Investigating Morphological Trade-offs in Evolutionary Optimization of Robot Morphology and Control

Steinar Bøe

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

Department of Informatics
The Faculty of Mathematics and Natural Sciences

UNIVERSITY OF OSLO

Spring 2022

Investigating Morphological Trade-offs in Evolutionary Optimization of Robot Morphology and Control

Steinar Bøe

Investigating Morphological Trade-offs in Evolutionary Optimization of Robot Morphology and Control

http://www.duo.uio.no/

Printed: Reprosentralen, University of Oslo

Abstract

Robots need to be able to adjust to their environments. There will always be features or changes to the environments not anticipated by the designers. Developing robots for the exact specifications of the right environment and the task they will face is not intuitive. The field of evolutionary robotics, with the inspiration of Darwin's theory of evolution, attempts to construct artificial evolution capable of evolving robots adapted to these specifications. For this to be possible, the evolutionary algorithm needs to consider all interacting features. This includes both the control and the body of the robot. However, most research on evolutionary robotics does not evolve the robot body.

This thesis emphasizes that evolving control and morphology together are essential for a good environment adaptation. It demonstrates and investigates the possibilities of morphological adaptation to a fitness goal. This is done by co-optimizing both control and morphology for a 4-legged simulated spider robot with speed and stability as a multi-objective optimization problem. After extensive evolutionary training, the results show a significantly strong correlation between several morphology parameters and the fitness trade-off.

Acknowledgement

I want to express my sincere gratitude to my main supervisor, associate professor Tønnes Nygaard, for all the guidance and support throughout the work on this thesis. I am extremely grateful for your help. I would also like to thank my second supervisor, Professor Kyrre Glette, for all the feedback and guidance.

Also, a huge thank you to my fellow students, family, and friends for their endless support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

When physical robots are exposed to the real world they need to tackle different types of changing environments and obstacles. Each robot needs to be designed specifically for the challenges it will face. When developing robots today, we take inspiration from nature. We try to replicate the creatures found in nature and their abilities. We see what works and we can take advantage of it when designing our own solutions.

As an outcome of an enormously long natural evolution, we see animals all over the world highly adapted to their specific environments. We can find sahara desert ants tolerating extreme heat, emperor penguins surviving in the most ruthless arctic conditions, jellyfish that can reverse their life cycle, and some bacteria surviving all environments found on earth. We believe all life on earth steams from the same microscopic bacterial lifeforms, and have evolved over billions of years to become these highly specified creatures we see today.

These are all very inspiring, however, none of these animals have evolved to the tasks we want our robots to comprehend. The still small, but highly promising field of Evolutionary Robotics (ER) has a goal of doing this with robots. Instead of adopting the results from evolution, this field tries to adopt the process itself. While the fittest and most adaptable animals have survived and passed along their genes to continue evolution, we can set the premises of fitness for our robots and expose them to similar selection pressure. The surviving robots will pass along their genes to create the best robots for walking, flying, swimming, or the best ones at crossword puzzles, chess, car manufacturing, or stock predictions.

The long-term goal of ER is to develop and adapt robots that can outperform the best human-designed robots. Even autonomous systems producing such high-quality robots by themselves might be possible. One of the main advantages of ER is the possibility of exploiting the control, the body, the environment, and their interactions in ways not found by engineering. Using ER for creating robots gives us a powerful method allowing multiple aspects of a robot design to be considered simultaneously. However, most robots developed by these techniques

today are often limited to only evolving control, and not morphology. We believe that also evolving the robots' bodies is highly important during evolution. The performance of a robot will be determined by a lot of factors. The controller, the body, the environment, and their interactions are all crucial parts of the final product. A robot will be more adaptable to its task and environment if both the morphology and control can evolve.

So far, we have seen several exciting examples utilizing ER to achieve high-performing, adaptable controllers and morphologies in both simulations and reality [5, 24, 45, 54]. However, since one of the first evolutionary optimizations of both morphology and control, by Karl Sims in 1994 [54], the field has struggled to greatly exceed these results when co-evolving. One of the main challenges has been to avoid premature convergence in morphology. This is despite a great increase in computational power, and many new searching techniques. Some studies have suggested that this is caused by an underestimation of the morphological contribution to the final solution, and an underestimation of the complex interactions between control, morphology and environment [6, 7, 43].

1.2 Goal of this thesis

The goal of this thesis is to call attention to the importance of evolving the morphology along with the controller. We want to investigate the effect of evolving the robot body instead of just designing one by hand. By using a multi-objective fitness measurement, one could see if the morphology solutions tend to adjust to the trade-off between them. If this is the case, we know the morphology adapts to the goal at hand during an evolution. Previous studies have shown that evolving morphology can be beneficial, but we could not find any studies that really look if or how the morphology parameters adjust to a fitness trade-off. Therefore, this is our hypothesis:

> $Hypothesis(H_1)$: Co-evolution of morphology and control on a four-legged spider robot will produce individuals with a fitness-related trade-off in morphology.

To prove or disprove our main hypothesis, we need a conflicting null hypothesis: H_0: None of the morphology parameters will show a fitness-related trade-off.

We want to test this null hypothesis and will consider it rejected if we find a statistically significant correlation between one of the morphology parameters and the fitness. Since we run a statistical test on many of the parameters, the p-value requirement needs to be lowered accordingly. For this thesis, a p-value less than 0.01 will be considered significant. If we can produce results rejecting this null hypothesis, it would strengthen the alternative, our main hypothesis.

This would mean including morphological evolution is important. One important direct effect of including it in evolution would be to find the best bodies to the according fitness trade-off. We could also use the results to shrink and/or adjust the parameter ranges to only the essential part.

Making the search space smaller for further training. Also, using these results as a template to handpick great morphological solutions to the specific fitness could be possible.

1.3 Contributions

The main contribution from this thesis is to show that adapting morphologies are possible. This is shown in the results, which suggest a strong correlation between morphological trade-offs and the multi-objective fitness function. Also, this thesis shows that these morphological adaptations can be done while co-evolving with the control. This is essential to emphasize since we know there is a complex interplay between control, morphology, and environment. This means that including morphology during evolution is both important and preferable.

1.4 Outline

This thesis consists of 6 chapters: Introduction, background, tools and frameworks, implementation, experiments and results, discussion, and conclusion.

In chapter 2, the background, there is a general introduction and motivation for this field, an overview of the difficulties and challenges this field faces, as well as an overview of previous relevant research.

The third chapter introduces the software programs and tools used in the experiments. Both the simulation engine and the control system are introduced here. Chapter 4 describes the system implementation. An overview of the simulation framework, the robot design, and the evolutionary setup is also presented. The experiments conducted are presented in chapter 5, along with the results.

The second to last chapter contains a general discussion on the overall results, a deeper analysis of the experiments, and a section with suggestions for future work. Finally, you find the conclusion.

Chapter 2

Background

This chapter introduces evolutionary algorithms with some typical applications and techniques. Then a more in-depth overview of the evolutionary robotics field is presented.

2.1 Evolutionary algorithms

Evolutionary algorithms (EAs) are optimization strategies inspired by biological evolution, which have become increasingly popular and important over the last decades [55]. EAs have proven themselves to be a great optimizing strategy for a range of problems with varying complexity.

This approach is inspired by Darwinian evolution, where random solutions are initialized and biologically inspired mechanisms are used to search for better solutions. This makes EA a guided random search technique. Typically, a population of individuals is exposed to selection pressure where the strongest and most fit individuals are more likely to be selected for breeding the next generation.

The most popular subversion of EAs is genetic algorithms (GAs). This method is often used in machine learning and robotics among many others. GAs use biological inspired operators like mutation and recombination to generate new solutions.

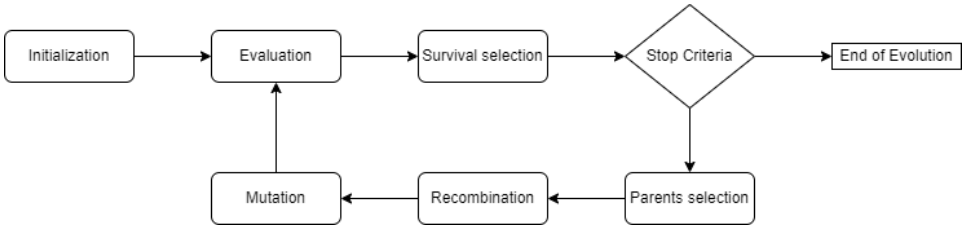


Figure 2.1: An overview of the standard process of a genetic algorithm

2.1.1 Process

A schematic of the general process of genetic algorithms can be seen in 2.1. At first, a population of solutions is initialized at random. The following

steps then go in a loop for each generation.

Each solution in the population is applied to the problem and evaluated. After evaluation, they all receive a fitness score according to how well they performed. A parent selection is done based on the fitness values, and the selected solutions get to breed new solutions through recombination and/or mutation. The new population is created from a survival selection of the most fit solutions from parents and the offspring.

This is one iteration of the EA, and it continues to cycle until we have a stopping criterion that terminated the search. This criterion could be achieved when the evaluation has a good enough fitness, it has reached the max number of generations, or we have too many generations without a sufficient improvement in fitness.

2.1.2 Genotypes and Phenotypes

In a genetic algorithm each solution is usually represented by a set of parameters. This set is called the genotypes, and each value in the genotype is an allele. The genotypes describe and can be decoded to the actual solution. The genotype is a digital representation of the solution, either binary or an array of numbers. The actual search during optimization happens in the genotype space, by tuning and changing these parameter values. When actually evaluating a solution, we need to expose the solution to the given task at hand and get back a fitness score. The genotypes represent these solutions, called phenotypes. When training a robot to walk, for instance, the robot itself will operate in phenotype space, while the parameters representing it operates in genotypes pace.

2.1.3 Exploration vs Exploitation

When running an EA it uses a guided random search for new solutions. Most search spaces are very rugged and difficult to navigate through. It is easy to get stuck at local optima, even though much greater parameters exist. To avoid this, it is important to choose good parameters which are neither too explorative nor too exploitative for this problem. By exploring an unknown search terrain, we will learn more about it as a whole. Exploiting is a more unfair use of resources to further investigate known areas of the search space and try to tune good solutions even better.

2.1.4 Generating new solutions

To generate new solutions and continue to improve the overall population in a GA, we need to introduce new solutions through recombination/crossover and/or mutate already generated solutions. This is how we do a random change to the population.

Crossover

Crossover or recombination is a way of breeding new solutions into the population from current solutions. These solutions are based on one or

two of the parent solutions. One of the simplest and most used crossover methods for floating-point numbers is the uniform crossover. Given two parent solutions and a mutation probability P, there is a chance P for each allele in the offspring to be equal to the allele at the same position as parent1 and a chance of 1-P for it to be equal to parent2's allele at this position. This results in two offspring solutions, which would be a mix of the two parents. A P equal to 0.5 will cause large exploration, while a low or high probability will create children very equal to their parents and a lower exploration is happening. An example of uniform crossover can be seen in figure 2.2.

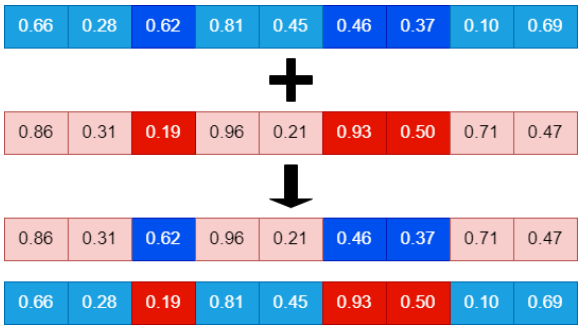


Figure 2.2: Uniform crossover example: The two parents at top get recombined into the two children. In this example the third, sixth and seventh alleles get switched.

Mutation

Mutation usually happens after we have generated new solutions, so a mutation might also happen to the newly generated solutions. There are different tactics for doing mutation on an individual. In general, they are small changes to parts of a solution with a certain probability of occurring. Which mutation method is chosen for a specific problem is depended on the solution representation. There are also different mutation algorithms for binary values, discrete integers, and floating-point numbers.

In 2.3 we see an example of a Gaussian mutation. This is a much-used mutation for floating-point numbers. If this mutation occurs, each number in the gene is added with a random number picked from a Gaussian distribution with zero mean. The standard deviation for this distribution is set at a fixed value set as a hyperparameter before the training starts. Usually, this mutation is done with a low value for standard deviation, making the algorithm very exploitative.

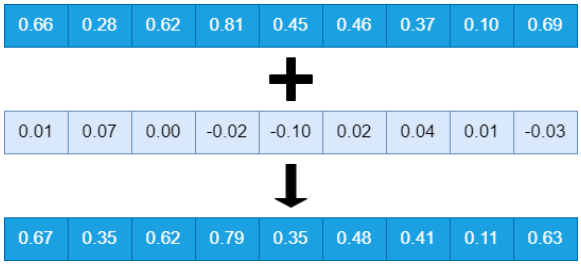


Figure 2.3: Gaussian mutation. The original gene (at the top) is added with random numbers drawn from a Gaussian distribution, creating the offspring (at the bottom).

Another highly used mutation algorithm is the random reset mutation. With a low probability of occurring, this mutation algorithm replaces one, or multiple, values from the gene with a new random value. Usually, this is used with a very low probability, as it is purely exploratory.

2.1.5 Selection

In each evolutionary run, we usually have two selections. One selection for choosing which individuals used for generating new solutions, a parent selection, and a selection determined which solutions should survive to the next generation. There is a range of different strategies on how these selections should unfold. Most algorithms will in general give better solutions a better chance of being selected. Some algorithms also use elitism, where only the best solutions are selected.

2.1.6 Fitness

After an individual is evaluated, they are given a fitness score. Different ways of calculating this fitness score will determine how the solutions develop. The algorithm will always try to achieve the greatest fitness score, without knowing the fitness function itself. When using EAs for robot development, a lot of research is about learning more about the process itself and how we can further develop it. Therefore, the finesse for robot development is often quite simple. Common fitness functions for walking robots will often consider traveled distance, speed, acceleration, stability, or weight.

2.1.7 Multi-objective optimization

The use of multi-objective optimization is very popular for genetic algorithms [26]. Some problems are multi-objective by nature, but several studies have also shown great benefits of using multi-objective optimization over single-objective in evolutionary robotics [42, 52]. A great benefit of multi-objective evolutionary algorithms (MOEAs) is greater diversity maintenance. The population will consist of outlying individuals great at only one of the objectives, and a frontier of individuals between

these solutions. Having this population diversity makes the evolution better at overcoming premature convergence.

When optimizing multiple objectives at once, it is not so straightforward to sort the individuals after fitness. We will no longer have a one-dimensional list, but rather several fronts of solutions. All solutions in a front will be considered equally good. The best solutions will belong to the Pareto front. An example of a population with two objectives, with an outlined Pareto front, can be seen in figure 2.4. There will be no other solutions that have a better fitness in both of the measurements than those that lie on the Pareto front. When investigating the solutions: An increase in one fitness will cause a decrease in the second fitness.

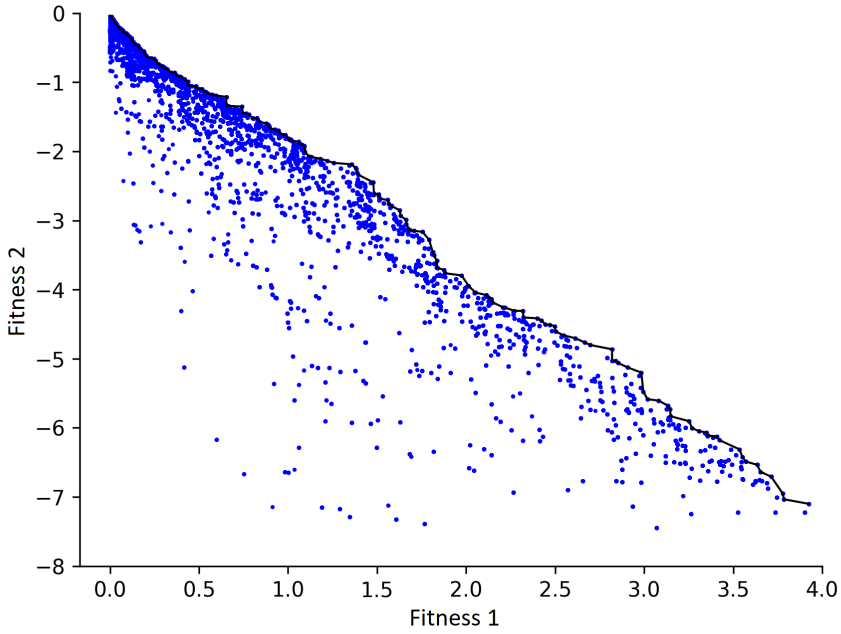


Figure 2.4: An example of a population trained by a multi-objective optimization algorithm. The blue dots in the plot represent solutions with both a score in fitness one and two. The black line consists of the Pareto optimal solutions in this population

NSGA-II

One of the most popular MOEAs, proven to find good results for multi-objective optimization, is the elitist NSGA-II algorithm [10]. It does selection by Pareto front optimization.Figure 2.5 gives an overview of the selection process, and an explanation of it as follows:

1. A Non-dominated sorting of the population occurs. This will divide the population into different front ranks according to their dominance.

2. The population will be made up of the best half of the old population. This is chosen by front-ranking.

3. When an entire front cannot fit into the new population, the solutions in the particular front get sorted by crowding distance. This is the average distance to a solution's two nearest neighbors. Solutions with large crowding distances are prioritized. When the entire new population is selected, the rest is rejected.

4. We then have a new set of parents. The rest of the population will consist of offspring from mutation and recombination of these parents.

Because of the elitism, this algorithm is quite exploitative. Just being a multi-objective optimization helps with exploration, but it often needs mutation or recombination to contribute more to the exploration part. Despite the NSGA-II being very good at two and three objective optimizations, it struggles when being introduced to higher dimensions. For this reason, an extension, NSGA-III, has also been developed [21].

