fitness and novelty with global competition, and both fitness and novelty with local competition. The results showed that the local competition discovered more diverse creatures in a single run than the global competition, while the global competition achieved higher fitness values. Novelty alone achieved the highest diversity of all but also the lowest fitness, while fitness alone achieved the second highest fitness and the lowest diversity.

Research in how different encodings (Section 2.3.5) affect the diversity when co-optimizing morphology and control has also been conducted [60]. Here they compared Direct Encoding, L-System, CPPN, and Cellular Encoding when evolving virtual creatures. They found that diversification varied across evolutionary runs for all of the encodings used. Direct Encoding tended to explore more local areas in the search space, while the indirect approaches made bigger jumps. The results also showed that the L-System and the direct encoding performed the best.

A different approach that is shown to preserve the diversity of morphology in modular robotics is MAP-Elites [13]. The MAP-Elites algorithm keeps an archive of solutions and is structured with cells representing specific combinations of feature descriptors. The algorithm does not have multiple objectives, but diversity is ensured through the archive. The results of [13] showed that MAP-Elites produced more morphologically diverse robots with higher performance than the two approaches they compared with. These approaches were based on objective-only EA and diversity-augmented multi-objective EA.

Co-optimizing robots' morphology and control in changing environments has also been shown to preserve diversity [61]. The study showed that when evolving a population of robots, higher diversity can be seen in populations that evolved in a scenario where a population of environments also evolved.

2.3.4 Impact of Environment

The environments in which robots evolve have a significant impact on both their control and morphology [48]. This is because the population gradually adapts to the environment throughout the course of evolution. Auerbach and Bongard [6] demonstrated that virtual creatures that evolved in more complex environments exhibited higher complexity than those that evolved in less complex environments.

Gradually changing an environment during the environmental run is also shown to lead to very different evolved robots compared to those evolved in just one of these environments [14]. This is referred to as a "genetic memory" of properties that evolved in the early stages. In the same study, Miras and Eiben demonstrated that gradually increasing the difficulty of the environment yielded better performance in the difficult environment compared to directly evolving with a static, difficult environment. In a later paper, Miras and Eiben [62] showed that starting in a difficult environment and gradually decreasing the difficulty also had a long-lasting effect on the evolved robots. Traits found in the population that evolved in just the target environment were missing from the population that evolved in decreasing difficult environments.

RoboGrammar [9] optimized robots in different terrains and clearly showed the differences in both morphology and movement strategies. They showed that their GHS together with MPC resulted in individuals well-adapted to their respective environments. A wide range of different robot designs were also found for each environment.

2.3.5 Encodings

When the morphology of a robot is to be changed through evolution, a representation of the body plan has to be made, the genotype. The choice of genotype representation is an essential part of any EA and varies vastly from problem to problem. Before an individual is evaluated, the information encoded within the genotype has to be mapped to a phenotype. This information can be encoded in different ways, and the two main approaches are *direct* and *generative* (indirect) encodings.

Direct encoding means having a complete one-to-one mapping of the features in the genotype. Every gene will therefore have a corresponding feature in

the phenotype. For modular robotics, every module in the robotic body is therefore expressed in the genotype and is used in research like [13, 51, 63]. Implementing direct encodings is very simple, and it is less computationally intensive compared to an indirect approach, as there is no need for complex developmental or generative processes.

Generative encodings offer advantages over direct encodings, like a smaller, less complex genome. This is because not every part of the phenotype has to be represented in the genotype. Parts of the genes will be reused, allowing for quicker evolution of decent morphology and control. Another advantage is that the generative nature allows for recursive structures, which are unlikely to occur with direct encodings. Examples of generative encodings that are used for virtual creatures / modular robots are rewriting systems like L-systems [3, 12, 14] and generative neural networks like CPPN [5].

Chapter 3

Implementation

This chapter provides an overview of the methods used and how they are implemented. The opening section, Section 3.1, details the tools utilized and how these tools are used in developing the system for the experiments. Section 3.2 focuses on the design and workings of the modules before Section 3.3 describes the control system.

Next, Section 3.4, describes how the modules are connected to form robots and how these robots are represented as genotypes. Section 3.5 then explains how the Morphological Evolution of Augmenting Topologies (MEAT) algorithm works before Section 3.6 explains the evolution process and the EAs used for the experiments.

Finally, Section 3.7 explains the methods used for analyzing the results before the chapter wraps up with Section 3.8 describing the experiment setup and the parameters used.

3.1 Tools and System Overview

The code used can be found on GitHub[1]. For this project, all of the simulations and evaluations are performed with the Unity game engine. The physics engine within Unity is Nvidia PhysX, which is also designed to be used for games. In order to use Unity for simulation and machine learning, the Unity Machine Learning Agents Toolkit (ML-Agents) [64] is used. ML-Agents is an open-source project for training intelligent agents for use in games and AI research.

[1] https://github.com/tobiaspaulsen/modular-robots

It provides a lot of implemented state-of-the-art algorithms, but agents can also be trained using custom ones. The toolkit also provides a Python API to train and control the agents, which is used for this thesis. The Python package *mlagents* allows Python to interact directly with the Unity environment in either an executable or the Unity editor. This is done by controlling the simulation loop and sending actions and observations back and forth between the Unity executable and the Python script. The EAs are implemented using the *Distributed Evolutionary Algorithms in Python* (DEAP) framework [65], and the software versions used in all experiments are listed in Table 3.1.

The training environment created in Unity is configured to have a fixed delta time of 0.01 second. This delta time is the interval between physics updates. By having a very low delta time, the physics is more accurate, but simulations also take longer to complete. The decision period for the agents is set to five, so for every fifth physics update, the agents will ask for a new action. This equates to 20 actions per second.

Python connects to the Unity executable or editor by creating an instance of the class *UnityEnvironment* from the *mlagents* package. Through this class, the simulation can be controlled, but it is not possible to send info for the creation of robots. To do this, a side channel has to be used. By using the provided *SideChannel*-class, messages can be sent between Python and Unity before the simulation begins. When an individual is to be evaluated, a JSON representation is sent from Python to Unity through this side channel. This file contains information about every module, the type, angle, parent, connection site, and GUID. This file is then decoded by doing a Breadth-First Search on the received graph. The described modules are then created and connected together as an agent in the environment. If there are any collisions between modules during the creation, the colliding modules are removed. After the robot is created, a message is sent back through the side channel to the python-side with the GUIDs of the successfully built modules. When this message is received, the simulation loop can begin.

Within the simulation loop, actions are calculated based on the delta time and the controller. One action per module is sent every 0.05 seconds of simulation time. During the simulation, the robot's position is sent back to Python, where it is used to track the movement and calculate the fitness value. After the chosen length of simulation time, the individual object in Python is assigned a fitness value, and the environment is reset to prepare for the subsequent evaluation.

Name	Version
Unity	2020.3.26f1
Unity ML-Agents	2.0.1
Python mlagents	0.27.0
DEAP	1.3.3

Table 3.1: Software versions used.

3.2 Modules

The modules designed for this thesis are heterogeneous, and the collection of modules consists of one root, four body modules, and four limb modules. The designs of the modules are inspired by the parts used in RoboGrammar [9]. However, there are some significant differences. In RoboGrammar, the links, connectors, joints, and wheels all are separate parts that are assembled together to form the robot. For this thesis, every module, except the root, consists of both a link and a joint, thus simplifying and limiting the different ways things can be connected. The comparison of the different modules can be seen in Figure 3.1. There are four different types, *x-rotating* (blue), *y-rotating* (red), *z-rotating* (green), and *static* (yellow).

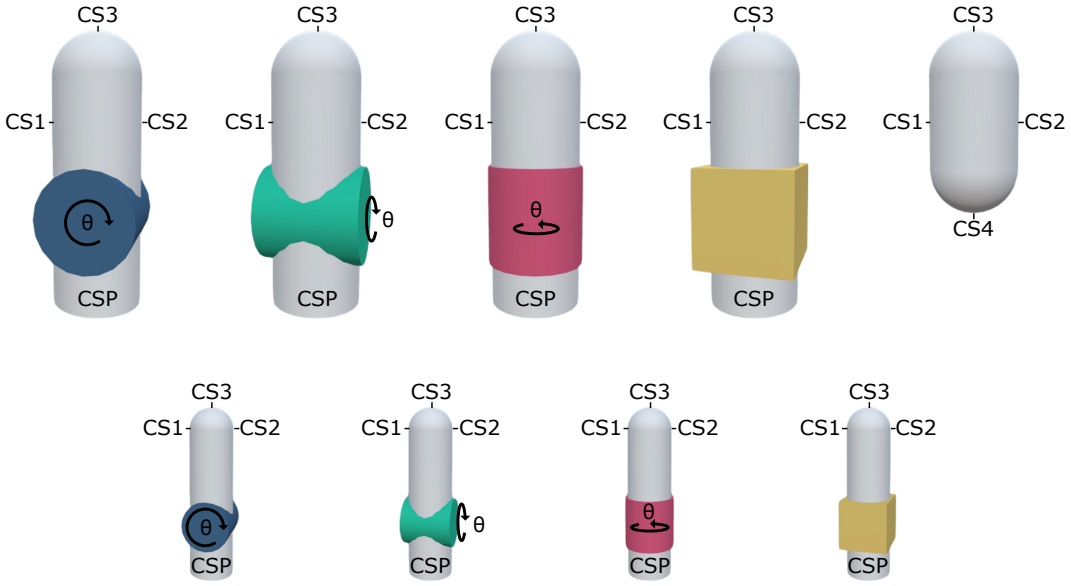


Figure 3.1: Connection sites and rotations of the different modules. The body modules are in the top row, with the root furthest to the right. In the bottom row, the limb modules are displayed. The parent connection site (CSP) is located at the bottom of all modules except the root, while the other connection sites (CS1 - CS4) are for children.

The modules take an input between -1 and 1, where -1 gives a target angle of $-90°$ and 1 gives $90°$. For the modules with static joints, the input has no effect.

All body modules have the same size. The length from the joint's center to the module's tip after scaling is 1 meter. These modules are always connected to either the root module or a different body joint. When connected, the lower part of the module will overlap with the parent to make the length between the joints 1 m. The widths of the modules are 0.5 m, but all modules are a bit wider where the actual joint is. For the limb modules, the length from the center of the joint to the tip is 0.65 m, and the width is 0.25 m.

To make the motors easier to tune, the masses and torques of both the body- and limb modules are set to the same values. The link part has a mass of 0.5 kg, and the joint part (the colored part) has a mass of 0.1 kg. The overlapping connection part has no mass, as it is there for aesthetic purposes. The joints have allowed angles between $-90°$ and $90°$ and are driven by a Unity Angular Drive. The specs of the Angular Drive were set to a spring value of 120, a damper equal to 25, and a maximum force of 40 N. These values were all tuned manually, with the goal of finding values that yield realistic physics. If the force and spring values are set too high, the system is prone to physics bugs and does not look realistic. However, if the values are too low, the robots cannot lift their own weight.

Every module, including the root, has four connection sites, as shown in Figure 3.1. The root has four children connection sites, while the other modules have three children- and one parent connection site. A child module is connected to its parent by connecting its parent site with one of the parent's child sites. For the body modules and the root, two of the child connection sites are on opposite sides in the middle between the tip and joint. For the limb modules, these two connection sites are almost at the tip. This is because when another limb module is connected to one of those two sites, there will be a smooth $90°$ between the modules.

3.3 Controller

The robots are controlled by a decentralized system consisting of several Coupled Oscillators (COs). Each module contains its own oscillator, and

the phases are coupled with an offset to its parent. This control system is inspired by the one used in Crespi and Ijspeert's paper [66]. The snake robot in this paper was controlled by a CPG modeled as a chain of coupled nonlinear oscillators. The CPG was designed to produce waves traveling through the body. As the robots in this thesis also can have limbs, the waves will travel through both the main body and the limbs. These traveling waves are also inspired by the metachronal rhythm exhibited in all legged invertebrates [67]. These rhythms resemble traveling waves easily observed in many-legged myriapods such as centipedes and millipedes, where the wave travels through the body and legs from the rear to the front.

The CO is very simple and takes no inputs, so the oscillations will be consistent no matter what. Therefore, it has no way of adjusting to a changing environment. The focus of all the experiments is the evolution itself, so the controller is kept very simple.

The phases of the COs are coupled with their parents' oscillators to help synchronization between the different joints. This coupling gives an advantage over a non-coupled oscillator because the children will still have the same offset if the parent oscillator's offset is mutated. Without this coupling, the synchronization between the parent and its children will have changed. This makes the control system more robust compared to a simple sine wave controller.

The output of one oscillator is given as follows:

$$y(t) = A \cdot \sin(\omega \cdot t + \varphi) + D \tag{3.1}$$

$$\varphi = \varphi_{parent} + \varphi_{offset} \tag{3.2}$$

where A is the amplitude, ω is the angular frequency, φ is the phase, and D is the joint angle offset. In Equation 3.2, the phase is calculated based on the parent's phase and the phase offset. If a module doesn't have a parent, the phase is equal to the phase offset.

Table 3.2 lists the allowed values for the different variables. The frequency is fixed to 4 radians per second (rad/s), which equates to 0.637 Hz. This is done to enable easier synchronization between the different modules. A higher frequency also correlates with faster movement, so with a fixed frequency, other strategies have to be used to speed up the locomotion. The fixed frequency

will also limit the search space and make finding optima easier. Another thing worth noting with the allowable parameters is that the amplitude is allowed to be zero, meaning that the joint will behave like the static joint even if the type is different. When a robot is initialized, the amplitude is set to at least 0.5 to always start off with some movement. When a new module is added to a robot through mutation, the control parameters of the parent are inherited by the new controller.

Variable	Allowed values
A	$0.0 \rightarrow 2.0$
ω	4 rad/s
φ_{offset}	$-\pi \rightarrow \pi$
D	$-1.0 \rightarrow 1.0$

Table 3.2: Allowable parameters for the controller (Equation 3.1 and 3.2)

The modules' joint values are clamped between -1 and 1, and this translates to target angles between $-90°$ and $90°$, so the oscillators' outputs have to match this. An amplitude bigger than 1 will lead to a clipped output, while a smaller than 1 amplitude reduces the angle span. The offset will also impact the clipping of the sinusoid, and together with the amplitude, it helps decide which angles a joint will utilize. In Figure 3.2, a few randomly generated COs's outputs are displayed to show how the module angles change over time.

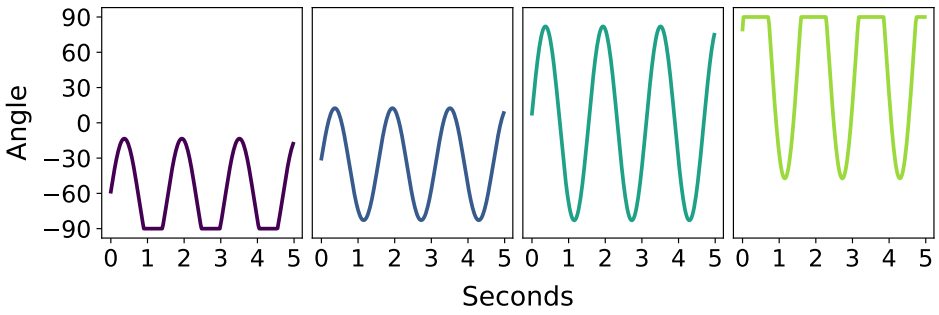


Figure 3.2: Controllers with random valid parameters and scaled outputs to show which angles will be utilized.

3.3.1 Mutation

Because the frequency of the controllers is fixed, only the amplitude, phase offset, and joint angle offset are mutated. For every module in the robot, there

is a p_c chance of mutation for every one of the parameters. These mutations are done using a Gaussian distribution random number generator and require a σ_c for the spread of the Gaussian curve. The size of σ_c will decide the size of the mutation; a small σ_c will give more gradual changes, and a large one will lead to more random values.

The spans of allowed values differ for the different parameters, so every mutation is normalized between -1 and 1. This is done to keep similar-sized mutations for all parameters. For amplitude and joint angle, the value after the mutation is clipped to be within the allowable spans. The sine wave is periodic, and therefore the phase offset is normalized between $-\pi$ and π.

3.4 Robot

A big source of inspiration for the robots is animals, not only in the designs of the control system and modules but also in how the modules are connected to form robots. The aim is to create robots exhibiting similar properties as living organisms. One of these properties is bilateral symmetry, which is explicitly enforced for the robots. The reason is that bilateral symmetry is a trait present in more than 99% of animals [68] and desirable for robots with targeted locomotion. A point of interest is to see whether the robots utilize the advantage of bilateral symmetry and learn to walk using the limbs, similar to animals, or if a more unnatural gait is found.

A robot consists of between one and thirty modules that are connected to each other at their connection sites, as described in Section 3.2. The genotype of the robots is a directed tree with *direct encoding*, so every node in the tree has a corresponding module. In Figure 3.3, a comparison between the genotype and phenotype can be seen. Every limb module also has a reference to the corresponding limb module on the opposite side of the body. This is because every change to the robot's morphology has to be mirrored on both sides to keep the body bilaterally symmetrical.

3.4.1 Rules

To limit the morphological search space and to ensure that the robots exhibit bilateral symmetry, a set of rules is used during initialization and mutation. The robots created will have a central body composed of body modules, and

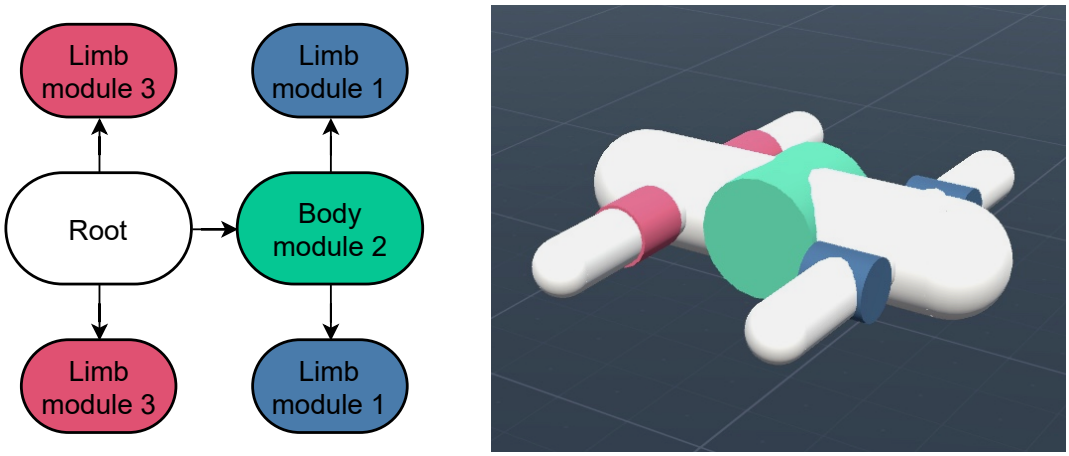


Figure 3.3: Robot genotype to the left and phenotype to the right.

the number of limbs will be either zero or an even number. These limbs are made up of limb modules, and branching in the limbs is not allowed. This is primarily to stay true to the concept of animal-inspired robots but also to reduce the likelihood of colliding modules.

The rules used are:

- Body modules can only have a parent that is the root or another body module. They can connect to connection site 3 of the body modules and site 3 or 4 of the root.
- Limb modules can be connected to body modules or the root at connection sites 1 and 2, and to all of the connection sites of other limb modules.
- Root modules can have min 0 and max 4 children.
- Body modules can have min 0 and max 3 children.
- Limb modules can have max 1 child.
- Only the limb modules can be connected at several different angles. The allowed angles are 0°, 90°, 180°, 270°.

3.4.2 Initialization

In this work, two initialization techniques were explored; simple and random. When a module is created, all the parameters of the controller are chosen

randomly from the allowed ranges (Table 3.2), except the amplitude that is chosen between 0.5 and 2.0.

Simple Initialization

The simple initialization creates minimal individuals consisting of the root, one body module, and no limbs. The type of body module is selected randomly, so there are four possible configurations in total. These configurations are pictured in Figure 3.4.

