Co-evolving morphology and controller in an open-ended environment

Emma Hjellbrekke Stensby

Thesis submitted for the degree of
Master in Robotics and Intelligent Systems
60 credits

Department of Informatics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Spring 2020

Co-evolving morphology and controller in an open-ended environment

Emma Hjellbrekke Stensby

Abstract

Designing robots by hand is often both costly and time consuming. In order to create robots automatically, without the need for human intervention, it is necessary to optimise both the behaviour and the body design of the robot. However, when co-optimising the morphology and controller of a locomoting agent the morphology tends to converge prematurely, reaching a local optimum. Approaches such as explicit protection of morphological innovation have been used to reduce this problem, but it might also be possible to increase the exploration of morphologies using a more indirect approach.

We explore how changing the environment the agent locomotes in affects the convergence of morphologies. Inspired by POET, an algorithm which evolves environments open-endedly, we create POET-M, an expansion of POET which includes evolution of morphologies. We compare the morphological change and diversity of agents evolving in a static environment, a curriculum of hand crafted environments, and in POET-M.

We show that the agents experience increased morphological change in response to environmental change, and that agents evolving in an open-ended environment exhibit larger morphological diversity in the population than agents evolving in a static flat environment or a hand crafted curriculum of environments. POET-M proved capable of creating a curricula of environments which encouraged both diversity and quality in the population. This might suggest that the open-endedly evolving environments in POET-M act as stepping stones for the agents, enabling the morphology to escape local optima and continue evolving past the early stages of the evolution.

Acknowledgments

I would like to thank my supervisor, Kyrre Glette, for guidance and encouragement, and my fellow students for making IFI a great workplace. I would also like to thank my family for generous support during the last three months working from home.

Parts of the experiments were performed on UNINETT Sigma2 - the National Infrastructure for High Performance Computing and Data Storage in Norway.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

When creating a robot's morphology it is common to mimic the body structure of creatures found in nature. Bipedal robots inspired by humans [1], [2], quadruped dog-like robots [3] and hexapedal robots similar to insects [4] are examples of such bioinspired robots. It is also possible to take inspiration, not from the animals directly, but from the evolutionary process that created them.

In 1994 Karl Sims published a study, "Evolving Virtual Creatures" [5], which showed virtual creatures evolving in an artificial world with simulated physics. In this artificial evolution the creatures evolved both their bodies and their behaviours simultaneously, and solved various tasks such as walking, swimming and competing against each other. Inspired by Sims' work many researchers took interest in creating robots that can evolve both their morphology and controller at the same time [6]–[10], and the field of co-evolution of robot morphology and controller emerged.

An ambitious goal in robotics is to create robots that find good body designs, produce themselves and learn to solve arbitrary tasks, without the need for human intervention. Reaching this goal would remove the costly design process of creating body designs by hand, and the designs reached could potentially enable the robots to reach higher performance than what is possible with human made designs. Lipson et al. [11] attempted to create a system that produces robots with as little human assistance as possible. Their algorithm evolve a robot's morphology and controller, and automatically 3D-print the robot. The only thing a human needs to do is to snap motors in place, in slots in the 3D-printed body, and the robots are ready to locomote. However, these robots have many limitations, and we are far from reaching the goal of automatic design. There are many challenges yet to be solved in this field. Finding efficient designs, and creating a diversity of useful and robust behaviours, is a difficult task.

Although computing power has increased significantly since Karl Sims' study was published, the morphological complexity of evolved agents has not increased as much as could be expected [12]. Deficiencies in the morphology encodings [13], deficiencies in the diversity maintenance of search algorithms [14], and that the environments used are not complex enough to encourage complex morphologies [15], have been suggested as sources for this problem. In 2012 Cheney et al.[7] proposed their theory on why it was so difficult to make further progress. When co-evolving morphology and controller, the morphology would often converge prematurely. Cheney et al. proposed that this is due to an effect called embodied cognition.

Embodied cognition is a theory stating that how a creature behaves is influenced heavily by their body. The body can be seen as an interface between the controller and the environment, and if the body changes, even just a little, it is as if the interface between body and controller has been scrambled. The controller adapts to its specific morphology, and when the morphology changes, the controller will have to re-adapt before it can manage to locomote with the new body.

Cheney et al. [8] continued their research, and studied how explicitly protecting individuals, that had just experienced morphological change, affected the evolution of the morphologies. They showed that when giving the controllers time to adapt to their new bodies, the new morphologies would overtake the old ones, and the algorithm would continue to explore new morphologies.

Several algorithms that reduce the problem of embodied cognition, without explicitly protecting novel morphologies, have also been proposed. One such algorithm is ALPS [9], where mating is only allowed between candidates that have experienced approximately the same number of earlier reproduction steps. This restriction divides the population into layers based on their age, and lowers the selection pressure on young candidates. Jelisavcic et al. [10], also take a more indirect approach to protecting new morphologies. In their work all controllers adapt to their morphologies, before being evaluated, through lamarckian evolution. Lehman et al. [14] do not allow the controllers time to adapt to their morphologies, but rather increase morphological diversity by optimising for morphological novelty in addition to performance with a multi objective evolutionary algorithm.

In traditional evolutionary algorithms it is common to optimise for better performance, but this approach can easily lead the algorithm to converge to a local optima prematurely. One way to increase the chance of finding good optima is to increase diversity in the population, with methods such as fitness sharing [16], [17], speciation [18], crowding [19] or novelty search [20].

In the field of open-endedness, the focus is not to move towards solutions with better performance, but to create novel and interesting solutions [21], often by optimising for diversity instead of performance. Counterintuitively, searching for novelty alone can sometimes find better solutions than what can be found by optimising directly for performance, as demonstrated by Lehman et al. [20].

Inspired by minimal criterion co-evolution [22], Wang et al. invented POET [23]. In POET environments evolve open-endedly, while agents are optimised to locomote within them. A minimal criterion ensures that the environments are appropriately difficult for the agents, increasing in complexity as the agents learn to walk more efficiently. Wang et al. show that the many environments are used as stepping stones, enabling the agents to learn new skills, and escape local optima.

We create POET-M, where agent morphologies are co-evolved with their controllers. We use this to explore whether the stepping stone environments, that enabled agent controllers to escape local optima in POET, will also enable agent morphologies to escape local optima in POET-M.

1.2 Research Questions

When co-evolving morphology and controller of a locomoting agent, the morphology tends to converge to a local optima prematurely. Our hypothesis is: "When co-evolving the morphology and controller of locomoting agents, evolving the agents in open-endedly evolving environments can prevent premature convergence of morphologies." From this hypothesis we create three research questions which we attempt to answer:

- How does changes in environments affect the morphological development of locomoting agents?
- What impact does increasing the difficulty of the environments, as the agent solves them, have on the morphological development of locomoting agents?
- How effective is POET-M in maintaining both quality and morphological diversity in a population, compared to populations evolved in a static environment or a hand crafted curriculum?

We perform one experiment for each of the three research questions.

1.3 Contributions

Our main contribution is showing that agents that are evolved in environments which are evolving open-endedly with POET-M, maintain larger morphological diversity in the population, compared to agents evolved in static environments, and in hand crafted curricula of environments. This may suggest that open-ended evolution of the environments decreases the problem of premature convergence of morphology in embodied agents.

1.4 Thesis structure

This thesis consists of five chapters: 1. Introduction, 2. Background, 3. Implementation, 4. Experiments and Results and 5. Conclusion and Future Work.

The background chapter presents relevant research and related work. We focus on four fields: Evolutionary algorithms, Evolutionary robotics, Neuroevolution and Open-endedness. We also describe the POET [23] algorithm, which our algorithm builds upon.

The implementation chapter describes the environment we use to test our algorithm, Bipedal Walker Hardcore, the evolutionary algorithm we use to evolve agents and environments, POET-M, and the hand crafted curricula of environments used in some of the experiments.

The experiments and results chapter presents three experiments, each tied to one of our three research questions presented in section 1.2, and the results and findings of each experiment.

The conclusion chapter summarises what we have done, and discusses possible future work.

Chapter 2

Background

2.1 Evolutionary algorithms

Evolutionary algorithms [24] are a type of meta heuristic search algorithm inspired by evolution found in nature. Solutions to a problem is found through trial and error, and information about previously tested solutions is used to decide which new solutions to test.

There are several variations of evolutionary algorithms. We focus on the genetic algorithm, which we use in our experiments, but we will also briefly look at evolutionary strategies, which is used by Wang et al. in POET [23]. A large challenge in evolutionary algorithms is the balance between searching for new solutions and exploiting good solution that have already been found. In section 2.1.3 we look at techniques to achieve this balance.

2.1.1 Genetic algorithms

The genetic algorithm [25] is a branch of evolutionary algorithms where solutions are represented as genomes, similarly to how DNA encodes creatures in nature. Animals and plants have evolved, from simple cells to complex beings, through *reproduction* and *mutation*. Similarly, in the genetic algorithm, a population of potential solutions is created, and the search is performed by creating new combinations of the previous individuals' genomes, and applying small random changes to them. Both in nature and in genetic algorithms there are not enough resources to keep all individuals alive. In nature, and often in genetic algorithms, the best performing individuals are more likely to survive and reproduce. This creates a selection pressure that drives the population as a whole towards better solutions.

The flow of a genetic algorithm can be described by these steps:

1. Create initial candidate solutions.
2. Evaluate the candidates with a *fitness function*, to rank their performance.

3. Choose which candidates get to become *parents* and reproduce.

4. *Recombine* the parents to create new solutions, the *children*.

5. *Mutate* the children.

6. Choose which candidates will *survive*, and proceed to the next generation.

7. Repeat step 2 - 5 until a termination condition is met.

The steps above will be described in detail below.

Creating the candidates

The most common way to create the initial candidates is by drawing random samples from the search space. They can be drawn from a uniform distribution, or from another distribution of choice. Another way to choose the initial candidates is by spreading them out over the search space to cover area. This can for example be done by dividing the search space into a grid, and drawing one sample from each square.

The fitness function

The fitness function evaluates how well a candidate solution performs a task, and gives the candidate a fitness score based on how well it did. The candidates behaviour is often evaluated by simulating it in a task environment. If the task of the evolution is to create gaits for a robot, a fitness function could be how far the robot moves within a time frame. For evolution that optimises the shape of an antenna the fitness function could be the strength of the emitted signal. The fitness function has to be designed specifically for the problem that is being solved. When all individuals in the population have been evaluated by the fitness function and given a score, the fitness score can by used to easily compare the individuals' performance.

Choosing the parents

When all the individuals in the population have been assigned a fitness value by the fitness function, a subset of the population is chosen to become parents. Parts of the parents' genomes will be passed on to the new individuals that are created in the recombination phase. A common parent selection mechanisms is to choose the parents randomly, either with equal probability to choose each individual, or with a probability decided by their fitness. Another common parent selection mechanism is to hold tournaments, in which a set number of individuals are chosen at random, and the best among them becomes a parent.

Recombination

In the recombination phase children are created. The children's genomes are created as combinations of the genomes of two or more parents. Some common recombination operators are uniform crossover, k-point crossover, and partially mapped crossover. In uniform crossover each value in the genome is chosen from a random parent. In k-point crossover the genomes are divided into sections, and each section is chosen from a random parent. Partially mapped crossover can be used if you need individuals with non-repeating values.

Mutation

In the mutation phase the children that have been created in the recombination phase are modified. Through mutations, values that did not appear in any of the population's initial genomes can be explored. Common mutation operators involve randomly replacing or modifying values, or switching the positions of two or more values within the genome.

Survivor selection

When the children have been created the individuals that survive and move on to the next generation are chosen. Two common selection schemes is to either select only the children, or select the individuals with highest fitness. Selecting only the children will increase exploration of new solutions, while selecting the individuals with highest fitness will increase exploitation of the good candidates.

2.1.2 Evolutionary strategies

Evolutionary strategies [26] were first developed to solve real-valued optimisation problems, such as design problems. The general evolutionary strategy has a population of parameter vectors, and a fitness function measuring their performance. The flow of the algorithm is similar to the genetic algorithm with parents creating offspring, and selection of survivors for the next generation.

When creating children in the general evolutionary strategy all individuals are chosen as parents, and each of them creates one child through asexual reproduction. An individual is mutated to create a child by adding an offset, containing Gaussian noise, to the parameter vector. After all individuals have reproduced the population size has doubled. The half of the population with highest fitness is chosen as survivors for the next generation.

CMA-ES

Some variations of evolutionary strategies do not evolve a population directly, such as Covariance Matrix Adaption - Evolutionary Strategy [27], CMA-ES. Rather, it evolves a distribution from which candidate solutions are sampled.

For each generation the algorithm draws several candidates from the distribution. The mean of the distribution is then recalculated, and a new distribution is created. The new mean is calculated as a weighted average of the candidates, where each candidate is given weight equal to its fitness. A normal distribution is often used.

2.1.3 Exploration vs. exploitation

When using evolutionary algorithms to search a solution space it is important to consider the balance between creating novel solutions that explore new ways of solving the problem, and converging toward better solutions to improve the fitness of the candidates. With too much focus on creating the best solution, the algorithm will quickly converge to a local optima. With too much focus on exploration the candidates will take too much time to reach the peaks of the search space while exploring useless candidates. A balance between the two is difficult to achieve, especially in a rugged search space, where the algorithms can easily be deceived by a local optima. In this section we look at some techniques besides increasing mutation, that can be used to increase the exploration of candidates.

Niching

In evolutionary algorithms the whole population tends to converge towards a single solution, as the algorithm converges towards an optima. In niching the population is divided in some way, and candidates can only recombine with candidates from the same division. This causes the groups of candidates to collect on different peaks, and the algorithm can find multiple solutions in one run, as illustrated in figure 2.1.

Reaching more than one peak can lead to better performance. When a larger number of peaks are explored, the algorithm's chance of finding the global optima increases.

The population can be divided into groups in different ways. One method can be to simply separate the population into smaller sub populations, and evolve them separately. Individuals can sometimes be moved between the niches, to share information, which is called island hopping.

Crowding

There are also more indirect approaches that encourages the populations to form niches on its own. An example of such an approach is crowding [19].

When using crowding in an evolutionary algorithm, children have to compete against the already existing candidates for a spot in the population. Each child only competes with one other candidate. To choose the candidate that the child will compete against, w candidates are drawn randomly from the population. From the w candidates, the child competes

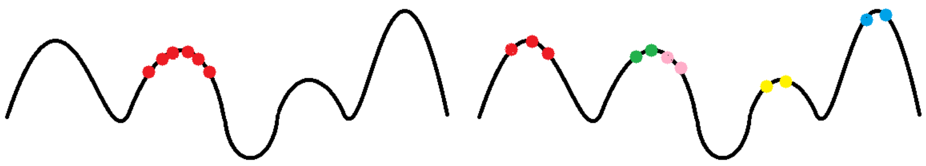


Figure 2.1: In the left picture all candidates collect at one peak. In the right picture the candidates are separated into groups that can collect at different peaks, as in niching.

against the candidate that is the most similar to itself. This causes fit candidates to only increase selection pressure for similar candidates, which causes the population to naturally crowd at different peaks.

Fitness sharing

Another method to decrease selection pressure and encourage population diversity is fitness sharing [17], where an individual's fitness is degraded through a sharing function. The theory behind this is that individuals in nature have to compete for resources with other similar individuals, while very different individuals find resources in different places. The sharing function in fitness sharing decreases an individual's fitness based on how many similar candidates are found in the population, creating an effect where the similar candidates get lower fitness due to having to share their resources with many others.

2.2 Neuroevolution

Artificial neural networks have been used to approximate solutions to various problems for a long time. A common way to find the network weights is through backpropagation [28]. Neuroevolution is the process of searching for optimal weights or topology of a neural network with evolutionary algorithms [29], and is an alternative to gradient descent based approaches when optimizing neural networks.