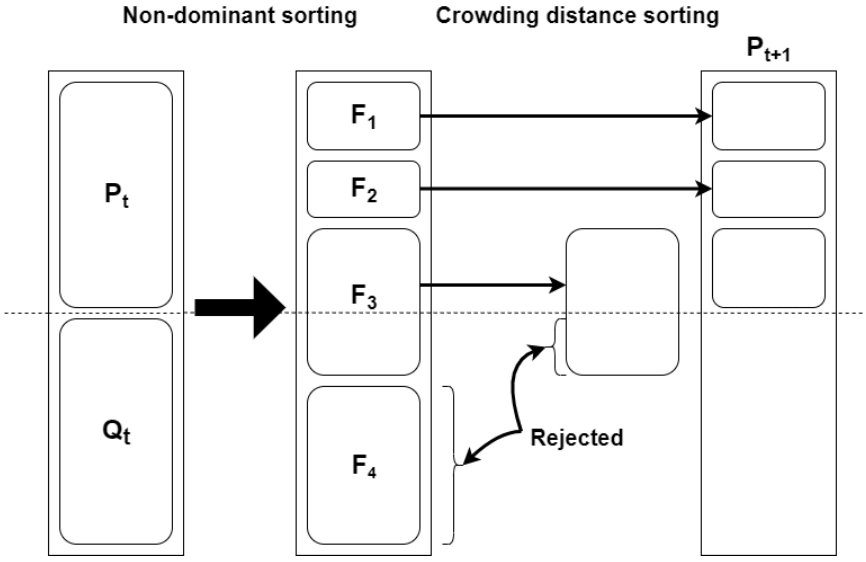


Figure 2.5: Overview of the selection done by NSGA-II

2.2 Performance evaluation

When researching a topic, one should not make too many claims without statistically significant results. One can discuss and come up with suggestions, but one does need statistical evidence to confirm or disprove a theory.

Evolutionary algorithms have a stochastic nature. The random initialization and the randomness during mutating and recombination will have a lot to say for the performance. Depending on the task at hand, one

might be looking to find only one really good solution, and therefore train with large exploration. In these cases, one does not care about the average performance, but when doing research on different methods, comparing algorithms or different sets of hypervolume, the average performance will have everything to say. Since randomness can cause large differences between each run, it is necessary to conduct a lot of runs with different seeds to provide statistical confidence. A general analysis of the data is needed before we can confidently say something about the features of the specific evolution setup.

Hypothesis testing

Hypothesis testing is a statistical inference method to test if a set of data can support a theory. When doing this, two conflicting two hypotheses are present. The null hypothesis, H_0, which we usually want to compute evidence against, and an alternative hypothesis, H_A, which we usually want the data to favor. For example, the null hypothesis expresses that there is no significant statistical relationship between two variables. While H_A state there is. The observed data can be used to compute a test statistic, with a corresponding p-value. Assuming H_0 is true, this p-value represents the probability of finding more extreme values than the ones observed. Therefore, a small p-value means a low probability that these results happened by chance, and it is strong evidence against the null hypothesis.

Bonferroni correction

A problem that occurs when studying multiple statistical inferences simultaneously, is the multiple comparison problem. If we want to test a set of hypothesis tests, and we use a signification level of 5%. The probability of each of them being incorrectly rejected would be only 5%, but the probability that at least one of them will be incorrect can be much larger. The more comparisons we do, the greater the probability of at least one wrong rejection. There are several ways of counteracting this problem. The simplest and most conservative method is the Bonferroni correction. This method simply multiplies the p-values with the number of tests. The result of such a conservative method can give a much higher probability of rejecting a false null hypothesis.

Correlation

Correlation is an interesting statistical measure often used when investigating dependencies between two features. There are two main measurements of the correlation between variables; the Pearson's and Spearman's correlation [51]. They both give a number between -1 and +1, where a large negative value means they have a strong decreasing correlation, and a large positive value means a strong positive correlation. A value close to zero means there is no correlation between the variables. The difference

is however in what type of correlation they measure. Pearson correlation measures the linear relationship between two variables, while Spearman on the other hand evaluates the monotonic relationship. A monotonic relationship occurs when both values increase or decrease at the same time, without necessarily being a linear trend. A logarithmic function would not score high on the Pearson's correlation, but the less strict Spearman's Correlation would find a strong correlation. Their use depends on what correlation we expect and want to discover.

LOESS

To better visualize a dependency, a regression model can be used. LOESS (locally estimated scatterplot smoothing) is a regression model good at noisy scatterplots, which often is produced by the randomness in EAs. LOESS builds a smooth curve between variables by looking at each point in one area at the time. It is computational heavy since it combines the features of a linear and a nonlinear least-squares regression in one. The degree of polynomial can be set when using this function, making it possible to vary how detailed and precise it should behave.

2.3 Evolutionary Robotics

When using evolutionary algorithms to produce robot solutions, we enter the field of evolutionary robotics. Evolutionary robotics (ER) stands out from traditional robotics since the solutions are not designed by us, but instead, it lets the search algorithm optimize the solutions [11, 12]. When designing a robot and its control system in traditional robotics, there are a lot of things to consider simultaneously which are closely related and will affect one another. To predict the movement of a given body it is necessary to consider the position, orientation, velocity, acceleration motor force, and torque, among others. This is usually done through kinematical calculations and usually grows in complexity as more components are added [53].

The great benefit of exploiting biological inspired evolutionary algorithms is the possibility of avoiding the calculation of each component's effect, but instead consider the system as a whole. This way there is not always a requirement to gain full knowledge of the environment, the control policy, the morphology, or the task dynamics [37]. The robot is more like a black box. The algorithm set some inputs and tries to optimize the input based on the evaluation scores.

ER is a fairly new research field, and evolved robot controllers and morphologies are not always capable of outperforming human-designed solutions today. Most tasks being looked at in ER today are relatively simple; locomotion, gait development, and obstacle avoidance, to state a few. Such simple tasks are chosen to easier compare different techniques of ER, and further develop evolutionary strategies. It is anticipated solutions created from ER will be able to exceed human-designed solutions in the

long term. If we could do this for these simple tasks, it would most likely be a lot better for tackling more complex problems as well.

The complexity of the task and environment should not increase much before our human-designed solutions struggle to comprehend the complexity and fail to provide adequate solutions. ER has the potential to develop autonomous robots capable of automatically learning how to deal with more complex tasks in more complex environments and situations. Autonomous robots might also have the potential of dealing with challenges not anticipated by the designer. Therefore, there is great value in work aiming to improve already achieved tasks, with the prospects of finding more robust and better performing evolutionary processes. Thereby, making it easier to investigate and achieve solutions for more complex tasks [37].

2.3.1 Long term benefits of ER

When considering the long-term benefits of evolutionary robotics research, the most obvious is engineering accomplishments. ER can be looked at as a tool for producing better robot designs. ER might help provide solutions to more complex tasks, but also to find more energy-efficient controllers and body designs. It also has the potential of providing a broader understanding of the infamous reality gap (a problem occurring when transferring a robot from simulation to the real-world) [22, 62].

In addition to engineering, biology can also benefit from ER. Evolution is too slow to be observed in nature, without manipulating it [11]. This makes it demanding to investigate. As the theoretical evolutionary biologist and geneticist John Maynard Smith said; "So far, we have been able to study only one evolving system and we cannot wait for interstellar flight to provide us with a second. If we want to discover generalizations about evolving systems, we have to look at artificial ones" (Maynard-Smith, 1992) [33].

Further studies and experiments in evolutionary robotics can hopefully be able to answer questions and give contributions to biological evolutionary studies. There are several examples where ER has been used to study essential evolutionary questions already [11].

2.3.2 Co-evolution

Throughout the history of evolutionary robotics, most studies have been limited to optimizing only the control of robots, and mostly in software. However, there are many examples of those that use EAs to also optimize the bodies of the robots, most often by co-evolving the control and body simultaneously [7, 30, 54].

A great motivation for including morphological evolution is the theory of embodied cognition. This theory describes the cognitive resources as being more than just the brain [6, 59]. Here they describe that the body, the control, the environment, and their interactions, all might as well be a part of the cognitive resource. They state that an individual's

behavior is a consequence of a mixture of all its task resources, and without extensive analysis of all these resources and their interactions, it is hard to tell each resource contribution apart. Therefore, they mean that both the environment, morphology, and brain are closely connected, and improving only one of them might worsen the absolute performance.

One of the first studies in this field actually used co-evolution of morphology and control. This study was published in 1994 by Karl Sims [54]. Throughout an artificial evolution, he evolved direct graphs describing the body and control of the robots. He also managed to do this for a range of different fitness measurements. Moving towards a target, walking, jumping, and swimming were all objectives considered. His work is often considered the beginning of the research field of evolutionary robotics.

Evolving control

In the 25-30 years of research in evolutionary robotics, many ideas have been suggested to further drive the field against the potential one believes it could have. There have been a lot of different methods to control a robot and evolve it. This differs from different simple gait trajectories controlled by a few parameters [14, 16] , to high-level control policies for movement trajectories including obstacle avoidance and target seeking [25]. To develop, train and evolve these controllers, a lot of different algorithms and neural networks have been used [37]. The preferred algorithms and parameters depend on the specific task, environment, and available computing power.

Evolving morphology

Creating a usable control policy for a robot is often difficult. We often need kinematics to predict features like position, orientation, velocity, and acceleration. A physical robot design is often more intuitive to humans than the control of a certain body. We can take inspiration from nature and simply try to copy existing bodies. It is relatively easy to design such morphologies.

How one can optimize a morphology to a certain environment is however not as intuitive. From the theory of embodied cognition, we know that control, body, environment, and the interaction between them affect the performance, but we do not necessarily know how each of them affects it [59]. If we could optimize both the body, the control, and their interaction, the overall performance could hopefully be improved. We have seen research demonstrating the importance of a good body design [8]. Here they added a small power source to a passive-dynamic bipedal walker and managed to create smooth, highly stable human-like walking gaits on flat terrain by utilizing very little power.

Thus, the potential of optimizing a robot's morphology and control can be extensive, but to do this we have to find ways of co-evolving the morphology and the brain, to best suit its environment.

Co-evolution in hardware

When including robot morphology during evolution, most work has been done in simulation due to the difficulties of designing a real-world evolvable body [44]. However, the technology is advancing quickly and recently we have seen more attempts at performing morphology evolution and co-evolution physically in the real world [24, 46]. In recent years we have seen premature research where one actually has the possibility of changing the morphological setup in hardware [39]. One of the major challenges is to these changes fast enough, and without too much help from humans. The robot DyRET used by Nygaard et al. in 2020 [39], is a four-legged robot with the ability to automatically change the length of its legs during evolution. By implementing such leg-adjusting abilities they manage to do continuous evolution of both morphology and control in the real world. This design has, however, its limitations with a long evolution time and the need for human supervision in the early stages of training.

Broadbeck et al. developed a preliminary systems that automatically assembles modular robot bodies [4]. They used an industrial robot arm to construct new individuals by gluing together active and passive modules.

Jelisavicic et al. introduced a proof-of-concept system for physical evolution of robots, but with some need for human intervention. They based their evolutionary framework on the Triangle of Life concept by Eiben et al. [13] This system lets individuals exist together, learn, be evaluated continuously, and have the ability to perform mate selection and reproduction. The experiment shows how such a proof-of-concept might be implemented. The experiment is very simplified and still far from actually being a total anonymous system. In this case, only one life cycle is completed. [24].

This case of autonomous robot production was taken even further by Hale et al. They introduced a vision of an autonomous robot fabrication system where new robots can be created, trained, and reproduced all without human intervention. This system is also based on the Triangle of Life concept. To allow a large variety of robot bodies, they 3D-print individuals. New individuals are placed on a training ground to learn before they get transferred to more difficult environments, and further learning and reproduction happen there [19].

Although we have some inspiring research in morphological evolution, the state of the art is still not capable of producing a wide range of totally new morphologies fast and autonomous.

2.3.3 Robot variations

Karl Sims considered both swimming, walking, and jumping robots [54]. Since then, rigid walking robots are the ones most used in ER and robotics in general, but we have also seen some exciting new types of robots with different characteristics than the traditional ones [4, 18, 57], which gives us new opportunities.

Although we do not know which robot variation is going to be

preferable in the future, research and discoveries in all fields will conduct to a fellow valuable understanding and development of ER. What we learn from one field might be utilized in other fields. The research across different robot variations can also, hopefully, explain more about the interaction between morphology and control during evolution.

Rigid robots

Rigid robots are the ones most often preferred throughout ER research. We have seen several successful examples where ER manages to create great control policies for walking robots [9, 25], wheeled robots [31], swimming robots [56] and flying robots [17]. Rigid robots are also most typical in other robotics fields, mainly because they are easier to design and build.

Soft robots

An alternative is soft robots. These are partly or completely made of soft materials. The flexibility of soft robots makes them able to do things that are impossible for non-soft robots, like squeezing through tight spaces and maneuvering past challenging obstacles. A soft robot might also discover, and be able to utilize, different behaviors not possible for rigid robots. For instance, a cube-shaped robot finding rolling behavior to increase its speed and efficiency [5, 30].There are however some challenges with soft robots. How to control such a robot body is not intuitive. Adjusting the size of voxels has been used to create control [30], but this also changes the morphology. There are also difficulties in developing such robots in hardware. In recent years, there have been done some promising advances in technology allowing more possibilities for soft robots. We have seen soft actuators inspired by natural muscles [34], and a prototype of entirely soft autonomous robots developed by Wehner et al. in 2016 [57]. Such soft robots show new ways of allowing change to physical morphologies, and it is a large potential in combining soft robots with rigid robots to exploit the benefits of both systems. The field is still in an experimental phase, but so far it seems promising.

Modular robots

Modular robots consist of several different modules, with numerous different ways of being assembled. This way of connecting modules and building robot bodies can possibly open the door to a larger search space of complex morphologies. Most work on modular robots has been done in simulation [1], but since these robots consist of many small modules it could be easier to implement evolutionary morphology changes in hardware. To design such an autonomous system does however come with many challenges. Broadbeck et al. designed an autonomous system that glued together modules, creating different robot bodies [4]. This method allowed a large physical morphological variation compared to previous work, which is needed to successfully do evolutionary training

in hardware. This idea was taken a step further by Moreno and Faiña [35]. They created a platform with close to fully automated morphological evolution. They connected modules together by utilizing magnets, making the connection faster and giving the opportunity to disassemble and reassemble the same modules.

Swarm robots

With inspiration from swarming insects like bees and ants, swarm robotics was introduced. These are robot systems containing multiple small mobile robots working together to achieve their goal [49]. These types of systems provide the field of ER with very different utilities than seen previously, but also new challenges. Swarm robots has already helped to give some new insights into biological swarms [48] , and hopefully it could provide more in the years to come. Swarm robots will also be beneficial in co-evolutionary studies. There is work that utilizes autonomous mobile swarm of robots that can connect, and assemble to become modular robots [18]. Such autonomous self-reconfigurable robots give us a range of different morphological opportunities faster. A fast assembling of swarm robots can provide easier and more effective morphological changes during evolutionary training of modular robots.

2.3.4 Main challenges in ER

Reality gap

When transferring a robot from a computer simulation onto the real world, the reality-gap problem always occurs to some extent [62]. This problem is caused by a divergence between the simulation and the physical world. It is impossible to create a simulation completely equal to the real world. Therefore, the reality-gap problem occurs when robots in simulation take advantage of some properties that are not present to the same degree in reality, or the reality poses some properties not accounted for by the simulation. Because of this we often get much worse results when testing a simulated robot in reality. This problem is one of the greatest challenges in evolutionary robotics [22].

A lot of studies have been dedicated to decreasing or overcoming this challenge. One thought is to improve the simulators and thereby reduce the problem. In 1995 Jakobi et al. successfully created a robust control system that behaved almost the exact same in reality as simulated, by adding the right amount of noise to the simulation [22]. This has become a much-used method afterward, often in combination with other methods.

There is also work that has adapted their simulators by co-evolving the robots' simulated behavior with the behavior in reality, without an extended number of tests in hardware [61]. The suggested algorithm reduces the effect of the reality gap and solves problems like gait optimizations with fewer real-world evaluations than other algorithms at the time.

For some problems, we have found solutions that reduce the reality gap enough to not be a critical issue, but in general, this is still an unresolved problem. Simulations might never be good enough for making the transition to reality easy [36]. It is simply impossible to simulate all the elements and variables occurring in the real world. An evolutionary algorithm is created with the purpose of exploiting every shortcut it finds. As robots and environments grow in complexity, we will need EAs to dig deeper to find useful solutions, and thereby also need more realistic simulations. However, the degree of realism of simulations is a limiting factor, and the reality gap will just become a greater challenge the more EAs dig.

Since there are inaccuracies in a simulation compared to the real world, robust robot control is in general a goal for ER to better handle the reality transition. A robust controller simply reduces the overfit in simulation. Adding noise to the simulation is just one way of creating more robust solutions. Another way was shown in [3], where the evolution accomplished more robust behavior by progressively increasing the morphological complexity during the evolution.

One of the most successful ideas for crossing the reality gap, that does not tend to correct the simulator, is "The transferability approach" [28, 29]. Instead of improving the simulator, this approach tries to learn the limits of the simulator. Here they transferred controllers to reality and measured a disparity between the real environment and simulation. They evaluated this measure as the transferability of a controller and used this in a multi-objective evolution with both the fitness and the transferability measured as goals. This way the algorithm easily excluded solutions that exploited the simulation's weaknesses. This could also lead to a problem if the simulation is weak, then its accuracy may greatly reduce the search space. However, this method showed better results than noise-adding methods when compared [29].

The only way to actually avoid and completely circumvent the reality gap problem is to avoid the transfer from simulations. In other words, to only do evolution on a physical robot [24, 39]. There are, however, a lot of challenges by only evaluating and evolving in the real world [20, 44]. Testing in hardware is extremely time-consuming, each robot has to be built and maintained, damaged and worn-out parts have to be repaired or replaced, and the project might become expensive. It is not realistic to perform the same number of evaluations as possible in simulation. Time consumption is maybe the biggest challenge. Each evaluation has to happen in real-time. The only way of speeding up the process would be to scale up the population and train multiple individuals simultaneously.

Even if one finds easier ways of doing hardware evolution, the reality gap will still be an issue when transferring the robot to a new environment. The robot cannot be limited to just a lab setup but needs to be able to perform when exposed to the actual real world, where it is supposed to work. The real world will not always have the same conditions, in fact, it would contain a lot of noise and changing environments. Robust and maybe adaptive solutions are essential to perform well.

