Author:
Walker, Katt E

Title:
Evolving Morphological Adaption Methods in Compliant Robots

A dissertation submitted to the University of Bristol in accordance with the requirements for award of the degree of PhD in Robotics and Autonomous Systems in the Faculty of Aerospace Engineering.

Evolving Morphological Adaption Methods in Compliant Robots

Kathryn Elizabeth Walker

October 2020

Supervised by:	Word Count:
Dr. Helmut Hauser	30800

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

SIGNED: Kathryn Elizabeth Walker   DATE: 23/10/2020

Despite the huge success of robotics in general, there are very few machines that are capable to stably locomote in rough, unknown terrain. Whilst a few robots are just becoming able to deal with the uncertainty, complexity and variety typical of natural environments, in the most cases even these cutting edge robots fail. On the other hand, animals, including humans, have the ability to locomote in many environments un-accessible to robots. Additionally, animals are able to outperform robots in almost any category (with respect to locomotion), including in energy efficiency, stability, robustness, agility, and numerous others. There are many possible reasons as to why nature is able to outperform robotic designs. It is speculated that animals are able to successfully locomote without an exact environmental model by outsourcing some of this computation to their well designed morphology, achieved through evolution. Additionally, if adaption to new environments is required, animals not only adapt their behaviour but also in some cases their morphology. They appear to be able to learn from their interaction with the environment and use this information to adapt. Whilst many researchers have improved robot morphology through artificial evolution, recreating adaptive morphology is relatively unexplored. This leads to the question, can artificial evolution be used to find optimal methods of morphological adaption by exploiting feedback from the environment in order to successfully locomote in a wide range of environments?

In an attempt to partly answer this question, this thesis forms two parts. Firstly, the best methods to adapt a simulated Spring Loaded Inverted Pendulum Model (the SLIP model). The SLIP model has a unique property that if the combination of its morphological and control parameters are in a particular range it is self stabilising. The aim of the first part of the thesis is to find ways that the SLIP model can adapt its parameters accordingly to become stable, based on its interaction with the environment. Two approaches are explored; an offline approach, where adaption of the SLIP model occurs between episodes and the model is allowed to fail after each episode and an online approach where instead the adaption takes place between strides. Not only did both methods expand the range of parameters for which self-stability could occur (when compared with a basic SLIP model that has no capacity for learning) but in the case of the online learning the model was now able to withstand environmental changes, such as a decrease in ground level of up to 14 times the length of the spring.

In the second part of the thesis, the optimal morphological adaption of a soft robot is evolved. The evolved morphological adaption method use the distribution of kinetic energy throughout the entire robot to determine which parts to harden and which to soften. If part of the robot becomes too soft, it is removed. Thus an optimal morphology, adapted for the specific environment is sculpted. Here, the results show that as the kinetic energy of the different robot parts depends on the interaction the entire robot has with the each environment, many task-specific final morphologies can be sculpted but crucially they are all created using one single general method of adaption.

Overall, it is hoped that the research presented in this thesis showcases the potential that artificial evolution has for enabling robots to learn to adapt their morphology, based on interaction with the environment, in order to achieve robust locomotion in a wide range of environments.

Contents

List of Figures

List of Tables

1 Introduction

In the past decades the advancement of robotics is undeniable. However, whilst the use of robots is now common place on factory lines and in industrial settings, few robots have been able to succeed outside these fixed, controlled environments. As of yet, robots are unable to successfully deal with unexpected situations and tasks that they have not been trained for.

These limitations of current robots are especially pronounced in the field of locomotion. Natural environments present a level of uncertainty that most robots can't deal with, there are some exceptions such as Big Dog [94].

There are numerous instances where using robots capable of adaptive locomotion, rather than humans, would be highly beneficial if not necessary. For example, robots capable of adaptive locomotion, could be used for search and rescue tasks relating to disaster relief, carrying out tasks unsafe for humans due to the uncertain dangerous environments. Another example as to where using robots instead of humans is in exploration, for example new planets [6]. Harsh environmental conditions such as lack of oxygen make planet exploration challenging for humans, whereas robots, especially those capable of adaptive locomotion offer a promising alternative. Furthermore, nuclear decommissioning is another situation where working surrounded by hazardous radiation would be damaging to human health [50, 27]. Yet another example of where adaptive locomotion in robots would be beneficial is reaching places where, although not hazardous, humans are simply unable to access, for example inspecting and clearing blocked pipework [55]. Therefore, the motivation for robots capable of robust, stable adaptive locomotion spans a large range of disciplines.

In contrast to robots, animals are particularly good at adapting themselves to different environments, including ones they have not yet experienced. This adaption to new tasks and environments occurs over multiple different timescales; i.e., almost instantaneous changes in the nervous system (learning), slower changes over the lifetime of the organism in both its morphology and behaviour (postnatal development), as well changes over numerous generations (evolution). In the last decade, advances in biology have shown the importance of environmental feedback at every stage of development, not just at evolution and learning, as commonly utilised in engineering.

Additionally biological systems do not rely on changes in their behaviour to adapt; they also alter their morphology. This is perhaps most obvious in the evolution timescale, where, for example, organisms living in water have evolved to have fins for swimming, whereas those evolved on land have legs. However, morphological adaption can also occur during the lifetime of the organism.

One example of this postnatal development is the adaptive growth of plants [76, 32]. Photo-convertible molecules found in the cells of plants above ground can be activated by specific wave lengths of light, stimulating sensitive and rapid growth of the stem to position leaves away from shaded areas[32, 103]. Thus, depending on location of a light source the final "morphology" of the plant would appear different, it has adapted to its environment. The same mechanism is triggered by submergence of the plant in water, triggering rapid growth and thus removal of the leaves from underwater [117]. In these instances, the same plant genotype has been exposed to vastly different environments and has used postnatal development to adapt accordingly in order to survive.

Morphological developmental as a response to environmental changes occurs not only in plants but also in animals. For example, Passerine birds that change their musculature to cope with winter [67]. The Arctic fox changes its texture and colour of coat in response to changes in season [76]. The tiger salamander is capable of radical metamorphoses if its aquatic environment becomes uninhabitable, and when a male bluehead wrasse is removed from his harem, a female will change its phenotype completely and become a male [76].

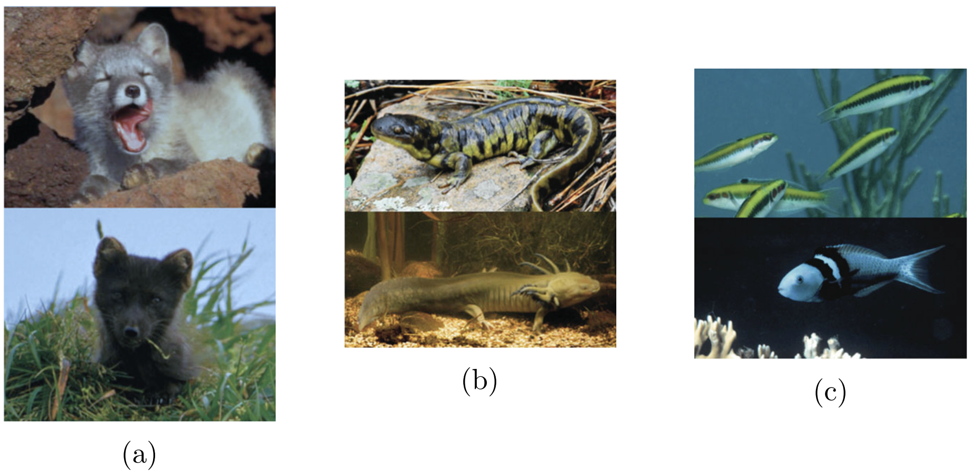


Figure 1.1: Examples of morphological development in nature; all images taken from reference [76]. a) The arctic fox changes the colour and texture of its coat depending on seasonal conditions. b) If the aquatic environment of the tiger salamander becomes uninhabitable it will metamorphose to adapt to the new conditions. c) If the male is removed from the harem of blue wrasses (the females of which are yellow) a female will phenotype completely and become a male.

Whilst changing morphology to adapt to new environments or situations is relatively common place in nature, traditionally is not so common in robotics in any timescale. Typically, robots are built with a predefined and fixed morphology and a corresponding suitable controller is found [88]. Additionally then if a new behaviour is required, usually only the controller is changed, but the morphology of the robot is kept as it is. This works well in traditional robotic applications where the environment is well known and can be controlled, like assembly lines or under lab conditions, but it has its limitations when complex behaviours are required to deal with unexpected situations, e.g. changes in the environment, or new tasks. The required controller becomes often too complicated and the robot is likely to fail.

The concept of *embodied intelligence* promises an alternative approach by placing higher emphasis on the morphology of the robot and its interaction with the environment [90]. Often, designing a more appropriate morphology can significantly improve the performance of a robot, whilst still employing either a very simple control system [19], or in some cases none at all [72, 120]. Furthermore in nature, especially animals, the morphologies are often soft. Compared with robots, which are traditionally made of stiff rigid body parts, animals have soft compliant bodies which they are able to exploit through their interaction with the environment. The dynamics of these soft, compliant morphologies, have been shown to be able to contribute to the overall performance in an animal's locomotion [41]. This is often referred to morphological computation [42].

Researchers are now considering how changes in morphology can be used to adapt robots to new environments or tasks, either on its own or in conjunction with changes in control systems and across varies timescales. Within this area, the most commonly researched timeline for morphological adaption is evolution. However, if the morphology of a robot is adapted to environment via evolution, this morphology is then fixed at the initialisation of the robot. Therefore, it will perform well in one environment but if during the lifetime of the robot the environment changes it will more likely fail.

Therefore, a new approach named Evo-Devo combines adaption on two timescales; evolution and over the lifetime of the robot (development). In the Evo-Devo field of research, the way the robot adapts over its lifetime could also be evolved; rather than a single adapted morphology or behaviour. Whilst there any many examples of using and Evo-Devo approach to design adaptive control systems, there a very limited examples of evolving methods of altering robot morphology throughout its lifetime.

In this thesis, I take an Evo-Devo approach and use evolutionary algorithms to evolve the way a compliant robot develops/changes its morphology over its lifetime in order to adapt to different environments. Furthermore, in my research the way the robot adapts its morphology is based on its interactions with the environment, relying on its embodied intelligence in order to successfully adapt.

In the rest of this chapter, I further introduce the key concepts of this thesis. I first discuss Morphological Computation in section 1.1 and provide additional evidence in support of the importance of an optimised robot morphology. Then I give an overview of evolutionary robotics in general in the section 1.2.

I then provide a literature review of the existing research relevant to this project. First, I consider the different ways that researchers have adapted the morphology of robots to different environments via evolution. In the case of evolutionary robotics there is a vast amount of research, so here I mainly consider examples where the researchers have focused on morphology. I then provide an overview of the existing research related to postnatal morphological adaption, followed by a section of Evo-Devo design. In this later section (i.e., the Evo-Devo section). Here, I also provide examples where researchers have evolved methods of adapting robot control.

After the literature review I contextualise my thesis aims and provide an overview of the structure of the rest of the document.

1.1 Morphological Computation

In this subsection, the concept of *morphological computation* is explained in more detail and the relevant literature surrounding this field of research is presented. I also discuss how the work detailed in this thesis relates to the current research area.

As previously mentioned, traditionally when designing robots, researchers have placed more emphasis on sophisticated control systems than the morphology design. It should be noted here that the word morphology does not just encompass the shape of the system (be that a biological organism, or robot) but also the dynamic properties, for example the stiffness of the individual parts. Typically, researches first design a morphology and then work to find a suitable controller. This design methodology, featuring non-compliant robots, has worked well when robot is required to perform a single task, where precision, speed and little interaction with the environment are desired, but it appears to fall down in situations where more adaptability is important [89]. Therefore with the aim of increasing adaptability, researchers keep the morphology of the robot the same and try to change or improve the complexity of the controller. In many cases this causes the robot to fail. Furthermore, traditionally, when designing the morphology of a robot, rigid body parts are favoured over soft compliant structures which are deemed less "good" versions of their stiff counterparts. This could be in part because soft morphologies have a larger number of degrees of freedom and are in many cases non-linear, therefore they are hard to model and control. Thus the design process is more complicated.