Neuroevolution can be effective when dealing with a complex search space due to its ability to explore a wide variety of solutions. Some common neuroevolution algorithms are conventional neuroevolution, NEAT and HyperNEAT.

2.2.1 Conventional neuroevolution

Conventional neuroevolution [30] is a simple form of neuroevolution. The algorithm searches for network weights while the topology remains fixed. Common evolutionary algorithm operators such as crossover, mutation and speciation are applied.

2.2.2 NEAT

NeuroEvolution of Augmenting Topologies [31], NEAT, is an extension of common neuroevolution to include the evolution of network topologies. The evolved topologies start simple, and gradually become more complex as new weights and nodes are added through mutations.

To make crossover between different topologies possible the genes are tracked with innovation numbers. Each time a new node or weight is created it is given a unique number. This number is inherited along with the weight in the recombination phase. When recombining the parents the genes from the parents are separated into two categories. The first category is the genes that exist in all parents. A gene exists in all parents if a gene with the same innovation number exists in all parents. The value of a weight can be different for all the parents, even if it has the same innovation number. For this category of shared weights the value of each weight is inherited from a random parent. The second category is the genes that do not exist in all parents. From this category only the genes from the most fit parent are inherited.

2.2.3 HyperNEAT

HyperNEAT [32] does not evolve neural networks directly, but evolves Compositional Pattern-Producing Networks [33], CPPNs, which encode neural networks. The CPPN uses a combination of functions to approximate the values of the network, and is evolved with NEAT. The CPPN takes the positions of two nodes as input, and outputs the weight between them. Because the CPPN knows the positions of the nodes in the network it is said to be able to take geometric relations into account.

2.3 Evolutionary robotics

Evolutionary robotics is the method of applying evolutionary algorithms to automatically generate the morphology or controller for robotic systems. In evolution of robot morphology the whole shape of the robot can be evolved, such as in soft robots [34] or modular robots [35]. However, it is also possible to only evolve small parts of the morphology, such as the length of the robots legs [3], [6], [36], [37], or the placements of the robots sensors [38], [39].

A common problem when evolving both morphologies and controllers for robots at the same time is premature convergence of morphologies [7]. Section 2.3.1 will present research on difficulties in co-evolution of morphology and controller, while section 2.3.2 will present research on how to overcome these challenges. Section 2.3.3 looks at research on environmental effects on robot morphologies.

2.3.1 Difficulties in co-evolution of robot morphologies and controller

Cheney et al. [7] suggest that there is a fundamental problem in co-evolving robot morphology and controller, causing premature convergence of morphologies. They hypothesise that changing the morphology scrambles the interface between the controller and the body, due to robots being embodied agents. This embodiment would create an effect where control optimization on an existing morphology should be more effective than morphological optimisation on a fixed controller.

To test this they conduct an experiment where they co-evolve the controller and morphology of robots. They freeze either the morphology or the controller at various points in the training, and continue training the other part. They observe that freezing the controller early has a bigger negative impact on final fitness than freezing the morphology early, and conclude that this supports their hypothesis. They suggest that when co-evolving morphologies and controllers it is necessary to protect candidates after a morphological change so that the controller gets a chance to adapt to it's new body.

2.3.2 Methods in co-evolution of robot morphologies and controller

Scalable co-optimization of morphology and control in embodied machines

Cheney et al. [8] continue their work on co-evolution of morphology and controller, and present a technique they call morphological innovation protection. Because robot controllers and morphologies are co-dependant a robot's fitness drops if either of them is changed while not allowing the other time to adapt to the changes. Protecting individuals that have recently experienced a big change, through keeping them in the population even if they are performing poorly, allows the morphologies and controllers to adapt to each other. When allowed time to adapt, an individual which experienced large changes can potentially reach a higher fitness than the individuals which did not experience large changes.

In morphological innovation protection innovation is protected by tracking the age of each individual. An individuals age is the number of generations that has passed since an individual last experienced a large change. An individual will only be removed from the population if there exists an individual that has both higher fitness and lower age than it. Selection pressure is reduced for candidates that recently experienced large morphological change, as they only compete with candidates that are younger than themselves. Their results show that when using morphological innovation protection the robots are more diverse, and reach higher performance than when innovation is not protected.

To test whether protecting innovations in the controller has the same effect on the training as protecting morphologies, they compare the two approaches. They find that the algorithm performs significantly better when morphologies are protected, compared to when controllers are protected. This result show an asymmetry in the difficulty of evolving morphologies and controllers, and supports the theory of embodied cognition in robots.

Reinforcement Learning for Improving Agent Design

Ha et al. [6] evolve agents locomoting in the bipedal walker and ant environments found in OpenAI Gym [40]. They evolve both the controller and the morphology of the agents. The agents are optimised with the policy gradient algorithm REINFORCE [41]. They show that the agents achieve higher fitness, and solve the environment faster, when allowed to change their morphology in both the bipedal walker and the ant environment.

They also evolve the agents while using a modified fitness function, which gives the agents extra reward for having smaller legs. The agents learn to walk efficiently even with tiny legs. The agents evolved larger legs for difficult environments than they did for easy environments, finding the smallest legs that were still capable of tackling the challenging environments.

Lamarckian Evolution of Simulated Modular Robots

Jelisavcic et al. [10] co-evolve the morphology and controller of modular robots. In addition to the change the robots experience through reproduction, they can optimise their control algorithm as they are being evaluated. This on-line optimisation is called *lifetime learning*.

Jelisavcic et al. look at the difference between baldwinian and lamarckian evolution, and perform one experiment for each of the two evolution schemes. In the first experiment, where baldwinian evolution is used, the child candidates inherit the weights that their parent had before undergoing lifetime learning. In the second experiment, which uses lamarckian evolution, the children inherit the resulting parent weights from after the lifetime learning.

Jelisavcic et al. observe that in most cases the robots evolved with lamarckian evolution performed better than the robots evolved with baldwinian evolution, especially in the earlier generations of the training. If they let the experiments run for many generations the baldwinian robots tended to catch up to and eventually surpass the lamarckian robots. They therefore conclude that lamarckian evolution is only better than baldwinian evolution if low computational cost is a requirement.

The robots evolved with baldwinian evolution covered more of the morphology search space compared to the robots evolved with lamarckian evolution, suggesting that exploration of morphologies is larger when using baldwinian evolution.

ALPS: the Age-Layered Population Structure for reducing the problem of premature convergence

Hornby et al. [9] propose using an age-layered population to slow down convergence in evolutionary algorithms. The population is divided into layers, where all candidates in a layer has the same age. The first layer contains new randomly initialised agents with age 0. After reproduction the age of a child is set to the age of its oldest parent plus one. Thus the age describes not the age of the individual, but the age of the genetic material it contains. Candidates can only mate with other candidates that are in their own layer or in a neighbouring layer.

This layering reduces selection pressure on young candidates, as they only compete with candidates of similar age. The selection pressure increases through the layers, and the elitist selection scheme ensures high selection pressure in the oldest layers. As young candidates discover new and better optima, than those found by the older population, they move up the layers until they replace the old candidates.

Hornby et al. compared their algorithm to a traditional evolutionary algorithm, a multi-start evolutionary algorithm, and two evolutionary algorithms with diversity maintenance schemes. They found that ALPS produced better solutions than all of them, with higher reliability.

2.3.3 Environmental effects on robot morphology

Environmental influence on the evolution of morphological complexity in machines

Biological creatures have increased tremendously in complexity over time, from single cells to giant organisms. However the reason for the increased complexity is still unknown. Auerbach et al. [15] explore how environmental complexity affects the morphological complexity of evolved agents. They evolve the morphology of their agents as three dimensional meshes. The genetic encoding produces regular patterns, which tends to produce symmetric phenotypes.

The environments in which the agents evolve are flat, high friction surfaces, which contain varying amounts of low friction "ice" blocks. Within one environment the frequency of the ice blocks is constant. A cost for morphological complexity was included in the fitness function, encouraging the agents to evolve the least complex form that was capable of solving the environment. When this cost was introduced the agents locomoting in complex environments evolved more complex forms than the agents locomoting in simple environments.

Effects of environmental conditions on evolved robot morphologies and behavior

Miras et al. [42] evolve morphologies and controllers for modular robots, and try to observe how changes in the environments affects the robot morphologies and gaits. The robots are evolved in one of three environments: a flat surface, a surface with small poles or a tilted surface. They observe that the robots evolve mostly rowing gaits for the tilted plain, and rolling gaits for the other two environments. The robots evolved in the tilted plains tend to be small with several limbs, while the robots evolved in the two other environments tend to evolve larger robots with fewer limbs.

2.4 Open-endedness

In nature a large variety of creatures exist, and continually evolve, creating new diverse creatures. The creatures behave and survive in very different ways, each species evolving to become efficient within their niche.

Evolutionary algorithms do not manage to create a diversity similar to the one found in nature, and usually converge to find a single solution to a problem. Inspired by the diversity in nature researchers have sought to create algorithms that create a large variety of solutions, continually creating new interesting ways to solve a problem. This type of algorithm, generating new innovative solutions, are often called open-ended algorithms.

2.4.1 Novelty search

Evolutionary algorithms are usually optimized for a predefined objective, but rewarding solutions that have come the closest to the objective, can often cause the algorithm to be deceived by a local optima. This kind of local optima, can be illustrated in a labyrinth, such as the one shown in figure 2.2. A simple path going straight forward will lead the agent very close to the goal, but this path is deceptive as there is a wall blocking the path to the exit of the labyrinth. To solve the labyrinth, a complex path, that might lead away from the goal at times, will have to be chosen.

Lehman et al. [20] propose an algorithm, novelty search, that does not optimise for higher fitness. Rather, it searches for solutions that are as different as possible from previously found solutions, by optimizing for novelty in the behaviour space. To calculate the novelty of a solution it is compared to an archive of all previously explored solutions.

They showed that by searching for novelty alone, the deceptive labyrinth can be solved more efficiently than by regular evolutionary search. Their algorithm also managed to solve complex labyrinths where regular evolution did not reach a solution at all. A disadvantage to novelty search is that it can be difficult to create a method for comparing solutions based on novelty. It can be difficult to evaluate how different two behaviours are, and comparing the genomes that create the behaviours is not always sufficient, as many genomes can lead to similar behaviours.

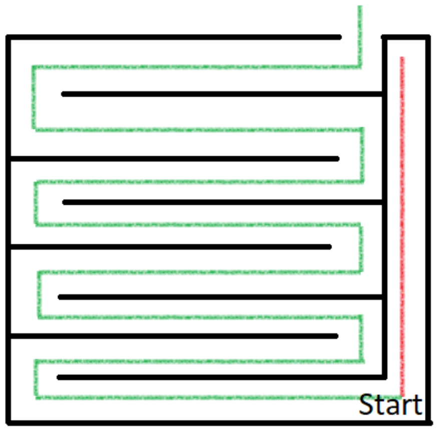


Figure 2.2: A labyrinth. A local optima can easily deceive the algorithm solving this labyrinth, as going along the red line can yield a very high fitness.

2.4.2 Quality-diversity search

Novelty search with local competition

Novelty Search with Local Competition [14], NSLC, is a multi objective algorithm that optimises for both novelty and performance. In problems with many incompetent solutions, such as robot morphology optimisation, searching for novelty alone can fail due to spending too much time exploring the large amount of incompetent variations. In these cases, optimising for both novelty and quality, at the same time, can help reach good solutions quicker.

NSLC has two fitness functions, one measuring performance, and one measuring novelty. A pareto-front with candidates that balance the trade off between novelty and quality differently is created. The novelty of an individual is measured similarly to how it is measured in the Novelty Search algorithm, by comparing it to an archive of all previous solutions. By including novelty as a separate objective, selection pressure is reduced on individuals with innovative solutions, while the individuals in well explored sections of the search space will need to rely on performance to survive.

Although adding novelty as a multi objective will increase the diversity of explored solutions, while still optimising for performance, it does not create a wide range of interesting behaviours. To evolve a diversity of different behaviours at the same time, Lehman et al. divide the population into niches, by only comparing an individuals performance to the performance of similar individuals. This reduces the selection pressure within niches where the performance potential is low.

MAP-elites

Multidimensional Archive of Phenotypic elites [43], MAP-elites, is an illumination algorithm, which attempts to visualise regions of high and low fitness in a search space.

The search space of a problem is often high-dimensional, so to visualise it the user chooses a set of features to be explored. For optimisation of robot morphologies the features could be the height and weight of the robot. A low-dimensional map of the features is created and divided into a grid of cells. The algorithm then searches for the best performing individual within each cell. This results in a map that visualises the fitness potential for different combinations of the features, and shows what areas of the feature space are the most interesting.

Mouret et al. compared their algorithm to random sampling of the search space, a regular evolutionary algorithm and NSLC. The algorithms were tested in three problem domains, evolution of neural networks, soft robots, and robot arm control. They showed that MAP-elites tended to perform better than all the comparison algorithms in all the problem domains, both with regard to final fitness, and with regard to exploration and illumination of the search space.

2.4.3 Minimal criterion co-evolution

Both Novelty search and NSLC require a novelty measure, and an archive containing all previously explored solutions. Novelty measures can be difficult to create, and archives can become very large as many behaviours are explored. In 2017 Brant et al. proposed minimal criterion co-evolution [22], a new type of open-ended evolution that does not have these two weaknesses.

Minimal criterion co-evolution co-evolves two interacting populations. Each of the two populations have a minimal criterion, and individuals that do not meet their population's criterion are not allowed to reproduce. In their experiments, Brant et al. test their algorithm by evolving mazes and maze-solvers. The minimal criterion for the mazes is that at least one of the maze-solvers have to be capable of solving the maze. The minimal criterion for the maze-solvers is that they have to be able to solve at least one of the mazes. To ensure that the all the mazes do not become variations of the same maze the population is divided into species based on similarity of the genomes. The parents that get to reproduce are chosen proportionally from all species.

This co-evolution evolves two diverse populations of problems and their solutions. The problems and solutions get more complex as the generations pass, thus generating endless interesting problems and their solutions.

2.5 POET: Paired Open-Ended Trailblazer

The Paired Open-Ended Trailblazer, POET, is an open-ended algorithm created by Wang et al. [23]. We build upon their work, and create a modified version of this algorithm, POET-M, which we use in our experiments. This section will describe the original POET, while our version POET-M is described in section 3.3, in the implementation chapter.

POET has a population of pairs, where each pair consists of one environment and one agent. The agents are optimized within their paired environment, and the environments are evolved with an open-ended algorithm optimizing for novelty. Wang et al. take experience from minimal criterion co-evolution, subjecting the environments to a minimal criterion that has to be fulfilled in order to be allowed to reproduce, and a minimal criterion to be allowed into the population. The minimal criterion regulates how difficult the environment is for the paired agent, and is checked by seeing if the agents fitness is within a range set by the user. The environments are pushed to increase in complexity as the agents get better at solving them. As the environments increase in complexity, the agents learn increasingly complex behaviours.

Wang et al. tested their algorithm in the OpenAI bipedal walker environment, and observed that the agents used the environments as stepping stones to learn behaviours and gaits they would otherwise not find. The pairs share their knowledge helping each other escape local optima.
The main execution loop of POET, shown in figure 2.3, has three main steps:

- Creating environments
- Optimizing agents
- Transfering agents between environments

Each of the three steps will now be described in detail.

2.5.1 Creating environments

When POET is initialised one initial pair is created. The pair consists of a very simple environment, and an agent chosen randomly from the agent search space. The agent is optimised within the first environment until it reaches the minimal criterion for reproduction. After the first pair has reached the minimal criterion, new environments are created at regular intervals decided by the parameters `transfer_generation` and `create_generation`, as seen in figure 2.3.