Early Convergence

When evolving robot control and morphology we need to traverse through a complex fitness landscape making it hard to navigate, improve the solutions and avoid getting stuck in local optima. This is also the case in reality with changing environments causing noisy and rough fitness landscapes. When doing co-evolution of control and morphology this fitness landscape gets even more rugged and often causes a premature convergence in the evolution of morphologies [6]. A lot of ideas have been explored to overcome this challenge of premature morphology convergence. An algorithm that combines an encouragement to novelty with local search, has shown promising results [32]. They managed to obtain a wider diversity of bodies during evolution than previous results with this technique.

Auerbach and Bongard published a study in 2014 suggesting that morphological complexity is dependent on the complexity of the environment, and premature convergence might happen because the environment is too simple [2]. Similarly, Cheney et al. suggested the task complexity might drive the morphological complexity during evolution [5].

A further study of this problem was done by Cheney et al. in 2016 where they were more skeptical of their earlier ideas and presented a new theory [6]. Here they evolved virtual soft robots by co-evolving the morphology and control, but by having two separate networks for control and morphology, and then optimizing both to easier analyze the differences during evolution. In this case, they managed to demonstrate premature convergence in morphology, without appearing to discover solutions close to the global maximum. At the same time, they could confirm this premature convergence did not happen during the evolution of control. They propose the reason for this problem is related to the theory of embodied cognition.

The theory of embodied cognition state that control and morphology are closely connected and affect each other. This might be causing the premature convergence problem in ER. When optimizing a controller for a certain body, the performance will gradually improve before converging. The problem then occurs when changing the body. The trained control will most often be worse for the new morphology. A small change in morphology could affect the control, and cause a decline in the overall fitness. This may lead to early convergence of the morphology [6, 7].

Nygaard et al. demonstrated a two-phase approach, by first evolving both morphology and control before locking the morphology and re-evolved the control [40]. This strategy showed better results at handling premature convergence than simply co-evolution. Here they also suggested a strategy to alternate between optimizing control and morphology. They believed this method is even less likely of getting stuck in local minima.

Another promising experiment was done by Cheney et al. in 2015 [7]. Here they continued the examination of the embodied cognition theory explanation in [6], Although many attempts have shown potential, we are

still not completely sure why the problem of premature morphological convergence occurs, and we have not found a general solution to the problem. Which one of the suggested methods works best, and if there are better-undiscovered solutions, is not known.

2.3.5 Evolution and learning

Another interesting aspect to consider during evolution would be to continue the adaptation and learning for each individual when facing new, unknown environments. Evolution and learning are both biological adaptive processes, but they operate on different time scales. Evolution is an overall adaptation over generations, and across individual variations, hence capturing long-term changes. Learning, on the other hand, is a process of adaptation in each single individual, hence capturing short-term changes [38, 53].

Baldwinian and Lamarckian learning are the two main approaches when considering evolution with lifetime learning. Both types do learning and adaptation during each individual's lifetime, but Lamarckian learning passes on the acquired change to its genome, while Baldwinian only stores the fitness value. The stored fitness value can act as a representation of this individual's potential. Throughout the history of ER, we have seen mostly Baldwinian evolution [38]. This method is based on the biological Baldwin effect, which is believed to smooth the fitness landscape [58]. According to the Baldwin effect, individuals with the ability to learn and adapt during their lifetimes could possibly speed up the evolutionary process [11, 38].

Lamarckian learning is not generally accepted in biological evolution , but ER is not limited by biological constrains. Lamarckian learning has shown promising results in recent years [23, 50].

An individual able to adapt could be more robust and therefore better at handling a changing environment. The reality-gap problem occurs when transferring a robot from simulation to reality, which equals a large change in the environment for the robot. Therefore, adaptable robots with more robust solutions might also be better at handling the reality-gap problem. For robots to be used for tasks in the real world, learning might also be necessary. A robot in the real world would experience different terrains and natural changing environments. The robot might also get damaged and needs to adapt to be able to continue its work without the help of human intervention. In addition to further developing the field of ER, one should gain more knowledge of how learning affects biological evolution by simulation it [53].

There is a lot of work suggesting that evolution combined with learning is needed to further drive evolution and produce adaptable solutions [38]. Several studies also suggest that an adaptable morphology is favorable during evolution [3, 23, 30]. Krigman et al. compared simulated soft robots both with and without development during their lifetime [30]. Here the development was not adaptive to the environment, but the parameters changed during their lifespan according to a predetermined goal value. They found that robots with development were a lot more robust to

mutations of the control than the non-developmental version. Also, the best robots with development had very little change in morphology. Hence, small morphological development was a lot better than non-developmental morphologies.

A simulation of robots developing from eel-like bodies into legged creatures was done by Bongard et al. in 2011 [3]. In this research, they discovered that evolution went faster, and the final creatures developed more robust gaits when the controller evolved as a consequence of body adaptation over evolutionary time combined with adaptation in each lifespan.

If we had adaptable robots, they might handle the reality gap better by simply adapting and learning how to change their behavior when transferred to the real world. For this to be the case, we actually need robots to be able to learn and adapt to hardware. We need sensors to capture and measure deviations, and we need a robot to evaluate these measurements relatively fast and also change its control and maybe also morphology according to this. There are a lot of technological challenges to make this happen, but we have seen some early successful attempts [45].

The main challenges are to do all this without too much human intervention and fast enough. Nygaard et al. managed to do learning in the real world when forcing a robot to move over changing surfaces and adjust both the control of a robot and the morphology by having self-adjusting leg length. The robot managed to find and adapt to more energy-efficient solutions in unseen environments by doing a continuous adaptation. They also showed that this could be done in relatively few evaluations [45].

2.3.6 How to deal with these challenges, by co-evolving

We have seen results suggesting co-evolution of morphology and control could help create more robust solutions [3], and therefore also solutions better at tackling changing environments and the reality-gap problem. A problem with the co-evolutionary approach, in general, has been the early convergence of morphologies [6]. If one could surpass this convergence, and develop solutions further, we could maybe come closer to a solution for the reality-gap problem as well. The question is then how one chooses to do the co-evolution. Which co-evolving strategy would be least exposed to early convergence, and which would produce the most robust solutions? Evolving morphology and control simultaneously, or locking one of them while training the other? The latter option has shown promising results before [7, 40]. By doing co-evolution this way, it may be harder to get stuck in local optima for both simultaneously.

In [6] they found positive results by giving the controller time to re-adapt to each new morphology. When considering learning in [30], they found robots more robust to internal changes. Maybe solutions created this way will be more robust to changing environment also. Maybe we need to include developmental learning during an individual's lifetime to drive the evolution past early morphological convergence. An interesting aspect could be to evolve morphology and control over different timescales. If one

considers a certain morphology to be an individual, and during its lifetime it tries to develop the best possible control. One can then save the fitness as a potential for this morphology and do an evolution with Baldwinian or Lamarckian learning. This way evolution might have time to re-adapt to new morphologies. Maybe this will easier overcome early convergence, or maybe it will just produce the most easily trained morphologies and not the overall best.

Chapter 3

Tools and frameworks

This chapter introduces some of the software frameworks utilized during this thesis. The simulation engine and the specific control system are presented. Table 3.1 shows an overview of the software tools and versions used.

3.0.1 Unity

The simulation and evaluations in this thesis are performed in Unity[1]. This is a free, popular, and relatively easy to use game engine supporting both 2D and 3D game development. It was developed by Unity Technologies, with an aim of making game development more accessible and easier to use. This makes Unity especially popular for indie game development, but it is also used in a range of other industries like film creation, architecture, engineering, and construction. The Unity physics engine provides rich environments with realistic and high-end physics simulation. This, and the easy-to-use game engine makes it a viable option for robot simulations.

Unfortunately, the Unity physics engine is not deterministic across different CPU-types. To visualize and achieve the same exact fitness values for a solution, it is necessary to not only use the same operating system but also use the same type of CPUs as done in this experiment. The exact same evolutionary runs cannot be recreated though, since every run used a random unsaved seed. The source code with Python requirements, CPU specifications, and all Unity settings can be found on the github of University of Oslo - Robotics and Intelligent Systems Research Group[2].

[1] https://unity3d.com/get-unity/download/archive. (February 2022)
[2] https://github.com/UiO-Robotics-and-Intelligent-Systems/master_steinboe

Table 3.1: List of software versions

Name	Version	Usage
Unity	2020.3.19f1	Evaluation solution
Unity ml-agents package	1.0.7	Smart agents in Unity
Python mlagents package	0.27.0	ML algorithms for Unity
Python mlagents-envs package	0.27.0	Python API to Unity

3.0.2 ML-agents

Unity Machine Learning Agents (ml-agents) is an open-source toolkit providing artificial intelligence options in Unity[3]. This is great for creating diverse and advanced behavior for non-playable game characters, without too much coding. This is also very useful for researching new algorithms and methods.

While the toolkit itself contains several PyTorch-based state-of-the-art algorithms to do machine learning, they also provide a Python API. Unity is C++ and C# based, but this interface allows Python to communicate with the Unity executable. This executable can be called from Python, with specified input. Throughout an evaluation, Python can send input to the executable Unity environment through both a call on the timestep-function and also through custom side channels. The Unity executable can return output through both the timestep-call and side channels. This allows game developers and researchers to implement fully custom design machine learning methods from Python as well, making it a good option for this thesis

3.0.3 Wave control

This thesis utilized the same wave controller as used by Nygaard et al. [40]. Nygaard used a closed-loop controller, producing a periodic and smooth gait pattern. Originally this wave controller was used to develop gaits for a 6-legged robot, but multiple studies have shown success in using similar types of controllers for 4-legged robots as well [15, 60].

Using regular sine waves would result in solutions with constant, steady movement throughout the cycle. To create a more natural and effective gait, allowing the legs to touch the ground longer, Koos et al. [27] proposed a method using a Tanh function, with amplitude and phase shift as parameters seen in equation 3.1. This parameterized, smooth wave produced solutions that were more stable as they allowed the joint angle to be constant for some time during each gait cycle.

$$\gamma(t, \alpha, \phi) = \alpha * tanh(4 * sin(2\pi * (t + \phi))) \tag{3.1}$$

Nygaard did some changes to the solution developed by Koos et al. by including an offset parameter β, allowing the wave's center to not necessarily line up with the center between the two angular limits. It might be beneficial to have some joints operating in the upper or lower part of their limitations. This also increases the search space and allows for more varied solutions.

Nygaard also substituted the amplitude and offset parameters with the two parameters v and w, representing the angular end movement. The functions in 3.2 both use v and w to express the amplitude and offset. This way it is easier to ensure that the generated function will say within its limits. As v and w will represent the end movement, the amplitude

[3]https://github.com/Unity-Technologies/ml-agents. (February 2022)

and offset can no longer cause the wave to reach beyond its range. The new parameterized controller was named the MinMaxPhase and is seen in equation 3.3. A comparison between the controllers developed by Koos and Nygaard can be seen in figure 3.1.

$$\begin{aligned} \alpha &= \frac{(v-w)}{2} \\ \beta &= \frac{(v+w)}{2} \end{aligned} \tag{3.2}$$

$$x(t,v,w,\phi) = \frac{(v-w)}{2} * tanh(4 * sin(2 * \pi * (t+\phi))) + \frac{(v+w)}{2} \tag{3.3}$$

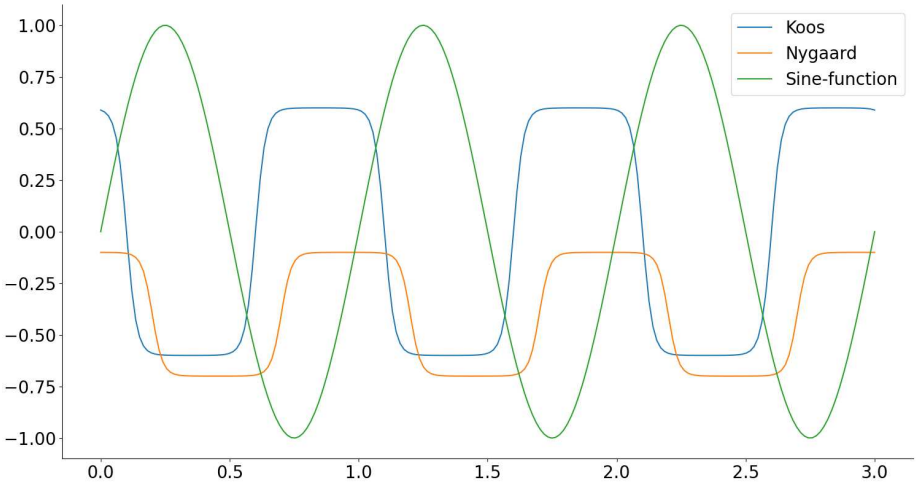


Figure 3.1: A comparison between the controller developed by Koos (blue), Nygaard (orange), and a simple sine wave in green for reference. The original controller by Koos et al. has amplitude $\alpha = 0.6$ and phase shift $\phi = 0.4$. The MinMaxPhase by Nygaard has $v = -0.1$, $w = -0.7$ and phase shift $\phi = 0.6$.

This simple and easy-to-implement controller needs relatively few trainable parameters, even after the changes made by Nygaard, and creates natural, smooth, periodic motion. Considering this, in addition to the fact that the controller itself is not the main research area for this thesis, makes it a good choice for this thesis.

Chapter 4

Implementation and experimental setup

This chapter gives an overview of the implemented evolutionary setup. First, the simulation setup and the robot are presented, then the genetic algorithm is described.

4.1 Simulation setup

4.1.1 Simulation framework

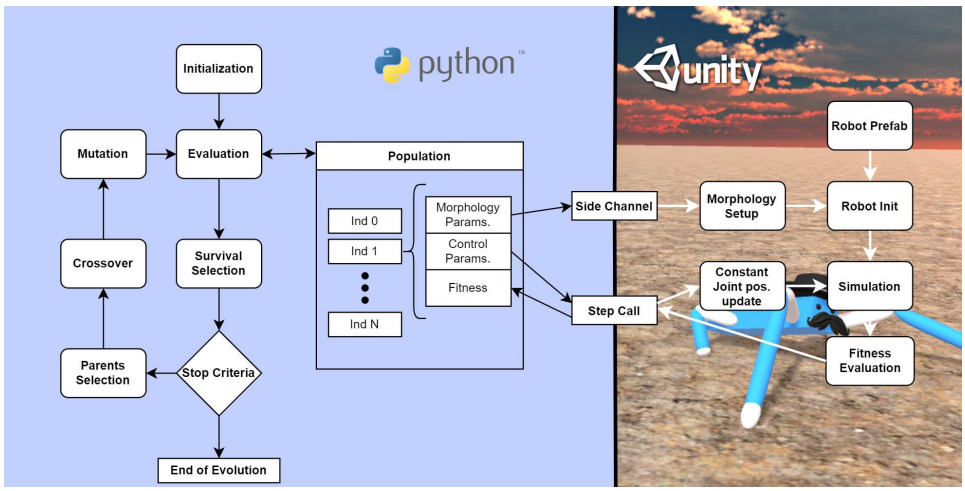


Figure 4.1: An overview of the complete simulation setup.

The setup in this thesis consists of a Python environment and Unity executable. An overview of the entire setup can be seen in figure 4.1. The robot is designed in Unity and the evaluation takes place in the Unity executable, while all other parts of the evolution and data processing are executed in Python. The communication between Python and the Unity executable is done through a side-channel and as parameters at each step-call. A call on step, refers to a new positional update in Unity.

Since we want to change and evaluate different morphologies we use the side-channel to send morphology-specification strings from Python and these morphology-specifications get extracted from the strings by the C# code and the correct morphologies are created. To allow for changing morphologies, without the change itself affecting the evaluation, the default robot design is stored as a prefab model and not as a game object in the Unity scene. Upon initialization, the Unity environment only holds an empty game object. Only when the Unity environment receives a morphology-string, and then a reset call, the prefab model will change according to the morphology specification of the input string, and the robot is initialized and spawned.

After initialization, the Python code sends direct controller input to the executable at every timestep. This way the Python environment can constantly send control input to set each joint angle during evaluation, and the Unity-executable will constantly return output data after each step-call. During the evaluation, the C# scripts in the executable will calculate the fitness-values and return it along with other output data. After a certain evaluation time, the evaluation terminates, output data get returned to Python and the Unity-executable wait for a new configuration solution to evaluate. The Python environment saves the output data to the belonging individual object. By using this Python API we can have multiple parallel instances of an Unity environment with the agent simultaneously. Hence, speeding up the total time of the evolution.

4.1.2 Robot design

The robot is a modified version of the Unity ml-agents crawler robot. Three evolved morphologies examples can be seen in figure 4.2. This version has 4 legs where each leg consists of two links connected by a joint. The length of each link and their base positions are trainable during evolution. Since we want to find realistic solutions, we use a constant density for the robot legs making the weight proportional to their lengths. To allow for a variety of different base positions, the robot has a long cuboid-shaped body.

We use a symmetric body and a symmetric walking gait for the robot. There are many reasons why a symmetric body and gait are preferable: Most natural bodies and gaits are symmetric, and we want a realistic approach. One of the fitness measurements, speed, is only calculated along one axis, so walking straight forward is favorable. A stable, symmetric gait with a symmetric body tends to move straighter, hence scoring higher on this fitness measurement. Symmetry might also lead to more stable gaits, hence also scoring high in the stability fitness measure.The main reason for using symmetry, however, is the heavy reduction of trainable parameters that follow.

Figure 4.2: Three different morphology specifications generated through evolution.

4.1.3 Morphology configuration

A lot of initial testing was conducted to figure out which morphology setup to use. Initially, we only used evolvable parameters to determine the leg length. Since the parameter space for the controller is so much larger, we also decided to include parameters training for the base positions of each leg. After these changes, the parameter space for control is still a lot larger, but to further introduce possibilities for evolvable morphology would be a lot more advanced. This uneven search space between control and morphology is also found in similar studies, and is hard to avoid.

A figure describing the possible changes to the morphology can be found in figure 4.3, and the limit of each trainable parameter can be seen in table 4.1.