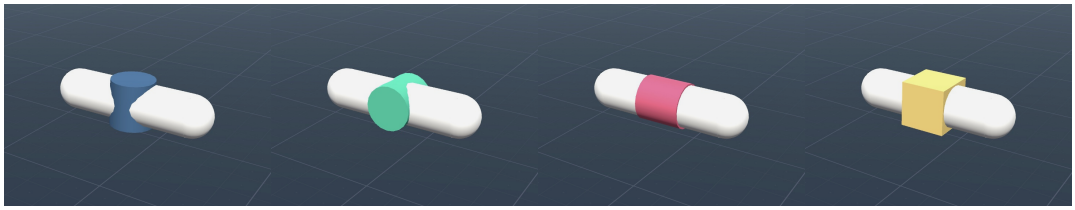


Figure 3.4: All possible simple initialized robots.

Random Initialization

The random initialization creates individuals with a varying number of modules. First, the number of modules, n, is chosen as a uniform random number between four and the maximum allowed number of modules. For all the experiments, the max was set to 30. The root is then created, and the remaining $n-1$ modules are added in accordance with the rules (Section 3.4.1). The probability of adding a body/limb module is normalized to the proportion of connections that allow for a connecting module of that type. This means that if there are more connection sites for body modules available than there are for limb modules, it is more likely to add a body module. This approach makes it less likely to get extremely long individuals and makes it more likely for existing limbs to grow than to add an entirely new set of limbs. Since the robot must be symmetrical, two limb modules are always added each time, counting as 2 of the remaining $n-1$ modules. This is balanced by only considering the connections on one side of the body when normalizing.

As shown in Figure 3.5, the complexity of the randomly generated robots varies a lot. Many of them also have parts that will be challenging to utilize when moving, like having limbs pointing straight up.

The advantage of random initialization is that there is a lot of diversity within the initial population (see Experiment 1, Chapter 4). The more of

the search space represented in the initial population, the better. Having a lot of different morphologies present early in the search will help avoid low-quality local optima and instead focus on the more high potential parts of the search space. Spending time adding several modules to make an individual capable of movement is also time-consuming, as is necessary for the simple initialization. There might therefore be a trade-off between both approaches either in removing excessive modules vs adding new modules. If the time spent removing modules is too large, it might be better to slowly complexify like simple initialization.

3.4.3 Mutation

For this thesis, crossover is not used, only mutation. There are three different types of morphological mutations, *add*, *remove*, and *swap*. There is a combined morphological mutation rate, p_m, and when a morphological mutation occurs, one of the three mutation types is selected randomly. If one of these mutations fails (e.g. if add is chosen and the maximum number of modules is reached), one of the two remaining mutations is chosen instead. One of the three will always succeed. This means that for every morphological mutation, a module is either added, removed, or swapped. For all three mutations, when selecting a module, only the body modules and the limbs on one side are considered. Every mutation, be it add, remove, or swap, is mirrored on both sides to keep with the rules described in Section 3.4.1.

The add-mutation works the same way as adding modules in the random initialization. The chance of adding a body- or limb module is also normalized in the same way as described earlier. The parent, connection site, and module type are all selected randomly. For the limb modules, the angle is also selected randomly. To facilitate a larger mutation, there is a chance of calling another add-mutation when a module is added. This is also done to balance the size of the remove-mutation. A theory is that the larger mutation might help to escape local optima. For all experiments, this probability is set to 0.5, and the maximum number of added modules per add-mutation is set to six. When a module is added, it inherits the parent's controller. The thought behind this is that the parent's controller already has evolved some useful properties and that this will be better than starting off with a completely random controller.

The remove-mutation works by selecting one module randomly and removing that module, thereby removing all the descendant modules connected to it.

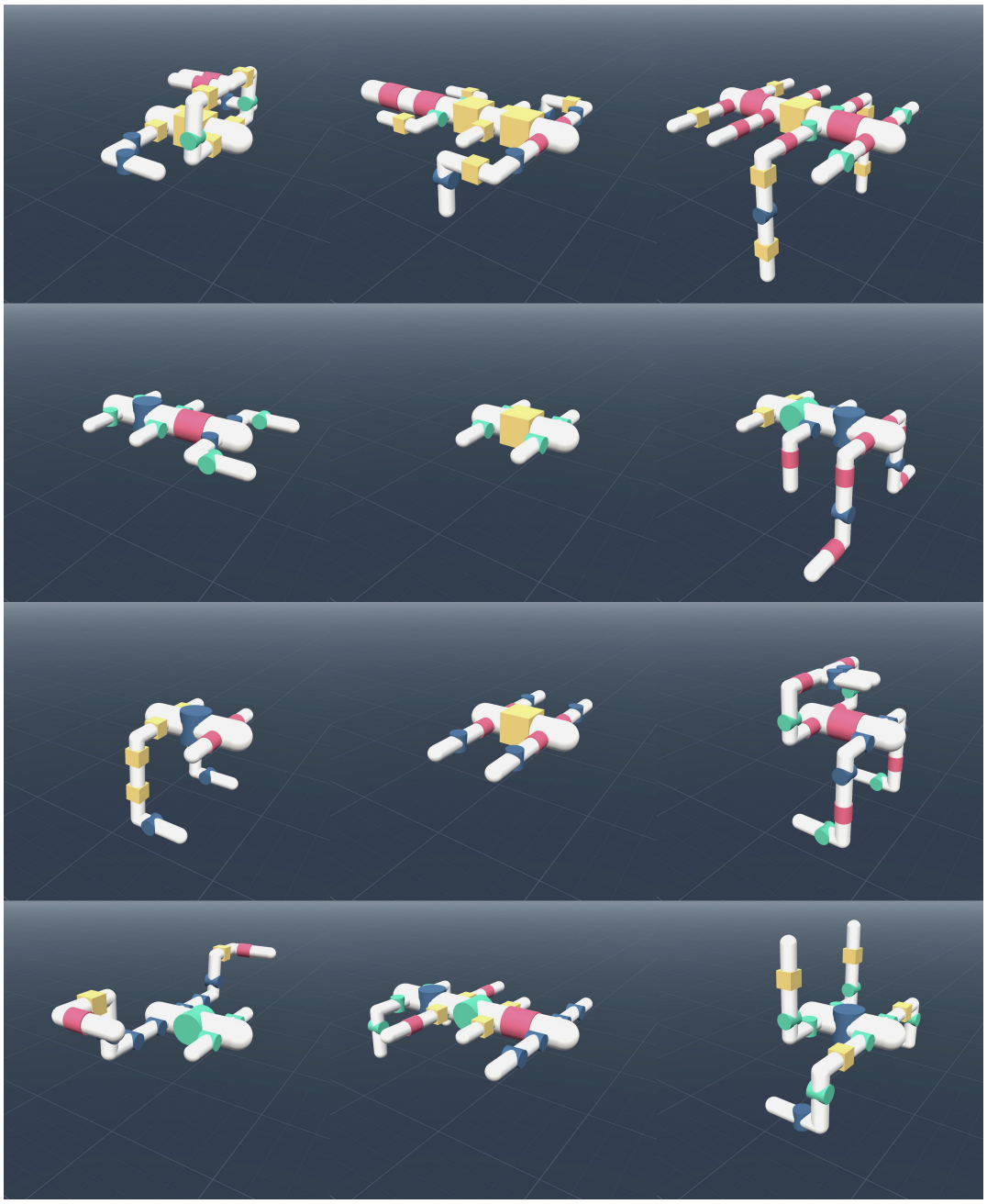


Figure 3.5: A selection of randomly initialized robots. Here every joint is in a neutral position of 0°, and gravity is turned off.

Again, if a limb module is removed, the symmetrical limb module is also removed. Finally, the swap-mutation works by selecting one module randomly and swapping the type of that module without changing the controller. For limb modules, the swap-mutation also selects a random angle, and if the parent is a limb module, the connection site can also change.

3.5 MEAT

The two key aspects of MEAT are simple initialization and morphological protection. Similarly to NEAT [19], MEAT utilizes the concept of an initial population consisting of very minimal solutions. This means starting off with robots initialized with simple initialization (Section 3.4.2). This minimal initialization allows the robot morphology to grow slowly to a more complex structure throughout the evolution process. This means that there are no unnecessary parts in the initial individuals, and there is no need to waste time optimizing controllers in modules that are later removed. Starting with simple individuals means there are far fewer controllers to optimize in the early generations, which might help lay the groundwork for when the complexity has increased. Every module in the final evolved robot is also added with a purpose, so the hope is to find more effective solutions with fewer unnecessary modules. Randomly initializing the robots will increase the chance of having unnecessary modules, as they can be initialized with many already present. Non-hindering modules will have no reason to be removed, and optimization time has to be spent to remove the non-working or hindering parts.

3.5.1 Morphological Protection

One of the key aspects of MEAT is the morphological protection method. The thought is that it will be necessary to use some protection to avoid premature convergence due to the problem described in Section 2.3.1. If the controller has had sufficient time to adapt to the morphology, a mutation to the body plan will likely decrease fitness, reducing the chance of survival. However, by allowing newly mutated individuals to only compete against other newly mutated individuals, they will be given more time to optimize their controllers to the new body plans. This hopefully will have the effect of helping escape local optima, thereby limiting premature convergence.

The protection method used is based on Morphological Innovation Protection [7] and AFPO [38]. In both these studies, there is a multi-objective optimization based on fitness and age. For an individual to dominate another individual, it has to be a Pareto dominance on both age and fitness, meaning both a higher fitness and a lower or equal age. This lets individuals compete with others of the same age or younger, protecting them from the older, more optimized solutions. Due to this, the morphological mutations are no longer detrimental

to the survival chances in later generations.

In AFPO, the age measures how long the oldest part of a genotype has existed in the population, and it is also increased after mutations and crossover. This is done to protect new random individuals that are inserted into the population during evolution. For Morphological Innovation Protection, a *morphological age* is used instead, which is a measurement of how many generations since the last morphological change. The same age measurement is used for MEAT, representing how many generations a controller has had to optimize to the current body. When the body plan changes through a morphological mutation, the age is reset, but not if the mutation does not lead to a change in the phenotype.

NEAT also utilizes a form of protection to avoid premature convergence, but that method is based on speciation, while MEAT is based on morphological age.

Pareto Tournament Selection

When there is both parent and survivor selection, the survivor selection usually is more deterministic and selects the n best individuals (Section 2.1). Cheney et al. used this deterministic survivor selection for both their protection and baseline, despite the lack of parent selection. The protection worked by selecting the n best individuals based on Pareto optimality. Because this is a very elitist approach, the exploitation is very high, and when testing their approach, the population quickly converged toward the elites, especially for the baseline without protection.

To lower the exploitation, the Pareto tournament selection used in ALPS was instead chosen, despite the different age measures. This selection performed better than the elitist method used by Cheney, as it introduced more noise, slowing down the convergence and exploring more of the search space. This selection removes individuals from a too-large population until the population reaches a size of n instead of adding to the next. By doing this, no duplicates will be selected, and more diversity will be preserved. Tournaments of size k are repeatedly selected, and dominated individuals from this tournament are removed from the population. This also guarantees the survival of the fittest individual.

Another reason for using the tournament-remove selection instead of

tournament-add is the issue of balance between selection pressure and tournament size when using morphological protection. A smaller tournament size leads to more noise and randomness in the system, thereby avoiding local optima. When using protection and a small tournament size, there is a high chance that no individual in the tournament will be dominated. This is especially the case in later generations when the individuals in the population exhibit a lot of different ages. When using tournament-add, this can be a problem as it can lead to selecting individuals almost randomly, leading to very low selection pressure. Increasing the tournament size increases the selection pressure but will also lead to more duplicates in the next generation. When using tournament-remove instead, the tournament size can be kept low because when no individuals are dominated, no individuals will be removed, and the next tournament is selected. This will take longer since more tournaments must be conducted to remove a sufficient number of individuals, but eventually, enough individuals will be removed. If no individuals in the population are dominated, everyone will survive, but this did not happen in any experiments.

$k \leftarrow$ tournament size
$n \leftarrow$ desired population size
$P \leftarrow$ current population where $lenght(P) > n$

while *length(P)* $> n$ **do**
  $tournament \leftarrow k$ random individuals from P
  $front \leftarrow$ age-fitness Pareto front of $tournament$
  for $ind \in tournament$ **do**
    if $ind \notin front$ **then**
      remove ind from P
    end
  end
end

Algorithm 1: Tournament-remove selection based on age and fitness. The requirement is that the population starts with more individuals than desired.

3.6 Evolution

3.6.1 Simulation

During all the experiments, the individuals are simulated for the same length of time, 15 seconds. This simulation time can, however, be interrupted early not to waste time on poor individuals. This happens either if the robot moves

backward or when it has not moved forward during the first two seconds. The evaluation will also be interrupted if the robot turns beyond ±90°, as it then moves in the opposite direction.

$$fitness = x_{max} - x_{start} \tag{3.3}$$

The goal is for the robot to travel the furthest straight forward during the simulation. The fitness function (Equation 3.3), therefore, is chosen to be the distance traveled. This is measured by finding the difference between the starting and maximum x-position for the root. If the robot turns around and the simulation is interrupted, the fitness value received will be the maximum achieved fitness until that point. This is because many individuals have a promising gait but tend to turn slightly towards one of the sides. An individual that moves fast can get a very low fitness score because it has turned around entirely and moved closer to the starting point. A minor correction to a gait like this can lead to a very high-performing individual, so keeping it in the population can be advantageous.

The two environments used in the experiments are *flat* and *stairs*. The flat environment is just a large flat plane, while the stairs have a flat plane with stairs starting 3 m away from the robots' spawning point. The steps of the stairs are 0.25 m high and 0.5 m deep, so the stairs are not very steep (Figure 3.6).

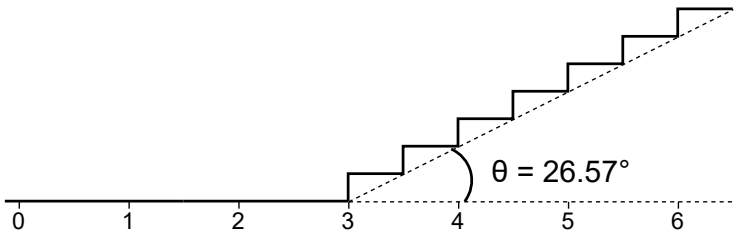


Figure 3.6: Cross section of the stairs with 0 being the starting point for the robots.

3.6.2 Evolutionary Algorithms

As mentioned earlier, the two key aspects of MEAT are simple initialization and morphological protection. The three other EAs are created to investigate different aspects of MEAT. *Simple Initialization No Protection (SNP)* is used to isolate the effects of the simple initialization alone, while *Random Initialization*

No Protection (RNP) and *Random Initialization Protection (RP)* serve as baselines.

All four EAs used in the experiments are based on the approach used by Cheney et al. [7]. The basic scheme is as follows: (1) n individuals are initialized and evaluated, (2) every individual reproduces asexually once, (3) either the controls or the morphology of every child is mutated, (4) all children are evaluated, (5) $(\mu + \lambda)$ tournament-remove survivor selection. After step 5, the next generation starts at step 2.

The selection scheme used is $(\mu + \lambda)$, so both parents and offspring can be a part of the next generation in contrast to generational replacement. This approach can be very elitist, and the diversity of the population can decrease quickly. The parent selection is therefore skipped to keep more of the diversity in the population. This means that every individual is allowed to create exactly one offspring. With parent selection, high-fitness individuals can be selected and allowed to reproduce several times, while low-fitness individuals will have a meager chance of reproduction. By skipping parent selection, low-fitness individuals can survive and have a higher chance of fulfilling their potential because of the decreased selection pressure on the elites. The selection operator is the tournament-remove selection introduced in Section 3.5.1, with a tournament size of $k = 2$. Because SNP and RNP do not use any protection, only fitness is considered instead of using Pareto dominance.

When the children are mutated, either their control or the morphology is mutated. There is an equal probability for both mutation types, and the reason for this is that Cheney et al. [7] tested different ratios, and equal chances showed the best results. A difference from the method used by Cheney et al. is that they only mutated half the offspring, while for this thesis, every child is mutated. By not mutating every child, there will be children which are duplicates of their parents, lowering the population diversity and increasing exploitation. After testing lower mutation rates, the conclusion was that mutating every individual led to better performance for all methods, especially when not employing protection. When a morphological mutation occurs, there will either be an add-, remove-, or swap-mutation as described in Section 3.4.3. For the controller mutation (Section 3.3.1), there is a probability of $p_c = 0.33$ for each controller parameter for every module in the robot. This probability was chosen so every module will have one parameter changed on average. For all experiments, $\sigma_c = 0.2$.

3.6.3 Genome Cleanup

As described in Section 3.1, information about the robot morphology is sent as a JSON file through a side channel to the Unity executable. After the modules are built, Python receives a list of GUIDs for the successfully created modules. If the length of this list differs from the number of modules in the genome, the genome is cleaned up to contain only modules that can be built in Unity. This is done to prevent a mismatch between genotype and phenotype. The genomes are also cleaned up after the population is initialized.

When removing the colliding modules from the genome, the morphological age will be set back to the age before the latest morphological mutation if all added modules have collided. This is because if there is a collision, it will be because of the latest morphological mutation. It is undesired that the individual has the advantage of having both an unchanged morphology *and* a morphological age of zero when using protection. It is crucial that the morphological age represents the actual age of the morphology and that there is no advantage to adding colliding modules.

There are several reasons why genome cleanup might be advantageous. One such situation is when an individual has reached the maximum allowed number of modules, but several of them collide. These collisions can hinder how future mutations happen and prevent the addition of new modules. By removing the colliding modules, the number of expressed modules and the number of modules in the genome will be the same, and an add-mutation is again possible.

Another reason for using genome cleanup is that individuals with a lot of bloating in their genome will have an advantage in the selection based on age. A colliding module might be added, removed, or swapped when this individual is mutated. New modules might also get a colliding module as a parent, and over time the bloating increases. These mutations lead to a reset of the morphological age without actually changing the expressed morphology. As a result, the individual can have several generations to adjust to its morphology and still have an age close to zero. This can also remove younger and better prospects from the population because they have not had the same amount of generations to adjust to their morphology.

3.7 Analysis Methods

3.7.1 Morphological Features

To simplify an individual's body plan, the robot graph is mapped to points in $\mathbb{R}^3$. Here, each of the three axes corresponds to a feature of morphology. This is done to make it easier to compare different configurations, thereby avoiding using the whole graph. It also creates bins for similar individuals to categorize them easily. The features chosen are the number of body modules, pairs of limb modules, and pairs of limbs. The reason for going with pairs of limbs and limb modules instead of the actual number is that every time a limb or limb joint is added, a complementary one is added to the other side, so it is impossible to have an odd number of limbs/limb modules.

$$MF = \begin{bmatrix} \text{body modules} \\ \text{pairs of limb modules} \\ \text{pairs of limbs} \end{bmatrix}$$

3.7.2 Diversity

To measure the diversity of an individual, x, in a population, p, the average distance to all other individuals in the population is calculated with a distance measure d. This is done in the same way as Samuelsen and Glette [69]:

$$D_d(x) = \frac{1}{|p| - 1} \sum_{y \in p} d(x, y) \tag{3.4}$$

Two main distance measures were considered, Graph Edit Distance (GED) and the Euclidean distance between the morphological features of two individuals. Both these measures showed similar information, so the Euclidean distance between features was chosen as it is much less computationally heavy (Equation 3.5). A comparison can be seen in Figure 3.7.

$$d(x, y) = ||MF(x) - MF(y)|| \tag{3.5}$$

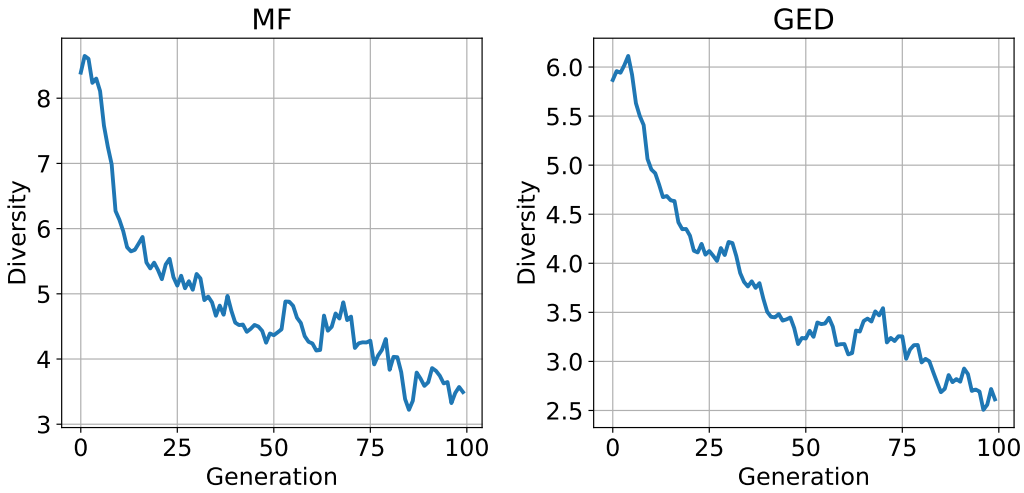


Figure 3.7: Comparison between diversity calculated with Euclidean distance with morphological features and graph-edit-distance.