When new environments are created, all the pairs are checked against the minimal criterion for reproduction. The environments from the pairs that satisfy the criterion are marked as eligible to reproduce. If there are no eligible environments the creation of environments is skipped, and the evolution proceeds to the next generation.

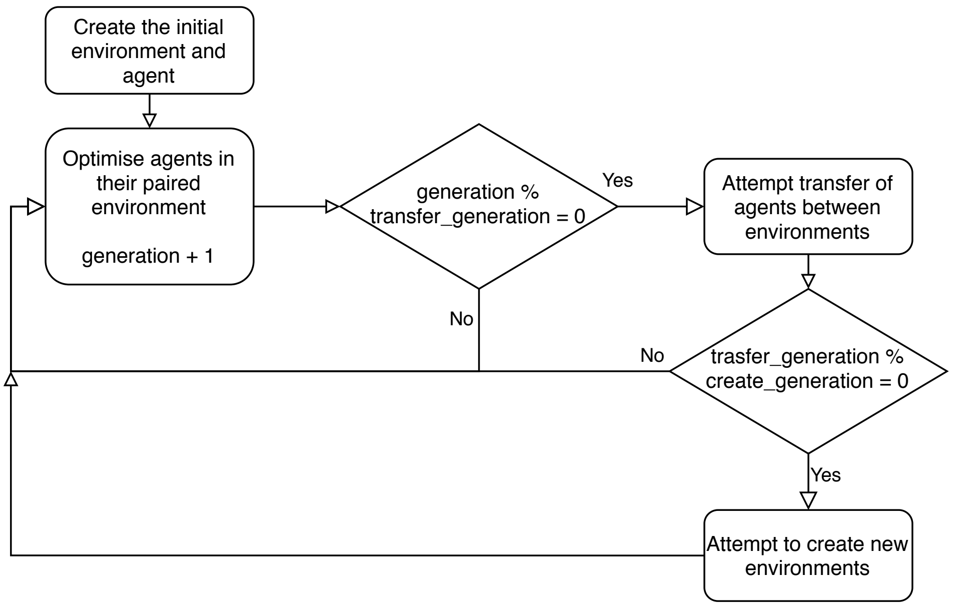

Create the initial environment and agent
Optimise agents in their paired environment
generation + 1
generation % transfer_generation = 0
Yes
No
Attempt transfer of agents between environments
trasfer_generation % create_generation = 0
No
Yes
Attempt to create new environments

Figure 2.3: The main loop of the POET algorithm. Transfer of candidates is attempted every `transfer_generation` generations, and creation of environments is attempted every `create_generation*transfer_generation` generations.

To create a child, one of the eligible environments is selected at random and mutated. The number of children created is decided by a parameter set by the user. To choose agents for the children, all agents that exist in the population are tested in the child environments. The agent that has the highest fitness in the child environment is chosen, and copied to the child to create a pair. The child pairs are then checked against a second minimal criterion, the minimal criterion of difficulty. This minimal criterion has an upper and lower boundary for agent fitness, and ensures the environment is not too difficult or too easy for its agent. The children that do not meet this minimal criterion are removed.

After this process we are left with a list of children that all meet the minimal criterion of difficulty. These children are sorted by environment novelty. The novelty of the environment is found by comparing it to an archive of all environments that have existed in the run. The novelty measure is the euclidean distance to the five nearest neighbours in the archive. If a child environment already exists in the archive, it is removed from the list of child pairs.

The most novel of the child pairs is added to the population until the maximum number of children that can be added each generation is reached, or until there are no more children left to add. The maximum number of children that can be added is controlled by a parameter set by the user. The POET population has a maximum population size. When this limit is reached the oldest pair is removed.

2.5.2 Optimising the agents

Wang et al. use evolutionary strategies to optimise their agents. Their algorithm is based on an algorithm used by Salimans et al. [44], which has been shown to be effective on reinforcement learning problems.

The algorithm optimises a parameter vector θ, which contains the parameters to a distribution. In each step of the evolutionary strategy a direction in which to move the distribution is found, and an offset is added to the parameter vector based on this direction. The direction is found by using the distribution to sample policies, calculating the fitness of the policies, and creating a rank-normalised weighed sum of the policies based on their fitness. The distribution used is an isotropic multivariate Gaussian.

2.5.3 Transferring agents between environments

In the transfer step all agents are cross tested in all environments. If any of the agents performs better in an environment than the environment's paired agent, the paired agent is removed, and is replaced by a copy of the agent that performs best.

There are two types of transfer, direct and proposal transfer. In direct transfer the agents are tested directly in the other pairs' environments, while in proposal transfer the agents are trained for five generations in the other pairs' environments before they are tested. Transferring of agents allows skills learned in one environment to be used in another environment, and in this way, the pairs trade experiences.

Chapter 3

Implementation

We co-evolve the morphology and controller of agents locomoting in the OpenAI Bipedal Walker Hardcore environment. The locomoting agents walk through courses, and the courses change, either through a curriculum, or through being evolved open-endedly with POET-M.

The first section of this chapter, section 3.1, describes a modified version of Bipedal Walker Hardcore. This environment has been modified both by us and by Wang et al. [23]. Section 3.2 describes the genetic algorithm used to evolve the locomoting agents. In section 3.3 we present a novel version of POET [23], POET with evolution of Morphologies, POET-M. Section 3.4 describes four curricula of bipedal walker courses, which we use in our experiments.

3.1 Bipedal Walker Hardcore v2

We use a modified version of the Bipedal Walker Hardcore environment to test our algorithm. We choose this environment because it is the environment Wang et al. use in their experiments with POET. Bipedal Walker Hardcore is a 2d environment from the OpenAI framework, in which a bipedal agent attempts to walk through a course, and reach a flag situated at the end. The course has various obstacles which the agent has to navigate past.

The version of the environment that we use was first modified by Wang et al.. They added the option to control the size, and thus the difficulty, of the obstacles that appear in the course. We have also modified the environment, adding the option to control the size of the bipedal agent's legs.

Due to its two dimensional nature the bipedal walker environment is not very computationally intensive, and therefore serve as a good initial test platform for new ideas. Figure 3.1 shows an example of an agent walking through a course in the Bipedal Walker Hardcore environment.

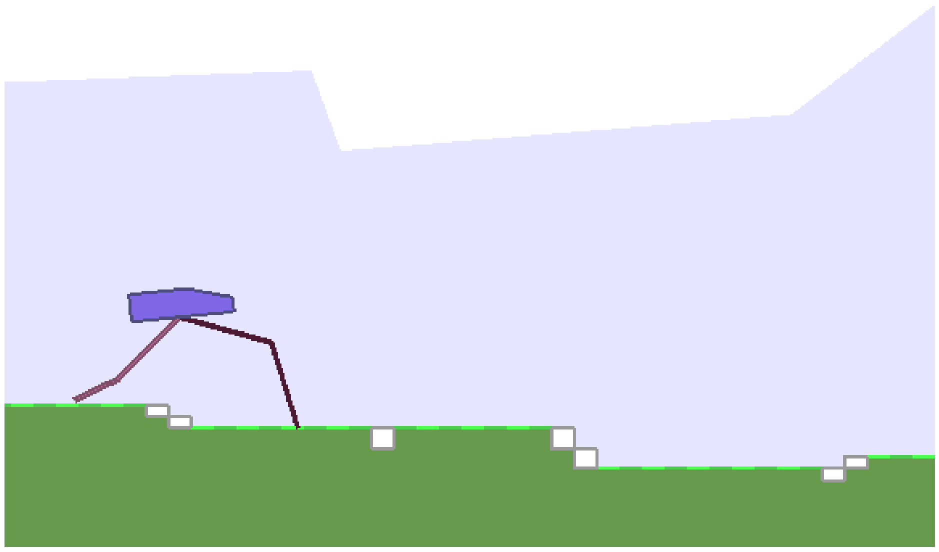


Figure 3.1: The Bipedal Walker Hardcore environment.

3.1.1 The bipedal agent

The bipedal walker agent has two legs and a head, and uses lidar measurements to observe its environment. The head has constant size and weight. The legs consist of two segments each, with one knee joint, and one joint connecting the leg to the hull. In the original environment the size of the legs is constant, however, after our modifications the size of the legs is controlled by eight parameters, described in section 3.2.2. The leg sizes are decided at environment initialisation, so to change the leg sizes the environment must be re-initialised.

Agent constraints

If the agent is allowed to evolve the most efficient leg sizes without any constraints a natural and simple solution would be to create legs with the same length as the course, and simply fall forward to the far end of the course. Ha et al. [6] demonstrated that this happened with their agents when they did not apply any restrictions. To avoid such solutions we apply the same leg size restraints as Ha et al. did in their study, which is $\pm 75\%$ from the original environment's leg sizes. The original, minimum and maximum sizes for the leg segments are in table 3.1.

Leg part	Original Size	Minimum Size	Maximum Size
Top segment width	8.0	2.0	14.0
Top segment height	34.0	8.5	59.5
Bottom segment width	6.4	1.6	11.2
Bottom segment height	34.0	8.5	59.5

Table 3.1: Original, minimum and maximum sizes for the bipedal walker agent's leg segments, the values are equal for the two legs.

3.1.2 The environments

The bipedal walker agent locomotes though 2d courses. The courses are filled with four types of obstacles: uneven terrain, stumps, pits and stairs. In the original version of Bipedal Walker Hardcore the size of the obstacles was constant, but after the modifications done by Wang et al. the obstacles are controlled by seven parameters:

- Stump height and width
- Pit gap width
- Stair height, width and step
- Roughness (The terrain curvature)

Some parameters consist of two numbers. These numbers define a range from which the size of the obstacle is drawn uniformly. As with the leg sizes described in section 3.1.1, the size of the environment features are set at environment initialisation. The environment features are spread randomly throughout the course, and are randomised each time the environment is reset and the agents starts walking from the beginning again. The placement and frequency of the features can not be controlled directly, but a random seed can be used to generate the same course multiple times.

Environment Constraints

To ensure the environment feature sizes do not become so small that they are almost non existent, or so large that is is not physically possible for the agents to move past them, we constrain the sizes to the minimums and maximums shown in table 3.2.

3.1.3 Inputs and outputs of the simulation

There are four inputs to and 24 outputs from the bipedal walker simulation. The values for the inputs and outputs are decided once per time frame. The four input parameters control the torque applied to each of the four leg joints of the bipedal walker agent. The 24 outputs describe the following information about the environment state:

Obstacle Type	Minimum Value	Maximum Value
Stump height	[0.1, 0.4]	[5.0, 5.0]
Stump width	[1.0, 2.0]	[1.0, 2.0]
Pit gap width	[0.1, 0.8]	[10.0, 10.0]
Stair height	[0.1, 0.4]	[5.0, 5.0]
Stair width	[1.0, 2.0]	[1.0, 2.0]
Stair step	1	9
Roughness	0.0	10.0

Table 3.2: The minimum and maximum values we use for the bipedal walker environment parameters.

- The angle, angular velocity, horizontal speed and vertical speed of the head.
- The position and angular speed of the leg joints.
- Whether each leg is in contact with the ground.
- 10 lidar rangefinder measurements.

3.1.4 Reward

The simulation is run until one of three termination conditions are met. The termination conditions are the agents head touching the ground, the agent reaching the flag at the end of the course or the simulation reaching a maximum number of allowed frames. We have set this maximum to 1000 frames in our experiments.

At the end of each time frame the agent gets a reward. The agent gets a positive reward proportional to how far it reached horizontally along the course within the time frame. It also gets a negative reward proportional to the torque that was applied to the joints. If the simulation ends due to the agents head touching the ground, it receives a negative reward of -100. This means that the agent that is the most torque efficient per unit moved horizontally gets the highest reward.

3.2 Evolution of controllers and morphologies

This section describes the genetic algorithm we use to co-evolve controllers and morphologies. The algorithm keeps a population of 192 individuals, where each individual consists of a neural network and a morphology. The neural network and morphology controls the behaviour and body of a bipedal walker agent.

3.2.1 The Neural network

The neural network has an input layer with 24 nodes, two hidden layers with 40 nodes each, and an output layer with four nodes. This gives a total of 2720 weights. The activation function used is the identity function. This network structure has been used in two other studies that also evolved agents locomoting in the bipedal walker hardcore environment [6][23]. The inputs to the network are the 24 environment state values. The four outputs from the network control the torque applied to the leg joints. See section 3.1.3 for more details about the environment interface. The neural network weights are initialised to random values, drawn uniformly between -1 and 1. Mutations can never increase the weights above 30, or decrease them below -30.

3.2.2 The Morphology

The morphology is a vector of eight floats. Each float controls one of the eight agent parameters defined in section 3.1.1, and thus each float controls either the length or the width of one of the agent's four leg segments. The minimum and maximum values for the floats are shown in table 3.1. The floats are initialised to random numbers between its minimum and maximum values.

3.2.3 Genetic algorithm

The Main Loop

Figure 3.2 shows the flow of the genetic algorithm. After initialisation the fitnesses of the individuals are calculated by evaluating them in the bipedal walker environment. Parents are then selected based on the fitness, and are recombined to create new candidates. The new candidates go through two stages of mutation, first some values are replaced, and then some values are modified. When the children have been mutated they are evaluated to find their fitness. Crowding is used to select survivors. After survivors have been selected the next generation begins, and the fitness of the candidates is re-evaluated. It is important to re-evaluate the fitness of the candidates at the beginning of each generation, as the environment the agent locomotes in may have changed, drastically impacting the fitness of the candidates.

The fitness function

To evaluate the fitness of a candidate it is tested in the bipedal walker environment. The candidate receives reward from the simulation as described in section 3.1.4. The fitness is the sum of the accumulated reward from all time steps.

The reward a candidate gets, when evaluated, varies a lot based on how the features are distributed in the environment. The candidate can be lucky or unlucky with the placement of features. This can make the fitness function very unstable. To make the fitness function more stable, the candidates are evaluated four times, and their fitness is calculated as the mean of the fitness it received in the four runs.

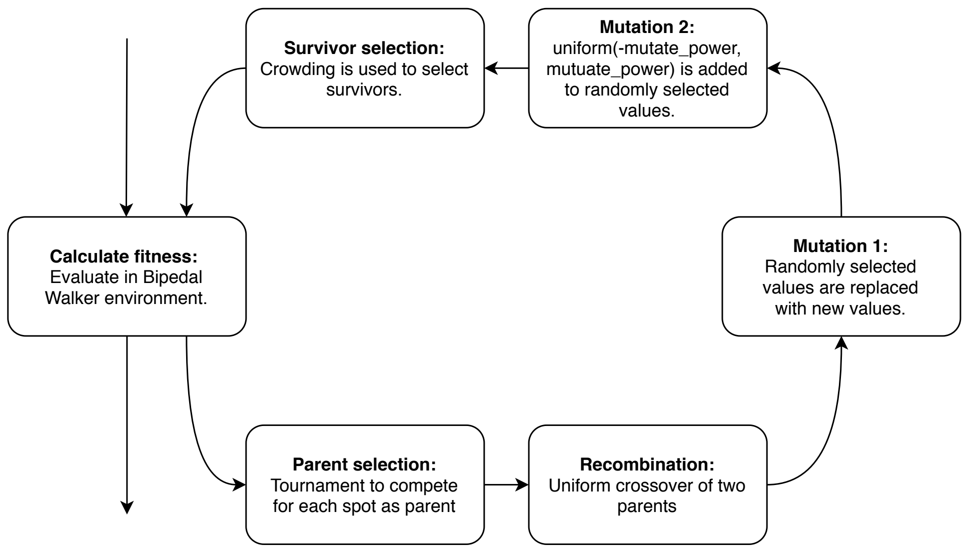

Survivor selection: Crowding is used to select survivors.
Mutation 2: uniform(-mutate_power, mutuate_power) is added to randomly selected values.
Calculate fitness: Evaluate in Bipedal Walker environment.
Mutation 1: Randomly selected values are replaced with new values.
Parent selection: Tournament to compete for each spot as parent
Recombination: Uniform crossover of two parents

Figure 3.2: Genetic algorithm training loop

Parent selection

The parents are selected by tournament. Five candidates are chosen at random from the population, and compete with their fitness to become a parent. This is repeated until 192 parents have been chosen. The same candidate can be chosen as a parent multiple times. The parents are then separated into 96 pairs, and the two parents from each pair is recombined to create two children.