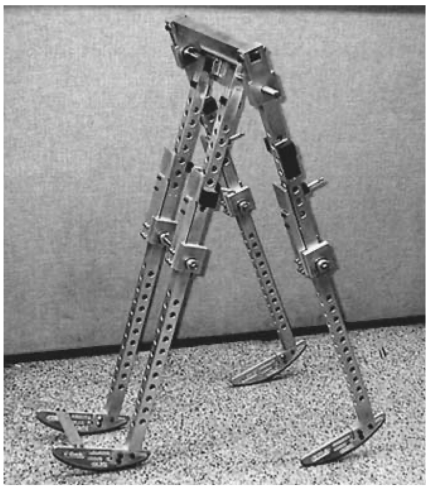


Figure 1.2: An replica of the first passive walker designed by McGeer at el. in 1990 [72]. The details of this replica are detailed in the work by Collins et al. [20] from which the image is also taken.

Researchers are therefore now looking for alternative ways to design better, more adaptive robots and thus placing the emphasis back on morphology, including considering the use of compliant bodies instead of rigid ones. Instead of viewing these soft morphologies as a hard-to-model nuisance, some researchers are now considering them to be a highly beneficial feature of the robot, able to outsource some of the required computation; i.e., they are a potential computational resource [40]. This concept, in robots, is referred to as *morphological computation*.

Here, I explore examples of the use of morphological computation to aid the design of *locomotive robots*. Note that there are other examples of areas of robotics where outsourcing computation to the morphology is also useful, such as passive grippers and vision [77, 65] that are not considered in this review.

One of the earliest and most researched examples of using the concept of *Morphological Computation* to design robots for locomotion is passive walkers, the first example of which was by McGeer at al. in 1990 [72], see Figure 1.2.

A passive walker is a simple robot that is capable of locomotion even though it has no motors or control electronics. Therefore, its ability to walk is solely due to the morphological parameters of the walker i.e., the leg segment lengths, mass distribution, and foot shape and the interaction this body has with its environment (a downwards slope) [77]. Therefore, it can be seen that although these passive walkers have no active control, when walking down slopes are able to exhibit intelligent behaviour, i.e., the balance of the robot is robustly maintained whilst walking. Additionally, passive walker designs have even been shown to have stability, finding stable locomotion despite errors in starting conditions [120]. In the case of McGeer et al. [72] the functionality of the walker is trivial, limited only to locomotion on a downward slope. However, building on this pioneering work, researchers such as Wisse (2006) and Collins (2005) [121, 19] have created robots now capable of locomotion also on horizontal ground by adding a very simple controller. Since most of the control of the robot is still down to the interaction between the body and the environment, the required control system is incredibly simple. Furthermore, utilising a "clever" morphology, i.e., a passive walker, can also improve a robots ability to learn function, as shown in the work by Tendrake et al. in 2005 [115], where the time required for the robot to learn to walk was reduced.

Another, more complex, example of where an appropriate morphological design aids the locomotion of a robot are the case of the quadruped "Puppy" by Iida et al. in 2006 [49]. This is shown in Figure 1.3 Here, simple springs are used as artificial muscles, allowing for not only fast but also highly robust locomotion with no sensory feedback.

A further example is a robotic fish, named "Wanda", created by Ziegler et al. also in 2006 [126]. In Wanda, the body of the fish is one dimensional and movement is achieved by wiggling its elastic tail fin back and forth. It therefore exploits the dynamics between its physical body and its environment and is able to achieve a number of behaviours (movement up, down left and right) through this simple interaction.

More recently, another example of morphological computation is the work done by Howison et al. (2019, 2020) [46, 47]. In this work they explore how by varying the parameters of a simple V shaped piece of paper, different and distinct behaviours can be achieved when the paper is dropped from a height of 3m. In this example, once again there is no direct controller and different behaviours are gained through how the morpologies interact with the environment.

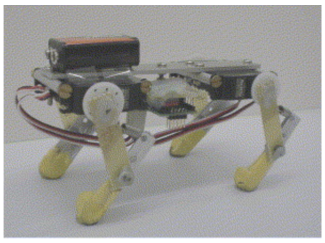


Figure 1.3: The “Puppy” robot provides and example of where an appropriate morphology can aid the locomotion of a robot. Image taken from [49]

A more abstract implementation of morphological computation is Tensegrity robots. Tensegrity robots are built from a specific combination of rigid struts and compliant strings [53]. It has been shown through numerous examples, e.g., Rieffel et al. (2008) [97], Paul et al. (2005) [86], Bruce et al. (2014) [13] and Iscen et al. (2013) [51], that evolutionary algorithms are able to be used as a powerful tool to evolve complex tensegrity structures that only require very simple controllers to induce locomotion. Locomotion is therefore achieved by indirectly exploiting the dynamics of the physical body. See Figure 1.4 for an example of Tensegrity robots.

Whilst the concept of morphological computation has been shown to be a powerful tool for designing robust robots capable of energy efficient locomotion there are two main points to consider. Firstly, designing suitable morphologies for a given task is challenging, especially, if soft compliant bodies are to be used. Since the bodies are non-linear with high degrees of freedom (which can be a benefit), it means they are hard to model, and therefore their desired interaction with the environment is not always obvious. One way to solve this issue is to evolve suitable morphologies, the methodology and examples of which is described in the next subsection (**Evolutionary Robotics, Section 1.2**).

Secondly, in all the above, the functionality implemented in a morphology is deemed to be fixed, i.e., they are specialised to one task that they perform well. However, truly autonomous robots should be highly flexible and therefore capable to adapt to changes in the environment and to new tasks [39]. In case of morphological computation, in order to change the desired computation to be carried out, the underlying morphology has to be altered, the investigation of which is one of the main themes of this thesis. Therefore the ideas of **morphosis** are discussed in Section 1.4.

1.2 Evolutionary Robotics

Even in traditional robot design, there are many different aspects to consider; its morphology, sensory apparatus, motor system, control architecture, and it not clear how best to combine these parts in order to produce the best robot [26]. Whilst morphological computation has the potential to outsource some of the necessary control to the morphology, and therefore also the potential to simplify the system as a whole, in actuality designing a suitable morphology is not trivial, especially when considering soft compliant bodies.

The idea of using artificial evolution to solve hard problems in optimization, modeling, and design is almost as old as computers themselves [26]. The concept takes its inspiration from natural evolution and Darwin's theory of evolution, a population of solutions to a problem is randomly created and each solution is tested against a particular criteria. Those that perform well at the task produce offspring (through a number of possible methods chosen by the algorithm designer, discussed below), those that perform poorly are discarded. This continues for many generations until the algorithm converges.

The field of Evolutionary Computing is vast and encompasses many areas, in this thesis I focus on Evolutionary Robotics (although even just the research surrounding Evolutionary Robotics considerable!). As alluded to above, since there are many aspects to robot, e.g. control, morphology, there are many potential ways to utilize evolutionary robots.

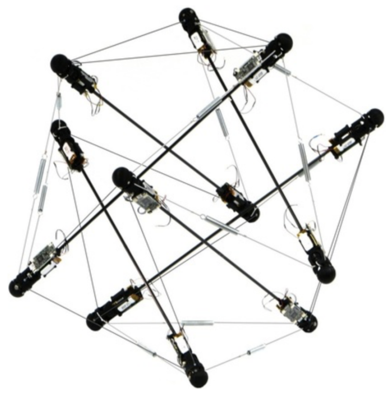


Figure 1.4: An example of a tensegrity robot. These robots are build from a specific combination of rigid struts and compliant strings. This image is taken from the Columbia University Creative Machine Labs website (https://www.creativemachineslab.com) accessed 14/04/2021)

No matter what is being evolved, be that the morphology, control or both, soft or rigid robots, the main ingredients of an evolutionary robotics experiment remain the same and are described in detail below.

First, an initial (usually randomised) *population* of candidate robots is created. However, finding a way to simply represent different robots making them suitable for use in an evolutionary algorithm is in its self challenging. The encoding of the robot is termed "genotypes". This genotype is then translated in a phenotype which is then tested. Note that if the genotype, analogous to genetic description of animals, the phenotype is analogous to the physical animal body [60]. The encoding used is down to the designer of the evolutionary algorithm and, as a result, how these robots are encoded in genotypes has been the subject of much consideration over the years. The simplest form of encoding is a "direct" method, where each the genotype directly maps to a phenotype and all components and attributes are simply listed. Of course if the robot required to be described is particularly complicated, direct encoding can have its disadvantages. Comparisons of encoding for evolutionary robotics can be found in [58, 116].

The fitness of each member of the population is then evaluated. This means firstly translating the genotype, i.e., the encoding, into a phenotype. Each phenotype is then placed in a particular environment and allowed to interact with said environment for some time. The resulting behaviour is observed and measured against a particular criteria to determine the fitness. The criteria used is again down to the designer of the evolutionary algorithm; and example fitness function could be distance travelled in a set time frame.

The final part of evolutionary robotics is that of the evolutionary algorithm. As previously mentioned, this is the part that takes the genotypes that translate into well performing phentopyes, and manipulates them to form the next generation. This manipulation is usually one of three types: reproduction, crossover or mutation. In reproduction the genotype is simply copied into the next generation unchanged. Often the genotypes selected for reproduction are those that have the best performing phenotypes, if this is the case it is normally termed *elitism*. Crossover forms a "child" genotype from combing two "parent" genomes. In mutation, an individual part of single "parent" genotype is randomly changed to create a new genotype. These different methods are used to generate a new generation of candidate solutions which are in turn tested against the fitness critera. Note that a new generation is usually created from a mixture of the methods, for example 50% new genotypes may be created by crossover, 40% by mutation and 10% by reproduction. This continues for many generations until an optimal solution is found – indicated by convergence of the algorithm. The parameters of the evolutionary algorithm, i.e, the percentages of mutation, reproduction and crossover, as well as the population size and number of generations are usually hand tuned by the designer. The first instances of evolutionary robotics, are those of Cliff, Harvey, and Husbands at University of Sussex [38] and Floreano and Mondada at EPFL [31], which both focus on evolving neural circuits for robots with fixed morphology operating in real environments.

Around the early 1990s, Karl Sims first used evolutionary algorithms to evolve both the morphology and an appropriate control system of "virtual creatures" [110], see Figure 1.5. This research was followed by many other studies, all investigating the best way to co-optimize the controller and morphology via evolutionary algorithms. These researchers include Dellaert et al. (1996) [25], Lund et al. (1997) [70], Eggenberger et al. (1997) [28],Pollack et al. (2000 and 2001) [91, 92], Bongard et al. (2001) [12], Ray et al. (2001) [95], and Shen et al. in 2021 [109]. The majority of these studies focused on how best to encode both the morphology and control system into a population suitable for testing and also manipulating for the next generation.