3.8 Experiment Setup

All experiments are performed using identical setups, but there are variations in which environment the robots are simulated and in which of the EAs (Section 3.6.2) that are used.

All the experiments used the parameter values specified in Table 3.3, and 25 independent runs were performed for each method.

Parameter	Value
p_c	0.33
σ_c	0.2
p_m	1
Tournament size	2
Population size	100
Generations	500
Simulation time	15 seconds

Table 3.3: Parameters used in experiments. p_c is the controller mutation probability for each controller parameter, σ_c is the spread of the controller mutation, and p_m is the morphological mutation probability.

Chapter 4

Experiment 1: Initialization

All experiments focus on the main research question: "What are the effects of gradually augmenting morphologies from a minimal body plan?". Additionally, this first experiment will specifically focus on the effects of MEAT's initialization and aims to answer the sub-question: "How does the initial population affect the search process?". To do this, the simple and random initialization methods will be compared by evaluating the performance, morphological convergence, complexity, and diversity.

A hypothesis that will be explored for this experiment is that a simple initialization will lead to less complex optimized creatures. This is because it is expected that the morphology will converge before the complexity of the random initialization is achieved when no morphological protection is employed. Furthermore, this early convergence will reduce the performance and also the variation of elites between the different runs. Another hypothesis is that the initialization will impact the diversity of a population. Specifically, the low-diversity initial population will lead to less diversity in later generations.

This chapter will begin with a description of the experimental setup, followed by the presentation of results, and conclude with an analysis of the findings.

4.1 Setup

Given that the goal of the experiment is to investigate the impact of the initialization of the population on the search, there are two methods compared in this experiment, SNP and RNP. These are chosen to focus on the effects

of initialization without the impact of morphological protection and see how MEAT would perform with just the simple initialization. The individuals are simulated in the flat environment (Section 3.6.1), and the parameters used are described in Section 3.8.

4.2 Results

4.2.1 Fitness

Figure 4.1 depicts the average fitness of the best individuals across the 25 runs. The plot shows that RNP starts off with a higher average fitness. However, around generation 10, SNP catches up, and they perform very similarly, as indicated by the overlapping standard errors. The fitnesses of the elites in the last generation are plotted in Figure 4.2, overlaid with a box plot to show the distribution. Here the results are also quite similar, although RNP exhibits slightly higher minimum-, max-, and median values.

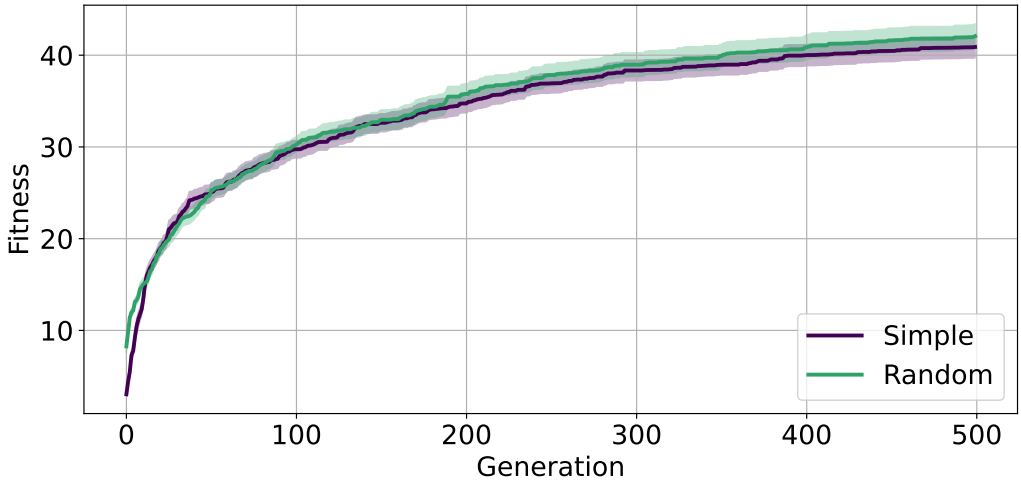


Figure 4.1: Comparison of average max fitness for simple and random initialization, where simple is SNP and random is RNP. The shaded area is the standard error.

To confirm whether the differences between the two approaches are statistically significant, a two-sided Mann-Whitney U test was utilized. The null hypothesis H_0 assumes that both sets of values are sampled from the same distribution. The test was performed at generation 0, 50, and 500, and the resulting p-values can be found in table 4.1. At a significance level of 0.05, there was only a significant difference at generation 0.

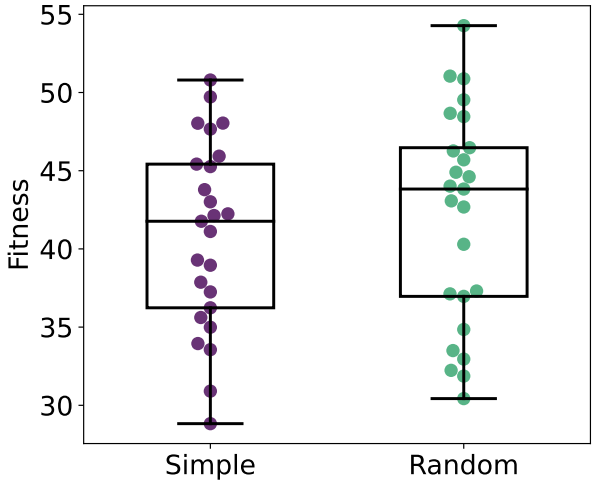


Figure 4.2: Box plots of the highest fitness found for each run.

Generation	P-Value
0	$\mathbf{1.416 \cdot 10^{-9}}$
50	0.786
500	0.497

Table 4.1: P-values of the Mann-Whitney U tests for maximum fitness of each run. The bold value is significant at a significance level of 0.05.

4.2.2 Morphology

One of the hypotheses was that simple initialization would lead to a quicker convergence to a morphology than random. To test this, the generation of convergence for morphology was calculated. This was measured by finding the generation the morphology of the elite last changed. After this generation, only the control system of the elite could have changed. The convergence generation for each run is shown in Figure 4.3, presented as a swarm plot. The evolutionary runs with random initialization converged significantly later than those with simple initialization (p-value = 0.034, two-sided Mann-Whitney U).

The average number of modules per individual in the population was calculated at each generation to see whether the different methods found different solutions. The average numbers of all the 25 runs were also calculated, and the results can be viewed in Figure 4.4. The hypothesis was that the simple initialization would lead to smaller robot configurations with fewer modules on average. In the figure, SNP has a lower average throughout all generations.

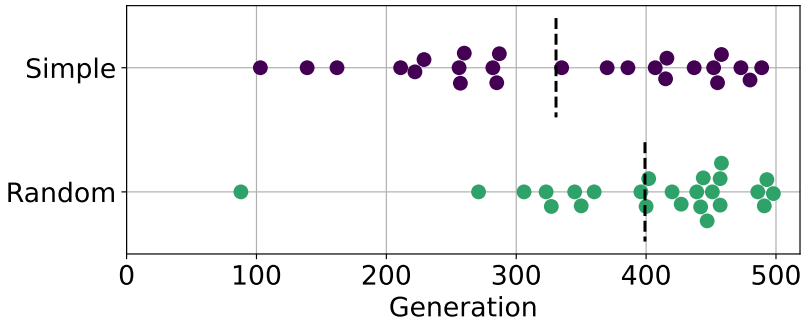


Figure 4.3: The generations when the morphology of each of the elites converged. The dotted line is the mean.

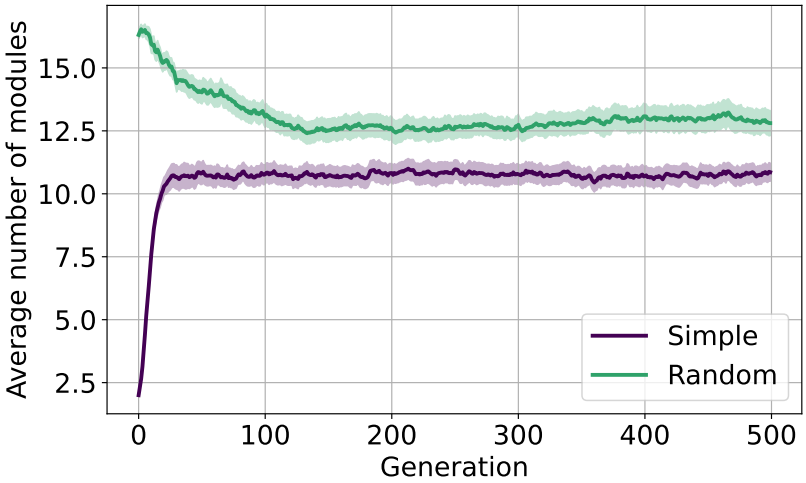


Figure 4.4: Comparison of the average number of modules for all individuals in the populations at each generation. The shaded area is the standard error.

In Figure 4.5, the elites' fitness and their number of modules are plotted as a scatter plot. This plot shows that the elites of the runs with SNP never have more than 12 modules, while the elites of RNP have 10 out of the 25 elites with 13 or more modules. The average number of modules for the elites can be found in Table 4.2, and RNP has significantly more modules per elite (p-value = 0.0027, two-sided Mann-Whitney U)

To illustrate the differences in the body configurations of the elites, the elites are mapped to the three morphological features introduced in Section 3.7.1. In Figure 4.6, these features are plotted pairwise to illustrate the different elites' body plans. This figure shows that most elites from both SNP and RNP have two pairs of limbs. Especially for RNP, this is prominent, as only two of the elites are not quadrupeds. RNP also has more concentrated peaks, 20 of the

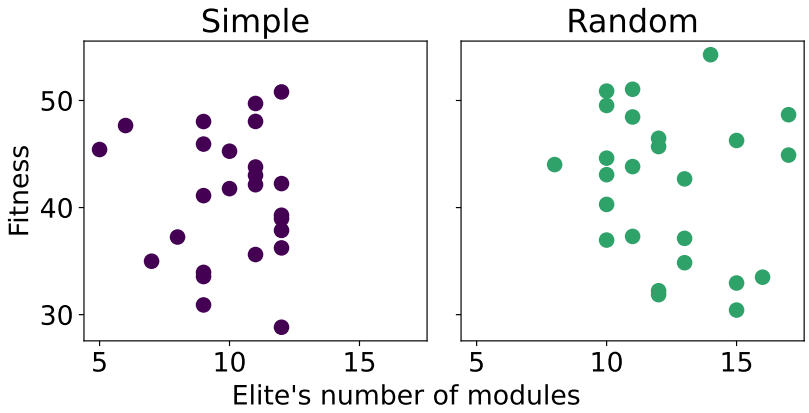


Figure 4.5: Scatter plots of the elites' number of modules and respective fitnesses. The left is the elites from SNP, while the right is from RNP.

Elites	Method	Avg. n. of modules	Diversity
All	SNP	10.0	4.08
	RNP	12.32	3.49
Top 10	SNP	9.5	4.54
	RNP	12.9	3.35

Table 4.2: Average number of modules and diversities of elites. The top section is for all 25 elites, while the bottom is for the top 10 of each respective method.

25 elites ended up with three or four body modules and four limbs. The most variation here is found in the length of these limbs. This figure also illustrates what can be seen in Figure 4.4 and 4.5, that SNP ends up with smaller body plans. But it also shows that this is primarily because of fewer and shorter limbs and not necessarily because of fewer body modules or the number of limbs.

To have a numeric measure of whether or not simple initialization leads to less variation in the elites, the diversities of the elites were calculated with respect to the other elites of the same method. The results can be seen in Table 4.2, and there is a clear difference between the methods. The numbers indicate that SNP, on average, leads to more varied elites. When looking at just the top 10 elites, the difference between SNP and RNP is even more prominent.

One question that arises is why SNP leads to fewer modules. Is the reason that the search area is less explored compared to RNP? To try to answer this question, all the different body configurations explored throughout the runs

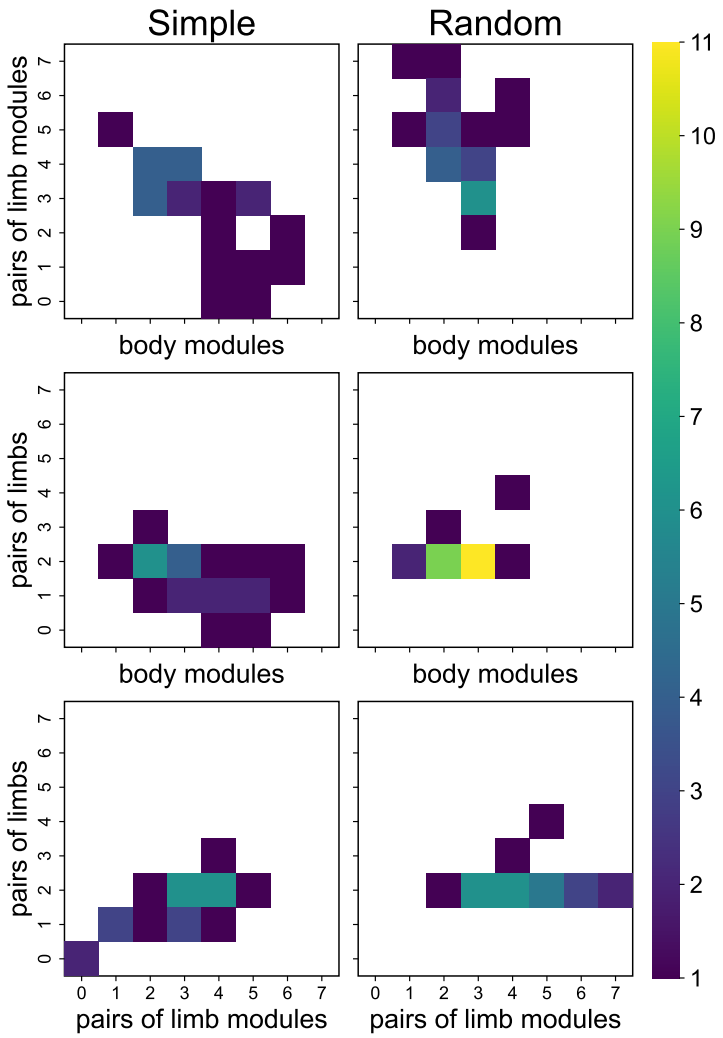


Figure 4.6: Comparison of morphological features for elites. The color bar represents the number of elites, so the color at each square is how frequently that combination of features occurred.

were tracked.

$$explored = gens \cdot pop\_size \cdot runs = 500 \cdot 100 \cdot 25 = 1,250,000$$

This means that a total of $1,250,000$ individuals were tracked per method. The morphological features of each of these individuals are plotted in the same way as the elites in Figure 4.7. From this figure, it looks like the different methods explore the search space similarly. The main differences seem to be that again, SNP explore more in the area with few limbs and limb modules, while RNP explores solutions with a higher number. A high concentration of the solutions explored by RNP seems to be in the area of three and four body modules and

four limbs. This was also seen when looking at the elites.

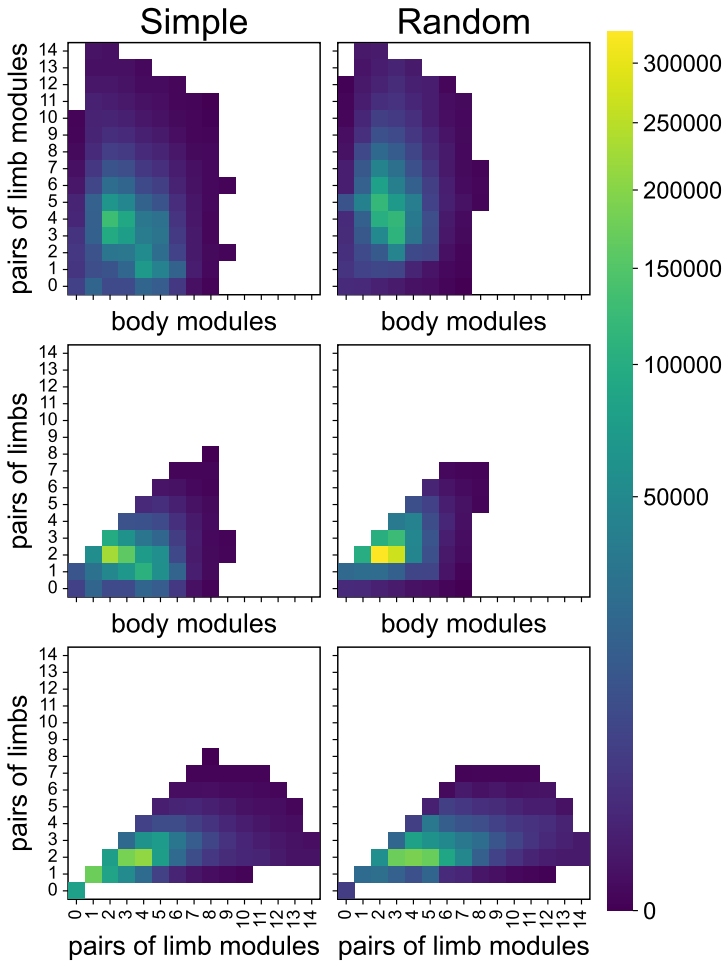


Figure 4.7: Comparison of morphological features explored throughout all runs. The color bar shows the number of individuals, so the color of each square in the grid is decided by how many individuals have that combination of features.

4.2.3 Diversity

The average diversity over the generations was calculated for all of the runs using the Euclidean distance between features described in Section 3.7.2. The averages of the runs are shown in Figure 4.8. The simple initialization leads to a diversity of 0 in the first generation, as every individual consists of one joint and the root module. This quickly rises to a maximum average of above 5 at generation 12 before it slowly decreases again. For RNP, it starts with a diversity measurement of 8 and decreases rapidly at the start. After generation 50, both methods have similarly diverse populations, and after generation 100, the diversity stays almost stagnant for both methods.

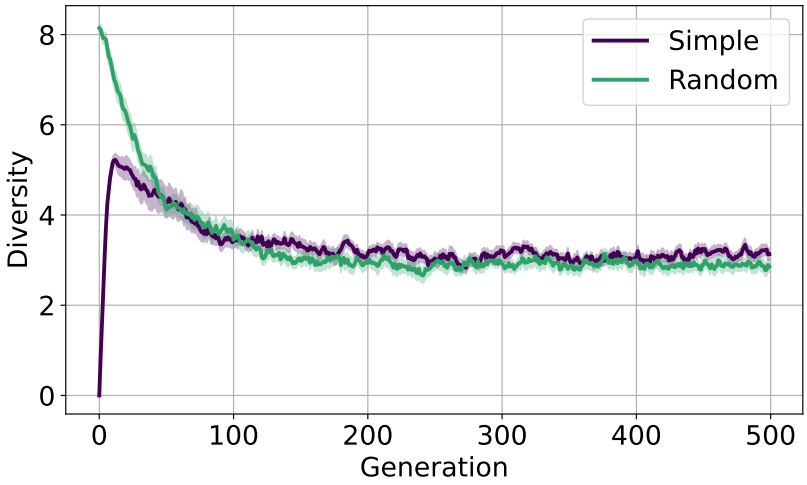


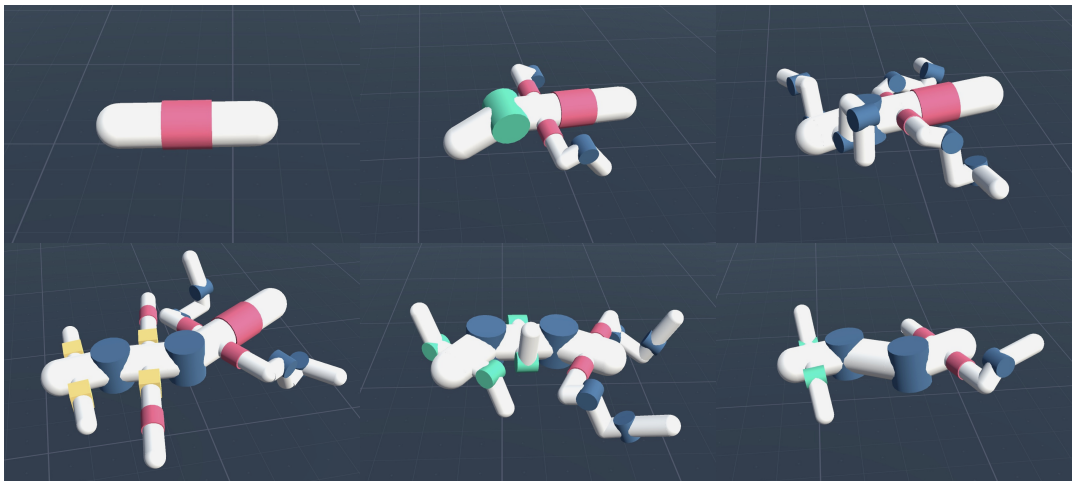
Figure 4.8: Average population diversity of each run over time calculated with the Euclidean distance between features (Section 3.7.2)

4.2.4 Qualitative Results

Ancestry

Another difference between simple and random initialization is clear by looking at the ancestry of the elites of the different runs. Simple initialization obviously starts off simple and grows over the generations. Interestingly, in some instances, like the example in Figure 4.9a, the morphology grows rapidly in the first generations and then starts shrinking. Through the ancestry of the elites from random initialization, it seems like a large part of the initial robot is present in the later generations. Often there are no significant changes and more minor changes to streamline the original "concept". For the robot in Figure 4.9b, there are minimal changes to the main body but more changes to the limbs.