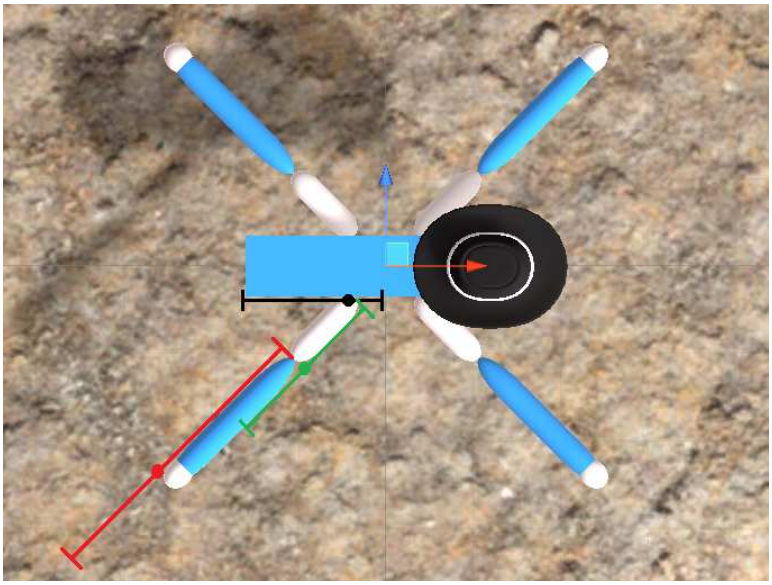


Figure 4.3: A top view of the default robot configuration. The black line visualizes the possible base position for the rear right leg. The green line shows the possible femur length, and the red line is the possible tibia length. In this case, the base position=0, femur length=10cm, and tibia length=20cm.

Parameters	Range (cm)
Base position	[-2, 80]
Femur length	[3, 30]
Tibia length	[3, 30]

Table 4.1: Min- and maximum leg lengths and positions. The robot has the same limit for all four feet. The lengths are measured in cm

4.1.4 Angular limits

For each leg, we have three joints that the controller can move. This is visualized in figure 4.4. The femur can move in two directions, hence there is two joints between the body and the femur. The first joint (the top, red angle in figure 4.4) has a range of 60°, reaching from 90°to 30°out from the side of the body. The second joint, seen in the middle figure in green, stands 45°out from the body, and can move between 35°to 55°. The third joint has most available movement, from 0°to 150°.

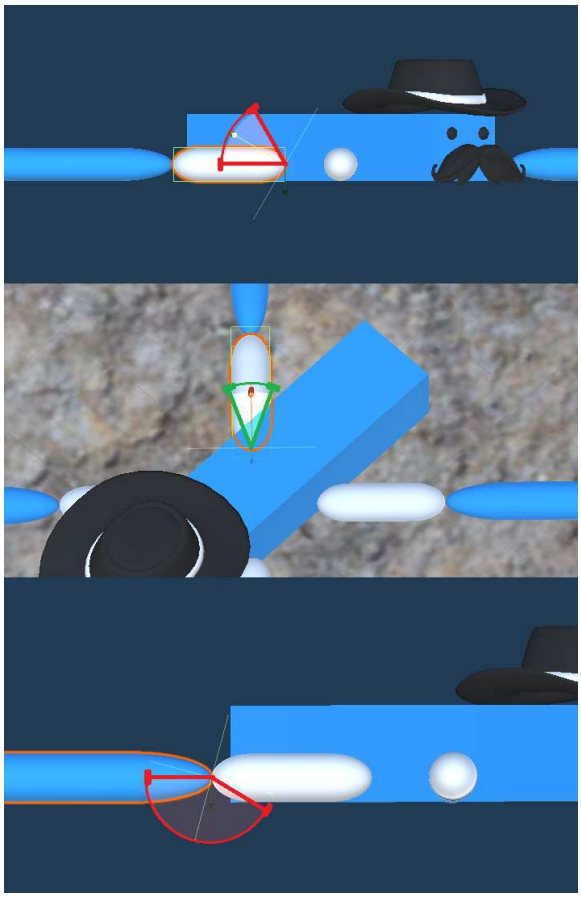


Figure 4.4: A visualization of the movement range for each joint in a leg. The joints are at 90°, 45°, and 0°respectively in this figure.

4.1.5 Controller

Reducing number of control parameters

We use the same MinMaxPhase controller as presented by Nygaard et al. [41]. This controller needs 3 parameters for each joint. If we use a unique control signal for each joint, the total of trainable parameters in the control space is 36 parameters, given 4 legs with 3 joints each.

Since the main goal is to investigate morphology optimization throughout the evolution, the size of the control space needs to be manageable. The control space is larger than the morphology space, but by reducing the number of controller parameters, hence increasing the proportion of morphology parameters, it is easier to investigate how the morphology differs.

A simple way of reducing the total of trainable control parameters is by using symmetric gaits. This does not necessarily increase the proportion of morphology parameters compared to control parameters, but it does reduce both search spaces greatly.

The three controllers

Three symmetrical controllers are implemented, using 15, 18, and 24 control parameters. These are developed to find the best balance between search space size and controller diversity. The v and w parameters for each joint on the right side, is the same as the according joint on the left side. In other words, the joint waves are mirrored across the body. This alone results in 12 parameters. The difference between these controllers is the use of the phase shift parameter. An explanation to these controllers follows, and table 4.2 provides an illustration of the differences.

The controller with 15 parameters (denoted as Cnt-15 in table 4.2) has 3 phase parameters. One for each joint in a leg. The rear-left and the front-right leg use the same phase parameters, while the rear-right and the front-left use the opposite of these phase shifts. This way the diagonal legs are in phase, half a cycle ahead of the other two, and only 3 phase parameters need to be optimized.

The 18-parameter controller (denoted as Cnt-18 in table 4.2) also use the same phase shift parameters for each set of diagonal legs. However, each of these two sets have their own trainable shift parameters. Making the total of phase parameters equal to 6. Compared to Cnt-15, this allows for more diverse walking gaits, but also a larger search space.

The last controller, with 24 parameters, simply has a unique phase-shift for each joint. Resulting in 12 phase-shift parameters in addition to the 12 parameters for v and w. This controller has far more possibilities than the other two, but it has also a much larger search space.

Leg	Cnt-15	Cnt-18	Cnt-24
Rear-right			
Rear-left	+0.5		
Front-left			
Front-right	+0.5		

Table 4.2: This table visualizes the differences between each controller. Each leg has three joints needing a phase parameter. Each color represents a unique set of 3 such phase-shift parameters. We see that each leg at Cnt-24 has unique phase parameters since they all have a different color. Cnt-18 has unique phase parameters at each diagonal, while Cnt-15 has the opposite phase parameters between the diagonals.

4.2 Genetic algorithm

4.2.1 Genome

To represent each solution, the genotype for each individual consists of a list of control- and morphology parameters. These parameters are floating-point values between -1 and 1, and get normalized into their corresponding position in their respective range, to create the wave controller and the morphology setup.

4.2.2 Evaluation

When evaluating solutions, one Unity executable is initialized for each thread. Before any evaluations take place, two warm-up rounds with default morphology and a simple, default joint control are executed. This is needed because the physics engine can behave a bit differently upon each initialization.

Before each solution is evaluated, the correct morphology parameters are sent through a side channel to the executable, and the body is built and initialized. Each solution is evaluated for 5 gait cycles, where each cycle is divided into a resolution of 50 timesteps. To make the robot move, the Python script sends the current position command as parameters at each timestep-call, which directly set the target joint position. The resolution and timestep-call frequency have been tuned to create a smooth and natural movement.

The numbers of CPU threads are the bottleneck for maximum throughput during evolution. Each evaluation needs to happen in real-time because a speedup would cause the Unity physics engine to behave differently and give different results.

4.2.3 Fitness measurement

To expose a potential trade-off in fitness, we consider two fitness measurements. They are speed and stability.

$$fitness = (fitness_{speed}, fitness_{stability}) \tag{4.1}$$

Each solution is simulated in 5 gait cycles of time, but the first cycle is not included in the fitness measurement. This is necessary since the robot spawns a bit above the ground, to have enough time for each joint to reach the correct starting position, and its first cycle is affected by this fall. The speed fitness is calculated from the distance walked, and the evaluation time after the first cycle and until the end, as seen in equation 4.2. Distance is measured as the difference in position of the robot body, along the x-axis. This is straightforward for the robot as it spawns.

$$fitness_{speed} = \frac{(pos_{end} - pos_{start}).x}{evaluation\text{-}time} \tag{4.2}$$

We consider a stable solution to have small orientation changes and small changes to the body's linear velocity. At each timestep during evaluation, Unity calculates and returns the body's linear velocity and orientation.

To ensure a low change in linear velocity, we sum the standard deviation over every sampled linear velocity during evaluation, as seen in equation 4.3. Both the standard deviation and variance measure variability over the data, but by using the standard deviation we have the same unit over all three directions.

$$stdsV = \sum_i \sqrt{\frac{\sum_j (v_{ij} - \hat{v}_i)^2}{N}} \tag{4.3}$$

The orientation measurements are compared against the mean orientation, by root mean square. This way the robot is not punished for constantly leaning. This value is also summed over the three directions.

$$rmsO = \sum_i \sqrt{\frac{(o_{i1}^2 + o_{i2}^2 + ... + o_{in}^2)}{n}} \tag{4.4}$$

$$fitness_{stability} = stdV + scalingFactor * rmsO \tag{4.5}$$

To ensure the linear velocity and orientation measurements contribute equally, a scaling factor was tuned based on initial training data. This scaling factor is set to 0.078 in this thesis.

4.2.4 Optimization algorithms

Because of these two fitness measurements, we use the NSGA-II algorithm for the optimization. As the rest of the evolutionary setup, the Python DEAP[1] (Distributed evolutionary algorithms in Python) implementation of NSGA-II is utilized. The parents is chosen by crowded tournament selection and the survival selection is based on front rank and crowding distance.

To later chose between the best solutions from a Pareto front, one needs to investigate the actual solution further. There will often be solutions

[1]https://deap.readthedocs.io/en/master/ (February 2022)

scoring very high on one fitness, but still low on the other. Such outliers can typically be rejected in the final evaluation. A motivation for this thesis is to have a realistic approach, and when handling the results after the evolutionary runs, outliers can typically be rejected for being unrealistic. Very high speed or really bad stability will typically be filtered out.

4.2.5 Evolutionary operators

This thesis uses a simple uniform crossover as the recombination operator. For each generation step, the selected parents get sorted into pairs, and the crossover probability is the probability for each parameter value to swap between the two parents. The resulting solutions are the offspring. The parameters for the control and morphology are separated and so is the recombination. This allowed us to have different recombination probabilities for control and morphology if we want.

A combination of two different mutation methods is used during the evolution. It is possible to have different mutation parameters for the control and morphology part of the genome as well. The main mutation method in this thesis is a Gaussian mutation. Two parameters control this mutation. One for mutation rate and one for the standard deviation of the Gaussian distribution. If the mutation occurs, all alleles in the genome get added with a random value drawn from this distribution. If a parameter value exceeds its limits after a mutation, a clipping will occur. This means that the value will simply be set to the most extreme value allowed.

Since these experiments use a relatively small value as the mutational standard deviation, this mutation technique causes only small mutational changes to the parameters. Therefore, it contributes most to the exploitation part.

To also make sure there is a sufficient exploration during evolution, we include a random reset mutation as well. To avoid an undesirably large exploration contribution, this value is also kept relatively low. We hope that the random reset mutation contributes to more divergence across the population. This decreases the probability of premature convergence.

Chapter 5

Experiments and results

This section contains a description of the final setup and the results of the main and supporting experiment.

5.1 Final setup after initial training

A lot of testing was done to decide on the final setup, before being able to produce the results. After early testing with all three implemented controller types, we chose to continue with the 18-parameter options. The 15-parameter option was a bit too limited, and it was difficult to develop useful solutions. Both the 18- and 24-parameter options seemed to allow for more freedom, but the latter option needed a lot more training to perform adequately. Since we want to keep the solutions relatively realistic, and we see there is a lot of symmetry in natural gaits, the 18-parameter option was deemed a better choice for this purpose. Keeping the number of evolvable control parameters low is also beneficial as it reduces the difference in the size of the control and morphology space.

Tuning of hyperparameters is important when comparing different methods of ER to achieve generalized results.

As a basis for the initial hyperparameters, we looked at a similar experiment done in [47]. Although this experiment was not done in the same simulation environment, it used much of the same evolutionary setup.

These hyperparameters were verified to be good enough after earlier supporting experiments.

5.2 Main experiments

Runs	**Mutation Probability**
25	0.3
Population size	**Mutation Sigma**
256	0.2
Generations	**Crossover Probability**
4096	0.1
Mutation Type	**Initial mutation Prob.**
Gaussian and random reset	0.01
Controller type	**CPU years**
Wave controller 18param.	Roughly 13.3

Table 5.1: An overview of the final configuration for the main experiments.

The main experiment in this thesis consisted of 25 runs of the co-evolutionary algorithm, training both the control and morphology of a robot with 256 individuals over 4096 generations. An overview of the specific hyper-parameters used in the main experiment can be seen in table 5.1.

To evaluate one solution on an already initialized environment, a single CPU uses about 16 seconds. When evaluating a population size of 256 over 4096 generations, this uses 4660 CPU hours for each run. Meaning that this main experiment of 25 runs used roughly 13.3 CPU years. We believe that such a large population size and long runs are necessary because of the difficult search environment.

5.2.1 General results of the main experiments

Since this thesis used a multi-objective optimization problem, we measured the performance of an evolutionary run in its hypervolume. This is calculated as the volume between -50 stability and the graph made up of each generation's Pareto front. In figure 5.1 you can see the fitness of every individual belonging to each run's Pareto front. From this graph, we see that the runs in general create a linear front. At low speed and high stability, the point spread is very narrow, and they tend to spread more as the stability decreases and the speed increases. We also see there are much more points in the upper-left part of the front, compared to the bottom-right half.

Appendix A includes a plot with separate lines for each Pareto Front in figure A.2. This figure shows that some runs tend to do better and worse at certain parts of the Pareto front than others.

The mean hypervolume over all runs and the area of plus-minus 1 standard deviation is plotted in figure 5.2. We see that every run has a strong increase in hypervolume during the first 1000 generations. Some

runs already converge at this point, while others increase more gradually throughout the entire run.

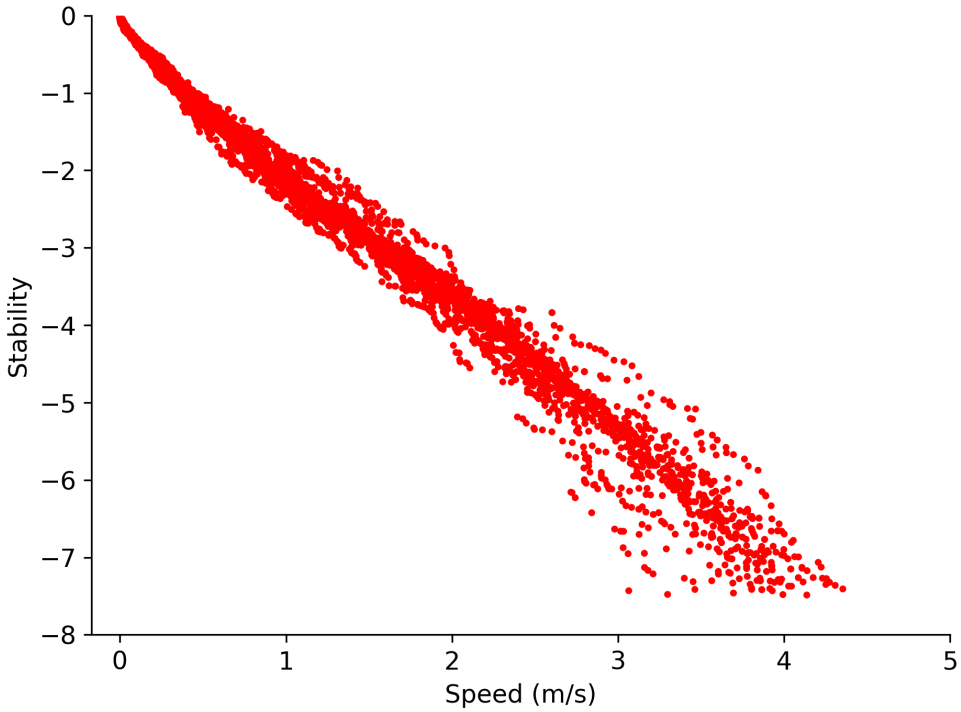


Figure 5.1: Pareto fronts for all runs in the main experiment.

We can see that all individuals with worse stability than -7.5 are not shown. Although several runs create solutions with worse stability, they are left out when analyzing the data. These solutions often tend to move in ways not considered realistic and would be filtered out for testing in the real world. All solutions are however allowed during training, as they possibly can lead to new and better, undiscovered solutions with acceptable stability. The hypervolume, and every other measure that follows, is also calculated when excluding solutions with worse stability than -7.5. Appendix A contains a Pareto front plot A.3 with no stability limit.

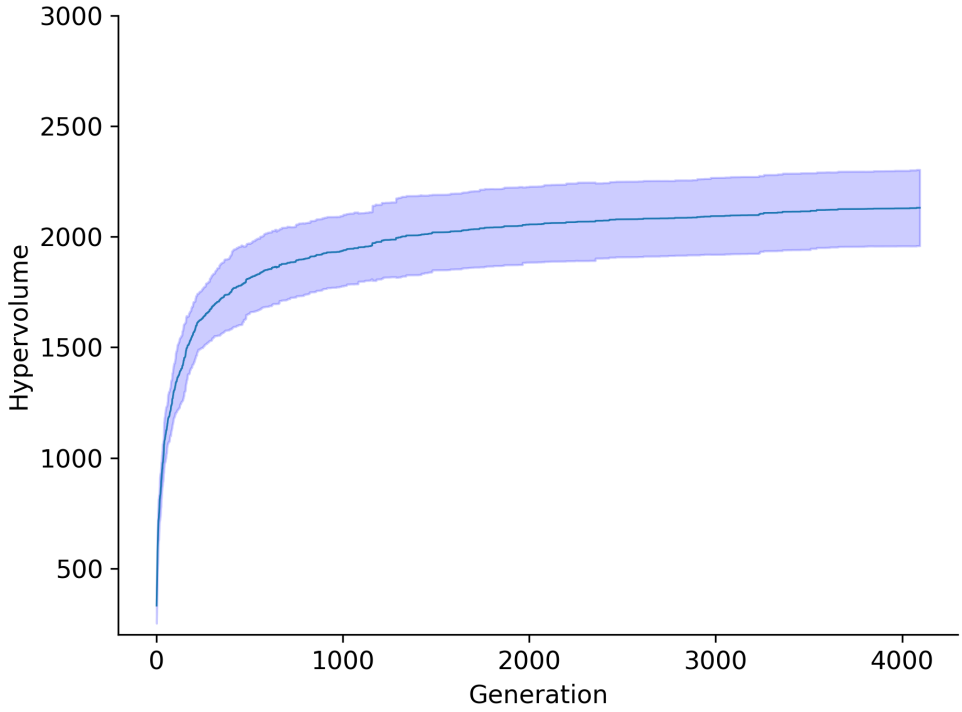


Figure 5.2: The blue line is the mean hypervolume over all runs in the main experiment. The shaded region shows the area within $\pm$ 1 standard deviation of the mean.