In this thesis I am especially interested in the idea of evolving robot morphology. Therefore, in the following sub-section I discuss specifically the area of research regarding evolution of morphology

However, it should be noted that in many cases it is hard to completely separate evolution of morphology with evolution of control. In the majority of cases a new morphology requires a new control system, and it is not a trivial challenge to co-

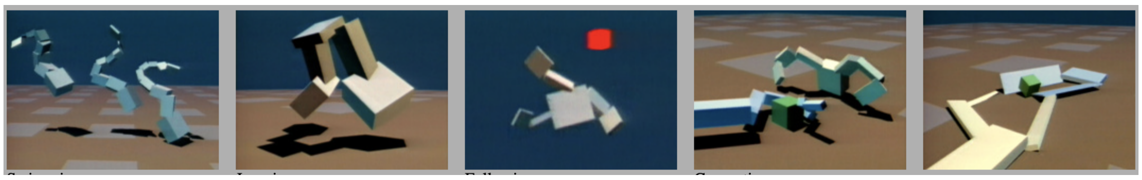


Figure 1.5: Examples of the pioneering work by Karl Sims. Sims used evolutionary algorithms to evolve virtual creatures capable of swimming, jumping, following, competing and collaborating [110]. (Image taken from http://www.karlsims.com accessed 14/04/2021)

optimise both aspects of the robot. In the rest of this literature review, and the rest of the thesis, I place the emphasis on evolution and adaption of morphology, but note that in some cases adaption of control is also required.

1.3 Evolving Robot Morphology

As previously discussed, using evolutionary algorithms to evolve robot morphology is something that has been investigated now for almost three decades. In this subsection I consider the more recent research into the area of just morphology evolution (rather than just evolution of control systems). Although, also as previously discussed, in many cases it is impossible to separate evolution of morphology and control.

Early research into evolving robot morphology focused on evolution of more traditional rigid bodied robots, but as the benefits of using soft robots became more apparent some researchers have also studied how the morphology and control of soft robots can be evolved using evolutionary algorithms, e.g., Hiller/Lipson (2010 and 2011) [43, 45] and Cheney (2010 - 2014) [17, 96].

Traditionally evolution of fixed morphology and control are all carried out in just simulation. However, what differentiates evolutionary robotics from just evolutionary computation is the requirement for embodiment. Therefore, there is a trend in this area to move from just simulation, to sim to real, or in some cases just evolve in the real world. This is discussed in the following subsection.

1.3.1 Sim-to-Real Evolutionary Robotics

One such example of where researchers have evolved in simulation and then transferred these designs to the real world is the work by Kriegman et al. in 2020 [63] where simple voxel based robots are evolved in simulation and then optimal designs are transferred to real life. The real robots are built out of silicone voxels and actuation is achieved by varying the air pressure in the voxels via syringes (see Figure 1.6a).

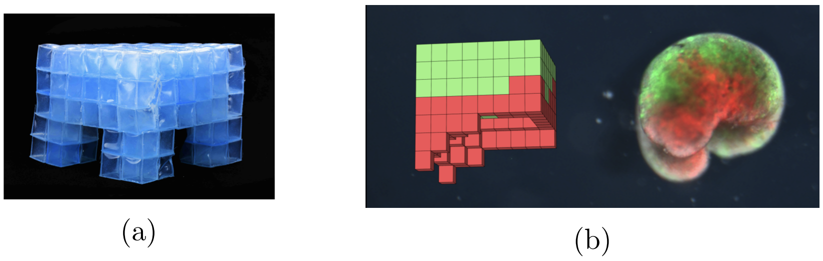


Figure 1.6: a) Example of the silicone robots made by Kriegman et al. The morphologies of the robots are first evolved in simulation and these designs transferred to real life. Image taken from reference [63]. b) Example of biological robot: xenobot. The design has been optimised through simulated evolution and transferred to the biological system. Image copyright Sam Kriegman, accessed from https://cdorgs.github.io

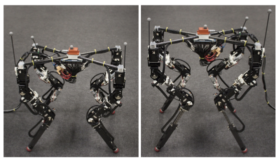
In the same year, Kriegman et al. also introduced a framework for transferring designs evolved in simulation to biological robots (Xenobots: see Figure 1.6b). These Xenobots have been evolved to walk, swim, push pellets, carry payloads, and can survive for weeks without food. They are also able to adjust their morphology to heal themselves after lacerations.

These are two examples of where traditional evolution of morphology in simulation, transfer to real systems has worked well. However, in many cases transfer from simulation to reality is challenging. This is especially true in the cases of evolutionary robots where evolution has the tendency to exploit artifacts in the simulation environment that may or may not be present in the real world.

A solution to this transfer-ability problem is to evolve robots entirely in the real world. Whilst this is usually costly both in terms of time (and if evolving morphology, hardware) there are a couple of examples in literature where this has worked well.

In particular, Nygaard Glette developed a quadruped system where the morphology of each leg is able to be altered, along with the control system [82, 83]. This robot configuration is shown in Figure 1.7a. With this platform, Nygaard et al. were able to explore real world (online) evolution of their robots morphology. They were able to evolve different morphologies depending on battery voltage but also different ground types (e.g. concrete versus grass). Their platform allowed the morphology to be changed for each genotype without completely rebuilding the robot. Whilst the only adaption that took place was through evolution, postnatal development could also be feasible.

Another examples of real world evolution is the ARE project [37, 36]. The main aim of the ARE project is to evolve in the real world both the control and body plans of robots, e.g., see Figure 1.7b. Again in this case the morphological adaption only occurs on an evolution timescale. However, in both the above cases the authors argue the importance of increased interaction with the environment where evolution


(a)

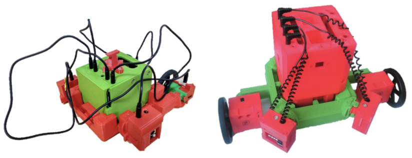

(b)

Figure 1.7: a) Nygaard and Glette developed a robot capable of morphological adaption to different environments. Image taken from reference [83]. b) Hale et al. are investigate how to evolve robot morphology and control in the real world. Image taken from reference [36].

is an online approach.

Another area of recent, particularly relevant, research is how evolution can be used to generate robot morphologies and control systems that are specifically adapted to different environments. They also aim to understand how different types of environment influence the final evolved morphologies. These examples are discussed in the section below.

1.3.2 Evolution in different environments: adaption via evolution

Traditionally, evolution of robot morphology has been carried out in a single environment (flat land). However, more recently researchers have been looking at evolving in multiple environments. This is usually done to observe the effects of the environment on the way these robots evolve and how they adapt to different environment. However, note that in these cases, adaption takes places over many hundreds (or thousands) of generations. Therefore, if the robot was then required to adapt to a new environment, the evolution process would have to be repeated.

One such example of evolving robot morphologies in multiple environments is the work by Auerbach and Bongard in 2012-2014 [5, 4] where they showed that more complex morphologies were produced when evolution took place in complex environment; when compared to evolution in simple environments. Examples of their work are shown in Figure 1.8.

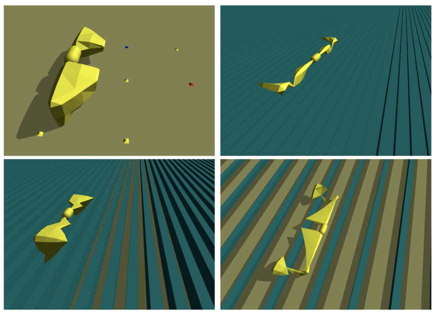


Figure 1.8: An example of the work by Auerbach and Bongard. This is one the first examples of researchers evolving robots for locomotion in different environments. They showed that the more complex the environment the more complex the robot morphology.

This finding was also supported by the work by
Miras et al. [74] later in 2020, where they compared evolution in two “static” environments with one “dynamic”. In the dynamic environment the environment the robot was evolved in changed throughout the robot’s lifetime. This also resulted in more complex robots being produced when evolved in the more complicated dynamic environment. Note however, that in this instance only one morphology per genotype was evolved even in the dynamic environment; this morphology was fixed throughout the robot’s lifetime but had adapted to cope with the changing environments.

A similar concept to Miras’s work was carried out by Corucci et al. in 2017 [22]. In this instance, virtual creatures were evolved separately on land and in water. Also evolved were robots that for the first half of the evolution (4500 generations) experienced one environment, and then for the second half of evolution (4500-9000) experienced another. They observed that the transition from water to land resulted in better performing land individuals than when evolved only on land. However, it is not clear whether these final morphologies are able to operate in both environments; it would appear that they lose the ability to locomote in water after a few generations of land evolution.

In 2018, Nygaard et al. evolved rigid legged robots in real world [83]. These legged robots were also evolved in a damaged condition (i.e., with lower voltage). They found that those robots evolved at the low voltage also performed well at optimal voltages. However, in contrast, the robots evolved at the higher optimal voltage suffered a loss of performance at the lower “damaged” voltage.

The above examples all show that the environment has a strong influence on the way the robots evolve, i.e., in both the control and the morphology of the robot. Whilst there are a number of positives to adapting via evolution there are a number of disadvantages also. First, is that adaption occurs over a large number of generations. Therefore, if the environment for which the robot is required to locomote in changes, the entire evolution process will need to be redone to re-adapt the robot. Nature however, adapts on a number of timescales, specially evolution and over the lifetime of the robot when considering morphology. A key aim of this thesis is to examine how adaption on two timescales (evolution and development) can be used to improve the robustness of robots in a wide range on environments. Therefore, in the next section I discuss examples of robots adapting after birth/initialisation.

1.4 Morphosis and Adaptive Robots

In this subsection, I explore the relevant literature surrounding robots that are able to adapt their morphology to different environments or tasks during the lifetime of the robot (i.e., after evolution).

The simplest case of morphosis for locomotion are robots capable of changing leg stiffness. A number of different variable compliant mechanisms have been proposed for this purpose, e.g., Quy et al. (2011), Tagliamonte et al. (2012), Vu et al. (2015), Li et al. (2019) [93, 114, 118, 66]. These examples attempt to mimic the behavior observed in humans and other animals, where they adapt their leg stiffness depending on ground conditions in order to locomote in an energy efficient manner.

Vu et al. [119] also built on the concept of variable compliant mechanisms by developing a robot platform to explore the idea of changing gait to enable robust locomotion. They use a crank-slider mechanism in the leg that translates the simple control signal (i.e., constant rotational velocity) into leg trajectories. This mechanism changes the way the rotational movement is translated, while the control remained unchanged. The result was a range of different end point trajectories that can be useful for different terrains.

There are a number of example of robots which aim to mimic that adaptive growth of plants. Whilst this is not locomotion in traditional sense it is still useful and relevant to this thesis. Using soft materials and additive manufacturing techniques these robots are able to navigate obstacles passively e.g., Sadeghi et al. (2017-2020) [100, 101], Greer et al (2020) [35] and Del et al. (2019) [24]. However, these examples do have the disadvantage in that, unless they have an infinite of resource to add to their bodies, or way of recycling the material, they will remain anchored to the starting location.

In 2015, Cacucciolo et al. [14] designed an adaptive simple robot with four soft, pneumatically actuated legs and a rigid backbone. They showed that by adjusting the angle of the legs and the pressure within the legs the robot was able to successfully locomote in two different environments. However, in these instances the morphological parameters were hand tuned in order to achieve this adaptive locomotion.

Modular robots have been also considered as a potential way to create robots capable of morphosis by moving identical modules relative to each other, e.g., consider work by Murata et al. (2002-2007) [78, 79], and Marbach et al. (2005)[71] and later in 2017 Veenstra et al. [116]. There are even some cases of researchers using evolutionary algorithms to evolve optimal methods of reconfiguration to adapt to new environments, e.g. Marbach et al. (2005), Yoshida et al. (2003), Klidbary et al. (2013) and Alattas et al. (2019) [71, 122, 57, 2, 69]. However, whilst in these examples the overall robot has the ability to change its morphology, they do not have the capability to determine when to adapt – this is still done by the robot designer. In this sense the robot does not develop, instead when the human designer puts the robot into a new environment the human reruns the evolutionary algorithm to determine the best new morphology, the new control system and then the way that that the individuals modules of the robot should move to realise this new morphology. In most cases the modules of the robot have no sensor – if they do, the results from the sensors are fed into an adaptive control system rather than to aid the development of the morphology to improve robustness.