For SNP, many elites either started off as snakelike or used two limbs to drag forward. As the generations went on, they often managed to add a couple more legs and effectively incorporate them. For many of the elites, the gait used in early generations is still very present in later generations even though new modules and limbs have been added.



(a) Simple initialization.

(b) Random initialization

Figure 4.9: Parts of the ancestry from first to last generation of a random elite from SNP at the top and RNP at the bottom.

Body Plan and Gait

The behaviors of the elites can be seen in the videos for SNP[1], and RNP[2].

A lot of the elites for both methods have many similarities. As mentioned previously (Section 4.2.2), many of them are quadrupeds with three or four body modules. These body modules are utilized for locomotion as well by maximizing the length of the strides, which in turn minimizes the time the limbs are on the ground. This makes the whole body swing from left to right quite dramatically. There is also a tendency for these quadrupeds to either have stiff front- or hind legs. The stiff legs often serve as stabilizers, while the

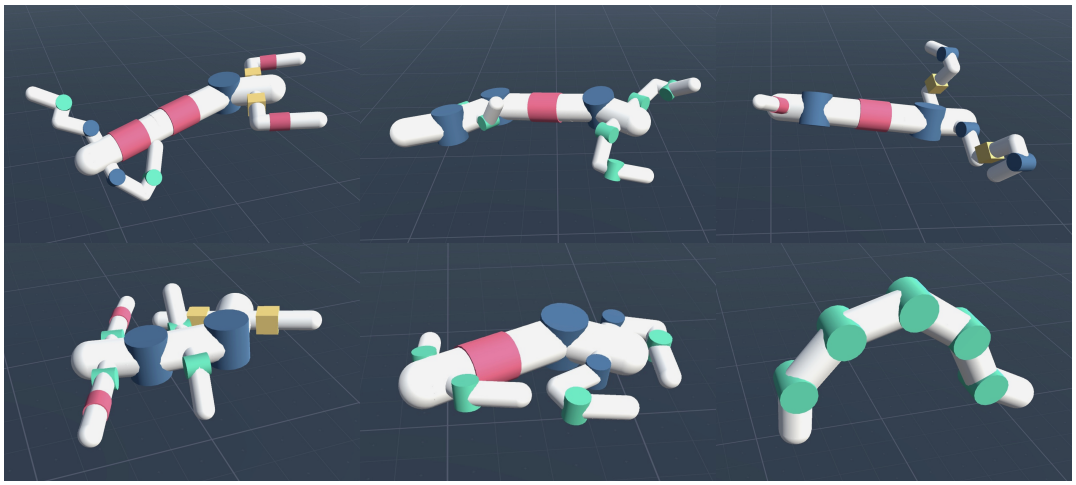
[1] https://youtu.be/0R6eBr7LWPM
[2] https://youtu.be/pHXu9WC8Ixk

other legs are primarily responsible for forward movement. Still, they can also contribute to locomotion with the help of joints in the body. For RNP, 14 of the 25 elites had stiff front- or hind legs, while for SNP, six of the 25 had this configuration. In Figure 4.10, the top six elites of both methods are shown, and the shorter, stiff hind legs are presented in quite a few of them. Both methods also had quadrupeds that managed to synchronize all four legs, but as this is a lot more difficult to optimize, it makes sense that it was fewer.

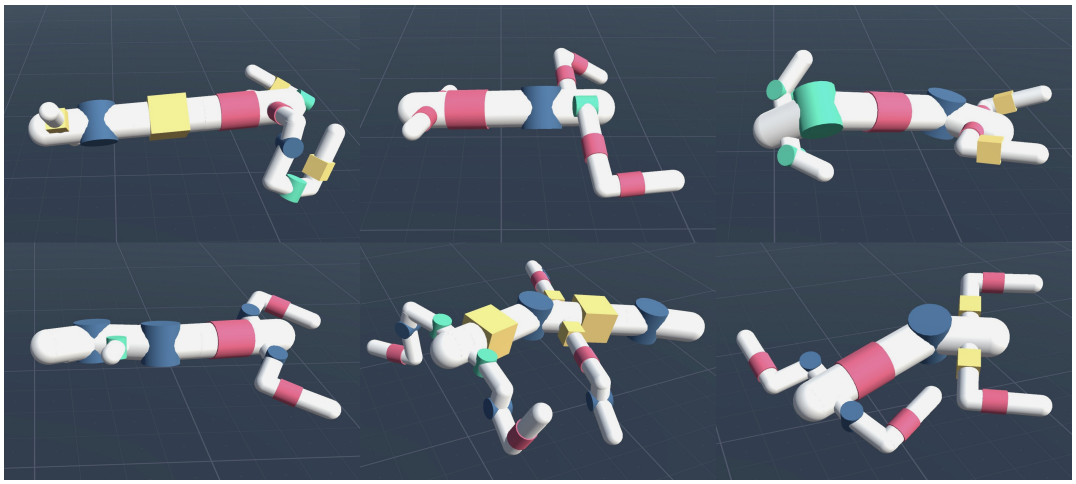
The main difference in elites, which is instantly noticeable, is that the RNP only had two elites that were not quadrupeds, while there was a lot more variation in the number of legs for SNP. Nine of these elites were either snakelike (with or without stabilizing limbs) or were clearly descendants of snakes. These solutions move like a wave, pushing from the ground to lift the body up. The last elite in Figure 4.10a can be seen moving in this fashion. There were also a couple of elites that only had two limbs and were using these limbs to jump/drag over the ground.

Quite a few of the elites had more potential in optimizing their control system. Many of them did not move straight, which meant that they slowly drifted to one side, and because the fitness function only rewarded locomotion in one direction, this led to less and less gain in fitness. By re-evaluating the solutions with a longer simulation time, it was found that most of the elites continued to have a good progression. Most of them also seemed quite robust and stable, using supporting limbs not to fall over.

As the fitness function only measures the distance traveled, many of the top elites had more of a jumping gait. The lower-fitness elites often looked more stable and had better synchronization, but moved a bit slower than the more jumpy solutions. The lower fitness elites also had more variation in their gait techniques. While the top elites often alternate the right and left legs, some of the lower fitness elites used both legs synchronously. This is less effective as the body can become almost stagnant between each “jump”. However, by alternating the limbs, the body can maintain a more fluid and continuous motion, allowing for greater forward momentum.



(a) Simple initialization. Fitness-values from top left to bottom right: 50.8, 49.7, 49.3, 48.0, 48.0, 47.7



(b) Random initialization. Fitness-values from top left to bottom right: 54.3, 51.0, 50.9, 49.5, 48.7, 48.5

Figure 4.10: The six best-performing elites of simple and random initialization.

4.3 Analysis

By looking at the performance of the two methods, the results do not show any significant difference in the fitness values achieved (Figure 4.1). This disproves the hypothesis that using a simple initialization would lead to significantly worse performance. Still, there are substantial differences regarding other aspects. SNP converges towards one morphology faster than the RNP does (Figure 4.3), confirming the hypothesis that using an initial population of minimal initialized robots leads to earlier convergence. This earlier convergence might indicate that starting with a simple initialization is more prone to premature convergence, or it may also mean that good solutions are discovered

more quickly. If the reason is premature convergence, the EA is unable to escape these local optima in the search space. Some of these local optima are snakelike, and as this body plan is quite close to how the minimal individuals are initialized, it is easy to find and optimize early in the search. The control of limbless bodies is very different from e.g. quadrupeds, so evolving legs will normally be detrimental later in the evolution.

The limbless solutions are probably an effect of copying the parent controller when adding new modules. When several identical modules are added with close-to-identical controllers, the movement of the modules will be identical, with an offset between each of them. By not copying the parent controller, there will be more difficult to find these wavelike movements, as the chance of adding several similar controllers successively is low.

The generation of morphological convergence is quite late for many runs (Figure 4.3), especially for random initialization, where 60% of the runs converged after 400 generations. One possible explanation is that it takes longer to optimize the larger configurations, as unnecessary or hindering parts need to be removed. This also suggests there is a benefit to running for more than 500 generations. Because of the later convergence for RNP, the difference between SNP and RNP might also be significant for fitness if the evolution is run for longer. Observing the gaits of the elites also supports the notion that longer runs are beneficial. A lot of the elites exhibit controls that need a bit more tuning to uncover their full potential. Having longer runs can potentially uncover more effective gaits that exploit the full capabilities of their body plans.

There is also a significant difference in the number of modules for the elites (Table 4.2). Because the fitnesses of the two approaches are insignificantly different, this might indicate that the simple initialization leads to more effective solutions. Another theory here is that fewer modules can be a result of premature morphological convergence, and this is also supported by other measurements. This confirms the hypothesis that simple initialization leads to less complex optimized creatures. More surprisingly, the diversity of the elites revealed that simple initialization results in greater variety among all elites and the top 10 elites. As a result of this, the hypothesis that simple initialization leads to less varied elites seems to be disproven.

One interesting observation is that despite the lack of diversity in the first generation, SNP manages to attain a good population diversity within a few

generations (Figure 4.8). By the 50th generation, the difference in population diversity is insignificant. While the hypothesis was that simple initialization would lead to lower diversity, this is only true for the earliest generations. During most of the evolution, the difference in diversity was practically zero.

To summarize the results, simple initialization leads to less complex body plans, more varied elites, and similar population diversity. This means that the simple initialization approach achieves performance on par with the random initialization while reducing the complexity of the robots. However, there is a significantly earlier morphological convergence, which may impede the search from attaining global optima.

Chapter 5

Experiment 2: Protection

The results of Experiment 1 (Chapter 4) showed a lot of promise for MEAT as the performance when using a minimal initialization method was higher than expected. However, there were indications that using simple initialization leads to earlier morphological convergence. This was not surprising, and a way to deal with this issue was already present in the design of MEAT. By enhancing the two methods used in Experiment 1 with the morphological protection method discussed in Section 3.5.1, the hope is that both methods will improve.

The focus of this experiment will thus be the research question; "What are the effects of using morphological protection based on age?". To answer this, the performances of methods with and without protection will be compared. Both the effects on fitness values, individual ages, morphological convergence, and diversity will be investigated.

A hypothesis suggests that MEAT will yield high-performing minimal solutions and avoid the early morphological convergence demonstrated in Experiment 1. The performance of MEAT will be similar or better compared to using the random initialization combined with protection, but also better than the methods without protection. It is also expected that both methods will gain higher performance when protection is employed.

5.1 Setup

This experiment aims to illustrate how the morphological protection mechanism performs compared to no protection. Therefore, the methods used in

Experiment 1, SNP and RNP, are compared to their counterparts with protection, MEAT and RP. The simulations are again done in the flat environment, and the parameters used are listed in Section 3.8. The results of SNP and RNP are the same as those presented in the previous experiment.

5.2 Results

5.2.1 Fitness

In the previous experiment, it was shown that there was no significant difference between the fitness values achieved for SNP and RNP. In Figure 5.1 and 5.2, a clear difference between MEAT and the other methods can be observed. When looking at fitness over the generations, all four methods overlap early on, but over time MEAT diverges from the rest. This difference grows throughout the rest of the generations. Looking at the individual runs' fitness values in Figure 5.2, MEAT reaches higher fitness values but also avoids the lower-fitness scores.

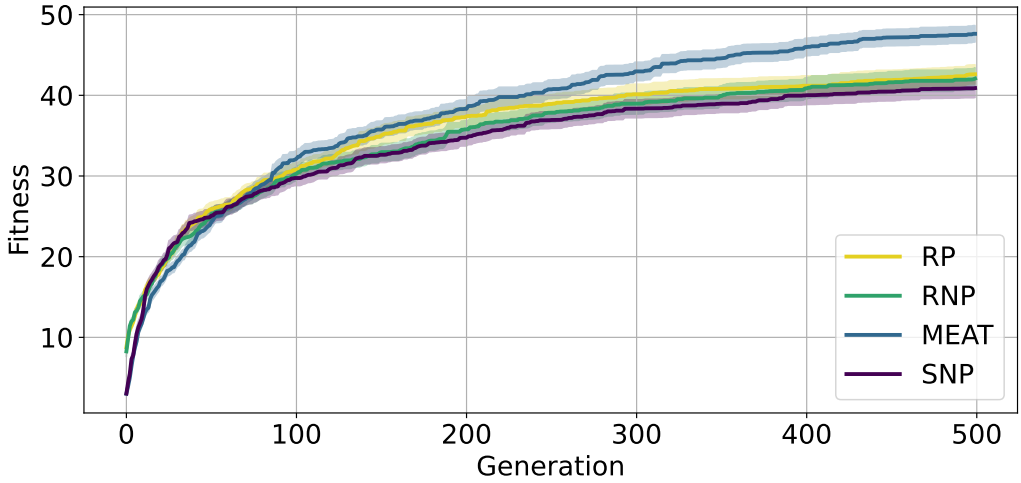


Figure 5.1: Comparison of average max fitness for SNP, MEAT, RNP, and RP. The shaded area is the standard error.

A two-sided Mann-Whitney U test is again utilized to test the statistical significance of what is visually observed from the plots. The null hypothesis H_0 assumes that the fitness values of all methods are sampled from the same distribution. The significance level chosen was 0.05, and a Bonferroni correction was done because of the several comparisons. This gives a value of $0.05/6 = 0.0083$. With this adjusted level, it is confirmed that there is no significant difference between SNP, RNP, and RP at the last generation.

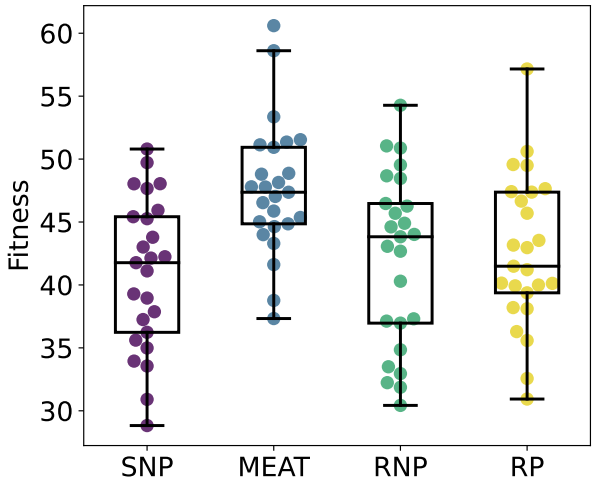


Figure 5.2: Box plots of the highest fitness found for each run.

However, it also shows that MEAT performs significantly better than the rest. At generation 250, the difference is insignificant for all the methods. The p-values for all the comparisons are listed in Table 5.1

Comparison	Gen 250	Gen 500
SNP vs. MEAT	0.0478	**0.0004**
SNP vs. RNP	0.5475	0.4971
SNP vs. RP	0.2072	0.4263
MEAT vs. RNP	0.2003	**0.0052**
MEAT vs. RP	0.3933	**0.0036**
RNP vs. RP	0.5220	0.9381

Table 5.1: P-values of the Mann-Whitney U tests for maximum fitness of each run. The bold values are significant at a Bonferroni corrected significance level of 0.0083.

5.2.2 Age

To further investigate the effects of morphological protection, the age of all individuals were collected in every run. Because age impacts the selection, it is evident that the population's average age is lower when using protection. What is more interesting is to see whether this also holds true for the best-performing individuals in a population. Because older individuals have had more time to optimize their controller, it is logical that the best individuals often are older.

Figure 5.3 displays the average age of the top 20 best-performing individuals in each generation, with and without protection. SNP is combined with RNP,

and MEAT with RP. This gives a total of 50 data points for each category. The effect of the protection is shown clearly here, as the average age without protection is far higher than with protection. The variation in age for protection is also much lower, while without protection, there are huge variations between the runs, illustrated by the percentiles in the plot.

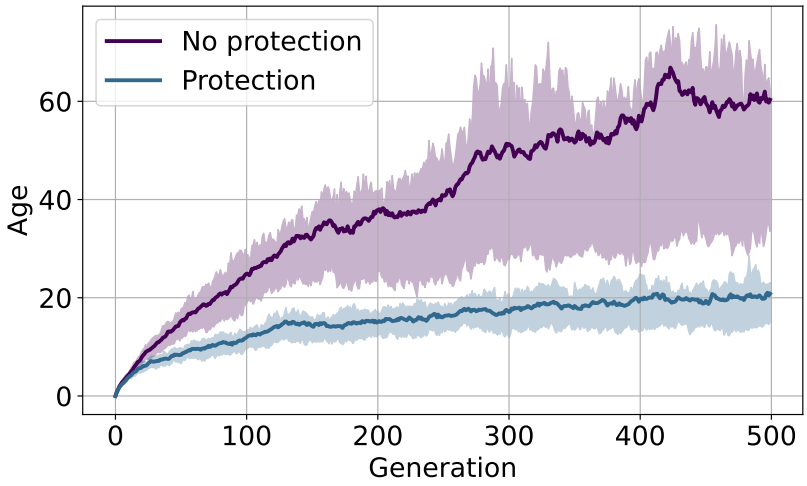


Figure 5.3: The average age of the top 20 individuals at each generation. *No protection* consists of the combined runs of both SNP and RNP, while *protection* is MEAT combined with RP. The shaded area is the area between the 25th and 75th percentile.

5.2.3 Morphology

One of the problems with using simple initialization, illustrated in Experiment 1, is that the morphologies converge earlier than they do when using random initialization. To compare if this is still the case when using morphological protection, the generations of morphological convergence were also calculated in this experiment. The comparison between the different methods can be seen in Figure 5.4. Here, an improvement can be observed for simple initialization, as MEAT on average converges later than SNP and that there is more of a cluster after 400 generations. For random initialization, more runs converged very late, but there were also more that converged early.

By doing a two-sided Mann-Whitney U test with a Bonferroni corrected significance level of 0.0083, there were no significant differences between any of the methods (Table 5.2). Still, there is an indication that morphological protection delays the convergence for simple initialization.

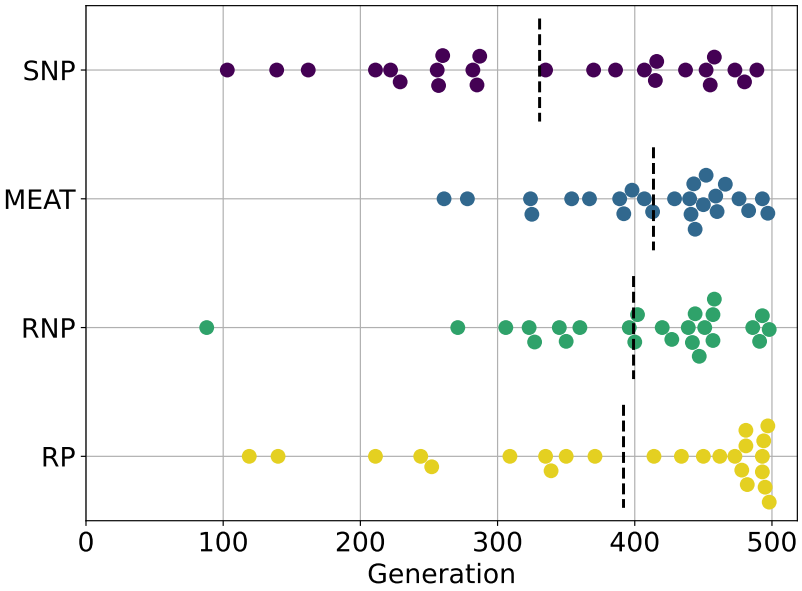


Figure 5.4: The generations when the morphology of each of the elites converged. The dotted line is the mean.