5.2.2 Morphology parameters

To investigate the hypothesis and see if the morphology parameters tend to adjust to the trade-off between stability and speed, we need to visualize the values of each morphology parameter according to their fitness.

The best way of comparing morphology parameters with the fitness trade-off would be to compare the parameter values directly against the position of the solution in the Pareto front. This way we could see if there would be a dependency between parameter values and the solutions' fitness values. The problem with such a frontier, however, is that it is not generalized. Each run achieved a unique Pareto front, and we wanted to investigate the general behavior across all runs.

We chose to compare the parameter values to only the speed fitness. Since we see a clear linear front in figure 5.1. We can assume that higher speed also led to lower stability. A greater spread in stability occurred as we reach a higher speed, but the general trend is strong enough that this is considered a safe choice. The plotting of each morphology parameter value with the solutions' speed fitness can be seen in figure 5.3.

From these six figures, we can see how the morphology parameters in general behaved at increasing speed. At the top, we see that the front and rear base positions tend to be quite spread out, especially at low speed, but it looks like they have more occurrences in the upper and lower range

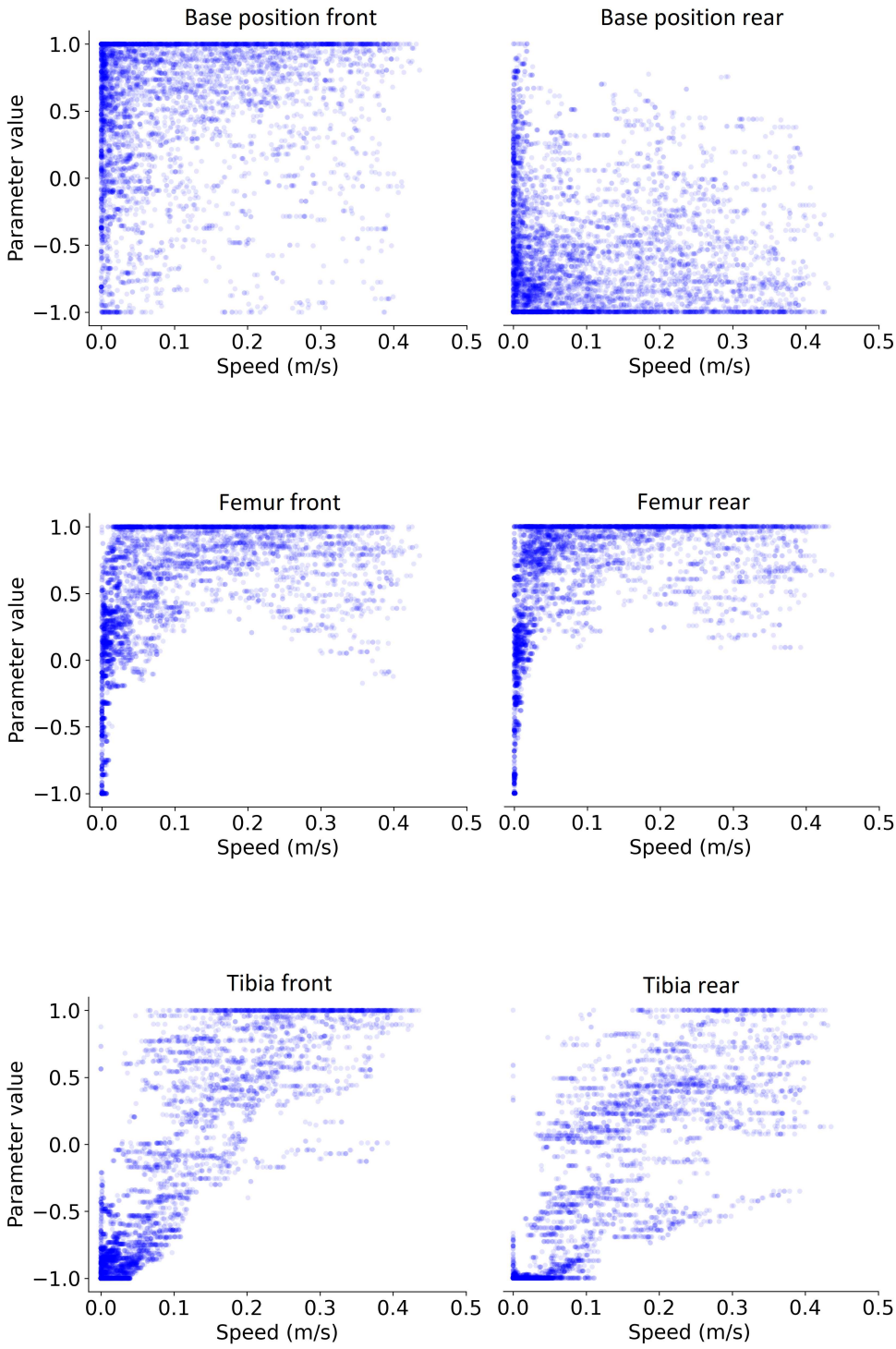


Figure 5.3: Morphology parameters for all solutions in the Pareto fronts over all runs. Their parameter value is plotted along the y-axis, while the solutions' speed fitness is plotted along the x-axis. This gives a visualization of the general change in parameter value as the speed fitness increases.

respectively. Both femur lengths, the two figures in the middle, have solutions covering the entire range at almost zero speed, but as the speed increases, all their solutions stay in the upper half of their range. The solution for front and rear tibia length can be seen in the two figures at the bottom. They both have very low values at low speed, but they seem to generally increase as the speed increases.

To help investigate and determine if there is a dependency between the morphology parameters and the speed fitness, we have fitted a regression model for each parameter. We chose to use the scikit-misc[1] implementation of the local regression model LOESS (locally estimated scatterplot smoothing). The results of the LOESS function on each parameter can be seen in figure 5.4.The regression models behave as expected from the parameter plots. Here it is easier to see the front and rear tibia length's dependency to the speed. The two lines have almost a linear trend.

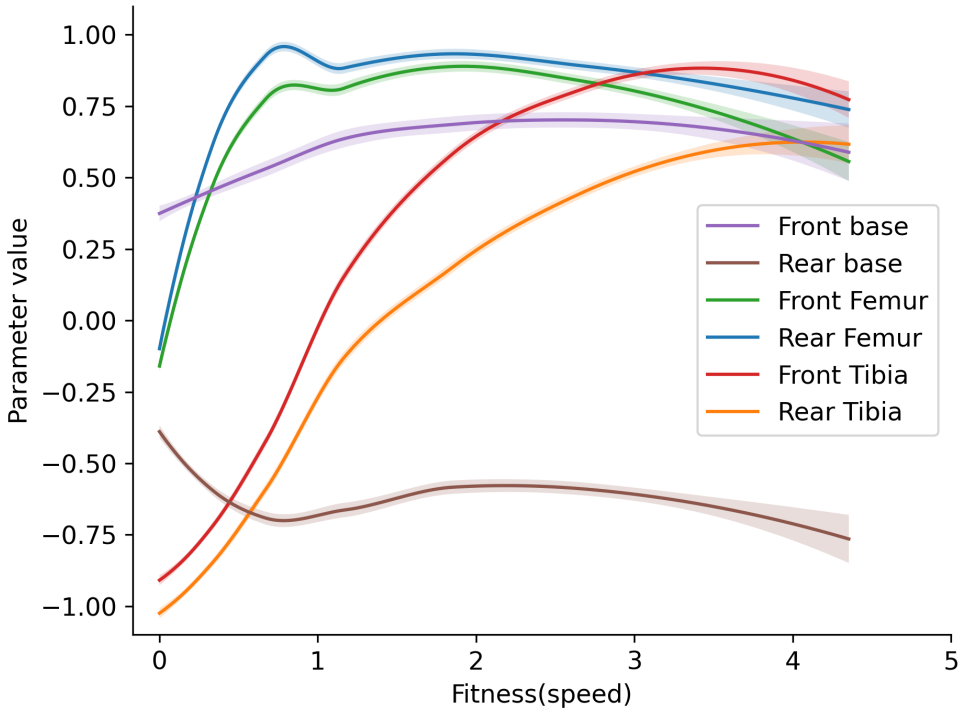


Figure 5.4: Each line represents a scikit-misc LOESS function fitted to the morphology parameter values and the speed fitness measurement, with 95% pointwise confidence intervals. This LOESS function uses a second-degree polynomial.

To get a statistical measure of how the morphology parameters affected the fitness, we ran a correlation test, both with Pearson's and Spearman's Correlation, between each parameter and the speed fitness. The results of

[1]https://has2k1.github.io/scikit-misc/stable/generated/skmisc.loess.loess.html (Mars 2022)

this test can be seen in table 5.2. Here we see that both the front and rear base position has a negligible correlation to the speed. Both femur lengths have a low linear correlation, but a moderate non-linear correlation. The Femur length shows a strong linear correlation to the speed.

Table 5.2: This table shows the results after Pearson's and Spearman's correlation are calculated between each morphology parameter and the speed fitness to the belonging solution. Even after a Bonferroni correction to every P-value (multiplied by the number of parameters), they all stay below 0.01, making them significant.

Parameter	Pearson	Spearman
Base front	0.204	0.259
Base rear	-0.115	-0.140
Femur front	0.489	0.623
Femur rear	0.475	0.632
Tibia front	0.860	0.861
Tibia rear	0.838	0.858

5.2.3 Control parameters

While the morphology parameters are the main research area of this thesis, it is also important to take the control parameters and their possible fitness adjustment into consideration. The fitness of a solution will be determined by both the evolved body and control. Since there are 18 control parameters, only 4 of them are plotted in figure 5.5. A figure with all 18 can be seen in Appendix A figure A.6. The plots of control parameters are in general lot more spread out than the morphology plots. Some tend to move towards the limit as speed increases, as parameter 2 does. Parameters 4, 8, and 9 are among those that show an interesting pattern. They seem to have a splitting into two groups as speed increases.

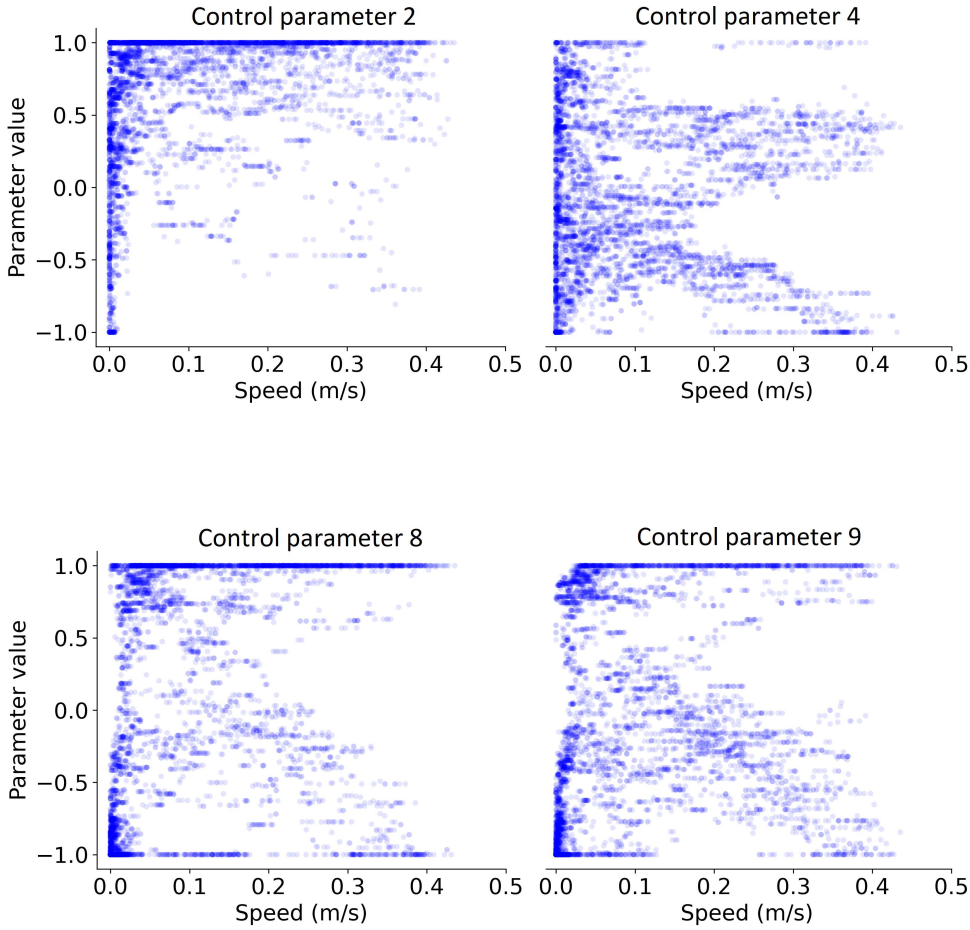


Figure 5.5: Four interesting control parameters are plotted in this scatterplot. A figure with all 18 control parameters can be seen in Appendix A figure A.6 . This is the same plotting used for morphology values. Their parameter value is plotted along the y-axis, while the solutions' speed fitness is plotted along the x-axis.

In addition to the scatterplots, we have also fitted a scipy LOESS function to all control parameters. The results can be seen in figure 5.6. These plots show few clear dependencies. Most regression functions tend to stay around the middle parameter value. The confidence intervals are quite thin all the way towards the highest speed, where they all become much wider.

The correlation between each control parameter and speed fitness value, with significance levels, is listed in table 5.3. The vast majority of control parameter plots show a negligible correlation to the speed. Only a few parameters have a low to medium correlation, and mainly non-linear correlation is found for them. These are marked grey in the table.

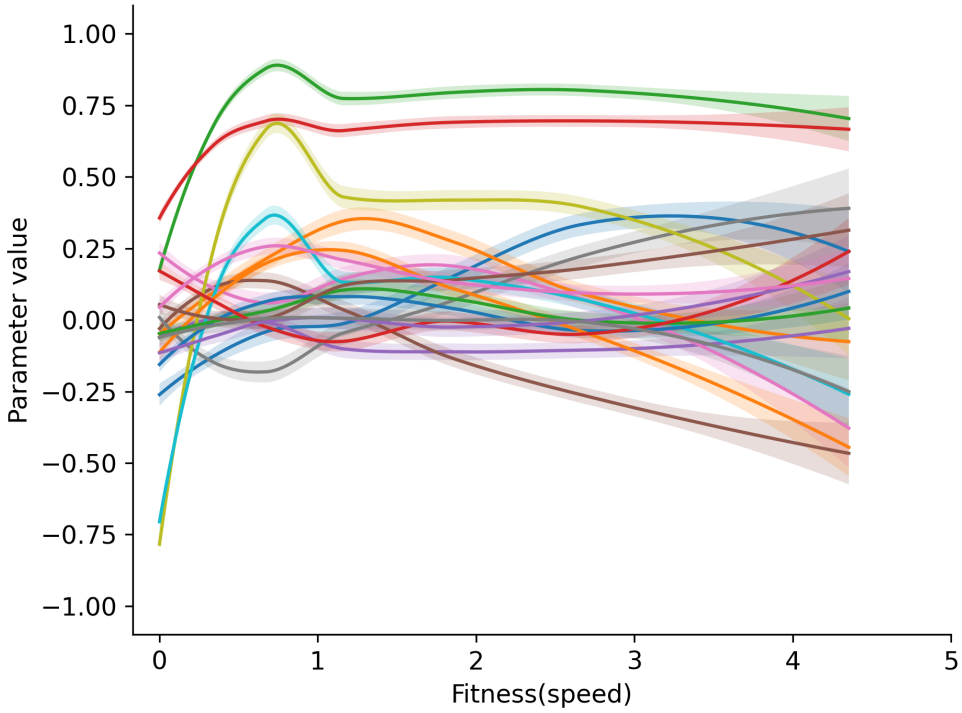


Figure 5.6: Each line represents a scikit-misc LOESS function fitted to the control parameter values and the speed fitness measurement, with 95% pointwise confidence interval. This LOESS function uses a second-degree polynomial.

Param.	Pearson's		Spearman's	
0	0.269	***	0.287	***
1	0.021		0.107	***
2	0.304	***	0.418	***
3	0.197	***	0.272	***
4	-0.051	**	-0.035	
5	-0.233	***	-0.174	***
6	-0.099	***	-0.057	***
7	0.179	***	0.145	***
8	0.298	***	0.474	***
9	0.179	***	0.383	***
10	0.048	**	0.077	***
11	-0.060	***	0.024	
12	0.019		0.035	
13	-0.093	***	-0.115	***
14	0.071	***	0.060	***
15	0.103	***	0.095	***
16	-0.010		0.018	
17	0.001		0.015	

Table 5.3: This table shows the Pearson's and Spearman's correlation between each control parameter and the speed fitness. The 3 parameters with significant non-negligible correlation are marked in grey.

5.3 Supporting experiment

This section presents the supporting experiment, which consisted of several runs with different hyperparameters, executed both before and after the main experiments, and act as support to the choices made in the main experiment and the analysis in the next chapter.

Number of Runs	**25**	3	3	3	3	3	3
Mutation probability	**0.3**	1.0	1.0	1.0	1.0	1.0	1.0
Mutation sigma	**0.2**	0.02	0.02	0.02	0.02	0.04	0.06
Crossover probability	**0.1**	0.0	0.1	0.3	0.5	0.1	0.1
Reset probability	**0.01**	0.02	0.02	0.02	0.02	0.02	0.02

Table 5.4: Overview of different configuration runs. They all have 256 individuals and are trained in 4096 generations. The first one, with 25 runs, is the main experiment.

In addition to the main experiments, six sets of hyperparameter configurations are tested in these runs. Because of the limited training time and resources, only three runs of each set are tested, making it a total of 18 additional runs. This is roughly 9.6 CPU hours in addition to the main experiment. When deciding on different sets of hyperparameters, we tried both more explorative and more exploitative parameters. The chosen

hyperparameters can be seen in figure 5.4.