Another area of morphosis or adaptive robotics is origami robots, e.g., Miyashita et al. (2017) [75], Kotikian et al. (2019) [59], Zhakypov et al. (2018) [124] and Taghavi et al. (2018) [113]. These robots aim to provide the same adaptability that can be gained from the morphology changes from the modular re-configuring without the need for many bulky electronics, which is generally seen as one of the main disadvantages of the self re-configuring robots. These robots are generally formed from a flat sheet of material which is activated, e.g. by magnetic fields or hinge mechanisms, to form differently shaped "exoskeletons." However, once again these do not necessarily respond to feedback from the environment in order to change shape. Also in these instances there are only a certain number of final morphologies able to be generated from the initial sheet and these are predetermined by the designer.

An interesting study, carried out by Corucci et al. [21] investigated using novelty as a fitness function to evolve morphing behaviour in the soft octopus robot the PoseiDRONE (see figure 1.9). The evolutionary algorithm determined optimal morphologies from which the robot could minimally change between to gain a maximum change in behaviour. Again this was completed without any real feedback from the environment – instead it was still up to the robot designer as to when the morphology should be adapted.

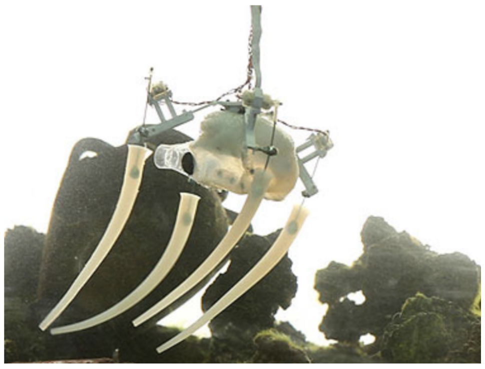


Figure 1.9: Corucci et al. used an evolutionary algorithm to determine optimal morphologies from which the robot could minimally change between to gain a maximum change in behaviour, this was implemented on the PoseiDRONE robot pictured here. Image taken from www.ideaconnection.com accessed 14/04/2021.

Another relevant study [52] considers how terminating embryonic development at different stages can allow for robust and adaptive behaviour suited to different environments. For example, a single genotype can be transformed into two different phenotypes just depending on for how long the genotype is allowed to develop. Furthermore, after "birth" these phenotypes have the ability to morph between the two morphologies. However, this study has the same problem as before, e.g., in [21]. The robot has no environmental sensing, it does not know when and how to change its body shape.

1.4.1 Morphological Adaption for Damage Recovery

The above examples mostly relate to research where the authors have adapted morphology in order to perform in a different environment or carrying our a different tasks. There is also an area of research that considers morphological adaption for damage recovery. Traditionally, researchers have used adaptive control methods to recover from even morphological damage, for example the work by Cully et al. in 2015 "Robots that can adapt like animals" [23]. However, with increased realisation of the importance of embodiment, some researchers have explored how adaptive morphology may also help in damage recovery.

One example of this is Kriegman et al. in 2019 [64] where they evolved soft voxel based robots who were shape to change the size of remaining voxels when others were removed. Additionally, to some extent this was replicated in the real world. Later, in 2020 Kreigman et al. [63] also designed biological robots capable of morphological regeneration with sliced (however in this instance the amount of damage is relatively low).

It has already been discussed in the previous section, that strong interaction between the environment and the robot is required for better performing adaptive robots. Whilst the above examples do show morphological adaption, in the majority of cases the adaption is instantaneous but also limited to only a few body types. Instead, I argue that combining evolution with development and allowing the robot to alter its morphology over its entire lifetime as a response to interaction with its environment would result in even more adaption.

There are some limited examples of where researchers have explored combing evolution and development of morphology; these are discussed in the next section.

1.5 Evo-Devo: Combined evolution and development of morphology

In the majority of the examples in the above section, adaption of robot morphology after initialisation or "birth are limited to one or two different body structures. These body structures are often hand designed and the trigger for morphological change is usually by a human. This therefore limits the amount of adaption that the robot can achieve.

Considering the example of the natural biological plants, adaption takes the form of morphological development and is driven by interaction with the environment. The plant grows towards a light source, if the light source moves, the plant changes the way it develops. This adaption is not instantaneous; adaption develops over the course of the plants lifetime.

There have been a number of examples of research were adaption takes place on two timescales; i.e., evolution and development.

More commonplace is the development of control systems. There are many examples where researchers have evolved starting neural networks to act as the control system for a robot. Then, the weights of the neural network over the lifetime of the robot are altered; enabling adaption through development. Examples of these are Najarro Risi [80], Oudeyer et al. [84], and Stanley et al. [111].

However, as previously discussed, there are comparably few examples of morphological development. Additionally, in many cases this development is used as a tool to evolve better performing final robots, rather than used for further postnatal adaption.

Bongard [9] designed robots capable of growing from an anguilliform into legged robots during their lifetime in early evolution runs. However, in later generations this postnatal development was removed and only legged robots were tested in the environment. Bongard found that allowing development in early evolution stages evolved better final robots.

Another such example of this is the work is by Kriegman et al. [62], where they showed the benefits of using even a small amount of morphological development, coupled with evolution, to create better performing voxel-based robots. Initially their approach did not consider how the environment could influence growth; but touched on the idea a year later [61]. In this later work, the stiffness of each voxel was changed depending on feedback from its interaction with the environment and this was shown to increase the robustness of the evolved robots.

Similarly, Corucci et al. [22], inspired by the adaptive nature of plants, investigated how individual voxels could alter their respective size to alter the overall virtual creatures morphology based on feedback from an artifical light source.

Therefore, it can be seen that morphological development (especially that driven by environmental feedback) of robots is something currently under researched. It is the aim of this thesis to expand upon this idea in the context of locomotion.

1.6 Thesis Context and Structure

From the literature in this chapter, it can be seen that using artificial evolution to evolve the way a robot develops and adapts its morphology based on its interaction with environment has not yet been researched. In this section, I discuss how my thesis relates with respect to the wider research field. Later in this section, I also present the structure of this thesis.

It can be seen, from the "Morphological Computation" section, Section 1.1, that there is benefit in designing soft compliant robots, capable of locomotion, that are able to exploit the way their morphology interacts with their environment. This exploitation has been shown to reduce some of the necessary computational expense required by the controller. However, designing suitable morphologies, especially when utilising soft bodies, is a challenge in itself.

Artificial evolution has already proved itself a powerful tool to design complex and interesting (albeit fixed) morphologies. However, previously it has been mostly used to produce fixed morphologies via evolutionary algorithms. Therefore these robots are only able to adapt on one timescale; over hundreds or thousands of generations. Since in these examples the robot morphology is fixed from initialisation, the amount of adaption they can achieve is limited. Whilst there are many example of robots that are able to alter their shape after initialisation, the amount of adaption is usually limited to two or three body shapes which have been pre-designed by a human.

There are also numerous examples of evolving developing control systems, some of which are based on feedback from the environment. However, evolutionary algorithms have only just begun to be used for evolving the postnatal adaption of robot morphology.

In this thesis, I present my research which combines the areas of Morphosis and Evolutionary Robotics. Using evolutionary algorithms, I evolve the way a robot should adapt/develop its morphology, based on feedback from the environment, in order to change itself into successful morphologies for a wide range of environments. The adaption takes the form of development, and the robot's morphology is gradually changed based on its integration with the environment. Therefore, the main hypothesis of this thesis is that evolutionary algorithms can be used to evolve optimal methods of morphological adaption/development in compliant robots allowing for more robust locomotion in a wide range of environment.

Furthermore, the key aims and objectives of this thesis are:

- To investigate the use of evolutionary algorithms to evolve methods of adapting the morphology of compliant robots in order to achieve robust locomotion.
- To carry out the investigation of evolving methods of adaption in two contexts. Firstly, using the theoretical Spring Loaded Inverted Pendulum model as a "base robot". Secondly, using a more complex voxel-based robot as a platform for adaption.
- To determine how transferable, and therefore robust, the evolved methods of adaption are in terms of performance in different starting environments and different starting morpologies.

Note that in this thesis, I concentrate on adaption/development of the robots morphology rather than control. A strong motivator for this thesis is to explore to what extent just adapting the morphology of a robot can improve its robustness in the context of locomotion. Whilst I do not anticipate that in the future, robots will require no control system, I aim to provide evidence as to the power of adaptive morphology. Furthermore, in many aspects of machine learning there is the theory of “no free lunch. In this thesis I do not present a single global algorithm capable of adaption in any possible scenario. However, I do aim to show that through evolution it is possible to generate robots capable of developing their morphology in order to be successful in a much wider range of environments than if they had a fixed body structure.

In the second and third chapters (i.e., the two chapters following this one), I consider methods of adaption of the both the morphology and control of the Spring Loaded Inverted Pendulum (SLIP model), which is a commonly used model to describe a wide range of animal locomotion, see chapter two literature relating to the SLIP model. One of the reasons the SLIP model is so widely used is that if the morphological and control parameters that make up the model have the correct values the model is self stabilizing. That is it will exhibit a stable locomotion pattern that is maintained even when the system experiences small perturbations (however, larger disturbances will make the model fall over). In chapter 2, I conceive a number of adaption rules that, when combined, uniquely describe how the SLIP model should adapt a single part of its self in order to find stability for a wide range of starting parameters. Here, I use an offline learning approach where the model allowed to fail and its parameters are changed upon reattempt; therefore, the rules are based on how the model improves between locomotion attempts. The rule sets are systematically tested to determine the optimal. In this chapter and optimal rule set is one that is able to update the widest range of initial morphology and control parameters so that, after a number of locomotion attempts the SLIP model exhibits stable locomotion.

The third chapter has two main parts. Firstly, I further investigate this offline learning approach by using evolutionary algorithms to evolve optimal rule sets that update both the morphological and control parameters simultaneously. In the second part of this chapter, I investigate how these rules can be changed to allow the SLIP model to adapt, to find stability, without the need to failure. Instead of being based on previous locomotion attempts, these “online” rules are now based on changes in energy between strides. As with the second chapter, the optimal rule sets are those which are able to adapt the widest range of initial morphological and control parameters into those capable of stable locomotion in the SLIP model. Additionally in this chapter, I explore how these new online rules allow the SLIP model to maintain stability in the face of changing landscapes, i.e., downwards steps.

In Chapter 4, the complexity of the simulated robot is increased and specifically, I consider using evolutionary algorithms to evolve methods of adapting a voxel based robot. The overall initial voxel based robot is formed of 216 individual modules, or voxels, arranged into a 6x6x6 cube. The internal voxels have the ability to expand and contract based on an external sinusoidal signal – this gives the robot the ability to locomote. However, in the initial morphological configuration, the robot is only able to locomote a very small distance in all three different environments for which it is tested in. As with Chapter 2, morphological adaption of the robot takes between a number of locomotion attempts. Whereas in the previous chapters, the adaption methodology is based on rule sets, in this chapter after each locomotion attempt a single neural network determines how the robot should adapt. More specifically, the neural network is based on the kinetic energy of the voxel, determines how much the stiffness of that voxel should change. If the stiffness of a voxel becomes too low it removed; thus an optimal morphology, adapted to the environment that the robot is in, is sculpted. In this chapter, it is the weights of the neural network that are evolved; an optimal neural network is one that is able to sculpt a final morphology capable of travelling the further distance in all three environments for which it is tested for. Note that the internal expanding and contracting voxels are not removed; thus only the morphology of the robot is adapted, not the control. Also included this fourth chapter is an investigation into how the evolved neural networks transfer into environments for which they were not initially evolved. I also investigate how these evolved neural networks can be used as a way of adapting the morphology of the robot to counteract morphological damage.

The final chapter contains a summary of the work presented in this thesis and discusses the arisen implications of the research. It also includes a section on potential further work. Note that the work carried out in this thesis is entirely in simulation, therefore the final chapter also contains a discussion as to how this work could be transferred to physical robots.