Recombination

The parents are recombined using uniform crossover. For each neural network weight, or morphology size, the parent contributing the value is chosen at random, with equal probability between the two parents. The first child gets the chosen values, and the second child gets the remaining values.

Mutation

The children are mutated using two types of mutation: replacement and modification. In replacement mutation, neural network weights, and morphology values, are chosen with a probability of 0.0075. The chosen values are replaced with new values. The new values are determined in the same way as initial weights and morphology values are determined when the agent is initialised, as described in section 3.2.1.

In modification mutation, neural network weights, and morphology values, are chosen with a probability of 0.075. An offset is added to the chosen weights and values. The offset is a random float drawn uniformly from (-x, x). For the neural network weights x is 0.2. For the morphology values x is 16% of the difference between the minimum and maximum values for the size of the respective leg segment, see table 3.1.

Survivor selection

To create niches of different solutions in the population, and to slow down convergence, deterministic crowding is used when selecting survivors for the next generation.

Instead of selecting survivors, the children have to compete with the older population in tournaments to replace them in the surviving population. For each child a certain number of candidates w are chosen at random from the old population. In our case w is 20. All the chosen candidates are compared with the child to find the one that is the most similar. The most similar candidate then competes with the child. If the child has higher fitness it replaces the candidate in the population, and the old candidate is discarded. If the child has lower fitness than the candidate the candidate keeps its place, and the child is discarded.

The difference between two individuals is the L1-norm of the individuals' morphologies. We compare only the morphologies, and not the neural networks, to encourage the niches in the population to explore different morphologies.

3.3 POET-M: POET with evolution of Morphologies

POET with evolution of Morphologies, POET-M, is very similar to POET, which is described in section 2.5. Therefore we only describe our additions to the algorithm. Sections 3.3.1, 3.3.2 and 3.3.3 correspond to the main steps in POET, environment creation, evolution of agents and transfer of agents between environments. These three sections describe the changes we have made to each step. Section 3.3.4 describes how the POET-M algorithm is applied to the bipedal walker environment.

3.3.1 Environment creation

POET has high computational cost, so to reduce this we create fewer environment children in the environment creation step. Wang et al. created 512 new potential environments in their POET runs, which all have to be simulated with all agents in the population to find their fitness. In POET-M the amount of environment children created is decided by the maximum population size, which in our case is 20.

This change might lead to less diversity in the environment population, as the likelihood of finding a very novel environment increases when more environments are checked.

3.3.2 Agent evolution

In POET an evolutionary strategy algorithm was used to optimise the agents, as described in section 2.5.2. In POET-M we use the genetic algorithm presented in section 3.2. Our genetic algorithm co-evolves the morphology and controllers of the agents, while the original POET only evolved the agent controllers.

3.3.3 Transfer of agents between environments

In the original POET algorithm there are two types of transfer, direct and proposal transfer, which are described in section 2.5.3. To reduce computation cost we only use direct transfer in POET-M. This may lead to fewer transfers, and thus less exchange of information between pairs. However, we observed that even with direct transfer only, transfers happened frequently.

3.3.4 Applying POET-M to the Bipedal Walker environment

There are a few parameters that exist in both POET and POET-M, which need to be set when applying the algorithms to an environment. These are the values we use for those parameters:

- Transfer frequency: 5. Transfer is attempted every fifth generation.
- Environment creation frequency: 40. New environments are created every 40th generation.
- Mutation criterion: 200. The minimal agent fitness for a paired environment to be eligible for reproduction.
- Selection criterion: 50-300. The minimum and maximum fitness of an agent for a child pair to be allowed to enter the population.
- Child pairs: 20. The number of child pairs created in the environment creation step.
- Children admitted: 2. The maximum number of child pairs that can be admitted to the population each time environments are created.
- Maximum population size: 20. The maximum numbers of pairs in the population.

We also need to decide how to mutate the environments. How we mutate the bipedal walker environments is described below.

Environment mutation

In the environment genome there are seven parameters controlling the environment features, as described in section 3.1.2. Five of the parameters can be mutated: the height of stumps, the width of pits, the height of stairs, the number of steps in a stair, and the roughness of the terrain. Each of the five parameters has a probability of 0.2 to be mutated. The first time a parameter is chosen for mutation it is initialised to its minimum value. After that, when a parameter is mutated, a mutation step is either added to or subtracted from it. The minimum values, maximum values and mutation steps for the parameters can be found in table 3.3.4.

When one of the stump or stair parameters are initialised, the other parameters describing the same feature will also be initialised to their minimum value, as the features cannot exists without all their parameters being initialised.

Obstacle Type	Minimum Value	Mutation Step	Maximum Value
Stump height	[0.1, 0.4]	0.2	[5.0, 5.0]
Stump width	[1.0, 2.0]	-	-
Pit gap width	[0.1, 0.8]	0.4	[10.0, 10.0]
Stair height	[0.1, 0.4]	0.2	[5.0, 5.0]
Stair width	[1.0, 2.0]	-	-
Stair steps	1	1	9
Roughness	uniform(0.0, 0.6)	uniform(0.0, 0.6)	10.0

Table 3.3: The minimum values, mutation steps and maximum values for the environment parameters in POET-M. Stump width and stair width are never mutated.

3.4 Environment curricula

3.4.1 Simple static environment

This curriculum consists of one environment that never changes. The environment is the simplest environment that can be created in Bipedal Walker Hardcore, and is the same environment as the one created when initialising POET-M. The environment is a completely flat course, with no features. The parameters for this environment is shown in table 3.4.

Feature	Env. 1
Stump height	[0, 0]
Stump width	[0, 0]
Pit gap	[0, 0]
Stair height	[0, 0]
Stair width	[0, 0]
Stair steps	0
Roughness	0

Table 3.4: Parameters for the **static environment**

3.4.2 Two environment curriculum

This curriculum consists of two static environments. The first environment is a completely flat course, with no features, while the second environment has small stumps. The environment is changed from the first to the second environment in generation 500. The parameters for the two environments are shown in table 3.5.

Feature	Env. 1	Env. 2
Stump height	[0, 0]	[0.1, 0.5]
Stump width	[0, 0]	[1.0, 2.0]
Pit gap width	[0, 0]	[0, 0]
Stair height	[0, 0]	[0, 0]
Stair width	[0, 0]	[0, 0]
Stair steps	0	0
Roughness	0	0

Table 3.5: Parameters for the **two environment curriculum**

3.4.3 Round robin curriculum

This curriculum has five environments. The first environment is flat, while the four other environments each have one of the four possible obstacles. The active environment changes every five generations, moving through the list of environments from environment one to environment five. When the sequence has finished it repeats from the first environment. The parameters for the five environments are in table 3.6.

Feature	Env. 1	Env. 2	Env. 3	Env. 4	Env. 5
Stump height	[0, 0]	[0, 0]	[0, 0]	[0.2, 1.0]	[0, 0]
Stump width	[0, 0]	[0, 0]	[0, 0]	[1.0, 2.0]	[0, 0]
Pit gap width	[0, 0]	[0.2, 1.0]	[0, 0]	[0, 0]	[0, 0]
Stair height	[0, 0]	[0, 0]	[0, 0]	[0, 0]	[0.2, 1.0]
Stair width	[0, 0]	[0, 0]	[0, 0]	[0, 0]	[1.0, 2.0]
Stair steps	0	0	0	0	3
Roughness	0	0	0.6	0	0

Table 3.6: Parameters for the five environments in the **round robin curriculum**

3.4.4 Round robin incremental curriculum

This environment is quite similar to the round robin curriculum described in section 3.4.3. In the beginning of this curriculum the environment parameters are as shown in table 3.6, and the environments are switched between as described in section 3.4.3. The only difference is that, in this curriculum, every time the agent reaches a fitness of 150 or greater in one of the environments, that environment increases in difficulty. The difficulty is increased by adding a mutation step to one of the parameters that describes the obstacle the environment contains. The mutation steps for each of the four obstacle types are shown in table 3.7. The stair environment has two possible parameters to mutate, stair height and stair steps, which are mutated every other time.

Environment	**Feature**	**Mutation step**
Env. 2	Pit gap width	[0.2, 0.2]
Env. 3	Roughness	0.6
Env. 4	Stump height	[0.2, 0.2]
Env. 5	Stair height	[0.2, 0.2]
	Stair steps	1

Table 3.7: Mutation steps for each of the four types of obstacles present in the environments found in the round robin incremental curriculum

Chapter 4

Experiments and Results

This chapter describes our experiments and presents our results and analysis. The first section, section 4.1, explains the methods we use to analyse our results. Section 4.2 describes the baseline we compare our results to. The last three sections, 4.3, 4.4 and 4.5, presents our three experiments and their results. Each of the three experiments corresponds to one of the three research goals from section 1.2.

- **Experiment 1:** To look for correlations between changes in environments and changes in morphology, the morphological change of two agents is compared. One of the agents experience environmental change, while the other does not.
- **Experiment 2:** The effects of increasingly challenging environments on morphological change are explored. Two curricula are compared. One of them has increasingly difficult environments, while the other does not.
- **Experiment 3:** We evaluate how evolving agents with POET-M affects their diversity and fitness.

4.1 Analysis methods

4.1.1 Morphological distance

We want to measure how much the morphology changes over generations. Because the values in the morphology genotype correspond directly to lengths and widths in the phenotype, we can measure change in the genotype and use it to represent change in the phenotype. In all our experiments we record the morphology every five generations. We call this period of five generations an epoch. The morphological change, for an epoch, is measured as the euclidean distance from the average of the morphologies registered in the current epoch, to the average of the morphologies registered in the previous epoch. The morphological change is calculated through the following two steps:

First we find the average of all the morphologies in each epoch. We call the average morphology in epoch i X_i. X_i can be written as:

$$X_i = \sum_{n=1}^{p} \frac{x_{in}}{p} \tag{4.1}$$

Where x_{in} is a morphology genome, n is the genome's placement in the population, i is the epoch, and p is the population size. The second step is to calculate the euclidean distance between X_i, and X_{i-1}, for every epoch. The formula for the morphological change, M, in epoch i, can then be written as:

$$M_i = \sqrt{\sum_{m=1}^{g} (X_{im} - X_{(i-1)m})^2} \tag{4.2}$$

Where g is the length of a morphology genome.

M only takes into account the change from the previous epoch. To observe how the morphology changes on a larger time scale, we calculate the average euclidean distance, from X in the current epoch, to X in each of the previous 20 epochs. We call this M^{20}:

$$M_i^{20} = \frac{1}{20} \sum_{j=1}^{20} \sqrt{\sum_{m=1}^{g} (X_{im} - X_{(i-j)m})^2} \tag{4.3}$$

4.1.2 Population diversity

Similarly to Samuelsen et al. [45] we define the diversity of an individual x to be the average distance from x to all other individuals in the population. We use euclidean distance as the distance measure between two individuals. The diversity of a population, X, is the average diversity of all individuals in the population. We call the diversity of a population X, D_X:

$$D_X = \sum_{n=1}^{p} \frac{\sum_{q=1}^{p} \sqrt{(x_n - x_q)^2}}{p^2} \tag{4.4}$$

Where p is the population size, and x_n and x_q are the morphology genomes of individual number n and q in the population.

4.1.3 Morphological feature maps

Inspired by Miras et al. [46] we define seven morphological descriptors, which are used to estimate how much of the morphology space has been explored:

- **Length:** The total length of the four leg segments.
- **Width:** The total width of the four leg segments.
- **Area:** The total area of the four leg segments.

- **L/R:** Left/right proportion. Length of the left leg divided by the length of the right leg.
- **UDL:** Upper/lower left leg proportion. Length of the upper leg segment, divided by the length of the lower leg segment, for the left leg.
- **UDR:** Upper/lower right leg proportion. Length of the upper leg segment, divided by the length of the lower leg segment, for the right leg.
- **W/L:** Width/length proportion. The total width divided by the total length.

We calculate the morphological descriptors for all morphologies, in all epochs, and plot two of the morphological descriptors against each other, in a feature map. Every morphology is represented as a circle in a 2D map. A circle's placement in the map is decided by the morphological descriptors, and its color is decided by what generation the morphology occurred in. The density plots show how many different combinations of the two descriptors have been explored by the algorithm, and in which generations they were explored.

4.1.4 Quality-diversity feature maps

The quality-diversity feature maps are created by plotting two of the morphological descriptors described in section 4.1.3, against each other, in a 2D map. The quality-diversity map is divided into a 20 by 20 grid of squares. All morphologies that were explored in the run are placed into their respective square in the grid. The color of a square shows the fitness of the best performing individual within the square.

4.2 Baseline algorithm

To create a baseline we run the genetic algorithm described in section 3.2 on agents locomoting in the static, flat environment, which is the curriculum described in section 3.4.1. The agents were evolved for 1000 generations, and the morphologies and fitness are recorded at the end of every epoch, which is five generations long. We perform five runs with different random seeds.

4.2.1 Results

The morphological change, M^{20}, from the baseline runs, is shown in figure 4.1. We can see that the morphology changes a lot in the first 200 generations, but the change gradually decreases and is very low from generation 500 and out. The baseline reached a fitness of almost 200 as can be seen in figure 4.2.

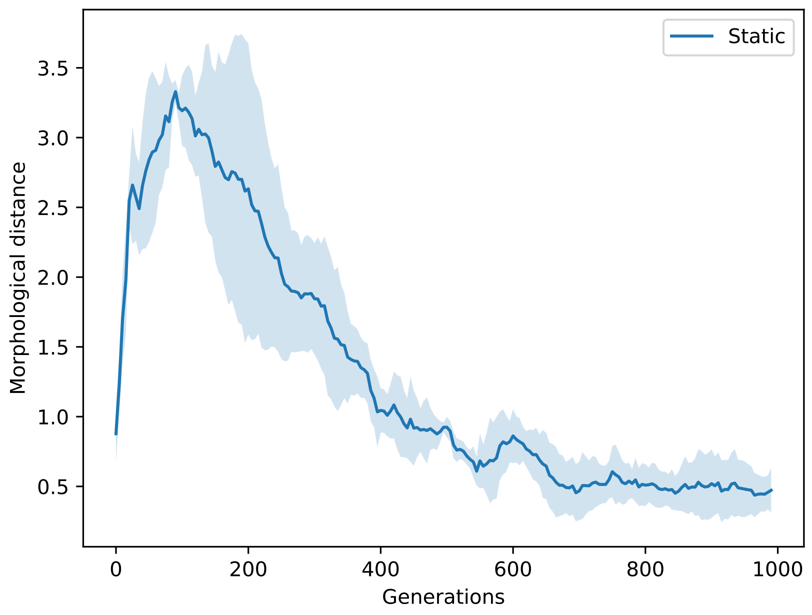


Figure 4.1: The morphological change M^{20}, described in section 4.1.1, for the baseline agent. The blue line shows the mean of five runs, and the coloured area shows the standard deviation.

Figure 4.4 shows several of the density plots for the baseline agent. The morphologies tend to collect around small areas in the feature space. The early morphologies, seen as purple circles, are quite spread out, while the last morphologies, seen as yellow circles are much more dense. The morphologies in the middle of the run, seen as green circles, are largely hidden behind the yellow sections, meaning that the search found the morphologies in the dense yellow sections of the feature space before reaching the halfway point of the run.