To compare each of these 18 runs with the main experiments, figure 5.7 shows a comparison across all hypervolumes achieved. Here we see the area of $\pm$ one standard deviation of the mean from the main experiments in blue and the hypervolume of each supporting run in black. We see that some runs from the supporting experiment are competitive with the main runs, and one run does even better than one standard deviation of the mean main run. However, most runs tend to do worse than -1 standard deviation.

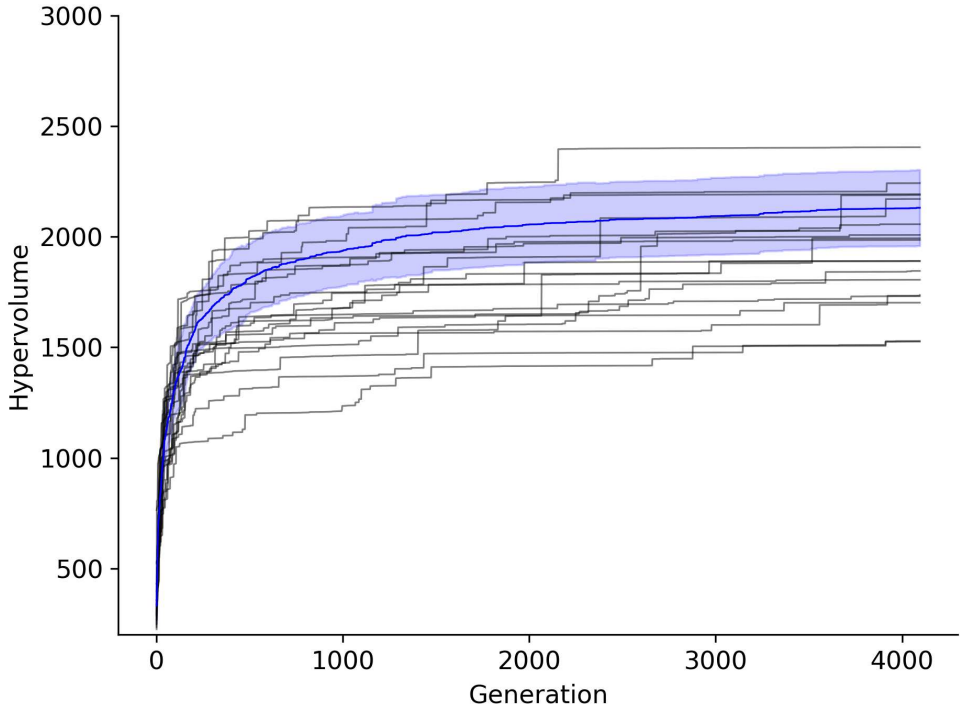


Figure 5.7: The black lines visualize the hypervolume of each supporting experiment throughout the evolution. The blue shaded area is within one standard deviation of the mean of the main experiments.

Chapter 6

Discussion

This chapter contains an analysis and discussion of the results, as well as the future work section.

6.1 General discussion

When discussing and analyzing the results of the experiments, a reminder of the null and alternative hypotheses is in its place:

> H_0: None of the morphology parameters will show a fitness-related trade-off.
>
> $Hypothesis(H_1)$: Co-evolution of morphology and control on a four-legged spider robot will produce individuals with a fitness-related trade-off in morphology.

From table 5.2, we see that both front and rear tibia lengths have a strong, positive correlation to the speed. The front and rear femur lengths show a moderate correlation. After the conservative Bonferroni correction of p-values, they all stay safely below 1% confident, meaning the correlation measures are significant. This is conflicting with the null hypothesis, hence, we can reject it. In other words, we can say that the experiment confirms the research hypothesis. By using co-evolution of morphology and control on this four-legged spider robot with speed and stability as fitness, the morphology does indeed adjust to the trade-off in fitness under the given setting.

This shows both the importance and the possibilities of co-evolving morphology and control when doing evolutionary optimizing of a robot. Now that we know it is possible to adapt the morphology to a fitness trade-off, we can say that including optimization of morphology is far from pointless. It can actually be quite favorable.

Further analysis and discussions of the results and their implications are given in the following sections.

6.1.1 The runs in general

It is worth mentioning the large variation over different runs, as seen in figure 5.2. From this plot, it is clear that some runs struggle more than others to find good solutions. For large and difficult search environments like this, there will always be a variation of how good solutions each initial random population can lead to. The variation seen here is a bit larger than expected. A reason for this might just be the choice of hyperparameters, causing an uneven distribution of exploration and exploitation. Unfortunately, it is not that easy to predict better hyperparameters by only analyzing the results. Also, testing a large range of different parameters is extremely time-consuming.

Even though the simulated robot and environment in this thesis are simple, the search landscape appears to be extremely difficult to search through. From initial testing, we saw that small adjustments can make large differences. This indicates that we will have large variation across different runs regardless of the chosen hyperparameters.

The observed variation in these main experiments is, however, not only a disadvantage. A large variety in fitness might suggest that several runs experience premature convergence by getting stuck in local optima. If we saw the solutions struggle to achieve adequate speed, this would be the problem, but to reach 0.3-0.4 m/s is relatively fast in this setting. From visualizing multiple solutions, we also saw that the solutions perform quite well. Within the -7.5-stability limit, we saw that the fastest solutions moved relatively fast and stable, without glitching or exploiting weaknesses in the simulation to do so.

Although it is not certain, a large variety in fitness could mean a large variety in solutions. Keeping up the solution diversity is something we want when investigating general trends after evolutionary runs. The more diverse data collected, the more accurate and generalized the data will be.

It is easier to find solutions that tend to have small movements and keep very stable, compared to solutions that reach high speed. We also see from the plotting in figure 5.1 that a large proportion of the individuals stay in the top left corner. Consequently, we will have more data from this region compared to the high-speed region. As the solutions train longer, the algorithm discovers more and better solutions for high speed, and those solutions are essential for creating a wider Pareto front. Hence, longer runs will have a greater chance of more diverse fitness adjusted morphologies.

In the Pareto front in figure 5.1, we see a nearly linear front. All fronts tend to have a linear trade-off between speed and stability. This is as expected, since it is a lot easier to be stable at low speed, and an increase in speed will most likely lead to worse stability.

6.1.2 Analysis of morphology parameters

We see that the morphology parameters tend to pairwise have the same features. This is not surprising, as the pairs are the corresponding front and rear parameters.

Both the LOESS plots in figure 5.4 and the correlation in table 5.2imply that the morphology parameter for front and rear femur length have a strong, positive, linear correlation to speed. The robot manage to achieve greater speed the longer the femur lengths are, in general. These parameters have few occurrences of low speed and high parameter value, and vice versa.

It is also shown a correlation between the morphology parameters denoting front and rear femur length, although these have only a moderate correlation and a less linear one. From the plots, we see that the values for both parameters are spread out at low speed but tend to stay in the upper area as the speed increased. They have very many values right at the largest possible value. When a value is mutated outside of its range, the algorithm sets the value to its maximum. It is likely that during evolution the algorithm finds this to be very beneficial and finds few solutions with low values for these parameters.

The legs' base positions does not show a significant correlation to the speed. We also see from figure 5.3 that the values are much more spread out, especially at very low speed. When increasing speed, the front base position has small tendencies to move and stay towards higher values, and the rear base position move towards and stay at a low value. A value of 1 for the front base position means it is placed as long forward as possible, while zero means a placement towards the middle. On the rear base position, an increasing value also means forward in the walking direction. Hence, a value of 1 on the rear would be a placement towards the middle, and a value of -1 would give a placement as far back as possible. This means the robots in general want to have their legs placed at the endpoints of the robot. This is not surprising, as this most likely contributes to better stability.

When analyzing the trends of the morphology parameters, it is important to remember what they represent, and what effect the controller has on the fitness. Given a controller that generates very small movement, and low speed, we expect it to easily score high in stability fitness for a range of morphology setups. We see that this is especially true for the base positions and the femur lengths as they have solutions all over their range for low speed. This also makes sense when looking at the robot design. Because of their angular range, the femurs will never touch the ground. They will at most be parallel to the ground. On the contrary, it will not take much movement or length for the tibia legs to touch the ground and cause the body to move and become more unstable, even though their positions are dependent on the femurs. This lifting will not happen if the tibias are extremely short, which probably is the reason the femur lengths are so short at low speed.

The LOESS functions in figure 5.4 summarize the main results of this thesis. Not only do the morphologies adjust to the fitness-trade off, but we can now also see a generalization of the morphologies. Such generalized results give us an idea of what a general fitness-adjusted morphology might look like. From this, we can much likely design our own morphologies fitted for great speed or low stability. The LOESS plot

also tells us how we could decrease the search space in a safe manner. We see that we can remove the lower range of the front base position and the upper range of the rear base positions. Also, the lower half of both femur lengths is not used much. We see that we cannot find usable solutions for these areas, and limiting the range more will decrease the search space considerably. Although this is not done in this research, it is an important observation and another feature favoring morphological evolution.

6.1.3 Analysis of control parameters

Very few control parameters show a significant correlation to the fitness, and those who do has only a weak correlation. This means the controllers do not correlate to morphology either. As opposed to the LOESS function of the tibia parameters, we do not see any LOESS functions on control parameters generating such a clear relationship to the speed.

This result is a bit surprising. We expected the controller to learn how to adjust to the fitness trade-off as well. It could be many reasons why this cannot be observed. With 18 parameters to train this search space is a lot more complex. The controllers might simply not be trained enough. They might struggle with premature convergence, as a consequence of imperfect hyperparameters for evolving control.

Although, after visualizing some of the evolved solutions, we saw several robots move quite well. This suggests that both the hyperparameters and the developed controllers are good enough.

Another possible explanation might be the choice of correlation measurements. The correlation measurements might not be able to detect how the control affects the fitness. If there is a non-monotonic function, neither Pearson's nor Spearman's correlation will detect this. When investigating the plotting of the four control parameters in figure 5.5, we see a similar tendency between parameters 4, 8, and 9. Only 8 and 9 show a weak correlation to the speed, while all three seem to split into two groups. One group of solutions achieves high speed with very low parameter values, while the other set achieves high speed with high parameter values. The same tendencies are seen in multiple control parameters in plot A.6 in Appendix A. We also see several confidence intervals of the LOESS function in 5.6 grow towards the end. It is more solutions in the low-speed area, but these intervals grow considerably more at the end than the LOESS function of the morphology parameters did.

Based on these plots it might seem like there actually is a correlation, but the measurements fail to discover them. The correlation might just be too complex to easily measure. This is a limitation of the analysis.

If a more complex correlation exists, it also means there is a similarly complex correlation between the control and the morphology.

6.1.4 Analysis of the supporting experiments

When investigating the hypervolume in figure 5.7, we see that some of the runs with alternative hyperparameters are competitive with the best

runs in the main experiment. Although we only have three evolutionary runs from each of the supporting experiments, we see that the main run has less variance across multiple runs than the others. To investigate the hypothesis in this thesis it is not preferable to maximize the hypervolume at the expense of the variance. We want to have a more stable variance in the runs of the main experiment, to create comparable data. This research is not aiming to find optimal solutions, but rather to analyze the effect the process has on its results. For this cause, the parameters chosen in the main experiments are the best ones of those we tested.

6.1.5 Limitations

Even though these experiments use a lot of processing power to train the robot over relatively many generations, we cannot generalize the achieved results too much. Both the robot, the environment, and the task in this study are very simple, but even such a simple setup has an extremely difficult search environment. During testing, we saw drastic changes to the search environment for even small adjustments. Given a completely different robot and/or environment, an absolute new search space will be present. The results and conclusions we can draw from the experiments might not be applicable for such other search spaces. The type of robot and fitness measurement will also play a crucial role. We use a four-legged robot with a spider configuration, together with speed and stability fitness. The amount of fitness trade-off between other morphologies and fitness measurements can differ greatly.

6.2 Future work

As this thesis try to find a correlation between the fitness trade-off and the training parameters, a lot of work could be done to investigate this further and use these results in other experiments.

Investigating more complex correlation. We do not manage to find much correlation between the control parameters and the speed fitness, but we do believe it exists a more complex correlation than we manage to find. To investigate such a complex correlation both between control and fitness, but also between morphology and control, would be very compelling. A deeper insight into this correlation would be very helpful to understand the general control-morphology relationship during evolution.

Evolving with pretrained morphology. These experiments produce a range of morphology solutions, partly fitted to an area of the Pareto front. With the right set of hyperparameters, it would be interesting to see if one could take these pretrained morphologies as a basis for a new evolution where only the control would evolve. This would further underline the morphology's ability to a Pareto front adjustment.

Handpick morphology solution. From the results achieved here, it would be interesting to see if the LOESS plots of morphology parameters in figure 5.4, could be used as a generalization for morphology solutions.

One may be able to handpick a great morphology for a specific goal based on these results.

More advanced morphology. The robot we use has quite a simple body, with simple changes. To create a robot with other morphological changes than adjusted leg lengths and leg placement would be a lot more advanced, but very interesting. Other options could be adjustable shapes for legs and body, adjustable joint limits, or a varying number of legs. Such changes could lead to a lot more advanced dependencies, and interesting fitness trade-offs could be seen.

Real-world robot. In the background section, we have seen some examples of adjustable morphologies on real-world robots. Although there are still few developed robots that manage this, and the adjustments are very limited, it would be very interesting to see if one could get such a robot to adjust to a trade-off fitness as well. It would be a lot more difficult to do this to a real robot, but if achieved, it would be an important result for morphological evolution.

More advanced environment. A very simple environment is used in this thesis. The robot train on flat ground, with no obstacles, and constant friction. If we introduce random noise or different terrain, we could maybe achieve more stable solutions. Maybe the morphology could adjust to both a fitness trade-off and the terrain.

Lifetime learning. We use an NSGA-2 algorithm for optimizing the solutions. As described in the background section, we have seen a lot of research and promising results from lifetime learning in ER. Doing the same experiment on a lifetime learning evolution would be very interesting. The controller of an individual could then have more time to adapt to its body, and from this, we might see very different and interesting trade-offs.

Other fitness measurements. The two fitness measurements in this thesis are speed and stability. Although we do see a morphological adjustment to the trade-off between these two, other choices for fitness might give completely different results. Trying to optimize three fitness measurements at the same time could also be very interesting.

Chapter 7

Conclusion

A simulated four-legged robot was implemented in this thesis. Both the control and morphology of this robot were trained by a multi-objective evolutionary algorithm with fitness measured as speed and stability. As most robots trained by evolutionary algorithms only optimize control, the aim of the study was to emphasize the importance of including morphology optimizing as well. Throughout extensive evolutionary runs, it was shown a strong, significant correlation between the fitness trade-off and the tibia length for both front and rear legs. The femur length, both front and rear, showed a moderate correlation to the fitness trade-off, while no correlation was found for front and rear base positions. With the correlation measurements used here, it was not found a significant correlation between the control and the fitness.

For this type of robot, under these conditions, we have shown that co-evolution of both control and morphology will lead to a morphology adaptation to the according fitness trade-off. Hence, allowing morphology changes during evolution does make an impact on the fitness and would be preferable in this setting.

The results achieved suggest that we can both find morphologies adapted to the fitness throughout co-evolution, and we have also achieved generalized results which we can be used as templates to design usable morphologies ourselves for this environment. These results underline the importance and the possibilities of doing morphology adaptation in evolutionary robotics.

Bibliography

[1] Reem J Alattas, Sarosh Patel, and Tarek M Sobh. "Evolutionary Modular Robotics: Survey and Analysis." In: *Journal of Intelligent & Robotic Systems* 95 (2019), pp. 815–828. DOI: 10.1007/s10846-018-0902-9.

[2] Joshua E. Auerbach and Josh C. Bongard. "Environmental Influence on the Evolution of Morphological Complexity in Machines." In: *PLoS Computational Biology* 10.1 (2014). ISSN: 1553734X. DOI: 10.1371/journal.pcbi.1003399.

[3] Josh Bongard. "Morphological change in machines accelerates the evolution of robust behavior." In: *Proceedings of the National Academy of Sciences of the United States of America* 108.4 (2011). ISSN: 00278424. DOI: 10.1073/pnas.1015390108.

[4] Luzius Brodbeck, Simon Hauser, and Fumiya Iida. "Morphological evolution of physical robots through model-free phenotype development." In: *PLoS ONE* 10.6 (2015). ISSN: 19326203. DOI: 10.1371/journal.pone.0128444.

[5] Nick Cheney, Josh C. Bongard, and Hod Lipson. "Evolving soft robots in tight spaces." In: *GECCO 2015 - Proceedings of the 2015 Genetic and Evolutionary Computation Conference*. 2015. DOI: 10.1145/2739480.2754662.

[6] Nick Cheney et al. "On the difficulty of co-optimizing morphology and control in evolved virtual creatures." In: *Proceedings of the Artificial Life Conference 2016, ALIFE 2016*. 2016. DOI: 10.7551/978-0-262-33936-0-ch042.

[7] Nick Cheney et al. "Scalable co-optimization of morphology and control in embodied machines." In: *arXiv* (2017). ISSN: 23318422. arXiv: 1706.06133.

[8] Steve Collins et al. "Efficient bipedal robots based on passive-dynamic walkers." In: *Science* 307.5712 (2005). ISSN: 00368075. DOI: 10.1126/science.1107799.

[9] A. Cully and J. B. Mouret. "Evolving a behavioral repertoire for a walking robot." In: *Evolutionary Computation* 24.1 (2016). ISSN: 15309304. DOI: 10.1162/EVCO_a_00143.

[10] Kalyanmoy Deb et al. "A fast and elitist multiobjective genetic algorithm: NSGA-II." In: *IEEE Transactions on Evolutionary Computation* 6.2 (2002). ISSN: 1089778X. DOI: 10.1109/4235.996017.

[11] Stephane Doncieux et al. *Evolutionary robotics: What, why, and where to*. 2015. DOI: 10.3389/frobt.2015.00004.

[12] Stéphane Doncieux et al. "Evolutionary robotics: Exploring new horizons." In: *Studies in Computational Intelligence*. Vol. 341. 2011. DOI: 10.1007/978-3-642-18272-3_1.

[13] A.E. Eiben et al. "The Triangle of Life: Evolving Robots in Real-time and Real-space." In: 2013. DOI: 10.7551/978-0-262-31709-2-ch157.