1.7 Publication List

1.7.1 Journal Articles

Walker, Kathryn, and Helmut Hauser. "Evolving optimal learning strategies for robust locomotion in the spring-loaded inverted pendulum model." International Journal of Advanced Robotic Systems 16.6 (2019): 1729881419885701.

Walker, Kathryn, and Helmut Hauser. "Adapting stiffness and attack angle through trial and error to increase self-stability in locomotion." Journal of Biomechanics 87 (2019): 28-36.

1.7.2 Conferences

Walker, Kathryn, and Helmut Hauser. "Evolution of Online Update Rules for Robust Locomotion in the SLIP Model." 9th International Symposium on Adaptive Motion of Animals and Machines (AMAM 2019). No. CONF. 2019.

2 Adaption of the spring loaded inverted pendulum - an offline learning approach

This chapter is based on the publication **Walker, K. and Hauser, H., 2019. Adapting stiffness and attack angle through trial and error to increase self-stability in locomotion. Journal of Biomechanics, 87, pp.28-36.**

2.1 Introduction

In this chapter and the next, optimal rule sets used to adapt a simulated "Spring Loaded Inverted Pendulum" (SLIP) model are investigated. The SLIP model is a prevalent model for analyzing the running and hopping leg motion of a wide range of species [7, 73]. The model consists of two main components, a point mass and a linear spring connected to it. It is due to its prevalence for modelling legged animal locomotion and its simplicity that the SLIP model was chosen as an initial platform to explore the evolution of optimal adaption methods.

In more detail, in its basic form, the SLIP model describes the motion of an animal by representing its body as a lossless, linear spring with constant stiffness k and constant rest length l_o as shown in Figure 1, the center of mass of the body is represented by a point mass.

Traditionally, the movement of the model is formed of two phases; the flight phase and the stance phase [105], as shown in Figure 2.1. During the flight phase the point mass follows a simple ballistic trajectory until the foot of the leg "touches down" on the floor. At this point the leg enters the stance phase; whilst the top of the leg, the point mass, continues to move with the same horizontal velocity, the rest of the spring first compresses and then expands; when the spring length returns to l_o the leg "takes off" and re-enters the flight phase. The angle at which the leg enters the stance phase is called the attack angle (α).

At particular combinations of attack angle (α) and spring stiffness (k) the SLIP model is stable. It is able to transfer from the flight phase to stance phase and back again indefinitely without the point mass coming into contact with the ground (i.e., without the SLIP model falling over). Additionally, at these combinations the SLIP model is able to withstand small perturbations, e.g. those introduced by small changes to ground level. However, if the perturbation is too large the model will fail. These combinations of attack angle (α) and spring stiffness (k) have been investigated by Seyfarth et al. (2002) [104] and are commonly referred to as the J-Figure [33] due to the shape of the region of stable combinations. An example of the J-Figure is shown in 2.2.

There is a wide range of literature relating to the improvement of the SLIP model, to widen the stable "J" region and therefore make the model more robust to envi-

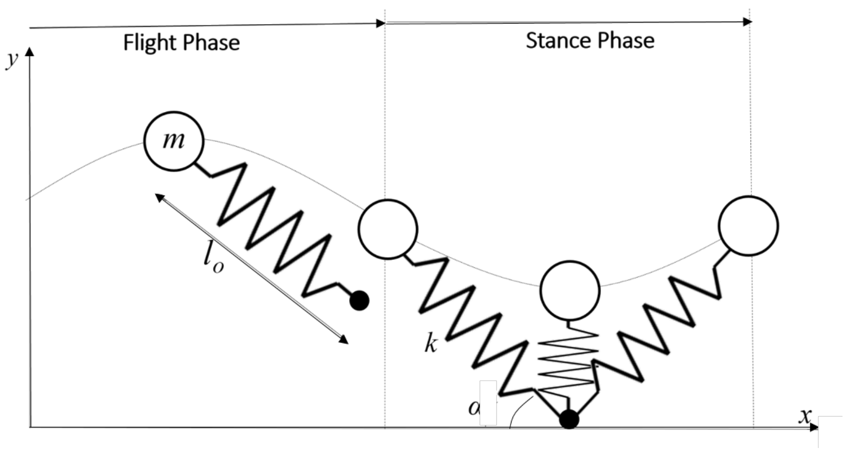


Figure 2.1: Figure showing the SLIP model in its basic form. The basic SLIP model consists of two main parts; a point mass (m) on top of a linear spring with constant, linear stiffness (k). Note the spring itself has no mass and is lossless, i.e., it is not damped. During locomotion the SLIP model has two phases, a flight phase and a stance phase. During the flight phase the point mass follows ballistic trajectory and the spring remains at a constant uncompressed length until the other end of the spring comes into contact with the ground, termed “touchdown.” At this point the “foot” end of the spring remains fixed. The spring first compresses and then starts to re-expand. When the spring length reaches the initial uncompressed length again the SLIP model re-enters the flight phase, i.e., it “lifts off.” The angle at which the SLIP model touches down is termed the attack angle (α).

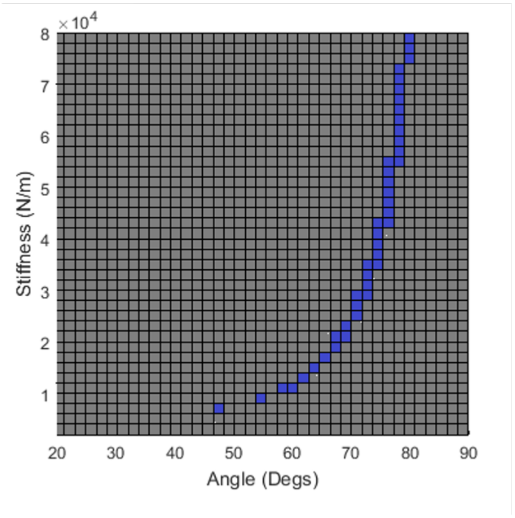


Figure 2.2: Figure showing the stability regions of a SLIP model. In this case the SLIP model has a leg length of 1m and a mass of 80kg. The SLIP model was tested for stability at 1600 different starting combinations of attack angle and stiffness. If the SLIP model is stable at these starting parameters it is shown in blue. From this the “J-Figure” as determined originally by [104] can clearly be seen. Note that if different masses and leg length were used the values for the stable combinations would be also be different but a J shape would still be observed.

ronmental changes.

One approach has been to extend or alter the design of the SLIP model in order to exploit the soft compliant morphology, i.e., the spring section. Note that in these cases the morphology of the extended SLIP model stays constant throughout the locomotion and any improvement in performance in robustness is simply down the extension in morphology. For example, Rummel et al. [99, 98] showed that an addition of an extra leg segment leads to an overall nonlinear stiffness function and as a results bigger region of self-stabilization. Karssen et al. in 2011 [56] optimized a non-linear stiffness profile function; optimizing the gait sensitivity norm (a measure for disturbance rejection, described in their paper.) Okwaki et al. in 2007 also carried out a similar approach [85], as did Yu et al. [123]. Other modifiers to the SLIP model, such as the inclusion of leg damping or hip torque have also been shown to improve the stability, e.g., the work from Shen et al. in 2021, Blickhan et al. in 2015 and Abraham et al. also in 2015. [109, 8, 1]. Additionally, Jun et al. (2011) [54] considered how curved legs affect the stability of SLIP based locomotion. As stated above, these examples modify the force profile of the SLIP model during the stance phase, either by introducing non-linearity of the spring or introducing damping, in order to improve robustness. These examples do not allow the morphology, or control (i.e., α), of the SLIP model to change either between strides or between locomotion attempts.

Some researchers have investigated changing the morphology from stride to stride. For example, Andrews et al. (2011) [3], Schmitt et al. (2009) [102] and Shen et al. (2014) [108] implement systems where the morphology, in this case the leg length, is varied from stride to stride (between two fixed leg lengths) to maintain stability over rough ground. However, in these cases the amount the the leg length is varied is irrelevant of any feedback from the environment.

Another method used to increase robustness is the addition of a control system which adapts the attack angle of the leg upon touchdown. For example, consider the works of Ghigliazza et al. (2005) [34], Hurst et al. (2004) [48], and Peuker et al. (2021) [87]. Whereas in the cases of morphological adaption (i.e., those described in the previous paragraphs) the amount that the attack angle is varied **is** dependant on feedback from the environment (albeit indirectly). Researchers have based their attack angle alterations on a variety of elements of the SLIP model, e.g. velocity (Sharbafi et. al (2016-2017) [107, 106] or force (Englsberger et al. in 2015) [30]. Note, however, that even though these angle changing systems do take into account the environmental changes experienced by the SLIP model (through velocity), the rules dictating how the angle is changed are hand designed; they have not been optimised.

In general, there is a vast amount of research into how to expand the robustness of the SLIP model. However, in the case of adapting the morphology, the systems do not take into account any feedback from the environment. Furthermore, in both the cases of adaptive control and adaptive morphology, none of the examples from the literature listed above use past experience, be it success or failure, to determine how best to adapt. Therefore, they all have one common flaw. If the SLIP model starts with a combination of parameters (stiffness k, and attack angle α) outside the range of stable parameter combinations, it has no way to learn to become stable. In this chapter, I present a learning methodology that adapts the attack angle (α) and spring stiffness (k) of the basic SLIP model based on a set of rules. This learning approach is offline; it uses locomotion attempts (named episodes) and allows the model to fail. The relevant parameters (be that spring stiffness or attack angle) are updated between such episodes. After each episode the distance travelled in that episode (D_i) is compared to the previous distance (D_{i-1}). This change in distance (ΔD) is the key measurement in each of the rule sets. Each rule set uses ΔD in a unique way to determine how to change either attack angle (α) and spring stiffness (k). Thus, an optimal rule set would, over a number of episodes, adapt either the control or the morphology parameter, from an unstable to stable one; i.e., the SLIP model would learn to become stable. Figure 2.3 shows the concept of episodes.

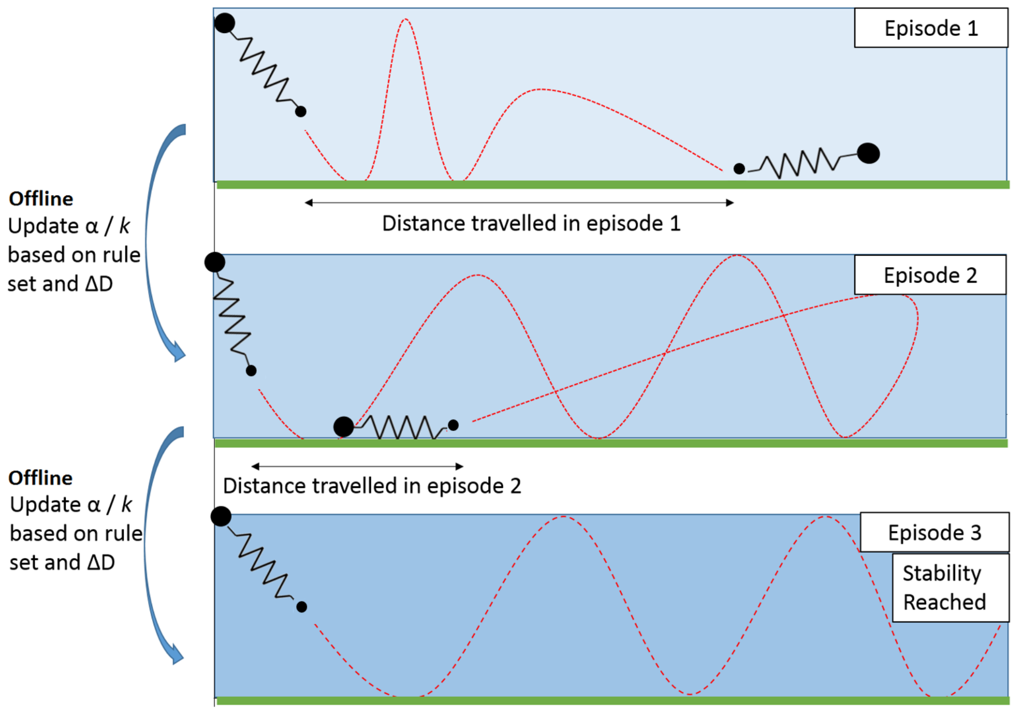


Figure 2.3: Figure showing how the attack angle (α) and spring stiffness (k) of a SLIP model are updated using the offline rule approach. Each rule set is based on the change in distance between each locomotion attempt (or episode) and dictates how either the control (α) or morphology (k) should change for the beginning of the next episode.