In figure 4.3 we can see how the agent's top performing individual's morphology changes over time in the first run of the baseline algorithm. The density plot of width and length for the same run can be seen in the top left corner of figure 4.4. Both the morphologies and the density plot shows that the agent evolves long thin legs over time. Four of the five baseline runs converged to this basin of attraction.

4.2.2 Analysis

As expected the baseline algorithm seems to converge to a morphology quite quickly. The morphological distance between generations is low in the second half of the runs in figure 4.1, and in most of the density plots shown in figure 4.4 the morphologies seem to collect in small areas.

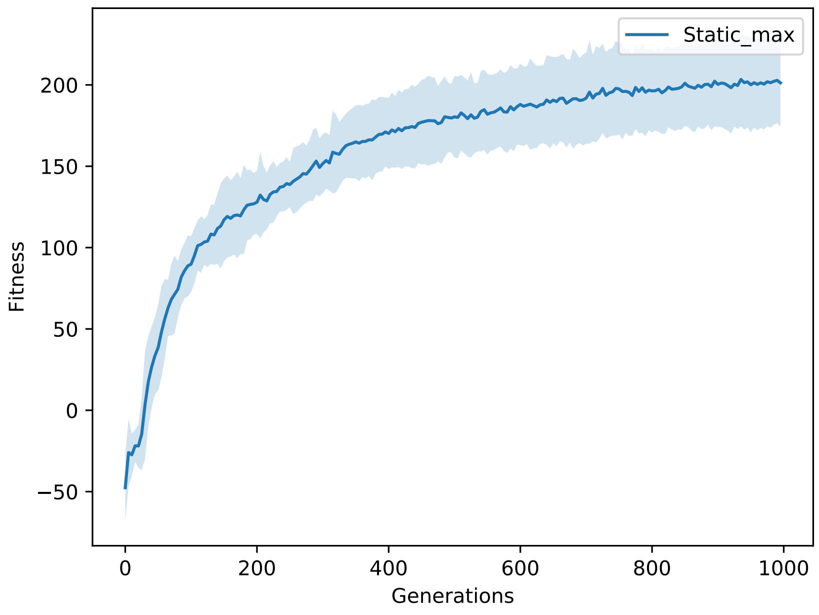


Figure 4.2: The fitness of the top performing individual from each generation for the baseline runs. The graph is the average of five runs, and the coloured area shows the standard deviation.

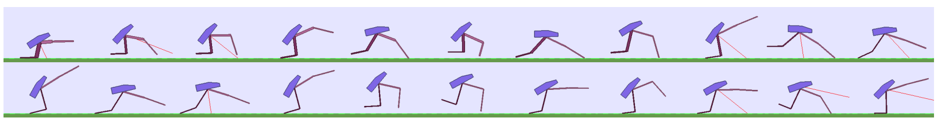


Figure 4.3: The morphology of the individual with the highest fitness for the baseline, captured every 40 generations. In the latter half of the evolution (the bottom row), the morphologies are nearly identical, all having long and thin legs.

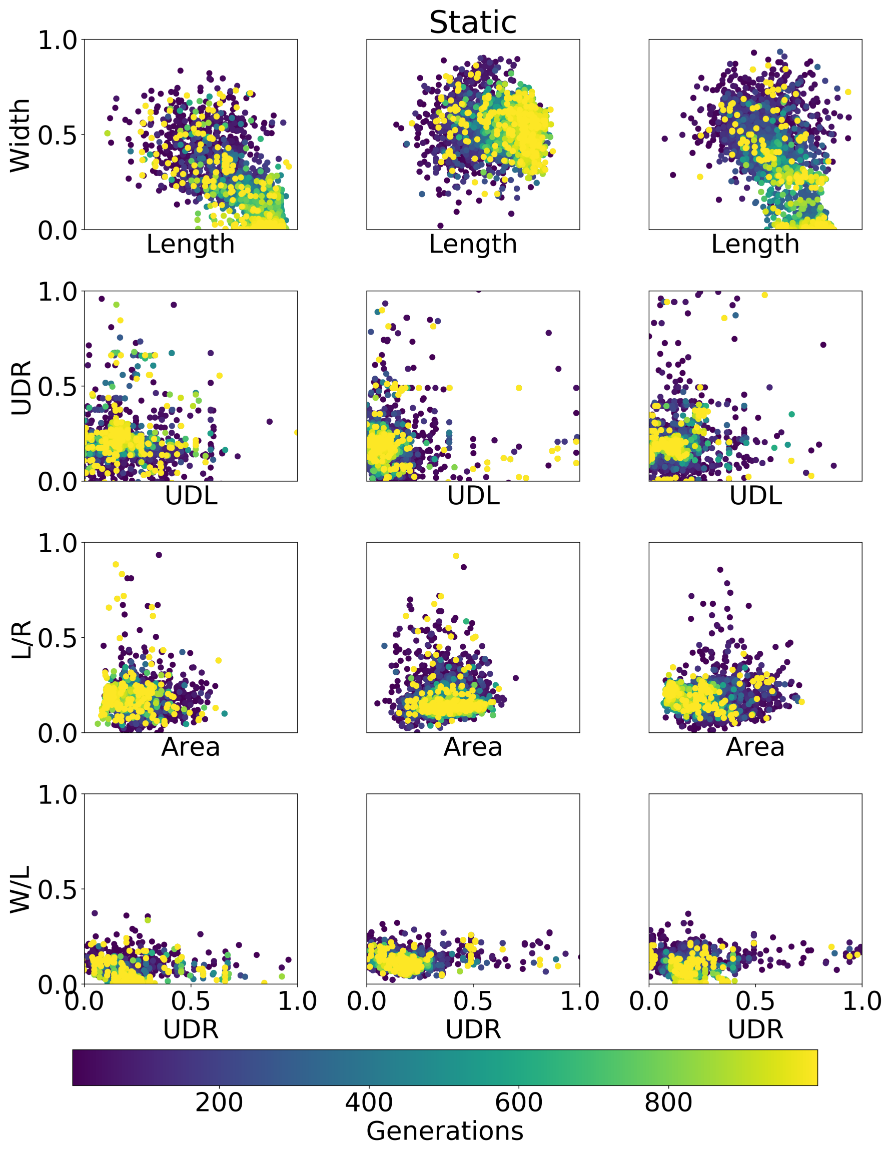


Figure 4.4: Morphological feature maps, as described in section 4.1.3, for three different runs of the baseline.

4.3 Experiment 1: Correlation between environmental and morphological change

In this experiment we want to find out how changes in environments affect the morphological development of locomoting agents, which is the first research goal described in section 1.2. We run the genetic algorithm, described in section 3.2, for 1000 generations, in the environment curriculum described in section 3.4.2. In this curriculum the environment is changed halfway through the evolution. We do five runs with different random seeds. The morphologies are recorded once every five generations. We expect to see changes in the morphology as a reaction to change in the environment.

4.3.1 Results

Figure 4.5 shows M^{20} for the runs performed in the first experiment. A spike in morphological change can be seen after generation 500, where the environment changes.

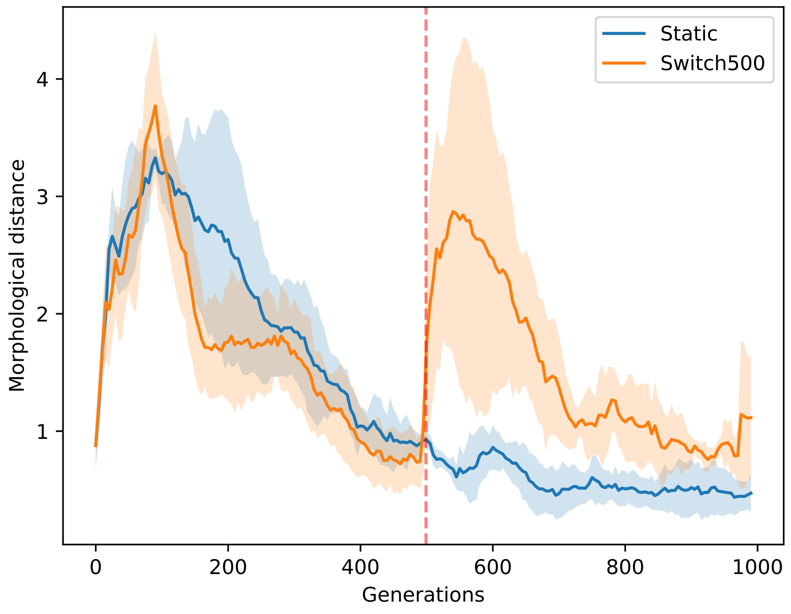


Figure 4.5: The morphological change M^{20} for the baseline (blue), and the two environment curriculum (orange), where the environment changes at generation 500. The dashed red line marks the generation where the environment changes. Both graphs are an average of five runs, and the coloured areas show the standard deviation.

4.3.2 Analysis

We can clearly see a spike in the graph in figure 4.5 right after the environmental change happened, which means that the agents experienced increased morphological change as a reaction to the change in environment. One possible explanation for the increase in morphological change is that the agents adapt their morphology to fit the new challenges it meets in the new environment.

Another possible explanation is that selection pressure could be reduced. When the candidates are evaluated in a new environment they are likely to experience a reduction in fitness, an they have not yet learned to tackle the challenges found in the new environment. This drop in fitness is likely to have a larger impact on high fitness candidates, as the low fitness candidates might not even reach far enough through the course to encounter the new challenges. When the high fitness candidates experience a larger drop in fitness than the low fitness candidates, the selection pressure for the low fitness candidates can be reduced. This reduction in selection pressure could potentially encourage the algorithm to explore in different directions than before the environmental change, as the low fitness candidates are more likely to reproduce than before.

This increase in morphological change when the environments change suggests that it is possible to use environmental change to increase morphological diversity in a population.

4.4 Experiment 2: The effects of increasing challenges on morphological change

Our second goal is to find out what impact increasing the difficulty of the environments, as the agent solves them, has on the morphological development of locomoting agents. We create two hand crafted curricula. The first curriculum, round robin, changes between five static environments, and is described in section 3.4.3. The second curriculum, round robin incremental, changes between five environments, similarly to the first curriculum, but the difficulty of the environments increase as the agents gets better at them. The second curriculum is described in section 3.4.4. Each of the agents are evolved for 1000 generations, and we perform five runs with different random seeds. The morphology and fitness are recorded every five generations.

If appropriate difficulty is necessary to prevent premature morphological convergence, the morphology of the agents in the non-incremental environment should converge, while the agents in the incremental environment should continue to experience morphological change. If appropriate difficulty is not necessary, then the environments changing back and forth, in the round robin curriculum, should be enough for continued morphological exploration.

4.4.1 Results

Figure 4.6 shows M^{20}, and the morphological diversity in the resulting population, for the baseline, the agent evolved in the round robin curriculum and the agent evolved in the round robin incremental curriculum. Both the agents evolved in the round robin curriculum and the agents evolved in the round robin incremental curriculum have higher morphological change than the baseline agents. However, they do not have higher morphological diversity in the population, despite the higher morphological change throughout the run.

The fitness for the baseline, round robin and round robin incremental agents is shown in figure 4.7. Both of the curricula agents have lower fitness than the baseline agent. When looking at the fitness for the agents evolved in the two types of round robin curricula, it is important to note that the environment they are evaluated in changes along with the curricula's change in environments. Both of the round robin curricula use the flat baseline environment every 5th epoch. Therefore the tops of the spikes in their fitness graphs, which are the point where the agents are evaluated in the flat environment, are the points that are representative for the agents' fitness.

Looking at the morphological feature maps in figure 4.8 we can see that the sections of covered feature space is larger for the two types of round robin agents, than for the baseline agent, especially in the length width maps. The round robin and round robin incremental curricula have explored quite similar sections of the feature space, the sections explored by round robin incremental slightly larger. Figure 4.9 shows quality-diversity feature maps, where we can see that the baseline agent has found more yellow high fitness solutions, than the two types of round robin agents.

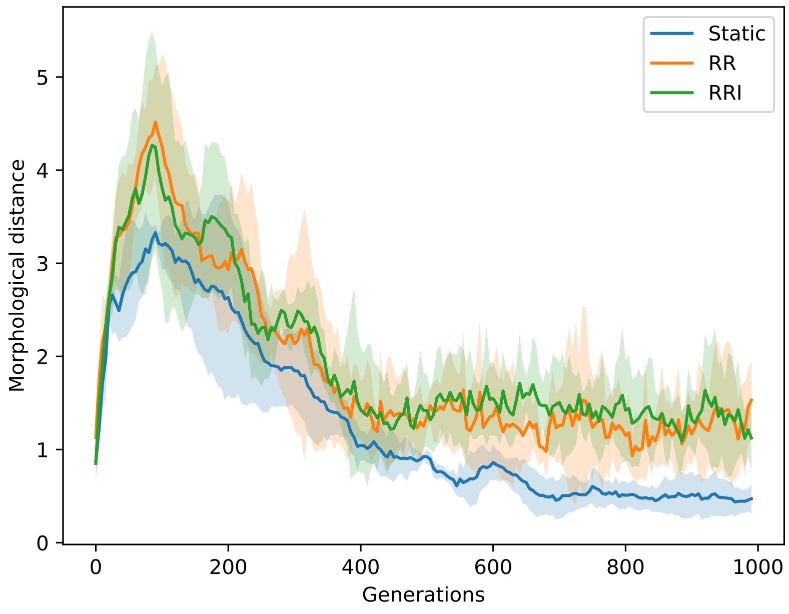
4.4.2 Analysis

Our expectations for this experiment was for the increasing difficulty in the round robin incremental curriculum to cause increased morphological exploration. However, the results for the round robin curriculum and the round robin incremental curriculum are very similar, both with regard to fitness and to morphological change. The morphological and quality-diversity feature maps are also quite similar, with the round robin incremental agents exploring slightly larger areas. Our results do not indicate any direct connection between increasing difficulty and morphological change. However, our results do not exclude the possibility of a connection.

The agents in the two round robin curricula have lower fitness than the baseline, with the gap up to the baseline fitness increasing gradually. The higher baseline fitness might be due to the baseline being able to focus all its resources towards optimising directly for the flat environment. In contrast to this the round robin agents spend resources adapting to other environments as well.

The agents from the two round robin curricula do not increase much in fitness at all after 400 generations. This might indicate that optimising for gaits in the environments that are not flat is detrimental to the search for a quick gait in the flat environment. The agents may have to slow down in order to overcome the obstacles in the other environments, which can lead them to slow down in the flat environment as well.

The round robin curricula had both lower diversity and lower fitness, in the flat environment, compared to the baseline. It is difficult to hand craft curricula that are balanced enough to encourage both diversity and quality, as the stepping stones that lead to greater diversity and quality are usually not intuitive. Intuitively, gradually increasing the difficulty of environments may seem like a well suited curriculum. However this is not necessarily the curriculum that causes the most effective learning.