[14] Hilmar Elverhøy et al. "Ankle Joints Are Beneficial When Optimizing Supported Real-world Bipedal Robot Gaits." In: *arXiv preprint arXiv:2105.10764* (2021).

[15] Kyrre Glette et al. "Evolution of locomotion in a simulated quadruped robot and transferral to reality." In: *Proceedings of the Seventeenth International Symposium on Artificial Life and Robotics* (2012).

[16] Daoxiong Gong, Jie Yan, and Guoyu Zuo. "A Review of Gait Optimization Based on Evolutionary Computation." In: *Applied Computational Intelligence and Soft Computing* 2010 (2010). ISSN: 1687-9724. DOI: 10.1155/2010/413179.

[17] Mario A. Gongora, Benjamin N. Passow, and Adrian A. Hopgood. "Robustness analysis of evolutionary controller tuning using real systems." In: *2009 IEEE Congress on Evolutionary Computation, CEC 2009*. 2009. DOI: 10.1109/CEC.2009.4983001.

[18] Roderich Groß et al. "Autonomous self-assembly in swarm-bots." In: *IEEE Transactions on Robotics* 22.6 (2006). ISSN: 15523098. DOI: 10.1109/TRO.2006.882919.

[19] Matthew F. Hale et al. "The ARe robot fabricator: How to (Re)produce robots that can evolve in the real world." In: *Proceedings of the 2019 Conference on Artificial Life: How Can Artificial Life Help Solve Societal Challenges, ALIFE 2019*. 2020. DOI: 10.1162/isal_a_00147.xml.

[20] Huub Heijnen, David Howard, and Navinda Kottege. "A testbed that evolves hexapod controllers in hardware." In: *Proceedings - IEEE International Conference on Robotics and Automation*. 2017. DOI: 10.1109/ICRA.2017.7989128.

[21] Himanshu Jain and Kalyanmoy Deb. "An improved adaptive approach for elitist nondominated sorting genetic algorithm for many-objective optimization." In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 7811 LNCS. 2013. DOI: 10.1007/978-3-642-37140-0_25.

[22] Nick Jakobi, Phil Husbands, and Inman Harvey. "Noise and the reality gap: The use of simulation in evolutionary robotics." In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 929. 1995. DOI: 10.1007/3-540-59496-5_337.

[23] Milan Jelisavcic et al. "Lamarckian Evolution of Simulated Modular Robots." In: *Frontiers in Robotics and AI* 6 (2019). ISSN: 2296-9144. DOI: 10.3389/frobt.2019.00009.

[24] Milan Jelisavcic et al. "Real-World Evolution of Robot Morphologies: A Proof of Concept." In: *Artificial Life* 23.2 (2017). ISSN: 15309185. DOI: 10.1162/ARTL_a_00231.

[25] Jérôme Kodjabachian and Jean Arcady Meyer. "Evolution and development of neural controllers for locomotion, gradient-following, and obstacle-avoidance in artificial insects." In: *IEEE Transactions on Neural Networks* 9.5 (1998). ISSN: 10459227. DOI: 10.1109/72.712153.

[26] Abdullah Konak, David W. Coit, and Alice E. Smith. "Multi-objective optimization using genetic algorithms: A tutorial." In: *Reliability Engineering and System Safety* 91.9 (2006). ISSN: 09518320. DOI: 10.1016/j.ress.2005.11.018.

[27] Sylvain Koos, Antoine Cully, and Jean Baptiste Mouret. "Fast damage recovery in robotics with the T-resilience algorithm." In: *International Journal of Robotics Research* 32.14 (2013). ISSN: 02783649. DOI: 10.1177/0278364913499192.

[28] Sylvain Koos, Jean Baptiste Mouret, and Stéphane Doncieux. "Crossing the reality gap in evolutionary robotics by promoting transferable controllers." In: *Proceedings of the 12th Annual Genetic and Evolutionary Computation Conference, GECCO '10*. 2010. DOI: 10.1145/1830483.1830505.

[29] Sylvain Koos, Jean Baptiste Mouret, and Stéphane Doncieux. "The transferability approach: Crossing the reality gap in evolutionary robotics." In: *IEEE Transactions on Evolutionary Computation* 17.1 (2013). ISSN: 1089778X. DOI: 10.1109/TEVC.2012.2185849.

[30] Sam Kriegman, Nick Cheney, and Josh Bongard. "How morphological development can guide evolution." In: *Scientific Reports* 8.1 (2018), pp. 1–10. ISSN: 20452322. DOI: 10.1038/s41598-018-31868-7. arXiv: 1711.07387.

[31] Wei Po Lee, John Hallam, and Henrik H. Lund. "Hybrid GP/GA approach for co-evolving controllers and robot bodies to achieve fitness-specified tasks." In: *Proceedings of the IEEE Conference on Evolutionary Computation*. 1996. DOI: 10.1109/icec.1996.542394.

[32] Joel Lehman and Kenneth O. Stanley. "Evolving a diversity of creatures through novelty search and local competition." In: *Genetic and Evolutionary Computation Conference, GECCO'11*. 2011. DOI: 10.1145/2001576.2001606.

[33] John Maynard Smith. *Byte-sized evolution*. 1992. DOI: 10 . 1038 / 355772a0.

[34] Aslan Miriyev, Kenneth Stack, and Hod Lipson. "Soft material for soft actuators." In: *Nature Communications* 8.1 (2017). ISSN: 20411723. DOI: 10.1038/s41467-017-00685-3.

[35] Rodrigo Moreno and Andres Faiña. "EMERGE Modular Robot: A Tool for Fast Deployment of Evolved Robots." In: *Frontiers in Robotics and AI* 8 (2021). ISSN: 22969144. DOI: 10.3389/frobt.2021.699814.

[36] Jean Baptiste Mouret and Konstantinos Chatzilygeroudis. "20 Years of reality gap: A few thoughts about simulators in evolutionary robotics." In: *GECCO 2017 - Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 2017, pp. 1121–1124. ISBN: 9781450349390. DOI: 10.1145/3067695.3082052.

[37] Andrew L. Nelson, Gregory J. Barlow, and Lefteris Doitsidis. "Fitness functions in evolutionary robotics: A survey and analysis." In: *Robotics and Autonomous Systems* 57.4 (2009), pp. 345–370. ISSN: 09218890. DOI: 10.1016/j.robot.2008.09.009.

[38] Stefano Nolfi and Dario Floreano. "Learning and evolution." In: *Autonomous robots* 7.1 (1999), pp. 89–113.

[39] Tønnes F Nygaard, David Howard, and Kyrre Glette. "Real world morphological evolution is feasible." In: *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*. 2020, pp. 1392–1394.

[40] Tønnes F. Nygaard, Eivind Samuelsen, and Kyrre Glette. "Overcoming initial convergence in multi-objective evolution of robot control and morphology using a two-phase approach." In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 10199 LNCS. 2017. DOI: 10.1007/978-3-319-55849-3_53.

[41] Tønnes F. Nygaard, Eivind Samuelsen, and Kyrre Glette. "Overcoming initial convergence in multi-objective evolution of robot control and morphology using a two-phase approach." In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 10199 LNCS. 2017. DOI: 10.1007/978-3-319-55849-3_53.

[42] Tonnes F. Nygaard, Jim Torresen, and Kyrre Glette. "Multi-objective evolution of fast and stable gaits on a physical quadruped robotic platform." In: *2016 IEEE Symposium Series on Computational Intelligence, SSCI 2016* (2016). DOI: 10.1109/SSCI.2016.7850167.

[43] Tønnes F. Nygaard et al. "Environmental Adaptation of Robot Morphology and Control Through Real-world Evolution." In: *Evolutionary Computation* (Mar. 2021), pp. 1–21. ISSN: 1530-9304. DOI: 10.1162/evco_a_00291.

[44] Tønnes F. Nygaard et al. "Experiences from real-world evolution with DYRET: Dynamic robot for embodied testing." In: *Communications in Computer and Information Science*. Vol. 1056 CCIS. 2019. DOI: 10.1007/978-3-030-35664-4_6.

[45] Tønnes F. Nygaard et al. "Real-world embodied AI through a morphologically adaptive quadruped robot." In: *Nature Machine Intelligence* (Mar. 2021), pp. 1–10. ISSN: 25225839. DOI: 10.1038/s42256-021-00320-3.

[46] Tønnes F. Nygaard et al. "Real-world evolution adapts robot morphology and control to hardware limitations." In: *GECCO 2018 - Proceedings of the 2018 Genetic and Evolutionary Computation Conference*. 2018. DOI: 10.1145/3205455.3205567.

[47] Tønnes Frostad Nygaard. "Evolutionary optimization of robot morphology and control - Using evolutionary algorithms in the design of a six legged robotic platform." MA thesis. Department of Informatics, University of Oslo, 2014.

[48] Randal S. Olson et al. "Predator confusion is sufficient to evolve swarming behaviour." In: *Journal of the Royal Society Interface* 10.85 (2013). ISSN: 17425662. DOI: 10.1098/rsif.2013.0305.

[49] Matt Quinn et al. "Evolving controllers for a homogeneous system of physical robots: Structured cooperation with minimal sensors." In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 361.1811 (2003). ISSN: 1364503X. DOI: 10.1098/rsta.2003.1258.

[50] Else Line Ruud, Eivind Samuelsen, and Kyrre Glette. "Memetic robot control evolution and adaption to reality." In: *2016 IEEE Symposium Series on Computational Intelligence, SSCI 2016* (2017). DOI: 10.1109/SSCI.2016.7850169.

[51] Patrick Schober and Lothar A. Schwarte. "Correlation coefficients: Appropriate use and interpretation." In: *Anesthesia and Analgesia* 126.5 (2018). ISSN: 15267598. DOI: 10.1213/ANE.0000000000002864.

[52] Carlos Segura et al. "Using multi-objective evolutionary algorithms for single-objective optimization." In: *4OR* 11.3 (2013). ISSN: 16142411. DOI: 10.1007/s10288-013-0248-x.

[53] Bruno Siciliano and Oussama Khatib. *Springer handbook of robotics*. 2016. DOI: 10.1007/978-3-319-32552-1.

[54] Karl Sims. "Evolving virtual creatures." In: *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1994*. 1994. DOI: 10.1145/192161.192167.

[55] Pradnya A. Vikhar. "Evolutionary algorithms: A critical review and its future prospects." In: *Proceedings - International Conference on Global Trends in Signal Processing, Information Computing and Communication, ICGTSPICC 2016*. 2017. DOI: 10.1109/ICGTSPICC.2016.7955308.

[56] Wei Wang, Dongbing Gu, and Guangming Xie. "Autonomous Optimization of Swimming Gait in a Fish Robot with Multiple Onboard Sensors." In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 49.5 (2019). ISSN: 21682232. DOI: 10.1109/TSMC.2017.2683524.

[57] Michael Wehner et al. "An integrated design and fabrication strategy for entirely soft, autonomous robots." In: *Nature* 536.7617 (2016). ISSN: 14764687. DOI: 10.1038/nature19100.

[58] Darrell Whitley, V. Scott Gordon, and Keith Mathias. "Lamarckian evolution, the Baldwin effect and function optimization." In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 866 LNCS. 1994. DOI: 10.1007/3-540-58484-6_245.

[59] Andrew D. Wilson and Sabrina Golonka. "Embodied Cognition is Not What you Think it is." In: *Frontiers in Psychology* 4.Article 58 (2013), pp. 1–13. ISSN: 1664-1078. DOI: 10.3389/fpsyg.2013.00058.

[60] Jason Yosinski et al. "Generating gaits for physical quadruped robots: Evolved neural networks vs. local parameterized search." In: *Genetic and Evolutionary Computation Conference, GECCO'11 - Companion Publication*. 2011. DOI: 10.1145/2001858.2001876.

[61] Juan Cristóbal Zagal and Javier Ruiz-Del-Solar. "Combining simulation and reality in evolutionary robotics." In: *Journal of Intelligent and Robotic Systems: Theory and Applications* 50.1 (2007). ISSN: 09210296. DOI: 10.1007/s10846-007-9149-6.

[62] Juan Cristóbal Zagal, Javier Ruiz-del-Solar, and Paul Vallejos. "Back to reality: Crossing the reality gap in evolutionary robotics." In: *Notes* (2004).

Appendix A

Additional experimental results

This section includes additional results and plots for the main experiment.

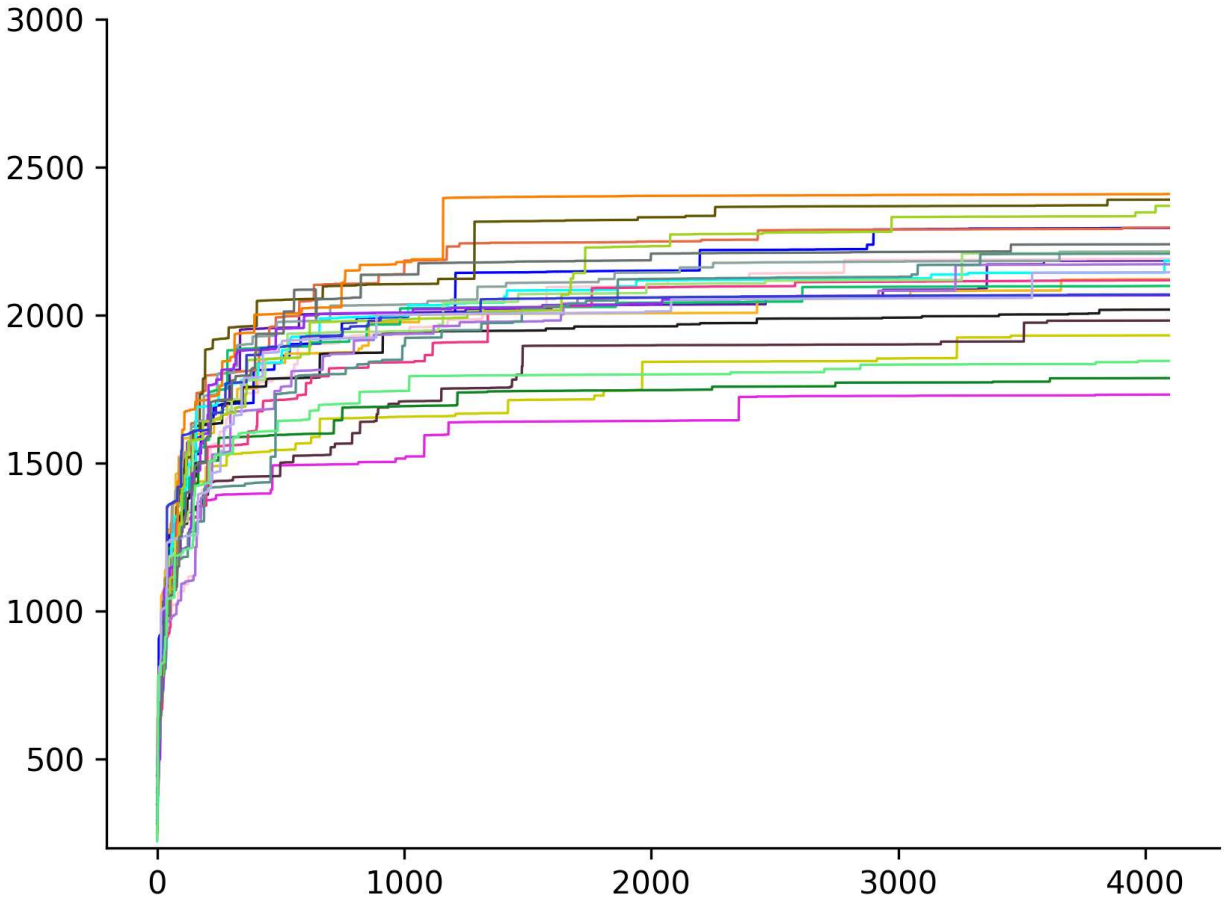


Figure A.1: Hypervolume over all the different runs.

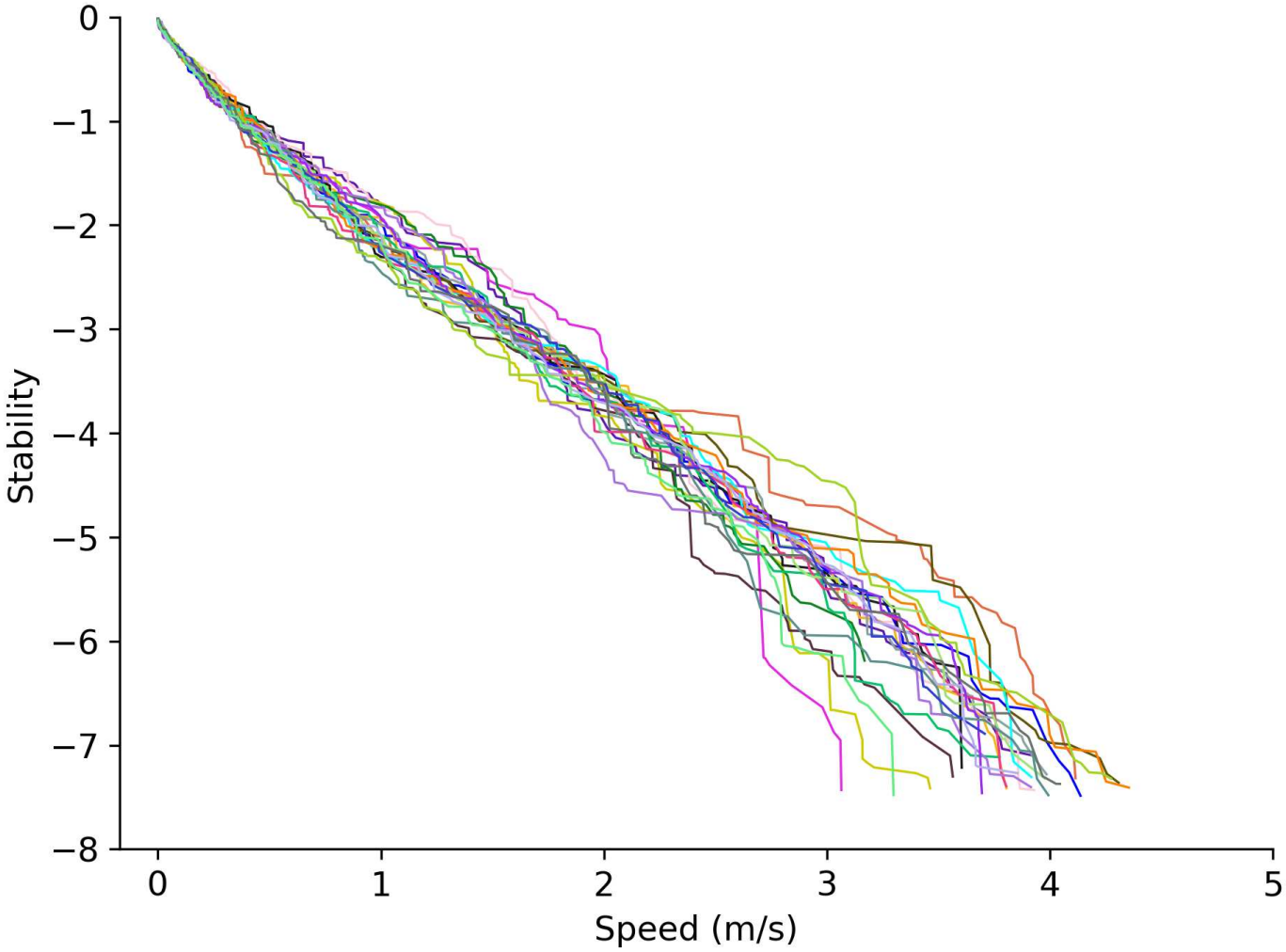


Figure A.2: Each run's Pareto front from the main experiment.