2.2 Methodology

2.2.1 SLIP Simulation

As previously discussed the SLIP model was first presented in 1989 by Blickhan [7]. It is a simple, generic model of legged locomotion, commonly used to model hopping or running. In the case of my research I follow the simulation layout from the 2003 work by Seyfarth et al. [105].

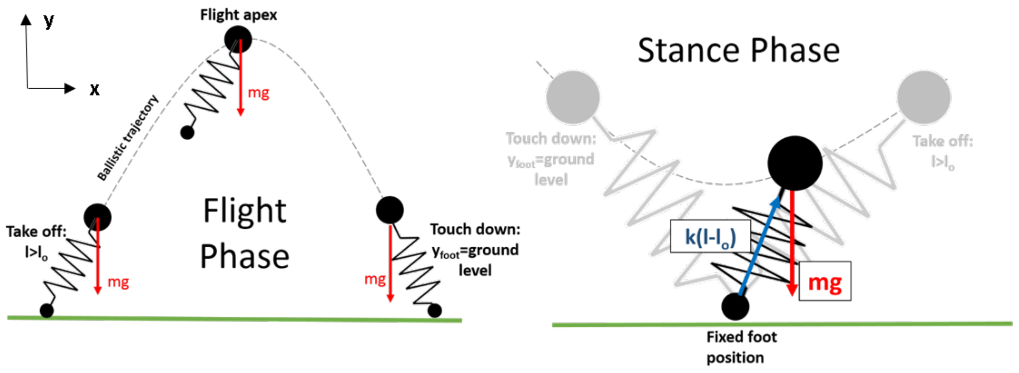


Figure 2.4: Figure shows the SLIP during the two phases, stance and flight, in more detail. During the flight phase the only force acting gravity g, the point mass follows a ballistic trajectory and spring remains at l_o. When the bottom of the spring hits the ground (i.e., the position of the foot is equal to the ground level) the spring starts to compress. This results in an additional spring force which provides an upwards force $k(l - l_o)$; when the spring length is at l_o once more take off occurs.

During the flight phase the leg is only represented by the point mass, since the spring is considered to be mass-less. This point mass follows a ballistic trajectory until the end of the spring (the foot) comes into contact with the ground at which point the leg enters the stance phase (this point is referred to as touch down). During the stance phase the foot of the spring is assumed to be anchored to the ground, the point mass continue to move in the direction of travel but its exact trajectory is no longer ballistic as a result of the anchored spring. When the length of spring returns to the resting length the stance phase ends and the flight phase begins once again (termed the takeoff point).

The behaviour of the SLIP model can be characterised mathematically with a few simple equations. During both phases the point mass follows the equations of motion shown in Equation 1.

$$\ddot{r} = F_l + mg, \tag{1}$$

where $\ddot{r} = [\ddot{x}, \ddot{y}]^T$, i.e. the acceleration of the point mass (in 2D), $g = [0, -9.81]^T$, the gravitational acceleration acting on the point mass, and F_l is the spring force.

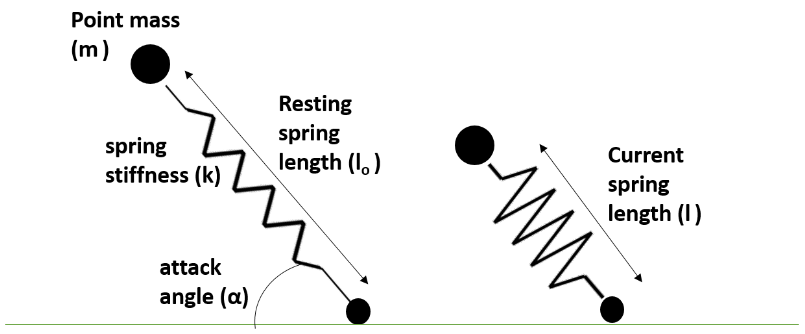


Figure 2.5: Figure shows the basic SLIP model and the parameters required to calculate motion. Initially the model is shown just at touch down, the attack angle (α), spring stiffness (k), point mass (m) and resting length (l_o) are labelled. In the second diagram the spring has compressed slightly and the current spring length is shown (l).

If the model is in flight phase $F_l = 0$. If the model; is in the stance phase the spring force is:

$$F_l = k(l_o - l). \tag{2}$$

Here, l_o is the resting length of the spring, as shown in Figure 2.5. Additionally, Figure 2.4 and Figure 2.5 show each of the parameters used in the equations. During the stance phase, first the spring force is calculated. This is then resolved into x and y using cosine and sine respectively. The angle that the spring makes with the ground during the stance phase is always measure from the same direction, therefore when the angle increases beyond 90 degrees and the spring is orientated the other way, the resultant force in the x direction changes sign. In the simulations, during the flight phase the spring is always angled against the ground surface by α, the attack angle, i.e., any retractions of the spring are not modelled. This means that the transitions between the two phases are simply triggered by the following rules:

- Entering Stance : y- $l_0$sinα $<$ ground level
- Entering Flight : Length of spring $> l_0$

The system was simulated in Matlab using the Forward Euler method for integration with a time step of $\Delta t = 0.001$ seconds. In each episode the system was simulated until either the spring fell over (the vertical height of the point mass was less than ground level, i.e. $y < 0$) or the maximum time of $t = 10$ seconds ($= 10'000$ simulation time steps) was achieved. If the spring had not fallen over after 10 seconds it was considered stable. An episode was defined as the time between when the leg begun to move and when it falls over (or the simulation reaches 10 seconds). Therefore, the maximum length of one episode was 10 seconds. It is worth noting that although that potentially there are parameter combinations of attack angle and stiffness that may lead to the leg falling over after 10 seconds. However, after extensive testing this situation never occurred and as a result this maximum episode length was a reasonable trade-off between simulation time and stability.

As a reminder, running the model for multiple episodes, and to update its parameters between episodes, allows the model to learn from previous experience. This means it can learn to become stable by updating its "unstable" parameters to stable ones. In the research presented here the simulations were allowed to run for up to a maximum of 100 episodes. If stability was found earlier the simulation is terminated. The number of episodes required is termed a "lifetime" and, therefore, these simulations have a maximum lifetime of 100 episodes. As with the choice of maximum episode length, again it should be noted that potentially there may be an adaption method that finds stability after this $100th$ episode (for example in episode 105). However the maximum lifetime length of 100 episodes was selected after extensive initial testing which showed that whilst potentially possible, the above case never actually occurred.

For the simulations the resting length of the leg was chosen to be 1 m and the point mass 80 kg. This is consistent with the values used in literature, see [105], and are representative of an average adult human male.

2.2.2 Adaption Methods

Next, I introduce the adaption methods that the leg (the SLIP model) uses to alter its morphology (spring stiffness, k) and control (attack angle, α).

The adaption method used in this research is termed a "rule set". Each rule set is made up of five individual rules that all dictate a part of the leg adaption. When the rules are combined, the rule set uniquely determines how the SLIP model will alter its parameters. When following a rule set, the SLIP model will either change the attack angle α, the stiffness k, or both. As discussed this adaption takes place after each episode, so that a new configuration of attack angle and stiffness is used in the next episode.

The rule sets are configured so that either just the attack angle, just the stiffness or both can be tested. The basic rule sets for angle change and stiffness change are shown in Figure 2.8. Initially separating the adaption of the morphological and control parameters allows a comparison of the two approaches. These rule sets are shown in Figures 2.8 and 2.7 (two separate figures shown for clarity).

A key aim of my research was to enable to SLIP model to learn from previous locomotion attempts. Therefore, *distance travelled,* D_i was used as a measure of how successful the previous episode had been, i.e. if the leg had managed to locomote a long distance before falling over, this was deemed to be a more successful episode than if the leg had fallen over almost instantaneously. Note that for a completely successful episode (i.e., where the model had not fallen over at all) the achieved

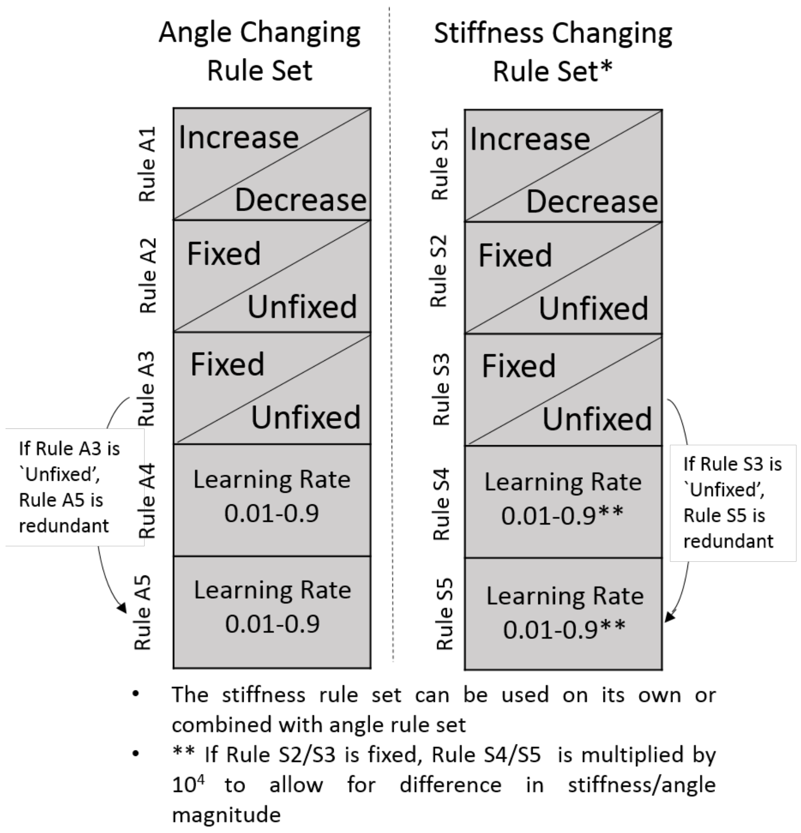

Angle Changing Rule Set
Stiffness Changing Rule Set*
Rule A1
Increase
Decrease
Rule A2
Fixed
Unfixed
Rule A3
Fixed
Unfixed
Rule A4
Learning Rate 0.01-0.9
Rule A5
Learning Rate 0.01-0.9
If Rule A3 is `Unfixed', Rule A5 is redundant
Rule S1
Increase
Decrease
Rule S2
Fixed
Unfixed
Rule S3
Fixed
Unfixed
Rule S4
Learning Rate 0.01-0.9**
Rule S5
Learning Rate 0.01-0.9**
If Rule S3 is `Unfixed', Rule S5 is redundant
• The stiffness rule set can be used on its own or combined with angle rule set
• ** If Rule S2/S3 is fixed, Rule S4/S5 is multiplied by 10^4 to allow for difference in stiffness/angle magnitude

Figure 2.6: Figure showing two basic rule set, one for adapting only the attack angle (α) and one for adapting only stiffness (k). These two basic rule sets can then be also combined to form a rule set capable of adapting both the attack angle and the stiffness simultaneously. Each rule set consists of five different rules. In this figure the possible values for each rule are shown. Refer to the text below for a detailed description of each rule.