Comparison	P-Value
SNP vs. MEAT	0.0137
SNP vs. RNP	0.0336
SNP vs. RP	0.0290
MEAT vs. RNP	0.7124
MEAT vs. RP	0.6979
RNP vs. RP	0.5604

Table 5.2: P-values of the Mann-Whitney U tests for morphological convergence of each run. At a Bonferroni corrected significance level of 0.0083, no values are significant.

Even though the morphological convergence is later for the MEAT compared to SNP, there is no indication that the elites' number of modules changes more in later generations. Figure 5.5 shows how the elites' number of modules changes over time and is sampled every 50th generation to avoid showing changes that are reverted shortly after. This illustrates how there are a lot of changes in the complexity early on, but quickly reduces to an average of about zero. The elites' complexity, therefore, on average, is pretty much unchanged from the 100th generation to the end of the runs for all methods.

Experiment 1 revealed that using SNP resulted in a lower average number of modules per robot. By using morphological protection, this average increased

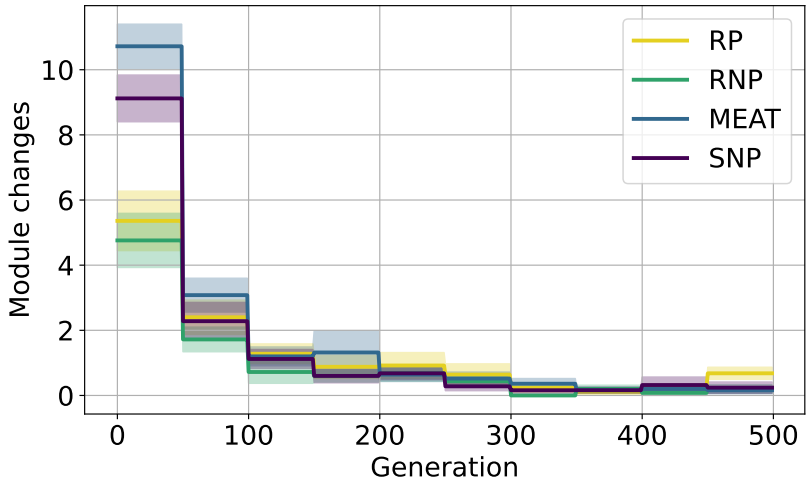


Figure 5.5: The average change in the elites' number of modules. Every 50th generation, the elites' numbers of modules are measured and the difference to the last measurement is calculated, illustrating how the complexity changes over time. The shaded area is the standard error.

by almost two modules, but when using random initialization, there was no difference. These results are depicted in Figure 5.6, which displays the average numbers of modules for every individual in the populations plotted over time. The average for MEAT is comparable to that of RNP and RP. Figure 5.7 illustrates the number of modules for each elite. The number of modules with MEAT ranges between 8 and 14, while RNP and RP have elites with up to 17 modules. Even if the average number of modules in the population is similar for MEAT, the random initialization still produces elites with more modules.

Elites	**Method**	**Avg. n. of modules**	**Diversity**
All	SNP	10.0	4.08
	MEAT	11.08	3.65
	RNP	12.32	3.49
	RP	11.88	3.72
Top 10	SNP	9.5	4.54
	MEAT	11.0	4.32
	RNP	12.9	3.35
	RP	11.9	3.94

Table 5.3: Average number of modules and diversities of elites. The top section is for all 25 elites, while the bottom is for the top 10 of each respective method.

To check if morphological protection significantly impacts the variation in

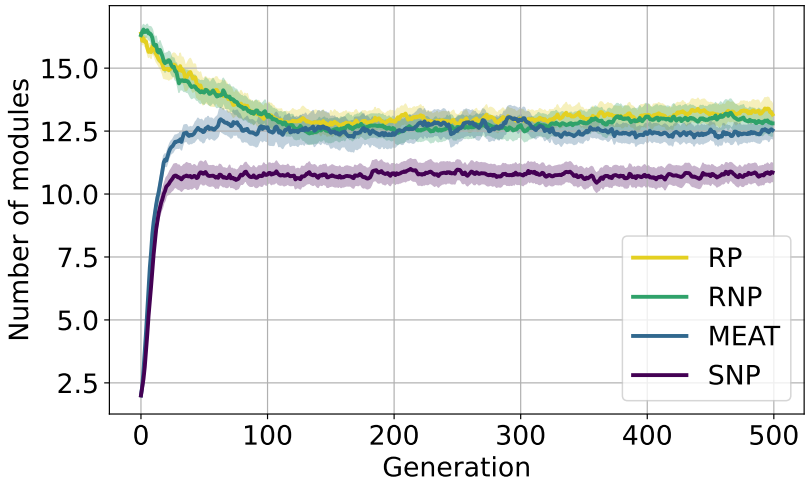


Figure 5.6: Comparison of the average number of modules for all individuals in the populations at each generation. The shaded area is the standard error.

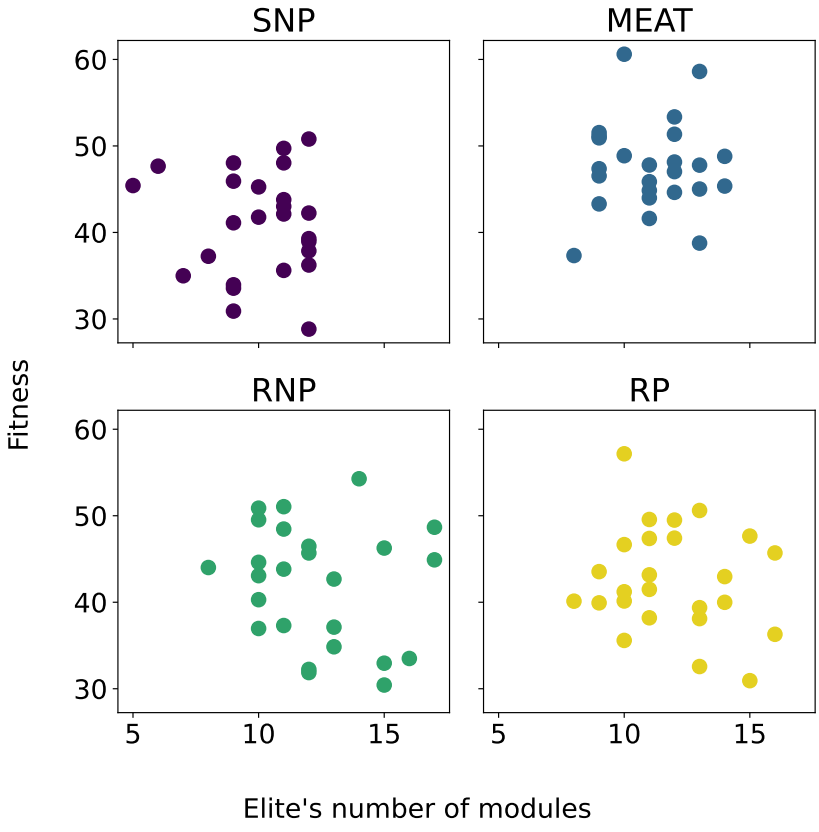


Figure 5.7: Scatter plots of the elites' number of modules and respective fitnesses.

the elites, the diversity and the average number of modules of elites were calculated for all methods (Table 5.3). The results show that protection lowers the average number of modules for random initialization and increases it for

simple initialization. This is true for all the elites and the top 10. Looking at the diversities of elites, protection increases the variation of the elites for simple initialization and decreases it for random initialization. There is no clear indication that protection lowers or increases the variation of the elites, as the observed differences are relatively small.

5.2.4 Diversity

The main effect of morphological protection is that individuals that would be eliminated because of lower fitness are allowed to survive when they are younger than the better individuals. This has the added effect of maintaining diversity in the population. In Figure 5.8, this effect can be observed by comparing the protection methods with those without protection. The diversities of both methods using protection are almost identical and significantly better than those of the two methods without protection.

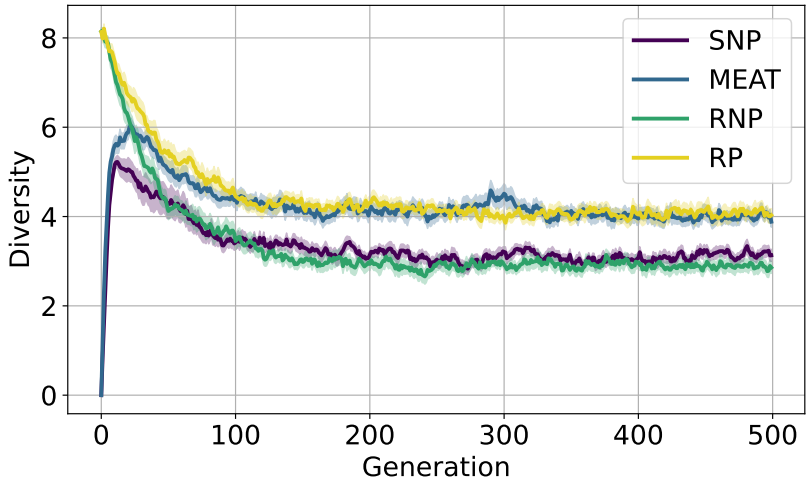


Figure 5.8: Average population diversity of each run over time calculated with the Euclidean distance between features (Section 3.7.2)

5.3 Analysis

The results show a significant increase in fitness when using morphological protection for simple initialization, while there is no improvement for random initialization (Figure 5.1). This lack of improvement is interesting because, as shown by the performance of MEAT, an improvement is possible. This disproves the hypothesis that both initialization methods would obtain increased performance due to using protection.

One theory as to why random initialization did not improve is that the complexity of individuals rarely decreases, as this seldom directly gives a higher fitness. The higher-than-necessary complexity gives more parameters to optimize for control, making things much more difficult compared to optimizing the more minimal solutions found using simple initialization. This theory is supported by looking at the average number of modules for all the elites when using random initialization (Table 5.3), as this number is higher than for simple initialization. Still, this table also indicates that morphological protection leads to a decrease in the elites' number of modules for random initialization.

Other reasons for the lack of improvement for random initialization can be the time it takes to optimize the COs and the limitations of the morphological search space. When modules are added, they inherit their parent's controller, speeding up the optimization time as it already inhibits some useful properties. Newly mutated individuals manage to survive long enough to optimize the controller even without protection, so protection is not needed. Because of the morphology rules (Section 3.4.1), the search space is also very reduced, and it is a lot easier to find working solutions. If these rules were less restrictive, like if there was no guaranteed bilateral symmetry, there might also have been more advantages to morphological protection for both initialization methods.

Even though there are indications that morphological protection delays the morphological convergence for simple initialization, there are no indications that the elites' complexities change more (Figure 5.5). This indicates that the elites converge to a number of modules and that this one rarely changes in the later generations. Contrary, Figure 5.4 shows that there normally are morphological changes in the final 100 generations for the methods with protection. The morphological changes, therefore, are either swap-mutations or mutations that cancel each other out. Mutations that cancel out can be when a module is added in one generation but removed shortly after removed. The individual will then have an age of zero while also having a controller optimized for the current body plan. This exploits the protection, thereby gaining an advantage in the selection without having a changed morphology.

As stated, the average number of the modules for the elites is higher for MEAT compared to SNP, and lower for RP compared to RNP (Table 5.3). This indicates that morphological protection helps to escape local optima and therefore manages to come closer to a more optimal number of modules. One morphology discussed in Experiment 1 was the snakelike one, which emerged

quite a few times for SNP and never for RNP. Similarly, for the methods with protection, some of the highest performing individuals from MEAT are limbless, but none of the elites for RP. When it comes to variety in the elites, both SNP and MEAT outperform RNP and RP, with a larger difference for the top 10.

One effect of protection is easy to observe when looking at the ages of the top 20 solutions in a population (Figure 5.3). These individuals dominate with high fitness values but remain younger than the top individuals without protection. The lower age indicates that protection has the desired effect of having top individuals that are not just old morphologies with fully optimized controllers. The relatively stagnant age over time also shows that new morphologies manage to break into the top 20. Still, this might also be because some individuals reset their age through two mutations that cancel each other out, as previously mentioned.

Another notable effect is that MEAT achieves similar results to RNP and RP regarding the average number of modules in the population (Figure 5.6). Because this number is higher than SNP, it suggests that MEAT traverses more of the search space. Comparing just the elites of the different methods shows that MEAT's elites have lower complexity and higher fitness compared to the methods with random initialization. MEAT, therefore, explores more complex solutions, but the less complex ones perform better and are easier to optimize.

The hypothesis suggested that MEAT would avoid the early convergence demonstrated in Experiment 1 while still keeping the complexity of morphology low. The results indicate this being true, even if the average number of modules for MEAT was higher than SNP (Table 5.3). MEAT still yielded less complex elites than both random initialization methods. The differences between the methods when it comes to convergence are too small to confirm or deny the hypothesis, as none of the methods converged significantly earlier than the rest. Still, on average, SNP converged earlier than the other methods. Another hypothesis was that MEAT would perform better than both methods without protection while matching the performance of RP. This is confirmed, as the protection had a pronounced effect for simple initialization, and MEAT outperformed the other three approaches (Figure 5.1).

From the results, it is confirmed that morphological protection leads to increased population diversity. When using simple initialization, the effects of the protection are also later morphological convergence, higher fitness, and

more exploration. There was, however, little to no effect of protection for random initialization. In this experiment, MEAT outperforms or matches the other methods in all metrics measured, resulting in higher fitness while maintaining a lower complexity.

Chapter 6

Experiment 3: Environment

This final experiment aims to answer the research question: "How does the environment impact the evolution of modular robots?". The previous experiments demonstrated that the initialization method influences the solutions found and that morphological protection is an effective tool for improving simple initialization. Thus, the question is whether this also holds true for other environments. Specifically, this experiment aims to determine whether the robots can adapt to a challenging environment by discovering new body configurations, or if the same morphologies that were successful on the flat surface also will succeed in a set of stairs. Another question is how the environment impacts other aspects of the development process, such as optimization speed, diversity, and convergence.

The results of Experiment 2 (Chapter 5) showed that MEAT outperformed the other methods. Is this also the case in a different environment, or can other methods perform better here? The effectiveness of protection might be completely different in a more challenging environment, and it might also improve random initialization. The hypothesis is that the advantage of using protection will be more prominent in a more challenging environment, as the optimization likely takes more time. Thus, the relative difference between SNP and MEAT would be larger when evolved in the stairs.

6.1 Setup

To investigate the impact of the environment, the results from the flat environment (Experiment 2) are compared to identical runs in the environment

with stairs (Figure 3.6). The parameters used are found in Section 3.3. The stairs are a significantly more challenging environment, and the direction of travel is angled at 26.57°. Still, the same fitness function as earlier is used, so the fitness is the distance traveled in only the x-direction.

6.2 Results

6.2.1 Fitness

Because of the different environments, comparing the fitness values achieved on the stairs directly with those in the flat environment does not make sense. Still, some interesting observations might be made by comparing the performances of the different methods in each of the environments. In Figure 6.1 and 6.2, the performance of every method in both environments is plotted. By looking at these results, the performances of the different methods are very similar in both environments, but with a large difference in fitness values achieved. MEAT performs the best in both environments, while the other methods perform similarly. The methods with protection did not gain a larger advantage in the more difficult environment.

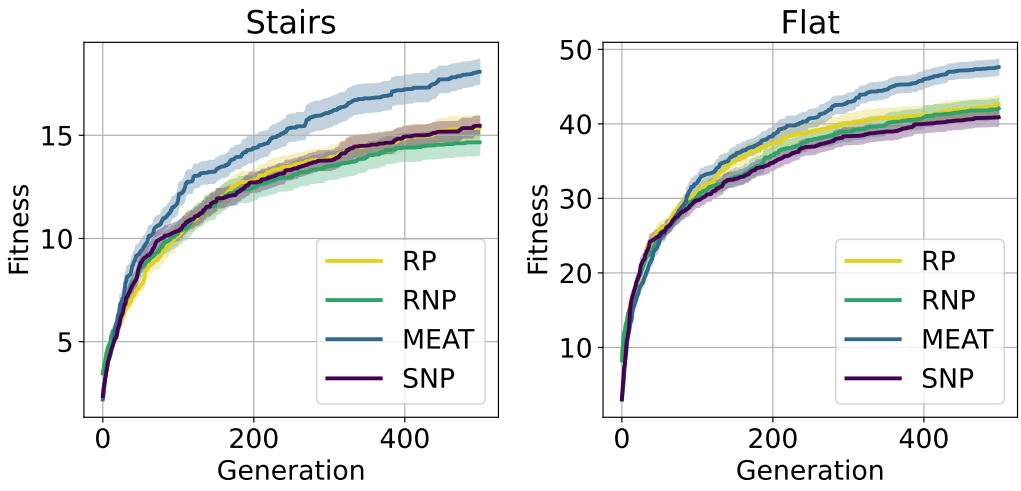


Figure 6.1: Comparison of average max fitness for SNP, MEAT, RNP, and RP. The shaded area is the standard error.

6.2.2 Morphology

Another way to measure the performance of the methods in different terrains is to look at the generation of morphological convergence. In Figure 6.3, there are

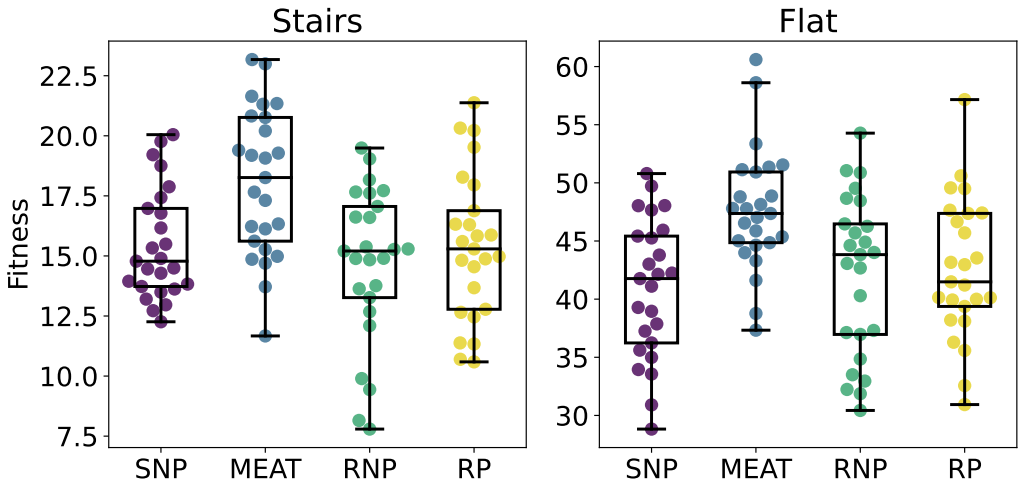


Figure 6.2: Box plots of the highest fitness found for each run, for the stairs to the left, and the flat to the right.

swarm plots of all methods for both environments. No method does significantly better than the rest, and the generations of convergence seem similar in both environments.

Looking at Figure 6.4, there seems to be a larger difference between the average number of modules for the methods in the stairs compared to the flat. In the early generations, all methods had a lower average than when evolved in the flat environment. Over time this difference decreases, but all methods, bar RNP, have a lower average number of modules per individual for the stairs.

Comparing just the average number of modules of the elites, the results are very similar across the environments (Table 6.1). When looking at just the top 10, the difference is larger for all the methods, and the random initialization methods ended up with an average number of modules similar to the other methods. RNP went from 12.9 to 10.4 average number of modules, and RP went from 11.9 to 10.1.

Table 6.1 also includes the diversities of the elites. This diversity says something about the variations of solutions found. For SNP and MEAT, the diversities are lower for the stairs than the flat, but for RNP and RP, the opposite is true. These values make it unclear whether the stairs lead to less or more varied elites. Diversity is decreased from flat to stairs for the top 10 elites of all methods.