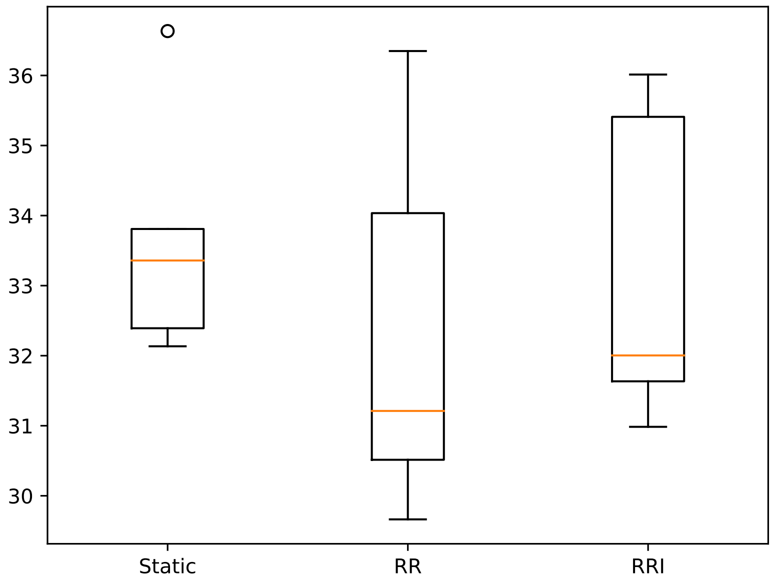
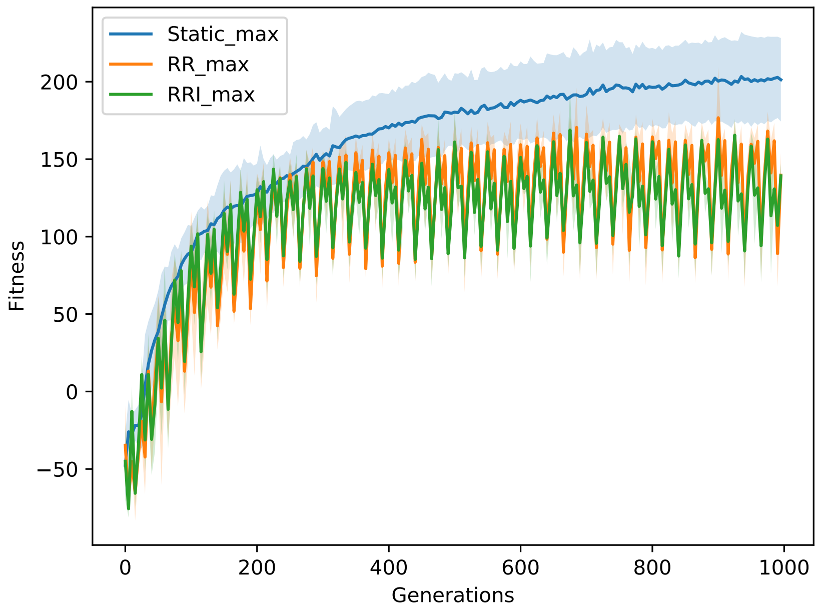


Figure 4.6: **Top:** The morphological change M^{20} for the agents evolved in the round robin curriculum (orange) and the round robin incremental curriculum (green). The baseline is included for comparison (blue). The graphs are averages of five runs, and the coloured areas show the standard deviation. **Bottom:** Diversity of the resulting populations at the end of the runs, for the baseline (Static), round robin (RR) and round robin incremental (RRI). The population diversity is measured as described in section 4.1.2.



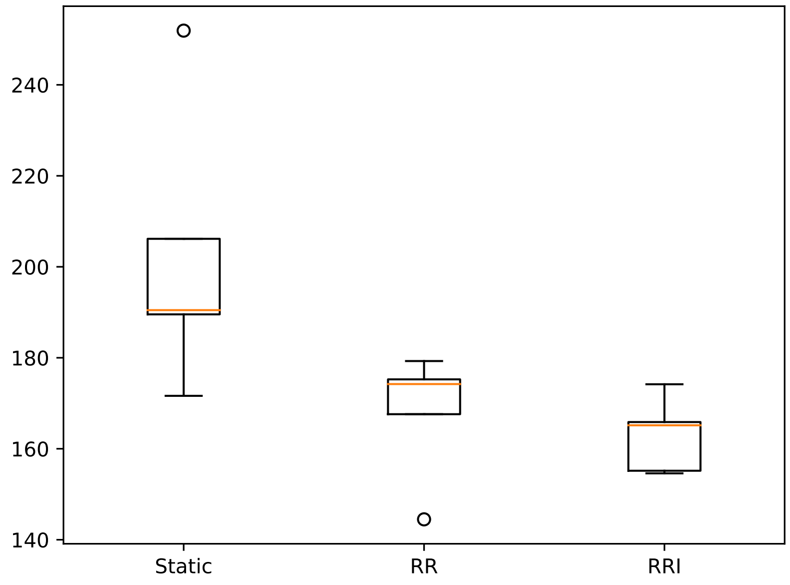


Figure 4.7: **Top:** Fitness of the best individual in the population for the baseline (blue), round robin curriculum (orange) and round robin incremental curriculum (green). The graphs are averages over five runs, and the coloured areas show standard deviation. The spikes in the RR and RRI fitness graphs are due to the changing environments. The tops of the spikes are the agents' fitness in the flat environment used for the baseline. **Bottom:** Fitness of the best individuals from resulting populations at the end of the runs, for the baseline (Static), round robin (RR) and round robin incremental (RRI). The fitness is recorded in generation 980, to ensure that all agents are evaluated in the same flat environment. In generation 1000 the RRI curriculum agents are training in a non-flat environment.

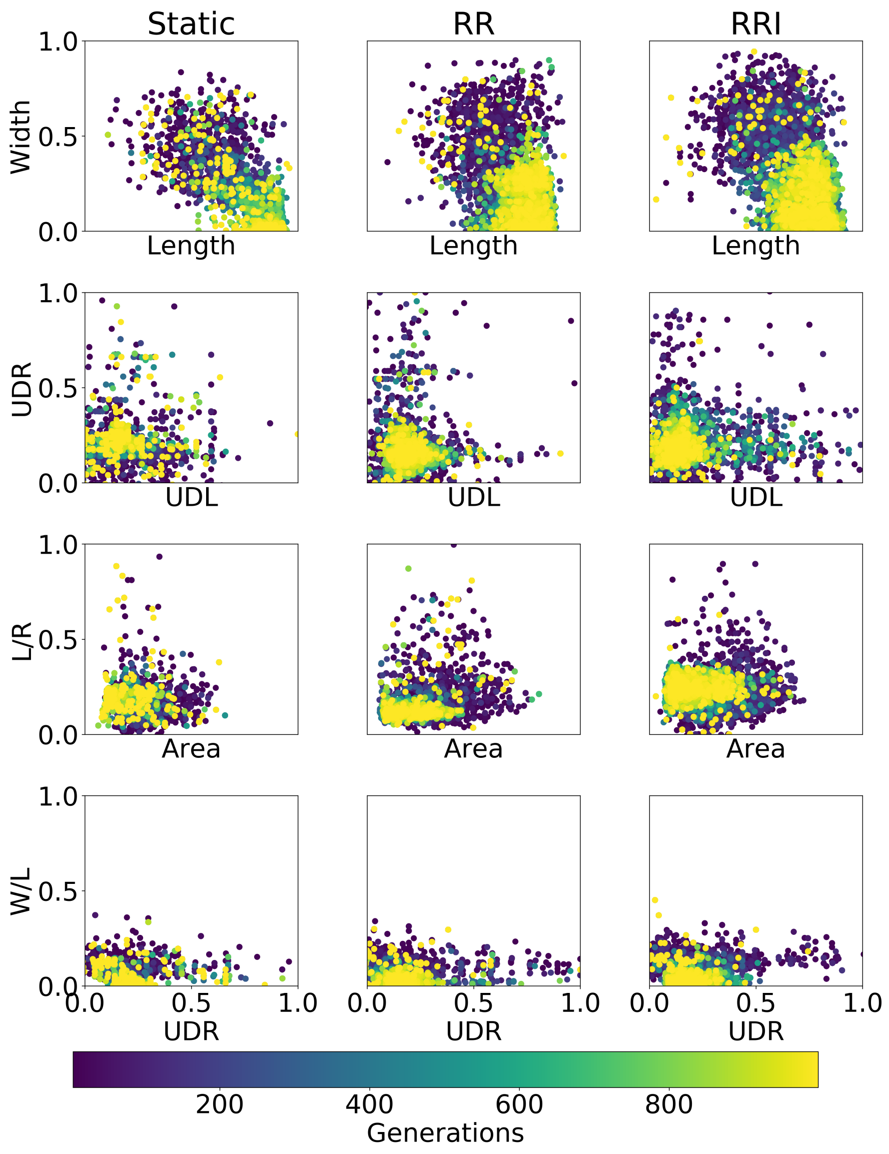


Figure 4.8: Morphological feature maps, as described in section 4.1.3, for the baseline (Static), the round robin curriculum (RR) and the round robin incremental curriculum (RRI).

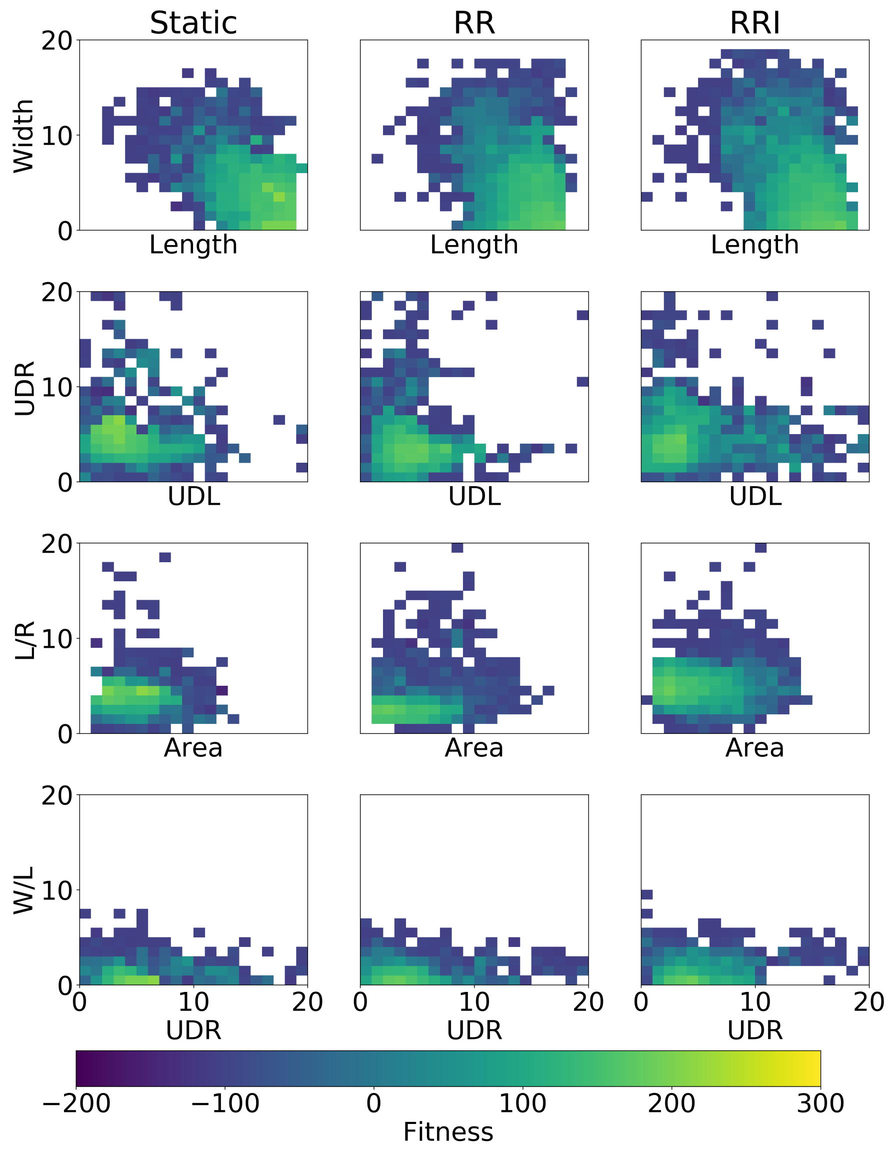


Figure 4.9: Quality-diversity feature maps, as described in section 4.1.4, for the baseline (Static), the round robin curriculum (RR) and the round robin incremental curriculum (RRI).

4.5 Experiment 3: Using POET-M to create a curriculum of challenges

In the third experiment we want to find out how effective POET-M is in maintaining both quality and morphological diversity in a population, compared to populations evolved in a static environment or a hand crafted curriculum. The POET-M algorithm, described in section 3.3, is run for up to 2000 generations. The POET-M agents evolve in many different environments in parallel, with agents jumping back and forth between them, but we follow the agents evolving in a flat environment, which is the first environment created by POET-M. The morphology and fitness, of all individuals in the population, is recorded every five generations, and five runs with different random seeds are performed.

Because POET-M evolves agents in parallel, in different environments, it uses a lot of computational power. The baseline and round robin incremental agents, which POET-M is compared to in this experiment, are run for 2000 generations. To achieve a fair comparison between POET-M and the curricula agents, POET-M is stopped when it reaches the same number of agent evaluations as the curricula agents use, even if it has not reached generation 2000.

If poet is effective in maintaining both morphological diversity and quality at the same time, we expect to see large sections of solutions with high fitness in the quality-diversity feature maps.

4.5.1 Results

Figure 4.10 shows M^{20}, and diversity of the resulting populations at the end of the runs, for the baseline, round robin incremental and POET-M. POET-M has high morphological distance throughout the run, similar to that of round robin incremental. However, POET-M has a lot more diversity in the resulting population compared to the other two runs. Fitness for the same runs is shown in figure 4.11. POET-M reaches a similar fitness as the baseline, which is significantly higher than the fitness of round robin incremental.

Figure 4.12 shows the morphological feature maps for the POET-M, round robin incremental and baseline runs. The baseline and round robin incremental populations start out in one area, shown in purple, and gradually moves towards a different area towards the end of the run, shown in yellow. In contrast to this, the purple sections in the POET-M feature maps are almost entirely covered by the yellow sections, meaning that POET-M is still exploring the same features in the end of run as those that were explored in the beginning.

If we look at the quality-diversity feature maps in figure 4.13, we see that the baseline and POET-M both find high fitness solutions, while round robin incremental finds low fitness solutions. In the width length maps POET-M's high fitness solutions are more spread out across the feature space compared to the baseline.

Figure 4.14 shows the morphology of the top performing individual of a POET-M run, sampled once every 40 generations, for the first 880 generations of the run. When comparing it to the baseline we can see that the best morphology is replaced many times in generation 440-880 for POET-M, while the baseline keeps the same morphology during these generations.

4.5.2 Analysis

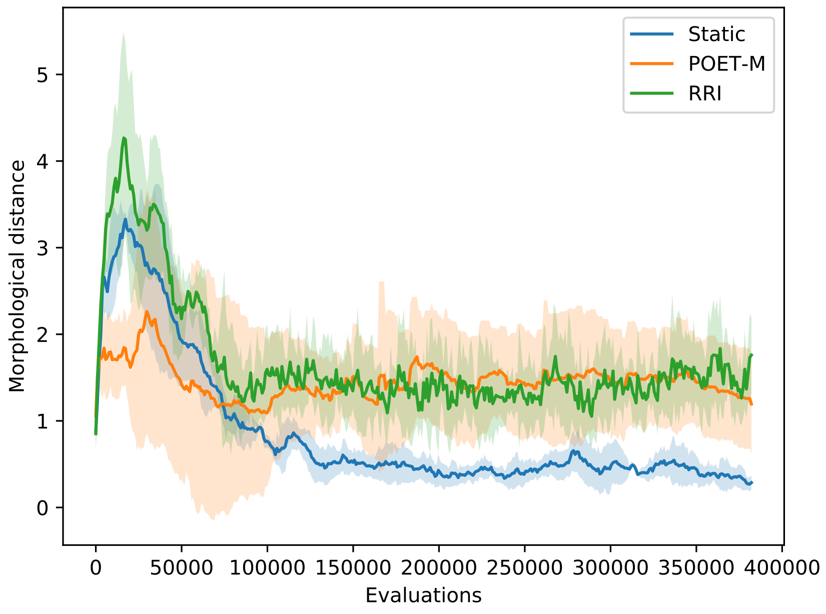
The POET-M agents have higher morphological change than the baseline agents, similar to the morphological change seen in the round robin incremental agents. In addition to this the POET-M agents have almost as high fitness as the baseline agents. While the environmental change in the round robin curriculum prevents the agents from reaching high fitness, the curriculum created by POET-M enables the agents to have both high morphological change and high fitness at the same time.