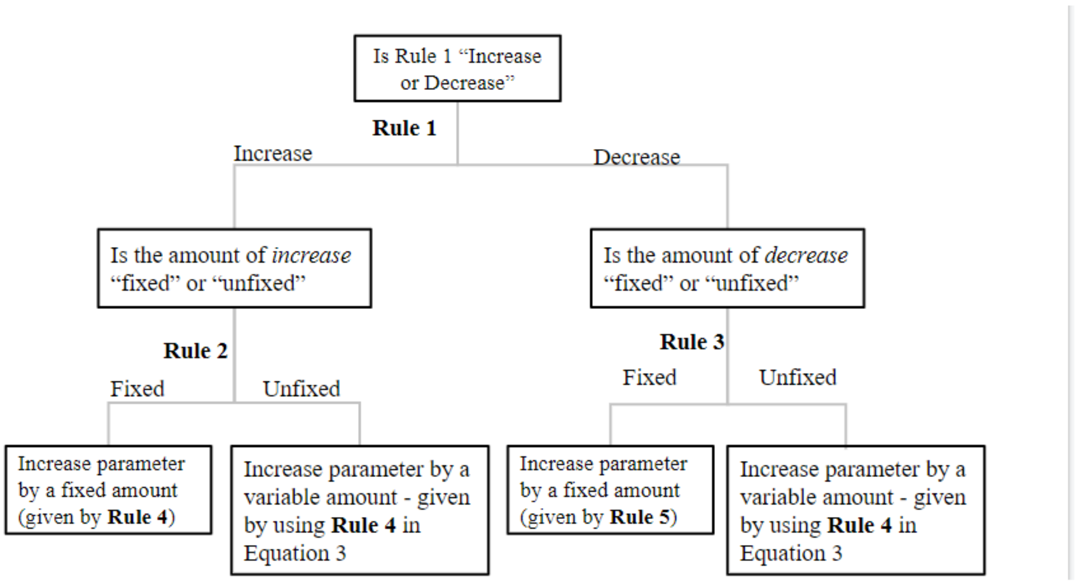
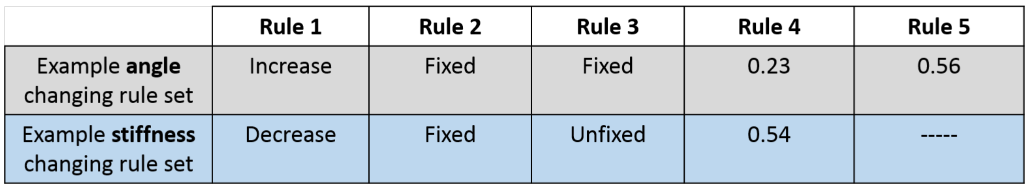

Is Rule 1 "Increase or Decrease"
Rule 1
Increase
Decrease
Is the amount of *increase* "fixed" or "unfixed"
Is the amount of *decrease* "fixed" or "unfixed"
Rule 2
Fixed
Unfixed
Rule 3
Fixed
Unfixed
Increase parameter by a fixed amount (given by **Rule 4**)
Increase parameter by a variable amount - given by using **Rule 4** in Equation 3
Increase parameter by a fixed amount (given by **Rule 5**)
Increase parameter by a variable amount - given by using **Rule 4** in Equation 3

Figure 2.7: Figure showing an alternative view of how each of the individual rules are combined to create an adaption method. It is this final rule set that is evolved via evolutionary algorithm.



	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5
Example **angle** changing rule set	Increase	Fixed	Fixed	0.23	0.56
Example **stiffness** changing rule set	Decrease	Fixed	Unfixed	0.54	-----

Figure 2.8: Figure showing two example offline angle changing rule sets. In the first example, if the difference in distance between the current and previous episode (ΔD) is positive, the attack angle is increased (rule 1) by the fixed amount (rule 2) of 0.23 rads (rule 4). If ΔD is negative the attack angle is decreased (rule 1) by the fixed amount (rule 3) of 0.56 rads (rule 5). In the second rule set if ΔD is positive the spring stiffness is decreased (rule 1) by the fixed amount of 5.4 kN/m (rule 4). If ΔD is negative the stiffness is increased (rule 1) by $\frac{\mu}{2D+1}$ where $\mu = 0.54$ as per rule 4. Note that in the second example rule set rule 5 is redundant as rule 3 is *unfixed*.

distance is the amount the SLIP model has traveled after 10 s. For an unsuccessful run it is the distance traveled before the model has fallen over.

The distance travelled in an episode is the key parameter used in the rule sets. In particular, the rules use the difference between in distance between the episode just finished and the one before that i.e., $\Delta D = D_i - D_{i-1}$. With this knowledge the individual rules can be detailed further, note that these rules can be used to adapt their the attack angle or the spring stiffness:

- Rule 1 dictates if ΔD is positive, the parameter (either α or k) should increase or decrease. This is symmetrical, so that if ΔD is negative the opposite occurs. To clarify further if rule 1 is set to "increase" and ΔD is positive either α or k increases, whereas if ΔD is negative the parameters decrease.
- Rule 2 dictates when the parameter is to *increase*, this increase should be dependent on ΔD or a fixed value, i.e. "fixed" or "unfixed".
- Rule 3 is similar to rule 2 and dictates whether, when the parameter is to *decrease*, this amount should be "fixed" or "unfixed".

If either Rule 2 or Rule 3 is *unfixed* an equation is required to calculate by how much either the attack angle or stiffness should change.

$$\text{amount of adaption} = \frac{\mu}{2 + \Delta D}, \tag{3}$$

where $\Delta D = D_i - D_{i-1}$ and μ is given by rule 4.

- Rule 4 has two possible functions. If Rule 2 is *fixed*, then Rule 4 determines by how much either attack angle or stiffness should increase. The second possible function is that if either Rule 2 or Rule 3 is set to *unfixed*, Rule 4 dictates the value of μ.
- Rule 5 is only used if Rule 3 is set to *fixed*. In this case Rule 5 is the amount that either α or k should be decreased by. If Rule 3 *unfixed* then Rule 5 is redundant as μ is dictated by Rule 4.

In addition, if the parameter is changed, according to the rules, to a value that is outside of the range ($< 0°$, or $> 90°$, for α, or a negative stiffness) the learning process is terminated (i.e., the lifetime has ended) and the rule set is determined unsuitable.

These five rules can be used to make up a wide variety of different sets, i.e., different methods the SLIP model can use to adapt with the aim of achieving stability.

Therefore, using these basic rules to build rule sets, a large number of potential adaption methods were able to be tested to determine the optimal. Considering that Rules 1, 2, and 3 are binary (i.e., either fixed/unfixed or increase/decrease) and Rule 4 and Rule 5 are set to be values from 0.01 to 0.9 in 10 discretitized steps, in total $2^3 * 10 * 10 = 800$ angle adapting methods and $2^3 * 10 * 10 = 800$ stiffness adapting methods were created. This enabled me to systematically test them all. However, it should be noted that the actual number of *unique* rule sets is 400, for each parameter. This reduced number accounts for the redundant Rule 5.

A metric of determining how successful a rule set was in terms of enabling the SLIP model to achieve robust locomotion was required. Therefore, the metric selected to determine the success of a adaption method (a rule set) was the proportion of tested starting parameter combination from which the rule set was able to find stability: the percentage success rate. To clarify, each rule set was tested 1,600 times, each time with a different starting combination of attack angle and stiffness. Within the selected testing range, the perfect rule set would be able to find stability, after a maximum of 100 episodes, for 100% of the tested starting situation. This hypothetical rule set would be described as having a success rate of 100%. In contrast a completely useless rule set would never be able to find stability regardless of starting parameters: it would have a success rate of 0%. In actuality, due to the set up of the experiments, this would never occur as in some cases the experiment would start with a parameter combination within the existing self stability region (i.e., the J-figure) for the SLIP model and, therefore, the experiment would be terminated after one episode. Using the metric of *success rate* allows a comparative measurement on the usefulness of any given set of learning rule.

The range of starting parameters(i.e., the ranges of attack angle α and spring stiffness k) were selected so that it fully encompassed the self stabilising area, but also had enough additional parameter combinations so that the validity as well the limitations of my approach could be full demonstrated.This area of stability is in literature often referred as J-Figure (e.g., [104] - also see Figure 2.2.) The range encompassed for α were values from $20°$ to $90°$ (in 40 discrete steps) and for k from $2,000$ to $60,000$ N/m (also in 40 discrete steps); resulting in 1,600 different angle/stiffness pairs.

Figure 2.9 shows how successful the basic SLIP model (i.e., one that has no capability for learning) is; this is used as a baseline for the experiments detailed in this chapter. A pink dot in the figure represents a starting condition where a stable solution could not be found. The success rate for the non-learning SLIP model was 3.75% (i.e. 60 out of 1600 starting parameter pairs were stable). This is the pre-existing self stabilizing area that is already a feature of the SLIP model. This basic SLIP model does not have the capacity to update its parameters between episodes. Therefore, if started with an unstable parameter it has no way of updating it to become successful.

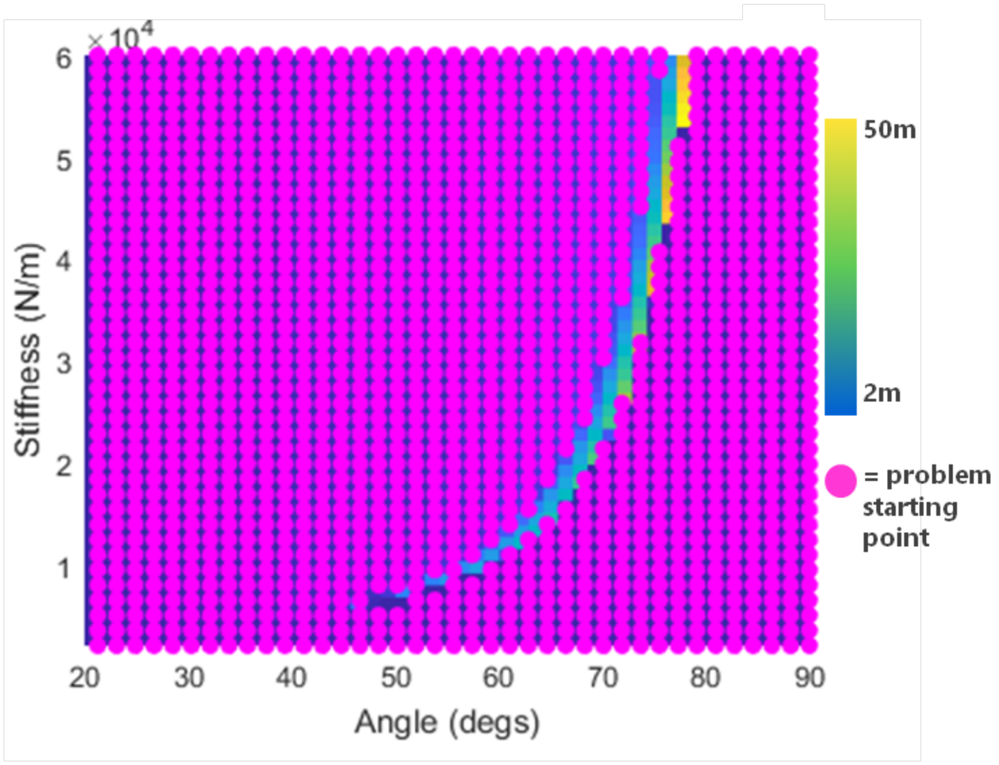


Figure 2.9: Figure showing a standard slip model (i.e, without the capabilities for learning). A pink dot represents a starting condition where a stable solution could not be found. For the tests detailed in this chapter 1600 different starting combinations were tested. From this figure it can be seen that a non learning SLIP model only has a small range of starting configurations where it can be stable (3.75%, 60/1600
) - i.e., the non learning SLIP model can only be stable if it is started in this small range of configurations. Please note that the background colours in this figure represent the distance travelled by the SLIP model in 10 seconds (one episode) at a particular starting configuration. The starting parameters for which the SLIP model travels furthest is indicated by yellow (50m), in contract the dark blue regions indicate starting combinations where the SLIP model travels the least (2m). However it should be noted that this is not necessarily an indication of stability. Both the light green and yellow parameters are stable, however the length of the strides are different due to the difference in parameter combinations.