To look closer at these results, all elites from the four methods are collected

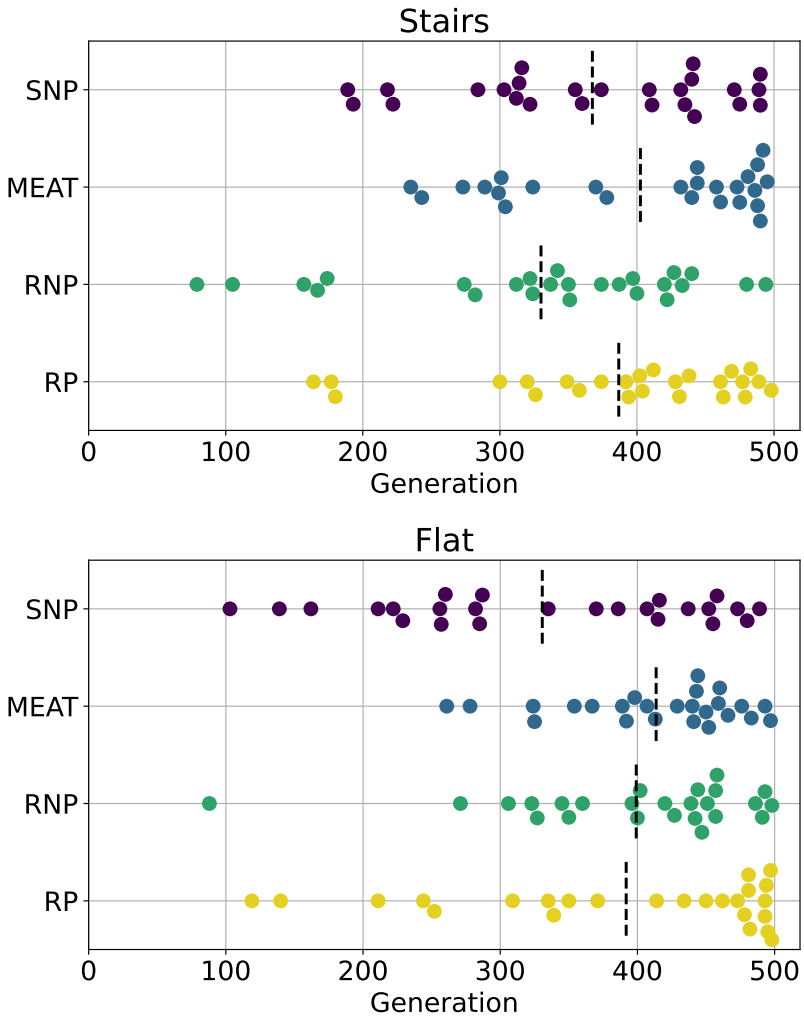


Figure 6.3: The generations when the morphology of each of the elites converged. The dotted line is the mean.

for each environment. Calculating these elites' average diversities resulted in a score of 3.998 for the ones evolved in stairs and 3.908 for the flat. Again the results are very similar. To investigate if elites from stairs and flat are different, the average difference to the elites from the other environment is calculated (using the Euclidean distance between features, Section 3.7.2). This means that for every elite who evolved in stairs, the distance to all the elites who evolved in the flat environment is calculated. The average of these distances is 4.091, insignificantly higher than the environment-specific distance. From this measure, there is an indication that both environments give rise to similar individuals. In Figure 6.5a, this similarity can also be observed, as the elites from both environments have very similar features.

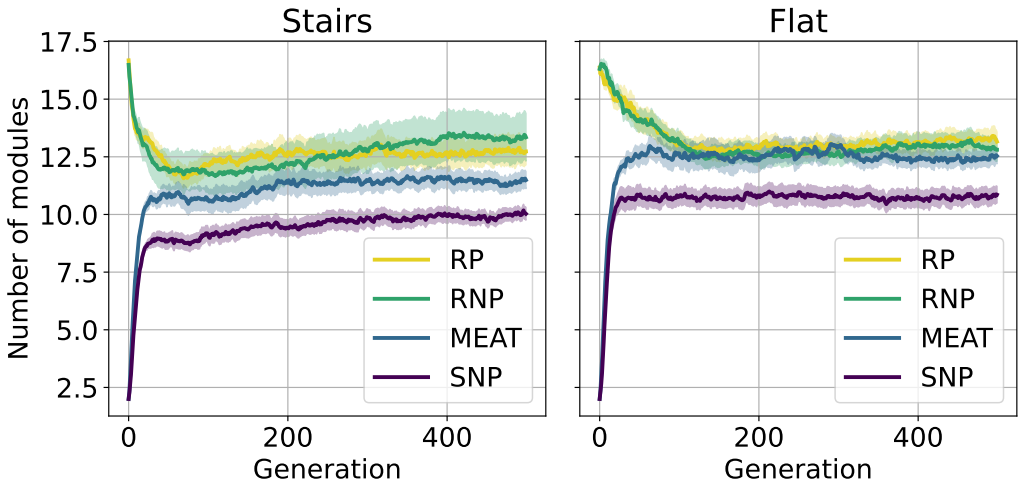


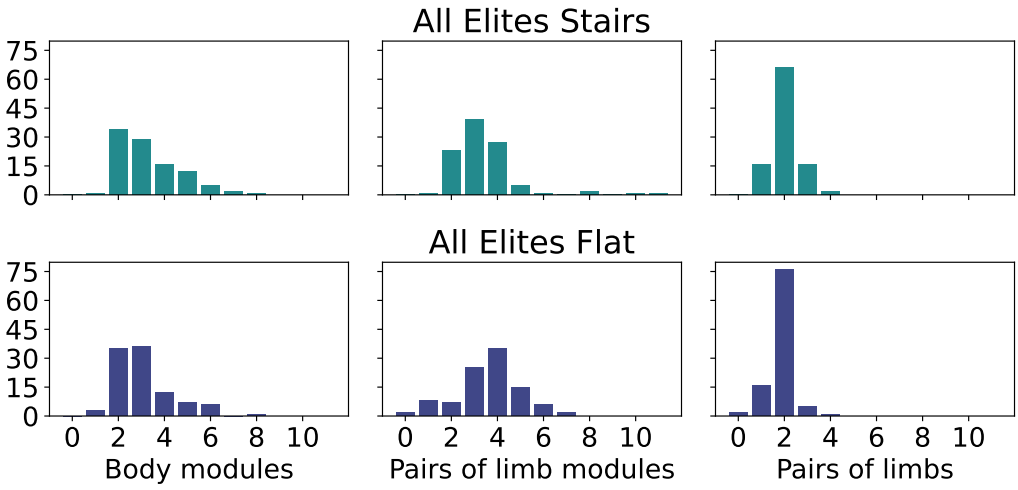
Figure 6.4: Comparison of the average number of modules for all individuals in the populations. The shaded area is the standard error.

Elites	**Method**	**Avg. n. of modules**		**Diversity**	
		Stairs	Flat	Stairs	Flat
All	SNP	10.04	10.0	2.95	4.08
	MEAT	10.4	11.08	3.39	3.65
	RNP	12.16	12.32	5.39	3.49
	RP	11.88	11.88	4.19	3.72
Top 10	SNP	9.9	9.5	3.57	4.54
	MEAT	9.8	11.0	2.13	4.32
	RNP	10.4	12.9	2.86	3.35
	RP	10.1	11.9	2.41	3.94

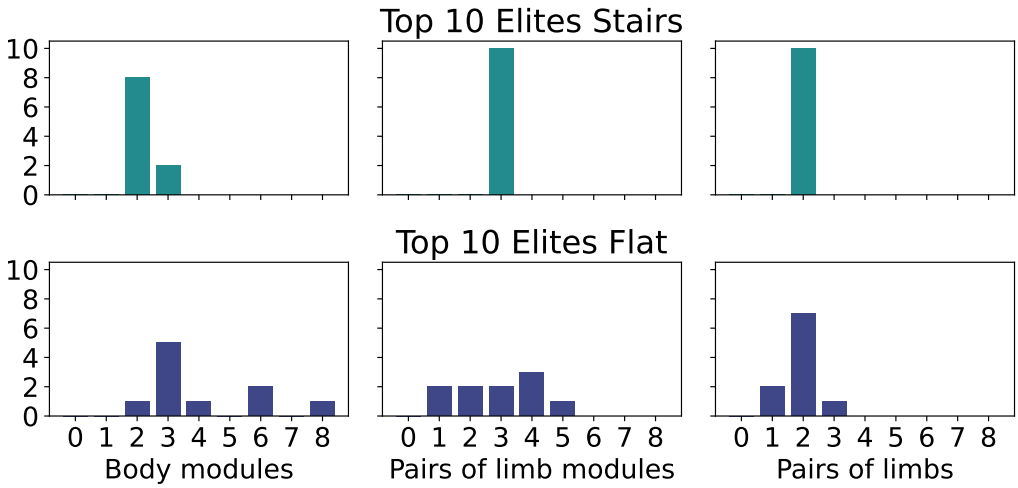
Table 6.1: Average number of modules and diversities of elites. The top section is for all 25 elites, while the bottom is for the top 10 of each respective method.

The average diversities of the top 10 elites were quite a lot lower for the elites that evolved in stairs (Table 6.1). To investigate the reasons for this, their morphological features were plotted in Figure 6.5b. Here it is clear that all top elites evolved in stairs have the same features. All of them are quadrupeds with six limb modules, and eight have two body modules. In the flat environment, the individuals have a lot more variation.

Figure 6.6 illustrates the number of modules most effective for the respective methods in the two environments. From these plots, it can be observed that the clustering is very similar in both environments. For the random initialization methods, the worst-performing elites also had the most modules when evolved



(a) All elites



(b) The 10 elites with the highest fitness values

Figure 6.5: Features of the elites across all methods for each environment.

in stairs. The best-performing individuals are clustered similarly to SNP and MEAT.

6.2.3 Diversity

Comparing the average population diversity for the two environments (Figure 6.7), there seem to be two clear groups in the flat environment and no clear grouping for the stairs. MEAT and SNP reach higher averages in the flat environment, RP perform similarly, while RNP actually maintains a higher average in the stairs.

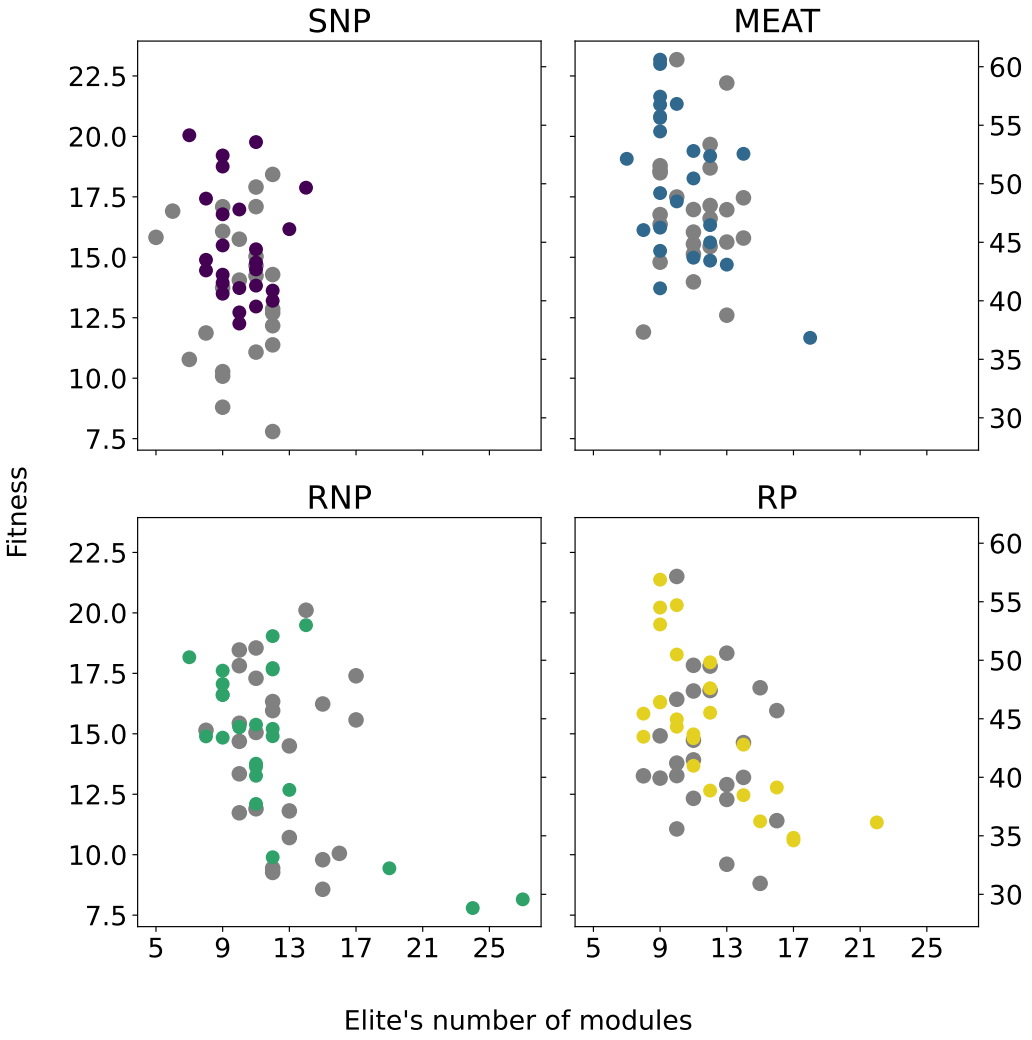


Figure 6.6: Scatter plots of the elites' number of modules and respective fitnesses. The colored dots are evolved in stairs, and the primary y-axis is their fitness. The grey dots evolved in the flat environment; the secondary y-axis is their fitness.

6.2.4 Qualitative Results

From the results in Section 6.2.2, no differences were found between the final elites' morphologies based on their morphological features. There was, however, less variation in the top-performing elites who evolved in the stairs. Looking at just the morphological features does have some drawbacks, as it is a very simplified version of the robot's actual body plan. The body plans might contain other qualities and differences not expressed through these features. To see if this is the case, a qualitative analysis of the body plans and gaits has to be done.

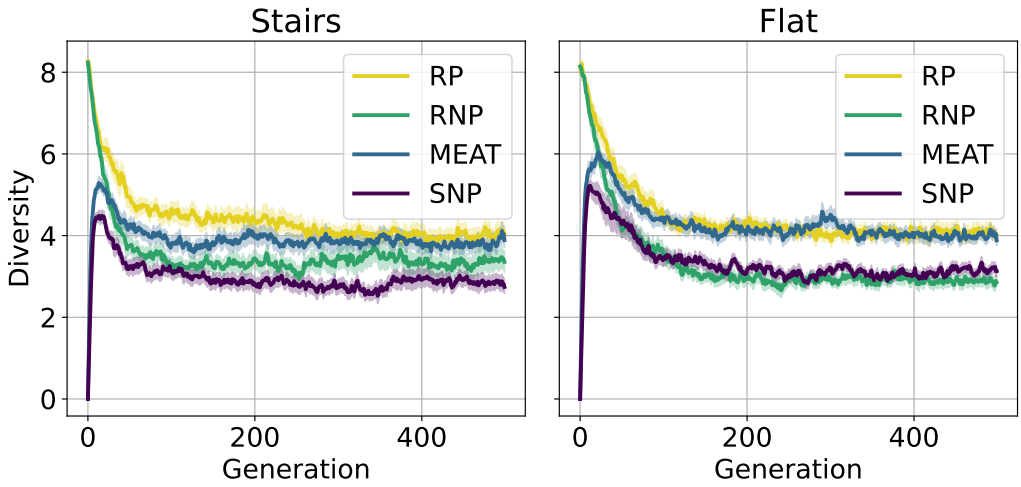


Figure 6.7: Average population diversity of each run over time calculated with the Euclidean distance between features (Section 3.7.2)

Body Plan and Gait

Looking at the different strategies the different elites use, it is clear that there are some differences between those that evolved in the stairs and those that evolved in the flat environment. Because of the angle of the stairs, individuals are much more prone to tipping over than they are in the flat environment. To mitigate this, the elites use different strategies. One of these is to use a "ballast" in the front of their body. This ballast is typically expressed as an extra body module in the front that acts as a counterweight to move the center of mass forward in the robot body, thus hindering the individual from tipping over. Figure 6.8 shows this for three of the pictured elites, top right, middle of the third row, and middle of the bottom row. Another strictly morphological way of dealing with the issue of tipping over is to have limbs that stabilize. This is generally expressed by having forward-facing front limbs or stiff rear-facing hind legs.

The elites from the stairs also use the gait effectively to avoid tipping over. Compared to the elites from the flat environment (Figure 6.9), they keep a lower center of mass when walking and do not jump up. Instead, they use small, more controlled steps to move forward. When looking at the different gait strategies, it seems to be a lot more variation from the flat environment, while the stairs have elites that utilize a few strategies often.

The less complex individuals tend to do better in the stairs. This was also

noted in Section 6.2.2. The top elites had fewer modules, and all the top 10 elites had the exact same strategy (Figure 6.5b). From visually comparing the top elites from stairs with the ones from flat, it is clear that there are more varied locomotion strategies and body plans for the flat environment. These strategies perform similarly, and no strategy dominates. This is not the case for the stairs, where it seems to be one dominating strategy that seems quicker and more stable than the alternatives. In Figure 6.8, the top 3 elites of each method are displayed. Even if the top 3 are not necessarily representative of a general trend, both methods with protection found individuals with the same configurations. These individuals have no unnecessary parts, as every module is utilized.

The gaits and morphologies of the top 20 elites can be viewed in the videos for the stairs[1], and the flat environment[2].

6.3 Analysis

The results of evolving in the stairs are similar to the flat environment. For both environments, MEAT outperformed the rest, with the other three methods performing similarly (Figure 6.1). The relative differences between the methods are no larger in the more challenging environment, so these results do not indicate that morphological protection is more effective in a more challenging environment. Thus, the hypothesis that evolving in a more challenging environment would lead to a greater advantage of using protection can be discarded.

A reason why there is no increase in the advantage of protection might be because there are fewer viable solutions in the stairs. This can be seen when looking at the variation in the top elites compared to the flat environment (Figure 6.5). There might therefore be very clear peaks in the search landscape, and the methods without protection can find well-performing morphologies before convergence.

Looking at the generation of morphological convergence (Figure 6.3), there is no indication of later convergence in the stairs. However, there is a steeper slope in the fitness improvements in the later generations for all methods. This reflects the greater difficulty of walking up the stairs compared to a flat surface, so

[1] https://youtu.be/dxFKTTmn03s
[2] https://youtu.be/HT6AngmX8io

Figure 6.8: Top 3 elites evolved in stairs environment for all 4 methods. The top row is SNP, next is MEAT, then RNP, and finally RP.

individuals need more time to adapt their control system to step up the stairs. In a flat environment, creating forward locomotion is much easier as there are no obstacles. This observation suggests that running longer might allow MEAT to reach a higher fitness relative to the other methods, as the fitness slope for MEAT is also steeper.