POET-M has significantly more morphological diversity in the resulting population than both the baseline and the round robin incremental curriculum. The curriculum created by POET-M managed to increase the population diversity, and not just the morphological change.

We can also see signs of high morphological diversity for POET-M in the morphological feature maps. POET-M explores approximately the same areas of the morphological feature space in the beginning and end of the evolution, suggesting that morphologies that are not easy to exploit are not removed from the population. Morphologies that are not easy to exploit are kept by POET-M even if the algorithm does not quickly find a good controller for it. From the feature maps it looks like the baseline and round robin incremental agents discard morphologies that do not have a good controller quite quickly, leading to the large purple sections in the morphological feature maps. This might create a bias towards morphologies that are easy to exploit, potentially causing the search to get stuck in a local optimum.

The quality-diversity maps, especially the width length maps, show that POET-M finds high fitness solutions for a larger amount of feature combinations than the baseline. This is likely due to the larger diversity in the POET-M population. These findings show that it is possible to use POET-M to create curricula that increases the morphological diversity of the population while still maintaining quality.

POET-M increases in fitness somewhat slower than the baseline agent in the last half of the runs. We believe that this is because POET-M uses a lot of its resources to search in other environments, while the baseline uses all of its resources in the flat environment. This causes the agent to converge more slowly to the good solutions for the flat environment, causing the slightly lower fitness. If the algorithms were run longer POET-M might surpass the baseline in fitness, by continuing to find better solutions after the baseline converges.



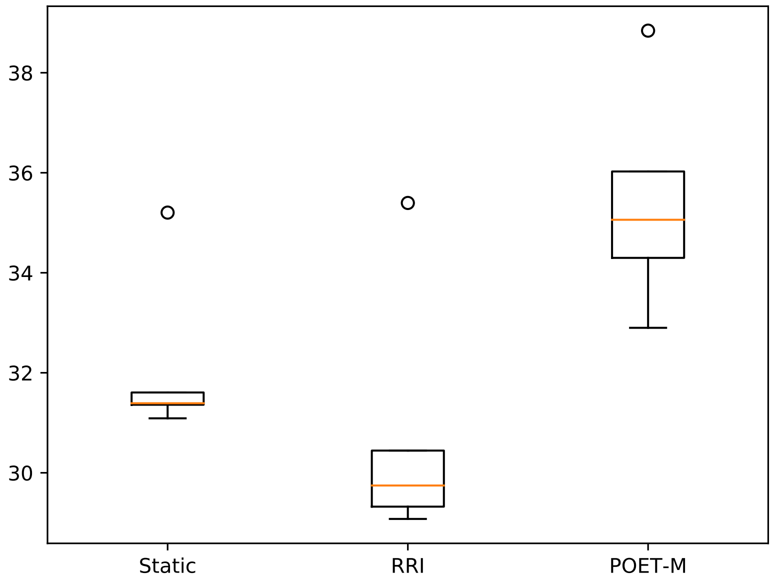
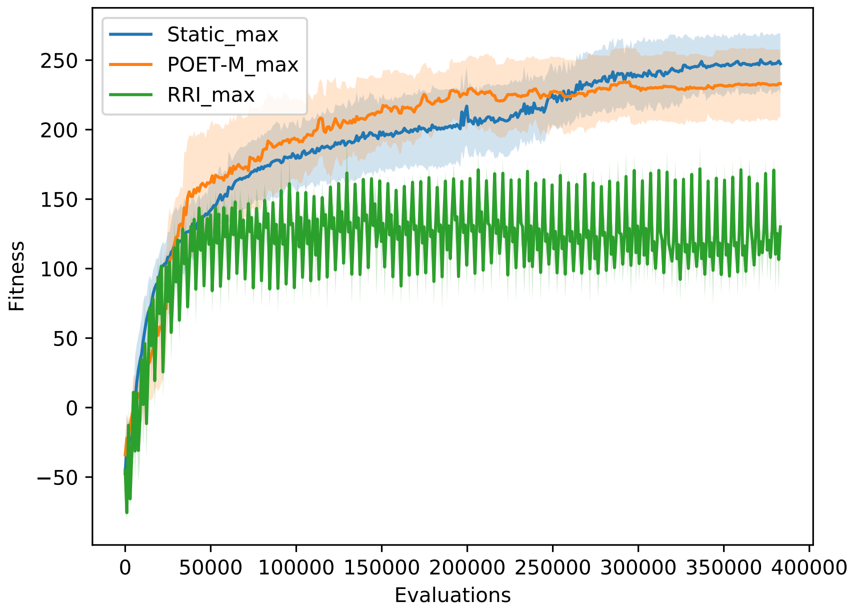


Figure 4.10: **Top:** The morphological change M^{20} for the agents evolved in POET-M (orange), the round robin incremental curriculum (green), and the baseline (blue). The graphs are averages of five runs, and the coloured areas show the standard deviation. **Bottom:** Diversity of the resulting populations at the end of the runs, for the baseline (Static), round robin incremental (RRI) and POET-M. The population diversity is measured as described in section 4.1.2.



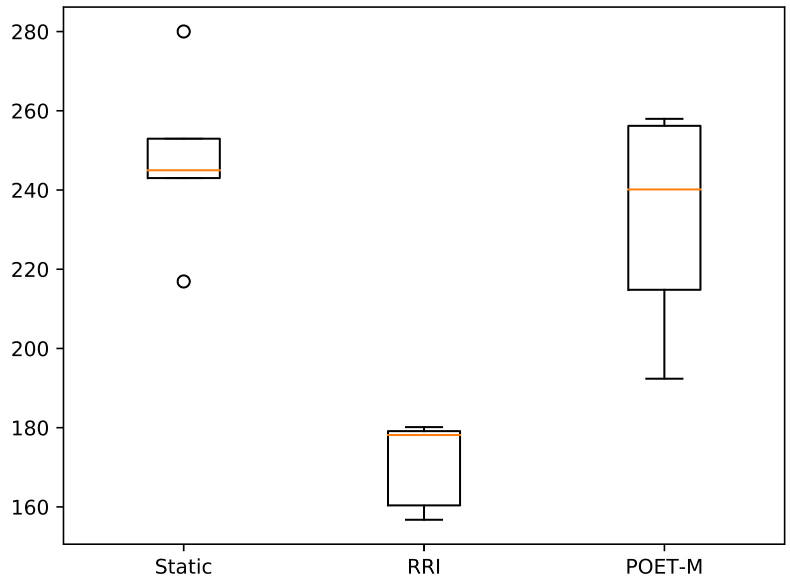


Figure 4.11: **Top:** Fitness of the top performing individual for the agents evolved in POET-M (orange), the round robin incremental curriculum (green), and the baseline (blue). The graphs are averages over five runs, and the coloured areas show standard deviation. **Bottom:** Fitness of the best individuals from resulting populations at the end of the runs, for the baseline (Static), round robin incremental (RRI) and POET-M.

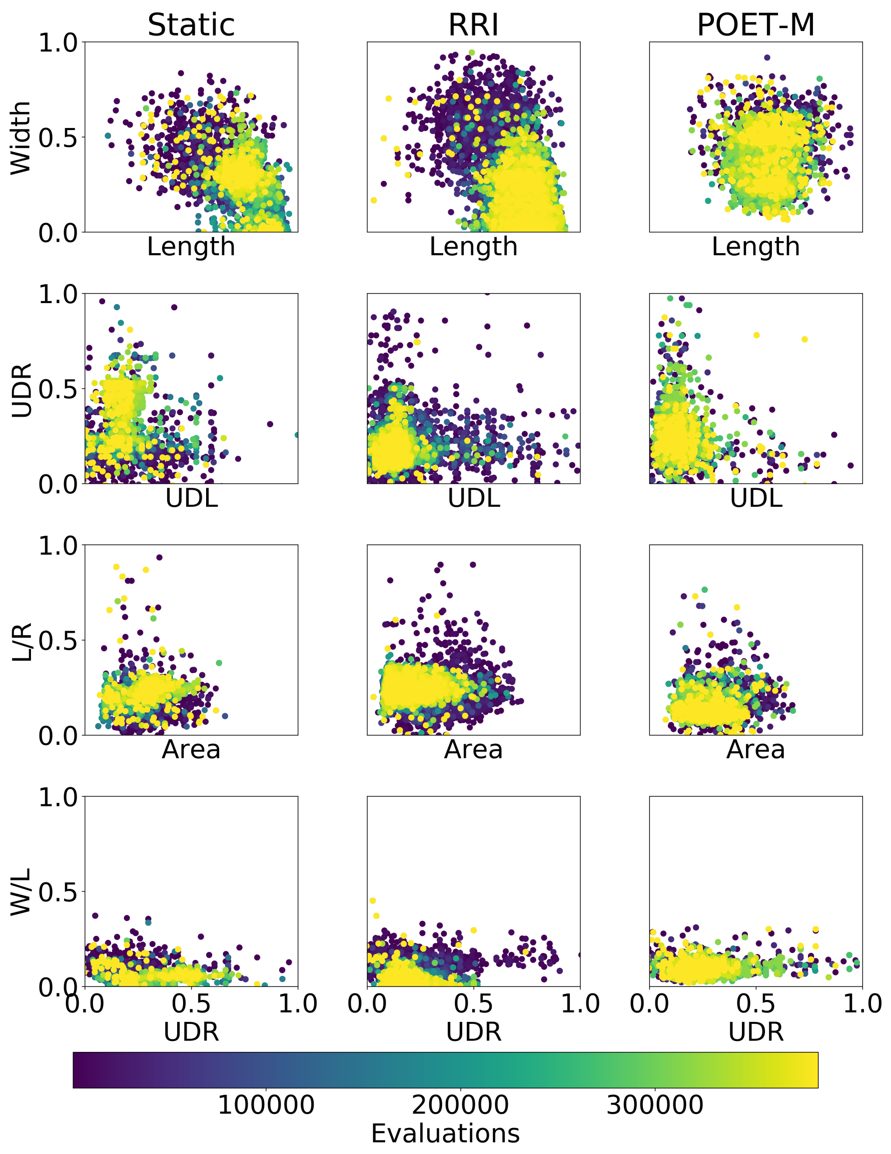


Figure 4.12: Morphological feature maps, as described in section 4.1.3, for the baseline (Static), the round robin incremental curriculum (RRI) and POET-M.

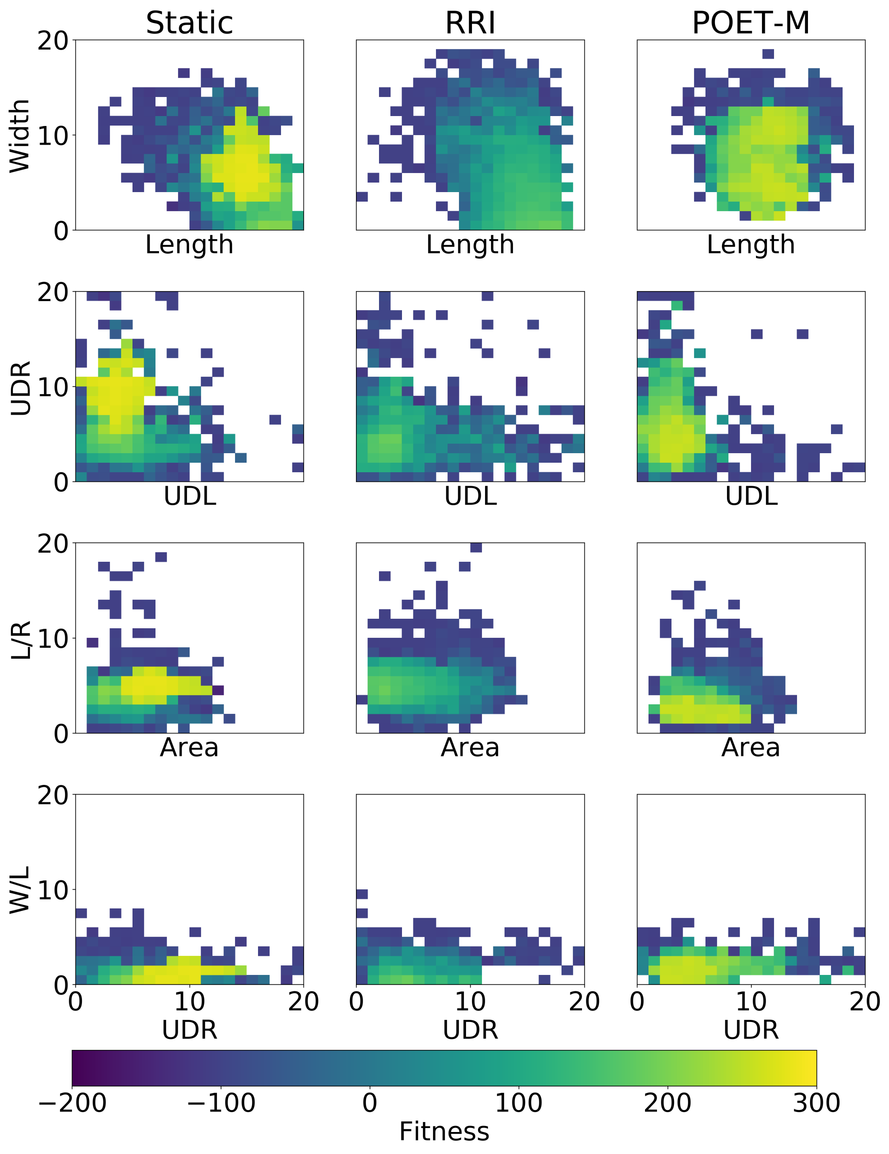


Figure 4.13: Quality-diversity feature maps, as described in section 4.1.3, for the baseline (Static), the round robin incremental curriculum (RRI) and POET-M.

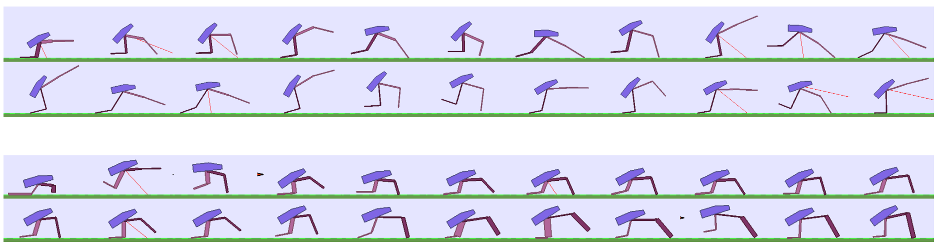


Figure 4.14: The top performing morphology captured every 40 generations. **Top:** The morphologies from the baseline. **Bottom:** The morphologies from POET-M.

Chapter 5

Conclusion and Future Work

5.1 Discussion

5.1.1 Computational budget

Both experiment 2 and experiment 3 could have benefited greatly from more time and a larger computational budget. POET-M was run on Sigma2, UNINETT's supercomputer, on 40-cpu nodes. The POET-M runs spent up to two weeks to reach 1000 generations, depending on how many environments were created. Because the POET-M runs took so long to complete, we were only able to perform five runs of each type. The statistics would have been more accurate if we had been able to perform more than five runs.

If we had more time we could also run both POET-M and the runs from the other experiments for longer. Only one of the POET-M runs had reached maximum environment population size when it terminated. As POET-M's population size increases the agents have more environments to transfer between. Reaching a higher environment population size might have enabled POET-M to create even better environments, as it would have more environments to choose from. It would also have been interesting to see whether the POET-M agents' fitness would have continued to increase beyond the fitness of the baseline, after the baseline converged.

Another interesting way to spend more time could have been to test other types of curricula in experiment 2, to get more insight into what kind of curricula is beneficial when attempting to increase diversity and quality in the population. By running the round robin and round robin incremental curricula agents for longer it may become easier to see any potential differences between these curricula.