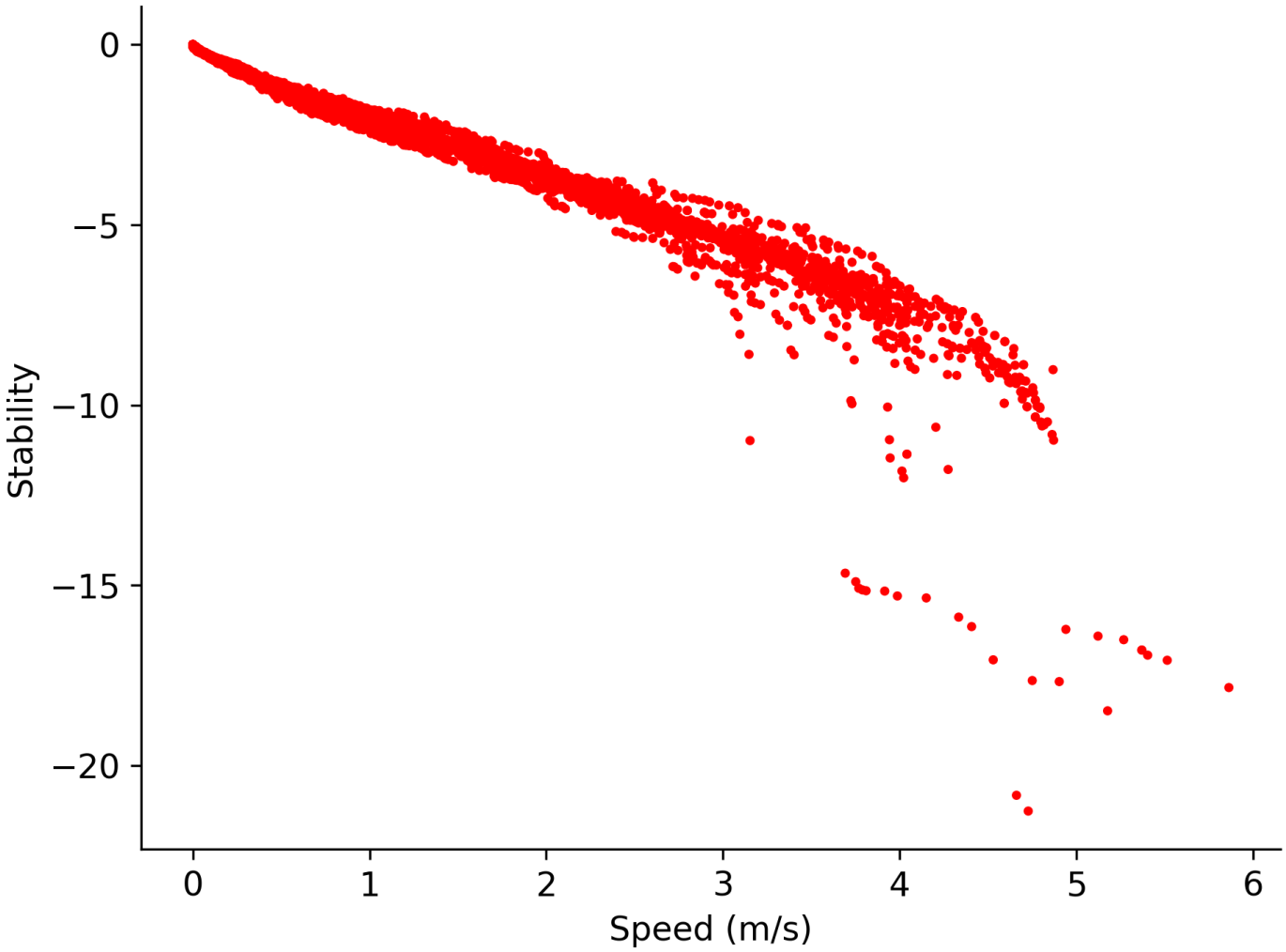
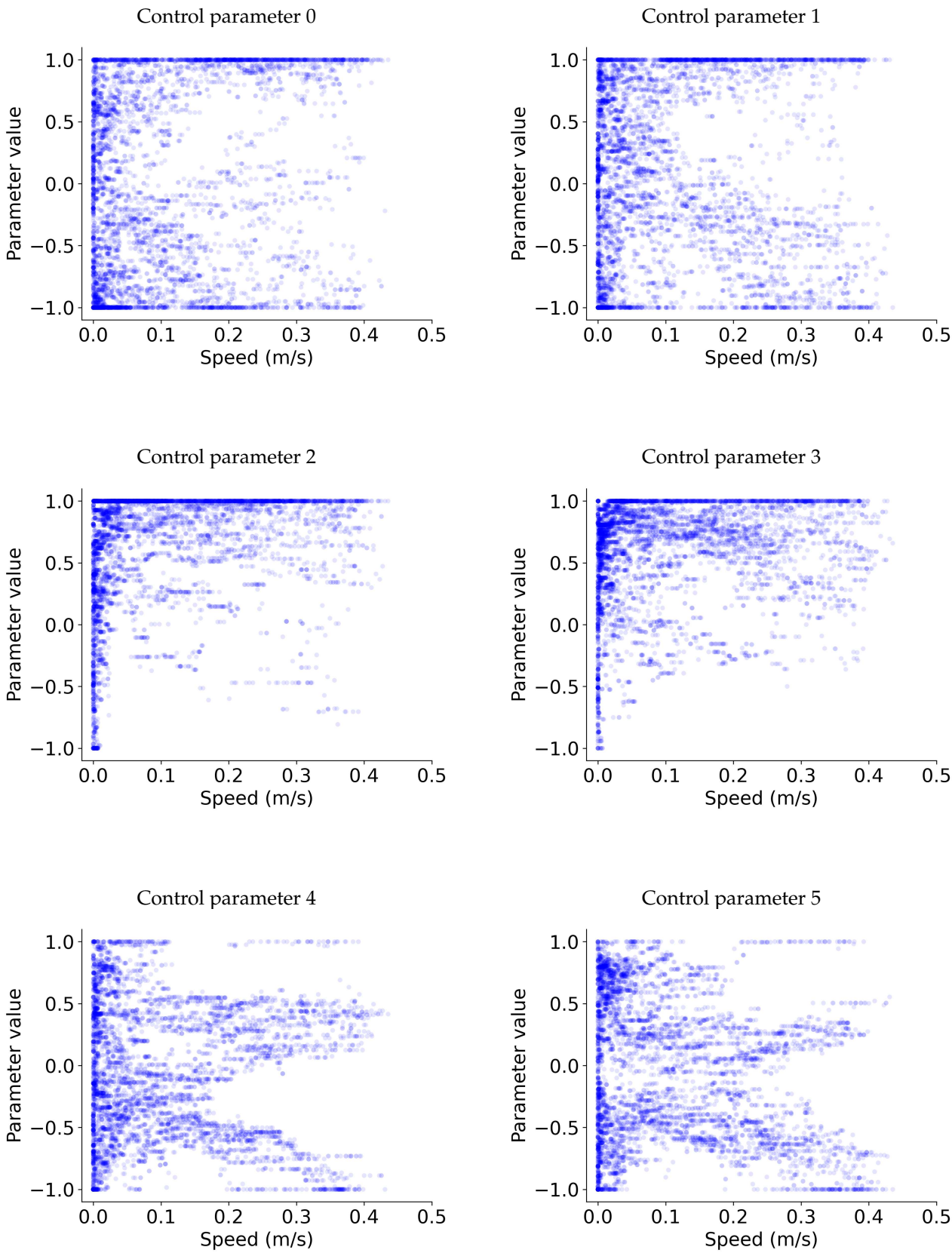
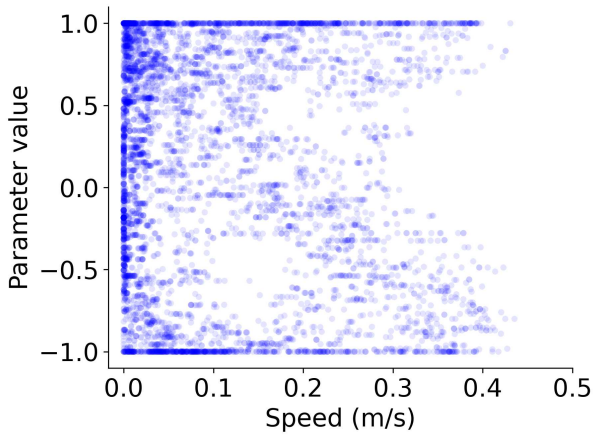


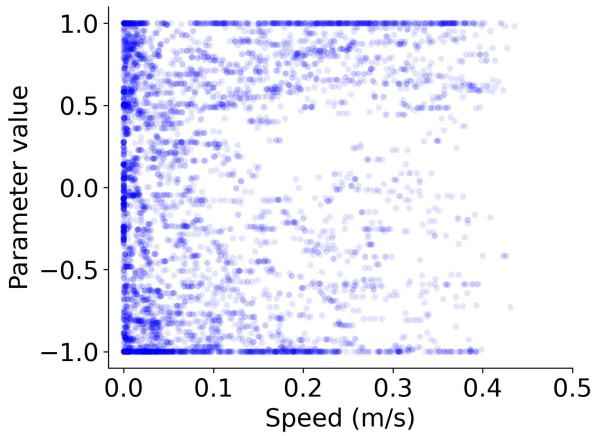
Figure A.3: All Pareto fronts from the main experiment without a stability limit.


Control parameter 0
Control parameter 1
Control parameter 2
Control parameter 3
Control parameter 4
Control parameter 5
Parameter value
Speed (m/s)
1.0
0.5
0.0
−0.5
−1.0
0.0
0.1
0.2
0.3
0.4
0.5

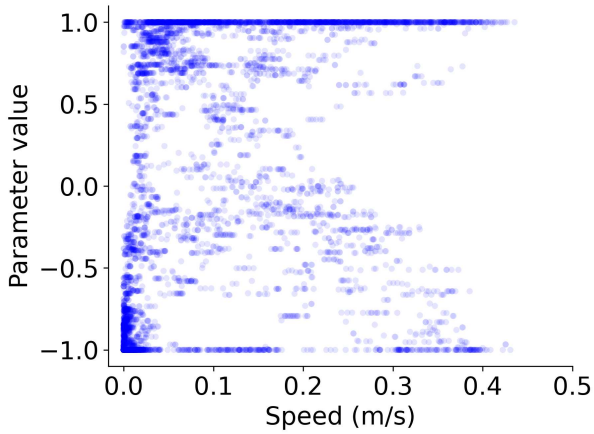
Control parameter 6



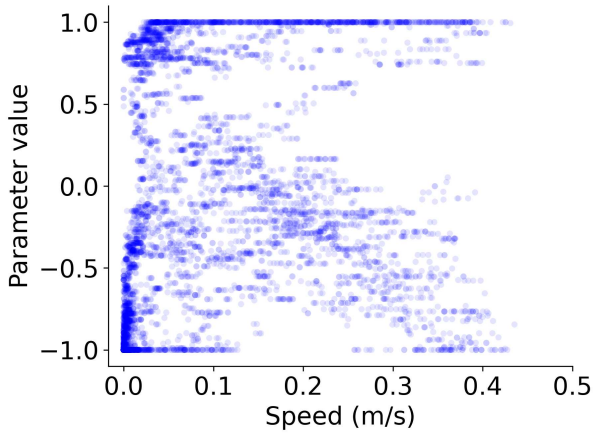
Control parameter 7



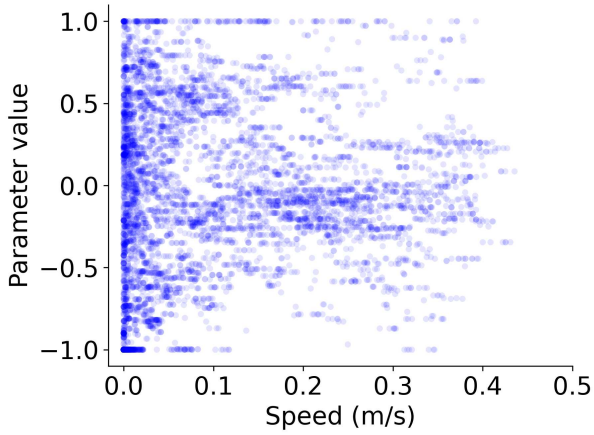
Control parameter 8



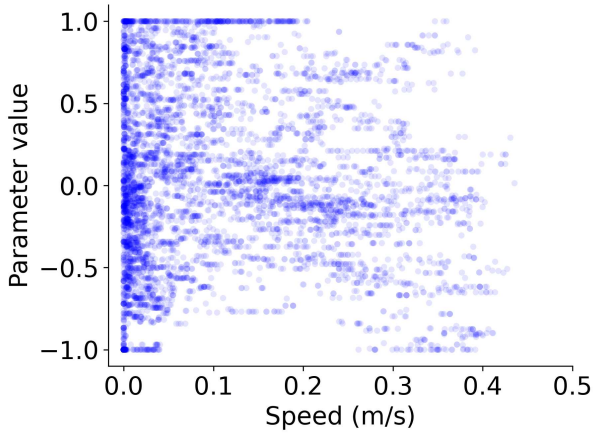
Control parameter 9



Control parameter 10



Control parameter 11



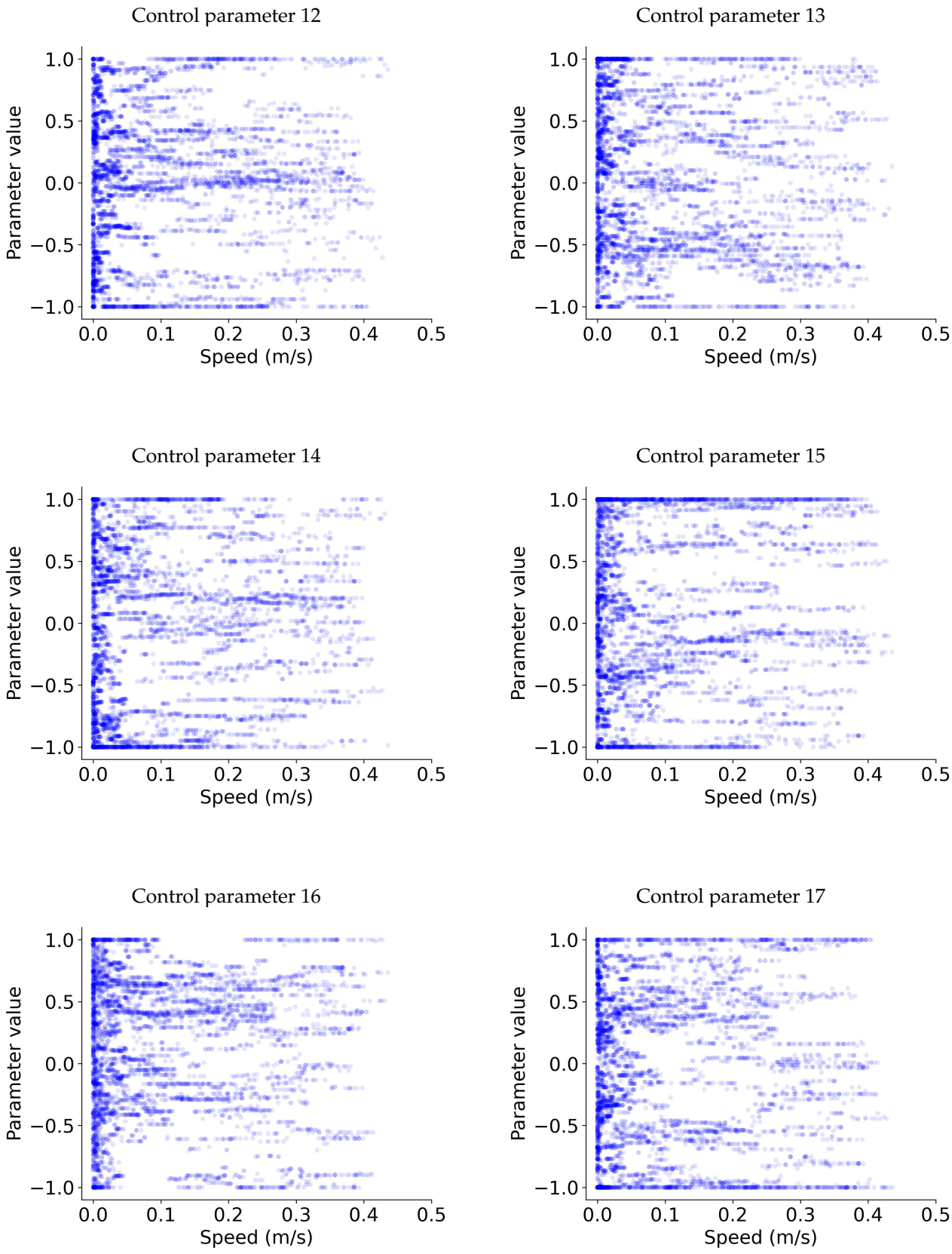


Figure A.6: Scatterplot of all control parameters