The most effective number of modules in the respective environments also differ (Table 6.1). It is more effective in the stairs to have fewer modules compared to the flat environment. One possible explanation for this can be that there is a clear dominating strategy in the stairs that have quite a low number of modules. This strategy might be dominating because fewer modules are more effective, or it can be the cause of the lower average number of modules. It does make sense that smaller robots are more effective on the stairs, as they are lighter and more agile, making them better suited for climbing the stairs. Additionally, smaller robots can be optimized quicker, which is advantageous

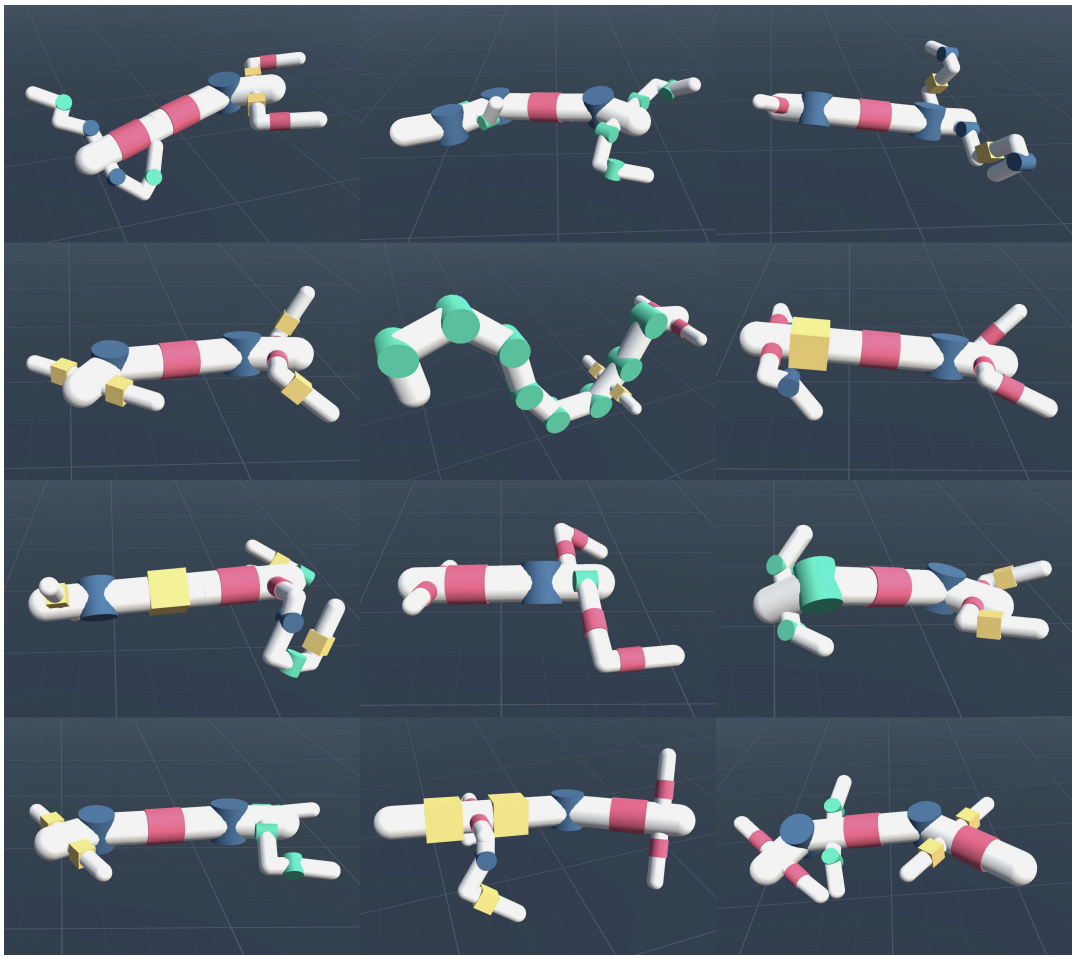


Figure 6.9: Top 3 elites evolved in the flat environment for all 4 methods. The top row is SNP, next is MEAT, then RNP, and finally RP.

in the context of the longer optimization time required in the stairs.

Over time robots with more modules might be better, but they might not have time to optimize during the first 500 generations. When looking at the populations' averages, there seems to be a slow increase in the number of modules throughout the generations for all methods in the stairs. In contrast, the number remained relatively stagnant in the flat environment. This is likely because it takes longer for the population to converge to one morphology in the stairs. In the last generations of both environments, the average number of modules per individual in the populations is similar to the average for the elites for each respective method. This implies that there is a complete convergence of morphology in the population, so every individual has the same number of modules. Still, there is an increase in modules for all methods in the stairs, so as the optimization process continues, larger configurations might arise. It would make sense that the higher effectiveness of the smaller configurations in

the stairs would translate to an advantage for simple initialization. There is, however, no indication of this from the results.

As expected, there are notable differences in body plans and gaits between the environments. Specifically, the individuals exhibit various strategies to adapt to the incline and steps of the stairs, as well as to improve their stability. Still, some solutions converged early and performed poorly. A few of these solutions are non-working and fail to get past the first step of the stairs. These non-working solutions are completely avoided in the 50 runs using simple initialization. There are also elites from both environments that use the same strategies. Some of these strategies perform well in both environments, while others struggle with stability in the more challenging environment. The variation of the elites was a lot lower for all the methods in the stairs compared to the flat environment. For the stairs, there was a lot of convergent evolution to similar solutions, even across the different EAs used.

The impact of evolving in a different environment was a lot less than initially expected. The different methods performed similarly with respect to each other as they did on a flat surface, and MEAT still performed better than the rest. The individuals adapted to the stairs, but there was less variation between the elites, with one dominating strategy.

Chapter 7

Discussion

Through three experiments, MEAT has been explored by investigating different aspects of the algorithm. The results showed a lot of promise, but many things can still be investigated further. This chapter discusses the experiments conducted, results achieved, and limitations, focusing on the research questions, robots evolved, and methods used. Following this, future work is discussed, looking into what can be further investigated, possible improvements, and the larger potential for evolutionary- and modular robotics.

Experiment 1 (Chapter 4) investigated the first research sub-question, "How does the initial population affect the search process?". In Section 4.3, the results are analyzed, and the results showed that the simple initialization achieved equal performance and diversity with less complex elites. Still, there was a significant earlier morphological convergence.

Following this, Experiment 2 (Chapter 5) focused on the question, "What are the effects of using morphological protection based on age?". The morphological protection increased population diversity and delayed the morphological convergence for simple initialization, leading to MEAT outperforming the other methods. However, the results also showed no increased performance with protection when using random initialization. In the analysis of the results (Section 5.3), it is theorized that this is due to the quick optimization of the COs and the limitations of the morphological search space.

The final experiment, Experiment 3 (Chapter 6), investigated the research sub-question, "How does the environment impact the evolution of modular robots?". The analysis of the results (Section 6.3) found that the individuals adapted well

to the stairs to create effective locomotion strategies. The performances of the different methods were very similar when evolving on a set of stairs compared to the flat environment. Again, MEAT performed the best. However, the more challenging environment did not accompany a greater advantage to using morphological protection. One possible explanation for this is that the fewer viable solutions in the stairs lead to more clear optima in the morphological search space.

The main research question for this thesis is yet to be discussed; "What are the effects of gradually augmenting morphologies from a minimal body plan?". In the larger context of this question, the results showed that there definitely are advantages to gradually growing the morphology from a minimal starting point. All the compared methods are outperformed when the protection is employed with simple initialization (Figure 6.1). Not only does MEAT perform the best in terms of fitness, but the elites also have more varied morphologies that, at the same time, are less complex (Table 6.1). Additionally, this is achieved without compromising population diversity (Figure 6.7). Therefore, the hypothesis that MEAT would avoid early convergence at the same time as it would perform better or equal to the other methods is confirmed. The results from both the simple and challenging environment also indicate that MEAT can generalize to more than one environment.

7.1 Elites and Strategies

There are several advantages to robots with less complex morphologies. One of them is that larger and more complex body plans require more modules, which in turn requires a more complex control system. As there are more parameters, it will be harder to optimize and make it less robust to changes. One such change can be mutations during evolution, and if too much time is needed to reoptimize the controllers, the mutated individual will likely not survive. Another change can be in the environment, and reoptimizing for a new or changed environment will take more time for a complex morphology. Lastly, an advantage of lower complexity is that it reduces the chance of unnecessary parts and locked-in imperfections.

Similar strategies were dominating across the different methods explored. These strategies were quadrupeds with three body modules for the flat environment and two body modules for the stairs. For the flat environment,

some high-performing solutions had no limbs, but other than this, all were quadrupeds or used only four of their limbs for locomotion. This indicates that having four limbs is an effective configuration, which is unsurprising as this is supported when looking at real animals. An explanation for this phenomenon is that adding more limbs would increase the controller search, as there are more controllers to optimize and synchronize. When going from two to four limbs, this increased search space is worth it, as this allows for more effective locomotion. The extra set of limbs lifts the robot's front or back half, reducing the surface contact time and thereby also reducing friction. If the physics were tuned differently, so the weight of the robots increased, adding a third set of limbs might have been justified, as this could allow for more effective locomotion.

These quadrupeds rarely utilized all four of their legs in the same way. Often either the front or hind legs were completely stiff, not using the joints in the modules at all. Because there is enough torque in the body modules, these two legs could still be utilized for locomotion when the body moves. Using a set of completely stiff legs reduces the number of limbs to optimize/synchronize, thereby shrinking the controller search space. This allows for simpler and faster optimization.

Many of these elites jump or move in ways that would be very difficult in the real world or a different environment, meaning a large reality gap. One idea was, therefore, to evolve for stability as well as fitness. The stability was measured by looking at how much the root module moved up or down, and the individuals were penalized based on this value. The result of this was that individuals that dragged their bodies along the ground would gain an advantage, as this is the most stable it is possible to move. This hindered the search early on, and overall the elites performed worse. An improvement can be to penalize the individuals based on both stability and body modules that make contact with the ground. The effects will be that only limbs can be used directly for locomotion, which might lead to a reduced reality gap.

The environment is also shown to influence the design of the individuals' gaits and body plans. Part of that environment is also the physics configurations and how often individuals receive actions. This was kept the same for both environments, but by changing these parameters, different strategies were dominating. The impact of the torque and controller frequency was especially influential. By having a higher torque or controller frequency, the diversities in

the elites decreased, and there was even more convergence to snakelike bodies. Low values made individuals struggle to move effectively, even if they had enough force to lift from the ground. This points to the fact that the control system is quite limited and that it struggles more to synchronize limbs when a "jumpy" gait is impossible.

7.2 Methods Used

All methods used in the experiments are very elitist. The best-performing individual will always reproduce and survive because of no parent selection and the tournament-remove survivor selection. The drawback of elitism is that it leads to too stable solutions, and the population will easily be trapped in local optima. Many other approaches are less elitist and do not guarantee the survival of the elites. These approaches introduce more noise in the search and lower the chance of getting trapped in local optima. Another effect of the noise is that the controllers can be made more robust. The values of the controllers will vary a lot, and they will need to deal with values within a specific range. A single fine-tuned combination of values has a lower chance of surviving, as small changes will disturb it. The tradeoff is that lower exploitation can slow down the search and also might discard high-performing solutions completely.

Cheney et al. [7] used an even more elitist approach than MEAT, and the advantages of morphological protection were even higher than what Experiment 2 (Chapter 5) showed. This indicates that the more exploitative the EA is, the more advantageous morphological protection is. The protection works by lowering the selection pressure, which will give more exploration that can help to avoid local optima. When using a less elitist approach, the protection will still lower the selection pressure, which might reduce it by too much and thereby hinder the search. The tournament-remove selection chosen for MEAT results from wanting to increase the exploration of Cheney's method, as testing showed that this method led to early convergence and low diversity.

Another reason for the design of the EAs used in this thesis was that other approaches tested showed worse performance. Figure 7.1 shows SNP and MEAT compared to an EA using generational replacement and tournament-add selection. This EA also uses a simple initialized population, and the mutation probabilities were selected based on the best-performing combination tested. What is clear from these results is that both MEAT and SNP perform

significantly better, and that the other EA converges quite a lot earlier. One possibility is that the other EA would perform much better with properly optimized parameters. Unfortunately, because of the stochastic nature of EAs, a lot of runs would be required per parameter combination to have enough data to say what combination performs the best. Another problem is that the runs last for 500 generations, so a parameter sweep for 100 generations would reward the parameters that lead to quicker improvement at the start. Therefore, long runs would have to be performed. Because MEAT has fewer parameters to tune, much time and computational power are saved.

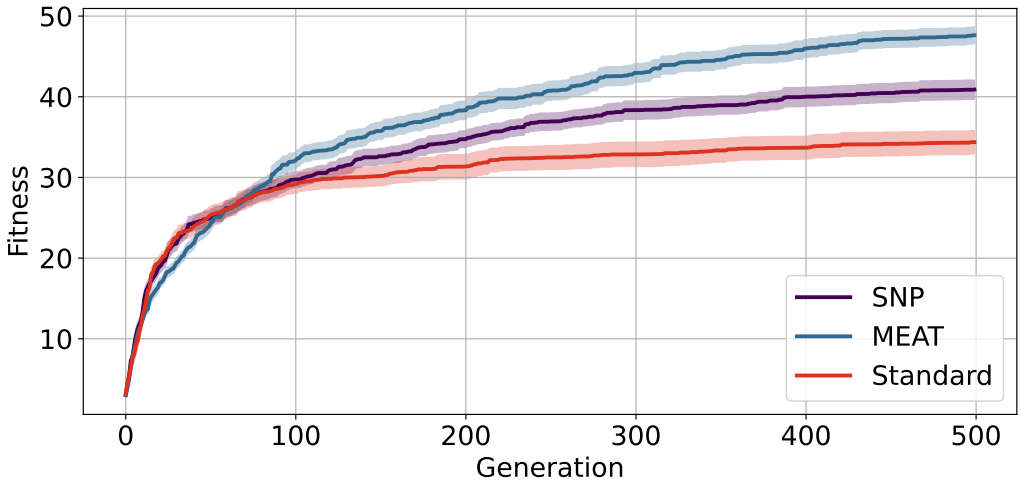


Figure 7.1: Comparison of SNP, MEAT, and more "standard" EA evolved in the flat environment. Standard uses simple initialization, a population size of 100, tournament parent selection with a tournament size of 3, elitism of 1, $p_c = 0.1$, $\sigma_c = 0.2$, and $p_m = 0.2$.

7.3 Future Work

7.3.1 Further Investigations

For random initialization, there was no difference in performance when using morphological protection (Figure 5.1) despite an increased diversity (Figure 5.8). The results of the experiments also showed indications of no complete convergence in the 500 generations. In this case, the benefit of protection will likely grow as the number of generations increases. Running until complete convergence might increase the differences between the methods and make insignificant differences significant. There are, however, very few changes in the number of modules for the elites in the later generations. This can indicate

that running for longer might not change the body plans too much. Still, for all methods, the fitness increases towards the end of the runs, so the control optimization has more potential. This is especially the case when evolving in the stairs.

The reason for not running for longer is time and cost of computation. Ideally, the runs would have lasted longer, and there could have been more than 25 repetitions to strengthen the statistical significance of the results. The problem is that this would have taken far too much time and computational power. A way to reduce the computational time is to reduce the simulation time for each individual. This would allow for more generations while not increasing the time a run takes. The problem with doing this is that short simulations will award high fitnesses for unstable behaviors, like leaping forward and falling over. A longer run will give a more accurate assessment of how robust the behavior is. The simulation time chosen was much lower in the early stages of the experiments, but the elites then tended to fall over or drift to one side when simulated for longer after evolution. This can possibly be avoided by utilizing a different fitness function or a more advanced controller able to take sensory inputs.

MEAT outperformed the three other methods, but further research is needed. This can include investigating the factors that influence the algorithm's success and research into modifications to the algorithm itself (Section 7.3.2). One question is whether the results are similar outside the chosen system. Experiment 3 (Chapter 6) demonstrated that the performance of the different methods was pretty much the same in both a very simple and a more challenging environment. This indicated some generalization as the performances are not specific to only one environment. Still, many things might change the outcome in other systems. Other modules or controllers might lead to entirely different results, and there is no guarantee that MEAT still performs the best. The same can be said about different morphological mutations and encodings.

By testing MEAT in different systems, much can be learned about the algorithm's generalization. A lot can be done, like changing the physics, environment, modules, controllers, genetic encodings, and variation operators. If the algorithm can perform similarly in other systems, it would strengthen the argument that MEAT performs better than the other methods presented in this thesis. It can also be interesting to compare MEAT with a method using an improved random initialization with better individuals. This can lead to

more insights into whether or not it really is beneficial to augment minimal solutions.

7.3.2 Possible Improvements

A lot can be done to try to improve the performance of all the methods used in the experiments. The controller used is very simple and cannot adapt to a changing environment because of the lack of input. Because of this, the chosen environments had to be uniform. The work done in this thesis can be a stepping stone to something more advanced by using a more sophisticated controller that takes sensor inputs from the environment. The robots might be able to perform better in the two presented environments but also perform well in a more rugged terrain. With the COs, the robot would have to be reoptimized in a new terrain because of the lack of inputs, while other control systems can be more adaptable. Another reason to use a controller that takes sensor inputs is that this can be used for targeted locomotions. This can help individuals correct the direction they are moving if they drift to a side, making the movement more robust. Targeted locomotion can also allow the robot to pursue moving targets.

Optimizing a more advanced controller will be more challenging than optimizing the COs. Thus, there might be significant changes in the morphological convergence and the impact of morphological protection. When a controller requires more time to optimize, a mutation to the body plan of an already optimized robot can be even more detrimental. This is because it is necessary to optimize for a higher number of generations to achieve the previous performance. Without morphological protection, there will be a higher chance of not surviving long enough to be optimized after a mutation. Therefore, the protection might improve the performance more than this thesis's results indicated, thereby also improving the performance with random initialization. It can also be interesting to investigate whether the simpler controllers help avoid premature morphological convergence because of their faster optimization time. This faster optimization time is due to a low number of parameters and the copying of the parent's controller when adding new modules.

For both the current controller and a more advanced one, different encodings can lead to very different behaviors and solutions found. Section 7.1 discusses the theory that the reason for the domination of quadrupeds is the increased control search space when increasing the number of limbs. An indirect

genotype-to-phenotype mapping (as in nature) might be advantageous to create more variations in the elites' number of limbs. The indirect mapping allows for the reuse of genetic information, meaning that the search space is not necessarily increased when adding extra limbs. This can help to create well-performing solutions that utilize more than just four limbs. In environments where it is beneficial to have more limbs, an indirect encoding might help create better-performing robots, as the reuse of genetic information allows for easier optimization.

Further work to specifically improve MEAT can also be exciting. Despite the late morphological convergence (Figure 5.4), there is evidence of little change to the elites' morphologies in later generations (Figure 5.5). One theory for this difference is an exploit of the morphological protection method, where an individual has two following mutations that cancel each other out (Section 5.3). These two mutations lead to a reset age despite no change in morphology. To improve upon the protection, the age could, for example, only be reset if there is a new unseen morphology instead of just a change.

Different ways of implementing the morphological protection were also tested during the development process of MEAT but did not perform better than the method presented in this thesis. As mentioned in Section 3.5.1, there is the issue of lack of selection pressure when using protection with tournament-add selection. When individuals are to be selected based on both age and fitness, there is a very good chance that no individuals in a tournament of three are dominated. Especially in the later generations, this is a big problem because the individuals' ages are pretty evenly distributed. The selection pressure is high for the newly mutated individuals, where everyone is the same age, while it is low for older individuals, where almost no one is the same age.

The solution to this problem can be to increase the tournament size, but this can again lead to too much exploitation as the low-fitness individuals never will survive. To mitigate this problem, different solutions were considered, like using bins of ages so individuals compete with others of similar ages. Another method was an increase in tournament size over time. This increasing tournament size was inspired by *simulated annealing* and was tested because of the lack of selection pressure in the later generations. Both methods improved on using standard tournament-add, but they did not perform better than the tournament-remove method chosen. Another protection method that might have potential is based on *truncation selection*, but there was not enough time

to test it thoroughly. This approach uses truncation selection within each tournament, allowing for larger tournaments without being too exploitative. If one individual in the tournament dominates on age and fitness, some dominated individuals will still survive, thus reducing the selection pressure.

A feature that might lead to improved performance is the addition of new random solutions to populations during the evolutionary process, similar to AFPO [38] and ALPS [37]. Both of these methods track age differently than what is done for MEAT, so there might have to be some modifications for how the age is measured. The reason this might be necessary is that new, completely random individuals will likely perform worse than mutated, optimized individuals. A solution can be that these new individuals are initialized with a lower age than the age after a morphological mutation.

7.3.3 Potential

Future work within evolutionary- and modular robotics might benefit from using concepts from MEAT. The achieved results suggest that there are advantages of starting with minimal solutions and that this can lead to well-performing robots without unnecessary complexity. This can mean that other algorithms and approaches also can improve by utilizing simple initialization methods and not wasting time and resources to create well-distributed initial populations.

The benefits of gradually augmenting an initial population of minimal solutions have been demonstrated for NeuroEvolution through NEAT [19]. The results of this thesis indicate that this is also transferable to Evolutionary Robotics. Therefore, the findings suggest that gradually augmenting solutions can improve performance and is a promising strategy for optimizing complex systems through different evolutionary methods.

There is also potential in using the morphological protection from this thesis outside of evolutionary robotics. The protection method can easily be used for other purposes with co-optimization by selecting when the age is increased. When increasing the age after changes for an objective, the protection will protect innovation for that objective. This will have the effect of preserving the diversity of solutions in the population.

In addition, other fields, such as physics-based animation, can also benefit from using and developing virtual creatures using modules. By using a control

system with sensory inputs, targeted locomotion can be achieved. Evolutionary modular robots, therefore, can help quickly model and animate creatures that can be controlled.