5.1.2 Poet dynamics

In our experiments evolving agents with POET-M seems to increase the population's morphological diversity. However, as the curricula the agents experience is different for each run of POET-M, it is difficult to know what the source of the increased morphological diversity is. An interesting experiment that could be performed in order to find out more about what causes the diversity could be to evolve an agent in the same sequence of environments that a POET-M agent experienced. This would not necessarily create the same effect, as the most effective curricula may be different based on what random perturbations happens to the morphologies and controllers in the mutation steps.

We believe the diversity in the POET-M agents may come from a bias towards robust agents. When POET-M transfers agents, the agents are copied, and the transferred agent is thus duplicated. Agents that are more robust to environmental changes are more likely to be transferred, as they drop less in fitness when they are introduced to a new environment. Agents that have a diverse population may be more robust to environmental changes, as having a diversity of different solution makes it more likely one of the solutions will not drop as much in fitness when the environment changes. Because of this the selection of agents to be transferred can cause a selection pressure favouring agents with diverse populations.

5.2 Conclusion

In this thesis we explored the possibility of using environmental change to encourage morphological diversity in a population when co-evolving the morphology and controller of locomoting agents. When evolving controller and morphology at the same time, premature convergence of the morphology is a common problem, and increasing the morphological diversity in a population is likely to slow down the convergence.

We performed three experiments each pertaining to one of the following research questions:

- How does changes in environments affect the morphological development of locomoting agents?
- What impact does increasing the difficulty of the environments, as the agent solves them, have on the morphological development of locomoting agents?
- How effective is POET-M in maintaining both quality and morphological diversity in a population, compared to populations evolved in a static environment or a hand crafted curriculum?

In the first experiment we found that changing the environment greatly affected the morphological development of the agents. The agents experienced increased morphological change as a reaction to the changes in the environment. We conclude that this morphological change could be caused by the candidates adapting their morphology to the new environment. However, another plausible explanation for the increased exploration of morphologies is lower selection pressure. When the high fitness candidates are introduced to a new environment their fitness drops for a while until they adapt to the changes. This can allow previously dominated candidates to reproduce and spread their genetic material.

In the second experiment we tested two curricula one with and one without increasing difficulty, expecting the increasing difficulty of environments to induce increased morphological change. However, we found no indication of a direct connection between increasing difficulty and morphological change.

In the third experiment we attempted to use our modified version of POET, POET-M, to automatically create a curriculum of environments. Wang et al. [23] showed that POET could enable agent controllers to escape local optima, and we expected this effect to also help morphologies escape local optima. We found that evolving the agents with POET-M increased the morphological diversity of the agents, and encouraged the agents to find high fitness solutions for a diversity of different morphological features. Diversity can help slow down convergence, enabling the search to explore multiple peaks in the search space, rather than stopping after converging to the top candidate on one peak.

It is very difficult to create a perfect environment curriculum by hand, and the curricula in experiment 2 were not capable of increasing the morphological diversity of the population, instead proving to be detrimental to the agent fitness. POET-M finds a curriculum automatically by searching curricula space with a novelty search algorithm, removing the need to search for a good curriculum by hand. Our results show that curricula created automatically with algorithms such as POET-M are quite good at maintaining the balance between diversity and quality.

5.3 Future Work

There are several interesting directions to explore in future work. Running POET-M until it reaches the most difficult environments takes a long time. If we had a larger computational budget we would have liked to run the algorithms for longer to see whether POET-M surpasses the baseline in fitness if given enough time, and also to see which morphologies POET-M would eventually converge to. We would also have liked to test the algorithms on other platforms, such as a on a three dimensional walker, or on modular robots.

In experiment 2 we did not find a connection between morphological change and curricula with increasing difficulty. An interesting path of research might be to test other curricula, and explore in depth what types of curricula encourage diversity and quality.

Another interesting direction could be to explore methods for automatically creating curricula without the time consuming processes used by POET-M. Perhaps information about the population such as diversity or morphological change could be used to predict which environments will be good stepping stones, and which should be abandoned.

Bibliography

[1] S. Wang, W Chaovalitwongse and R Babuska, 'Machine learning algorithms in bipedal robot control', eng, *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 5, pp. 728–743, 2012, ISSN: 1094-6977.

[2] X. Liu, A. Rosendo, S. Ikemoto, M. Shimizu and K. Hosoda, 'Robotic investigation on effect of stretch reflex and crossed inhibitory response on bipedal hopping', *Journal of the Royal Society Interface*, vol. 15, no. 140, 2018, ISSN: 1742-5689.

[3] J. H. Nordmoen, T. F. Nygaard, K. O. Ellefsen and K. Glette, 'Evolved embodied phase coordination enables robust quadruped robot locomotion', in, Association for Computing Machinery (ACM), 2019.

[4] M. J. Spenko, G. C. Haynes, J. A. Saunders, M. R. Cutkosky, A. A. Rizzi, R. J. Full and D. E. Koditschek, 'Biologically inspired climbing with a hexapedal robot', eng, *Journal of Field Robotics*, vol. 25, no. 4-5, pp. 223–242, 2008, ISSN: 1556-4959.

[5] K. Sims, 'Evolving virtual creatures', eng, in *Proceedings of the 21st annual conference on computer graphics and interactive techniques*, ser. SIGGRAPH '94, ACM, 1994, pp. 15–22, ISBN: 0897916670.

[6] D. Ha, 'Reinforcement learning for improving agent design', *Artificial Life*, vol. 25, no. 4, pp. 352–365, 2019, ISSN: 1064-5462.

[7] N. Cheney, J. Bongard, V. Sunspiral and H. Lipson, 'On the difficulty of co-optimizing morphology and control in evolved virtual creatures', *Artificial Life Conference Proceedings*, no. 28, pp. 226–233, 2016. DOI: 10.1162/978-0-262-33936-0-ch042.

[8] N. Cheney, J. Bongard, V. Sunspiral and H. Lipson, 'Scalable co-optimization of morphology and control in embodied machines', eng, *Journal of the Royal Society, Interface*, vol. 15, no. 143, 2018, ISSN: 17425689.

[9] G. S. Hornby, 'Alps: The age-layered population structure for reducing the problem of premature convergence', in *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '06, Seattle, Washington, USA: Association for Computing Machinery, 2006, 815–822, ISBN: 1595931864. DOI: 10.1145/1143997.1144142.

[10] M. Jelisavcic, K. Glette, E. Haasdijk and A. E. Eiben, 'Lamarckian evolution of simulated modular robots', *Frontiers in Robotics and AI*, vol. 6, p. 9, 2019, ISSN: 2296-9144. DOI: 10.3389/frobt.2019.00009.

[11] H. Lipson and J. B. Pollack, 'Automatic design and manufacture of robotic lifeforms', *Nature*, vol. 406, no. 6799, p. 974, 2000, ISSN: 0028-0836.

[12] T. Geijtenbeek and N. Pronost, 'Interactive character animation using simulated physics: A state-of-the-art review', *Computer Graphics Forum*, vol. 31, no. 8, pp. 2492–2515, 2012, ISSN: 0167-7055.

[13] G. Hornby, H Lipson and J. Pollack, 'Evolution of generative design systems for modular physical robots', eng, in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, vol. 4, IEEE, 2001, 4146–4151 vol.4, ISBN: 0780365763.

[14] J. Lehman and K. O. Stanley, 'Evolving a diversity of virtual creatures through novelty search and local competition', in *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '11, Dublin, Ireland: Association for Computing Machinery, 2011, 211–218, ISBN: 9781450305570. DOI: 10.1145/2001576.2001606.

[15] J. E. Auerbach and J. C. Bongard, 'Environmental influence on the evolution of morphological complexity in machines', *PLoS Computational Biology*, vol. 10, no. 1, 17. e1003399, 2014. DOI: 10.1371/journal.pcbi.1003399.

[16] P. S. Oliveto, D. Sudholt and C. Zarges, 'On the benefits and risks of using fitness sharing for multimodal optimisation', eng, *Theoretical Computer Science*, vol. 773, pp. 53–70, 2019, ISSN: 0304-3975.

[17] D. Goldberg and J Richardson, 'Genetic algorithms with sharing for multimodal function optimization', in *Genetic algorithms and their applications: proceedings of the second International Conference on Genetic Algorithms: July 28-31, 1987 at the Massachusetts Institute of Technology, Cambridge, MA*, Hillsdale, NJ: L. Erlhaum Associates, 1987., 1987.

[18] L. Trujillo, G. Olague, E. Lutton, F. Fernández de Vega, L. Dozal and E. Clemente, 'Speciation in behavioral space for evolutionary robotics', eng, *Journal of Intelligent and Robotic Systems*, vol. 64, no. 3-4, pp. 323–351, 2011, ISSN: 0921-0296.

[19] O. J. Mengshoel and D. E. Goldberg, 'The crowding approach to niching in genetic algorithms', *Evolutionary Computation*, vol. 16, no. 3, pp. 315–354, 2008, PMID: 18811245. DOI: 10.1162/evco.2008.16.3.315.

[20] J. Lehman and K. O. Stanley, 'Abandoning objectives: Evolution through the search for novelty alone', *Evol. Comput.*, vol. 19, no. 2, pp. 189–223, Jun. 2011, ISSN: 1063-6560. DOI: 10.1162/EVCO_a_00025.

[21] N. Packard, M. A. Bedau, A. Channon, T. Ikegami, S. Rasmussen, K. O. Stanley and T. Taylor, 'An overview of open-ended evolution: Editorial introduction to the open-ended evolution ii special issue', *Artificial Life*, vol. 25, no. 2, pp. 93–103, 2019, ISSN: 1064-5462.

[22] J. C. Brant and K. O. Stanley, 'Minimal criterion coevolution: A new approach to open-ended search', in *Proceedings of the Genetic and Evolutionary Computation Conference*, ser. GECCO '17, Berlin, Germany: Association for Computing Machinery, 2017, 67–74, ISBN: 9781450349208. DOI: 10.1145/3071178.3071186.

[23] R. Wang, J. Lehman, J. Clune and K. O. Stanley, 'Poet: Open-ended coevolution of environments and their optimized solutions', in *Proceedings of the Genetic and Evolutionary Computation Conference*, ser. GECCO '19, Prague, Czech Republic: Association for Computing Machinery, 2019, 142–151, ISBN: 9781450361118. DOI: 10 . 1145 / 3321707.3321799.

[24] Á. E. Eiben and J. E. Smith, 'Evolutionary algorithms', in *Handbook of Memetic Algorithms*, F. Neri, C. Cotta and P. Moscato, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 9–27, ISBN: 978-3-642-23247-3. DOI: 10.1007/978-3-642-23247-3_2.

[25] D. L. Hudson and M. E. Cohen, 'Genetic algorithms', eng, in *Neural Networks and Artificial Intelligence for Biomedical Engineering*, Hoboken, NJ, USA: John Wiley & Sons, Inc., 2012, pp. 215–224, ISBN: 9780780334045.

[26] K. D. Jong, 'Evolutionary computation', eng, *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 1, no. 1, pp. 52–56, 2009, ISSN: 1939-5108.

[27] N. Hansen, 'The CMA Evolution Strategy: A Tutorial', *arXiv e-prints*, arXiv:1604.00772, arXiv:1604.00772, 2016.

[28] D. E. Rumelhart, G. E. Hinton and R. J. Williams, 'Learning representations by back-propagating errors', *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[29] D. Floreano, P. Dürr and C. Mattiussi, 'Neuroevolution: From architectures to learning', *Evolutionary Intelligence*, vol. 1, no. 1, pp. 47–62, 2008, ISSN: 1864-5917. DOI: 10.1007/s12065-007-0002-4.

[30] J. Schaffer, D Whitley and L. Eshelman, 'Combinations of genetic algorithms and neural networks: A survey of the state of the art', eng, in *[Proceedings] COGANN-92: International Workshop on Combinations of Genetic Algorithms and Neural Networks*, IEEE Comput. Soc. Press, 1992, pp. 1–37, ISBN: 0818627875.

[31] K. O. Stanley and R. Miikkulainen, 'Evolving neural networks through augmenting topologies', *Evol. Comput.*, vol. 10, no. 2, pp. 99–127, Jun. 2002, ISSN: 1063-6560. DOI: 10.1162/106365602320169811.

[32] K. O. Stanley, D. B. D'Ambrosio and J. Gauci, 'A hypercube-based encoding for evolving large-scale neural networks', *Artificial Life*, vol. 15, no. 2, pp. 185–212, 2009, PMID: 19199382. DOI: 10.1162/artl.2009.15.2.15202.

[33] K. Stanley, 'Compositional pattern producing networks: A novel abstraction of development', eng, *Genetic Programming and Evolvable Machines*, vol. 8, no. 2, pp. 131–162, 2007, ISSN: 1389-2576.

[34] N. Cheney, J. Clune and H. Lipson, 'Evolved electrophysiological soft robots', in *Artificial Life Conference Proceedings 14*, MIT Press, 2014, pp. 222–229.

[35] E. Hupkes, M. Jelisavcic and A. Eiben, 'Revolve: A versatile simulator for online robot evolution', vol. 10784, Springer Verlag, 2018, pp. 687–702, ISBN: 9783319775371.

[36] E. Samuelsen, K. Glette and J. Torresen, 'A hox gene inspired generative approach to evolving robot morphology', eng, in *Proceedings of the 15th annual conference on genetic and evolutionary computation*, ser. GECCO '13, ACM, 2013, pp. 751–758, ISBN: 9781450319638.

[37] T. Nygaard, C. Martin, E. Samuelsen, J. Torresen and K. Glette, 'Real-world evolution adapts robot morphology and control to hardware limitations', eng, in *Proceedings of the Genetic and Evolutionary Computation Conference*, ser. GECCO '18, ACM, 2018, pp. 125–132, ISBN: 9781450356183.

[38] G. B. Parker and P. J. Nathan, 'Concurrently evolving sensor morphology and control for a hexapod robot', eng, in *IEEE Congress on Evolutionary Computation*, IEEE, 2010, pp. 1–6, ISBN: 9781424469093.

[39] G. Parker and P. Nathan, 'Evolving sensor morphology on a legged robot in niche environments', eng, in *2006 World Automation Congress*, IEEE, 2006, pp. 1–8, ISBN: 1889335339.

[40] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang and W. Zaremba, *Openai gym*, 2016.

[41] R. S. Sutton, D. A. McAllester, S. P. Singh and Y. Mansour, 'Policy gradient methods for reinforcement learning with function approximation', in *Advances in neural information processing systems*, 2000, pp. 1057–1063.

[42] K. Miras and A. E. Eiben, 'Effects of environmental conditions on evolved robot morphologies and behavior', in *Proceedings of the Genetic and Evolutionary Computation Conference*, ser. GECCO '19, Prague, Czech Republic: ACM, 2019, pp. 125–132, ISBN: 978-1-4503-6111-8. DOI: 10.1145/3321707.3321811.

[43] J.-B. Mouret and J. Clune, 'Illuminating search spaces by mapping elites', *arXiv preprint arXiv:1504.04909*, 2015.

[44] T. Salimans, J. Ho, X. Chen, S. Sidor and I. Sutskever, 'Evolution strategies as a scalable alternative to reinforcement learning', *arXiv preprint arXiv:1703.03864*, 2017.

[45] E. Samuelsen and K. Glette, 'Some distance measures for morphological diversification in generative evolutionary robotics', eng, in *Proceedings of the 2014 Annual Conference on genetic and evolutionary computation*, ser. GECCO '14, ACM, 2014, pp. 721–728, ISBN: 9781450326629.

[46] K. Miras, E. Haasdijk, K. Glette and A. E. Eiben, 'Search space analysis of evolvable robot morphologies', in *Applications of Evolutionary Computation*, K. Sim and P. Kaufmann, Eds., Cham: Springer International Publishing, 2018, pp. 703–718, ISBN: 978-3-319-77538-8.