Chapter 8

Conclusion

Premature morphological convergence is a common problem when co-optimizing the morphology and control of robots. This thesis considered whether augmenting simple body plans aids in evolutionary optimization or whether it leads to premature convergence. To perform these experiments, the MEAT algorithm was designed, and Experiment 1 investigated how the simple initialization would stack up against a random initialization method. The goal of Experiment 2 was to find out how morphological protection would influence the methods, while Experiment 3 focused on the effect of the environment on the robots' evolution and on the elites' strategies.

In Experiment 1, it was found that the simple initialization led to faster morphological convergence and less complex body plans compared to starting with a randomly initialized population. Still, the variation in the elites was higher for simple initialization, and the population diversity was similar. This shows that despite converging earlier, the simple initialization performs on par with random initialization.

When morphological protection was introduced in Experiment 2, the results showed a significantly improved performance with simple initialization. MEAT reached higher fitness, converged later, explored more of the search area, and maintained a higher population diversity compared to SNP. There were, however, minimal effects of protection for random initialization outside of improved population diversity. MEAT outperformed or matched the other methods in all metrics measured.

The final experiment, Experiment 3, showed that the environment had little

impact on the performance of the methods. MEAT continued outperforming the other methods, but the advantage of using protection in the more challenging environment was not larger. The evolved robots generally adapted well to the stairs, although there was a more evident dominating strategy that resulted in less variation among the elites.

In general, MEAT outperforms all other methods tested while also keeping the complexity of the robots lower. Despite the reduced complexity, there are no indications that MEAT explores less of the search area despite the initial population being concentrated at one single spot. The results also demonstrate that starting with an initial population with no diversity quickly improves and matches methods with higher initial diversity. These findings suggest that spending time and computational power to make an initial population evenly distributed in the search space will not always be beneficial.

The findings also give insights into the effects of morphological protection as a multi-objective selection operator based on fitness and morphological age. Maintaining a diverse population through morphological protection enables more exploration of the search space, thereby delaying morphological convergence. Regardless of the higher exploration, not all methods will experience increased performance by using protection, demonstrated by the random initialization methods.

Minimal initialization and morphological protection from using MEAT are demonstrably beneficial when optimizing the morphology and control when evolving robots. This approach can thereby be useful to be implemented in other evolutionary robotics approaches to explore the search space more effectively. Ultimately, the insights supplied by this thesis can aid in the rapid exploration of a variety of robot designs that, although now simulated, can eventually contribute to more effective robot designs in the real world.

Bibliography

[1] Charles Darwin. *On the origin of species, 1859*. Routledge, 2004.

[2] Karl Sims. "Evolving virtual creatures." In: *Proceedings of the 21st annual conference on Computer graphics and interactive techniques - SIGGRAPH '94*. the 21st annual conference. ACM Press, 1994, pp. 15–22. ISBN: 978-0-89791-667-7. DOI: 10.1145/192161.192167.

[3] Gregory S. Hornby and Jordan B. Pollack. "Evolving L-systems to generate virtual creatures." In: *Computers & Graphics* 25.6 (Dec. 2001), pp. 1041–1048. ISSN: 00978493. DOI: 10.1016/S0097-8493(01)00157-1.

[4] Joel Lehman and Kenneth Stanley. "Evolving a diversity of creatures through novelty search and local competition." In: *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*. Genetic and Evolutionary Computation Conference, GECCO'11. Jan. 1, 2011, pp. 211–218. DOI: 10.1145/2001576.2001606.

[5] Nick Cheney et al. "Unshackling Evolution: Evolving Soft Robots with Multiple Materials and a Powerful Generative Encoding." In: GECCO 2013 - Proceedings of the 2013 Genetic and Evolutionary Computation Conference. Jan. 1, 2013. DOI: 10.1145/2463372.2463404.

[6] Joshua E. Auerbach and Josh C. Bongard. "Environmental Influence on the Evolution of Morphological Complexity in Machines." In: *PLOS Computational Biology* 10.1 (Jan. 2, 2014), e1003399. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1003399.

[7] Nick Cheney et al. *Scalable Co-Optimization of Morphology and Control in Embodied Machines*. Dec. 12, 2017. arXiv: 1706.06133[cs].

[8] David Ha. "Reinforcement Learning for Improving Agent Design." In: *Artificial Life* 25.4 (Nov. 2019), pp. 352–365. ISSN: 1064-5462, 1530-9185. DOI: 10.1162/artl_a_00301.

[9] Allan Zhao et al. “RoboGrammar: graph grammar for terrain-optimized robot design.” In: *ACM Transactions on Graphics* 39.6 (Dec. 31, 2020), pp. 1–16. ISSN: 0730-0301, 1557-7368. DOI: 10.1145/3414685.3417831.

[10] T. Fukuda et al. “Self Organizing Robots Based on Cell Structures - CEBOT.” In: *IEEE International Workshop on Intelligent Robots*. IEEE International Workshop on Intelligent Robots. Oct. 1988, pp. 145–150. DOI: 10.1109/IROS.1988.592421.

[11] Nicholas Cheney et al. “On the Difficulty of Co-Optimizing Morphology and Control in Evolved Virtual Creatures.” In: *Proceedings of the Artificial Life Conference 2016*. Proceedings of the Artificial Life Conference 2016. Cancun, Mexico: MIT Press, 2016, pp. 226–233. ISBN: 978-0-262-33936-0. DOI: 10.7551/978-0-262-33936-0-ch042.

[12] Matteo De Carlo et al. “Influences of Artificial Speciation on Morphological Robot Evolution.” In: *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. 2020 IEEE Symposium Series on Computational Intelligence (SSCI). Dec. 2020, pp. 2272–2279. DOI: 10.1109/SSCI47803.2020.9308433.

[13] Jørgen Nordmoen et al. “MAP-Elites Enables Powerful Stepping Stones and Diversity for Modular Robotics.” In: *Frontiers in Robotics and AI* 8 (2021). DOI: 10.3389/frobt.2021.639173.

[14] Karine Miras and A.E. Eiben. “The impact of environmental history on evolved robot properties.” In: *Proceedings of the ALIFE 2019: The 2019 Conference on Artificial Life*. ALIFE 2019: The 2019 Conference on Artificial Life. July 1, 2019, pp. 396–403. DOI: 10.1162/isal_a_00192.

[15] H. Maaranen, K. Miettinen, and M.M. Mäkelä. “Quasi-random initial population for genetic algorithms.” In: *Computers & Mathematics with Applications* 47.12 (June 2004), pp. 1885–1895. ISSN: 08981221. DOI: 10.1016/j.camwa.2003.07.011.

[16] Heikki Maaranen, Kaisa Miettinen, and Antti Penttinen. “On initial populations of a genetic algorithm for continuous optimization problems.” In: *Journal of Global Optimization* 37.3 (Jan. 23, 2007), pp. 405–436. ISSN: 0925-5001, 1573-2916. DOI: 10.1007/s10898-006-9056-6.

[17] Pedro A Diaz-Gomez and Dean F Hougen. “Initial population for genetic algorithms: A metric approach.” In: *Gem*. 2007, pp. 43–49.

[18] Silvia Poles, Yan Fu, and Enrico Rigoni. “The Effect of Initial Population Sampling on the Convergence of Multi-Objective Genetic Algorithms.” In: *Multiobjective Programming and Goal Programming*. Ed. by Vincent

Barichard et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 123–133. ISBN: 978-3-540-85646-7.

[19] Kenneth O. Stanley and Risto Miikkulainen. "Evolving Neural Networks through Augmenting Topologies." In: *Evolutionary Computation* 10.2 (June 2002), pp. 99–127. ISSN: 1063-6560, 1530-9304. DOI: 10.1162/106365602320169811.

[20] Pradnya A. Vikhar. "Evolutionary algorithms: A critical review and its future prospects." In: *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*. 2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC). Dec. 2016, pp. 261–265. DOI: 10.1109/ICGTSPICC.2016.7955308.

[21] Xinjie Yu and Mitsuo Gen. *Introduction to evolutionary algorithms*. Springer Science & Business Media, 2010.

[22] Ágoston E. Eiben and James E. Smith. "Evolutionary algorithms." In: *Handbook of memetic algorithms*. Springer, 2012, pp. 9–27.

[23] A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing*. Natural Computing Series. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015. ISBN: 978-3-662-44874-8. DOI: 10.1007/978-3-662-44874-8.

[24] Leila Kallel and Marc Schoenauer. "Alternative random initialization in genetic algorithms." In: *Proceedings of the 7 th International Conference on Genetic Algorithms*. Morgan Kaufmann, 1997, pp. 268–275.

[25] Kalyanmoy Deb. "Multi-Objective Optimization Using Evolutionary Algorithms: An Introduction." In: (Feb. 1, 2011).

[26] Hisao Ishibuchi, Tsutomu Doi, and Yusuke Nojima. "Incorporation of Scalarizing Fitness Functions into Evolutionary Multiobjective Optimization Algorithms." In: *Parallel Problem Solving from Nature - PPSN IX*. Ed. by Thomas Philip Runarsson et al. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2006, pp. 493–502. ISBN: 978-3-540-38991-0. DOI: 10.1007/11844297_50.

[27] Joel Lehman and Kenneth O. Stanley. "Abandoning Objectives: Evolution Through the Search for Novelty Alone." In: *Evolutionary Computation* 19.2 (June 2011), pp. 189–223. ISSN: 1063-6560, 1530-9304. DOI: 10.1162/EVCO_a_00025.

[28] Justin K. Pugh, Lisa B. Soros, and Kenneth O. Stanley. "Quality Diversity: A New Frontier for Evolutionary Computation." In: *Frontiers in Robotics and AI* 3 (2016). ISSN: 2296-9144.

[29] Jean-Baptiste Mouret and Jeff Clune. *Illuminating search spaces by mapping elites*. Apr. 19, 2015. DOI: 10.48550/arXiv.1504.04909. arXiv: 1504.04909[cs,q-bio].

[30] Brad L Miller and David E Goldberg. "Genetic Algorithms, Tournament Selection, and the Effects of Noise." In: *Complex systems* 9.3 (1995), pp. 193–212.

[31] Matej Črepinšek, Shih-Hsi Liu, and Marjan Mernik. "Exploration and exploitation in evolutionary algorithms: A survey." In: *ACM Computing Surveys* 45.3 (June 2013), pp. 1–33. ISSN: 0360-0300, 1557-7341. DOI: 10.1145/2480741.2480752.

[32] A. E. Eiben and C. A. Schippers. "On Evolutionary Exploration and Exploitation." In: *Fundamenta Informaticae* 35.1 (Jan. 1, 1998), pp. 35–50. ISSN: 0169-2968. DOI: 10.3233/FI-1998-35123403.

[33] Marcus Hutter. "Fitness uniform selection to preserve genetic diversity." In: Evolutionary Computation, 2002. June 12, 2002, pp. 783–788. ISBN: 978-0-7803-7282-5. DOI: 10.1109/CEC.2002.1007025.

[34] De Jong and Kenneth Alan. "Analysis of the behavior of a class of genetic adaptive systems." In: (1975).

[35] David E. Goldberg and Jon Richardson. "Genetic algorithms with sharing for multimodal function optimization." In: *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*. USA: L. Erlbaum Associates Inc., Oct. 1, 1987, pp. 41–49. ISBN: 978-0-8058-0158-3.

[36] Robert J. Collins and David R. Jefferson. "Selection in Massively Parallel Genetic Algorithms." In: *Proceedings of the Fourth International Conference on Genetic Algorithms*. Morgan Kaufmann, 1991, pp. 249–256.

[37] Greg Hornby. "ALPS: The age-layered population structure for reducing the problem of premature convergence." In: GECCO 2006 - Genetic and Evolutionary Computation Conference. Vol. 1. Jan. 1, 2006. DOI: 10.1145/1143997.1144142.

[38] Michael Schmidt and Hod Lipson. "Age-Fitness Pareto Optimization." In: *Genetic Programming Theory and Practice VIII*. Ed. by Rick Riolo, Trent McConaghy, and Ekaterina Vladislavleva. Genetic and Evolutionary Computation. New York, NY: Springer, 2011, pp. 129–146. ISBN: 978-1-4419-7747-2. DOI: 10.1007/978-1-4419-7747-2_8.

[39] Tobias Friedrich and Markus Wagner. "Seeding the initial population of multi-objective evolutionary algorithms: A computational study." In: *Applied Soft Computing* 33 (Aug. 2015), pp. 223–230. ISSN: 15684946. DOI: 10.1016/j.asoc.2015.04.043.

[40] Shahryar Rahnamayan, Hamid R. Tizhoosh, and Magdy M. A. Salama. "A novel population initialization method for accelerating evolutionary algorithms." In: *Computers & Mathematics with Applications* 53.10 (May 1, 2007), pp. 1605–1614. ISSN: 0898-1221. DOI: 10.1016/j.camwa.2006.07.013.

[41] J. D. Schaffer, D. Whitley, and L. J. Eshelman. "Combinations of genetic algorithms and neural networks: a survey of the state of the art." In: *COGANN-92: International Workshop on Combinations of Genetic Algorithms and Neural Networks*. June 6, 1992, pp. 1–37. DOI: 10.1109/COGANN.1992.273950.

[42] Dario Floreano, Peter Dürr, and Claudio Mattiussi. "Neuroevolution: from architectures to learning." In: *Evolutionary Intelligence* 1.1 (Mar. 2008), pp. 47–62. ISSN: 1864-5909, 1864-5917. DOI: 10.1007/s12065-007-0002-4.

[43] Alan McIntyre et al. *neat-python*.

[44] Reem J. Alattas, Sarosh Patel, and Tarek M. Sobh. "Evolutionary Modular Robotics: Survey and Analysis." In: *Journal of Intelligent & Robotic Systems* 95.3 (Sept. 2019), pp. 815–828. ISSN: 0921-0296, 1573-0409. DOI: 10.1007/s10846-018-0902-9.

[45] M. Yim, D.G. Duff, and K.D. Roufas. "PolyBot: a modular reconfigurable robot." In: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings*. Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings. Vol. 1. Apr. 2000, 514–520 vol.1. DOI: 10.1109/ROBOT.2000.844106.

[46] Jordan B. Pollack et al. "Chapter 21 - Evolutionary Techniques in Physical Robotics." In: *Creative Evolutionary Systems*. Ed. by Peter J. Bentley and David W. Corne. The Morgan Kaufmann Series in Artificial Intelligence. San Francisco: Morgan Kaufmann, 2002, pp. 511–IX. ISBN: 978-1-55860-673-9. DOI: https://doi.org/10.1016/B978-155860673-9/50061-6.

[47] Joshua E Auerbach and Josh C Bongard. *Evolving complete robots with CPPN-NEAT: the utility of recurrent connections*. July 12, 2011.

[48] Karine Miras, Eliseo Ferrante, and A. E. Eiben. "Environmental influences on evolvable robots." In: *PLOS ONE* 15.5 (May 29, 2020), e0233848. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0233848.

[49] Jörg Conradt and Paulina Varshavskaya. "Distributed Central Pattern Generator Control for a Serpentine Robot." In: *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*. 2003, pp. 338–341.

[50] Frank Veenstra et al. "Evolution and Morphogenesis of Simulated Modular Robots: A Comparison Between a Direct and Generative Encoding." In: *Applications of Evolutionary Computation*. Ed. by Giovanni Squillero and Kevin Sim. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2017, pp. 870–885. ISBN: 978-3-319-55849-3. DOI: 10.1007/978-3-319-55849-3_56.

[51] Jørgen Nordmoen et al. *Quality and Diversity in Evolutionary Modular Robotics*. Aug. 5, 2020. arXiv: 2008.02116[cs].

[52] D.J. Christensen. "Evolution of shape-changing and self-repairing control for the ATRON self-reconfigurable robot." In: *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006. May 2006, pp. 2539–2545. DOI: 10.1109/ROBOT.2006.1642084.

[53] Deepak Pathak et al. *Learning to Control Self-Assembling Morphologies: A Study of Generalization via Modularity*. Nov. 21, 2019. arXiv: 1902.05546[cs,stat].

[54] Mia-Katrin Kvalsund, Kyrre Glette, and Frank Veenstra. *Centralized and Decentralized Control in Modular Robots and Their Effect on Morphology*. June 27, 2022. arXiv: 2206.13366[cs].

[55] Stephane Doncieux et al. "Evolutionary Robotics: What, Why, and Where to." In: *Frontiers in Robotics and AI* 2 (2015). ISSN: 2296-9144.

[56] Josh C. Bongard. "Evolutionary robotics." In: *Communications of the ACM* 56.8 (Aug. 2013), pp. 74–83. ISSN: 0001-0782, 1557-7317. DOI: 10.1145/2493883.

[57] T. Geijtenbeek and N. Pronost. "Interactive Character Animation Using Simulated Physics: A State-of-the-Art Review." In: *Computer Graphics Forum* 31.8 (Dec. 2012), pp. 2492–2515. ISSN: 01677055. DOI: 10.1111/j.1467-8659.2012.03189.x.

[58] Tønnes Nygaard, Eivind Samuelsen, and Kyrre Glette. "Overcoming Initial Convergence in Multi-objective Evolution of Robot Control and Morphology Using a Two-Phase Approach." In: *European Conference on the Applications of Evolutionary Computation*. European Conference on the Applications of Evolutionary Computation. Mar. 25, 2017, pp. 825–836. ISBN: 978-3-319-55848-6. DOI: 10.1007/978-3-319-55849-3_53.

[59] Charles Schaff et al. *Jointly Learning to Construct and Control Agents using Deep Reinforcement Learning*. Sept. 14, 2018. arXiv: 1801.01432[cs].

[60] Frank Veenstra and Kyrre Glette. "How Different Encodings Affect Performance and Diversification when Evolving the Morphology and Control of 2D Virtual Creatures." In: *The 2020 Conference on Artificial Life*. The 2020 Conference on Artificial Life. Online: MIT Press, 2020, pp. 592–601. DOI: 10.1162/isal_a_00295.

[61] Emma Hjellbrekke Stensby, Kai Olav Ellefsen, and Kyrre Glette. *Co-optimising Robot Morphology and Controller in a Simulated Open-Ended Environment*. Apr. 7, 2021. DOI: 10.1007/978-3-030-72699-7_3. arXiv: 2104.03062[cs].

[62] Karine Miras and A. E. Eiben. "How the History of Changing Environments Affects Traits of Evolvable Robot Populations." In: *Artificial Life* 28.2 (June 28, 2022), pp. 224–239. ISSN: 1064-5462. DOI: 10.1162/artl_a_00379.

[63] Hod Lipson and Jordan B. Pollack. "Automatic design and manufacture of robotic lifeforms." In: *Nature* 406.6799 (Aug. 2000), pp. 974–978. ISSN: 1476-4687. DOI: 10.1038/35023115.

[64] Arthur Juliani et al. "Unity: A general platform for intelligent agents." In: *arXiv preprint arXiv:1809.02627* (2020).

[65] Félix-Antoine Fortin et al. "DEAP: Evolutionary Algorithms Made Easy." In: *Journal of Machine Learning Research* 13 (July 2012), pp. 2171–2175.

[66] A. Crespi and A.J. Ijspeert. "AmphiBot II: An Amphibious Snake Robot that Crawls and Swims using a Central Pattern Generator." In: *Proceedings of the 9th International Conference on Climbing and Walking Robots (CLAWAR 2006)* (Jan. 1, 2006).

[67] F. Delcomyn. "Walking in Invertebrates." In: *Encyclopedia of Neuroscience*. Ed. by Larry R. Squire. Oxford: Academic Press, Jan. 1, 2009, pp. 479–484. ISBN: 978-0-08-045046-9. DOI: 10.1016/B978-008045046-9.01977-X.

[68] John R. Finnerty. “Did internal transport, rather than directed locomotion, favor the evolution of bilateral symmetry in animals?” In: *BioEssays* 27.11 (2005), pp. 1174–1180. ISSN: 1521-1878. DOI: 10.1002/bies.20299.

[69] Eivind Samuelsen and Kyrre Glette. “Some distance measures for morphological diversification in generative evolutionary robotics.” In: *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*. GECCO ’14: Genetic and Evolutionary Computation Conference. Vancouver BC Canada: ACM, July 12, 2014, pp. 721–728. ISBN: 978-1-4503-2662-9. DOI: 10.1145/2576768.2598325.