2.3 Angle Changing Rule Sets

In this section, I will first present the results from the testing of the rules that change attack angle α. The results for the rules that change the stiffness of the SLIP model are discussed later in this section (2.4). Every combination for the attack angle rule sets (a total of 800,400 unique when considering the redundant rule 5) was tested and the results assessed by their success rates, as previously defined), are presented in Figure 2.10. Note in Figure 2.10 that the rule sets are grouped according to the first three binary rules (i.e., increase/decrease, fixed/unfixed, fixed/unfixed), then by an increasing rule 4 and finally increasing rule 5. It can be seen that in the rule sets where Rule 3 is "unfixed" that usually as the value for rule 4 increases the performance of the set decreases. Since in these instances rule 5 is redundant it obviously has no effect on the success rate. In contrast, when rule 3 is "fixed" and thus rule 5 is not redundant it's value has a significant effect on the final set's success rate (arguably more than rule 4). For example, in the case where the first three rules are "increase, unfixed, fixed" there is a strong negative correlation between rule 5 and the overall success rate. Additionally, Table 1 summarizes the top 5 rule sets (they are also highlighted in red in Figure 2.10).

The best rule obtained a success rate of 78.93%, which is much higher compared to the standard SLIP model with 3.75% (i.e., the corresponding base line which is depicted by the red line in Figure 2.10). To clarify, a success rate of 78.93% means that this rule set was run for 1600 different starting combinations. For each starting combination, after each episode, the SLIP model would update the attack angle according this rule set. For 1216 out of 1600 (78.93%) of these starting parameter combinations by the time the lifetime of the model had been reached the attack angle had been successfully updated so that the system was now stable, i.e., within the J-Figure.

Rule Set	% Success Rate	x / total parameter combinations
Increase, Unfixed, Fixed, 0.67, 0.23	78.93%	1263/1600
Increase, Unfixed, Fixed, 0.78, 0.23	78.93%	1263/1600
Increase, Unfixed, Fixed, 0.45, 0.12	71.12%	1250/1600
Increase, Unfixed, Fixed, 0.34, 0.12	71.12%	1250/1600
Increase, Unfixed, Fixed, 0.56, 0.12	69.31%	1109/1600

Table 1: Top 5 successful angle changing rule set and their corresponding success rates. These top rules can also be seen in Figure 2.10 within the red circle. It can be seen that these top rule sets all have the same first three rules. The top rule sets increase the percentage success rate by at least 69.00% when compared with the basic SLIP model that does not have the ability to learn.

It can be seen from Table 1 that the best angle changing rule sets all follow a similar learning mechanism. This is that if the current distance D_i is larger than the previously achieved distance D_{i-1} the angle of attack is increased. The amount

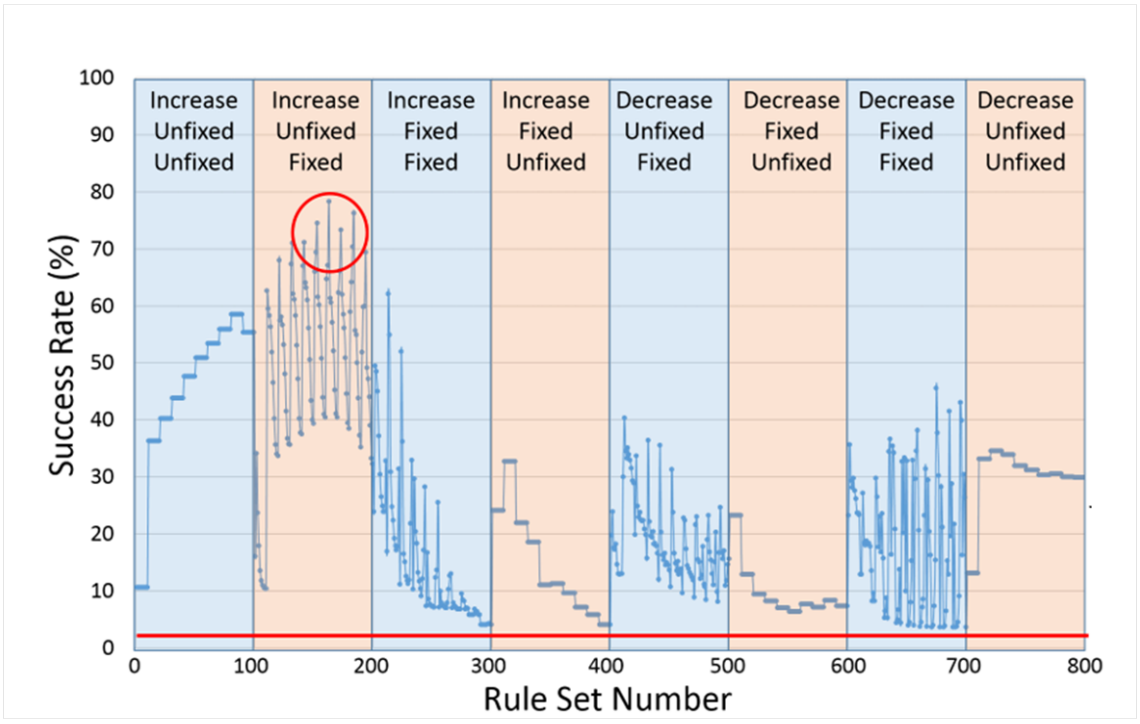


Figure 2.10: This figure shows all the results from the angle changing rule set testing. In total 800 different rule sets were generated and systematically tested, however there are only 400 unique rule sets due to the redundancy in rule 5. Note that the *rule set number* is actually an arbitrary number and just corresponds to the order the rule sets were tested in, but note that they are grouped by the first three rules, then the values of rules 4 and 5. Also shown in this figure is the base line percentage success rate for the basic SLIP model (this baseline is shown by the red line towards the bottom of the figure.) The red circle in this figure shows the top results for the angle changing rule set testing. Note that all the best performing angle changing rule sets have the same first three rules, *Increase, Unfixed, Fixed.*

is inversely proportional to ΔD. This allows the agent to climb the gradient of the SLIP model. Remember this global optimum is the point where the SLIP model is stable and will locomate indefinitely without falling over, so long as there are no extra perturbations to the system. Accordingly, if the leg performance was worse than the the previous run, i.e., $\Delta D < 0$, the angle was reduced by a fixed amount. Figure 2.10 also shows that any combination of the individual rules for changing the attack angle performs better than the default SLIP model, i.e., all success rates are much larger than the base line success rate of 3.75%.

In a similar way to that in Figure 2.9, Figure 2.11 shows the successful starting starting combinations of α and k for the best angle changing rule set. As a reminder, a pink dot represents a starting condition where a stable solution could not be found. Therefore, it can clearly be seen that the region where stable solutions can be found is much bigger (success rate 78.93%) than when its is compared to the standard, non-learning SLIP (success rate 3.75%, see the red baseline in Figure 2.9).

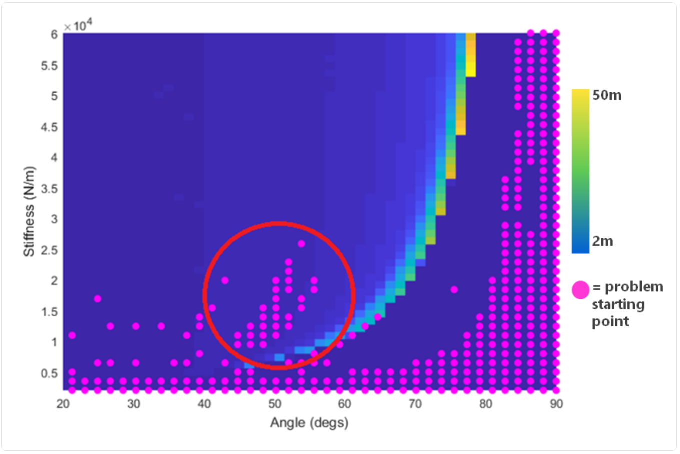


Figure 2.11: This figure shows the starting combinations that are successful for the best angle changing rule set found from the systematic testing. A pink dot represents a starting combination where a stable solution could not be found. Out of the 1600 starting combinations the best angle changing rule set was able to find stable solutions for 78.93% of the starting combinations. Also shown in this figure is a red circle which highlights an unexpected area of starting combinations from which the best rule set could not find a stable solution. These instability points are discussed in the text below. Please note, again that the background colours in this figure represent the distance travelled by the SLIP model in 10 seconds (one episode) at a particular starting configuration.

However, from Figure 2.11 it can also be seen that there are still some areas of the tested starting combinations where stable solutions cannot be found (pink dots). It seems for starting points with smaller angles, left of the J-Figure, most of the points reach a stable solution, while starting combinations with higher angle values are more likely to be unsuccessful. These “large-angle problem areas” were investigated further by looking in depth at the contrast between starting parameters on the right side and on the left side of the J- Figure. The change of α after each episode over the lifetime of the test was observed. These test results showed that if the starting angle is on the right hand side of the J-Figure, i.e., the starting angle was quite large, the learning mechanism will cause the angle to zig-zag *away* from the optimal region. This is shown in Figure 2.12.

By looking at the underlying gradient it can see that both sides of the peak stability region (low angle and high angle) the gradient is negative (as the angle decreases the maximum distance increases, see Figure 2.12). This is why, when the starting condition is on the right side of the J-Figure (it has a large angle) the best rule set will lead away from the stable region.

Given the arrangement of the gradient, a rule set that would be able to distinguish which side of the J curve the starting point was on, would perform even better. However, it this case that functionality it not available, because I assume the system does not have the information in which direction the stable region is located.

Another problem starting point area is shown by the red circle in Figure 2.11. Again, I investigated this area by studying how the attack angle changes after each episode. Here, I observed oscillation between an angle slightly below the stable region and one slightly above it. Interestingly, this behaviour is only seen in the top angle changing rule set; it is not present in the other top 4 angle changing rule sets (see Table 1). However, the other top rule sets in Table 1 have lower overall success rates due to more points at higher attack angles that are unable to find stable solutions due to the zigzagging behaviour described above. Therefore it may be more appropriate to select one of the slightly worse performing rule sets, .e.g. rules 2,3,4 or 5, as ff there were to be any perturbations to the system that would cause the initial attack angle to change, it is likely to only alter it slightly, i.e., into the unstable region observed in the top rule set (Figure 2.11). Although there are more unsuccessful starting parameters in rules 2-5 they are in more predictable areas, which could be more beneficial.

2.4 Stiffness Changing Rule Sets

In the previous section, I showed that allowing the SLIP model to adapts its attack angle between episodes increased the range of starting parameters for which it can achieve stability. However, I do not consider how morphological adaption could also expand this region. Therefore, in this section, I present the results from testing of the 800 stiffness changing rule sets. Figure 2.13 summarizes the obtained success rates from each of the 800 different stiffness changing rule sets that were testing, although note that due to the redundancy of rule 5, only 400 of these are unique. In this figure the rule sets are arranged in the same way as 2.3; first grouped by rules 1,2,and 3 and then by the values of rules 4 and 5. In this figure it can be seen that there is also a negative correlation between rule 5 and the overall success rate.

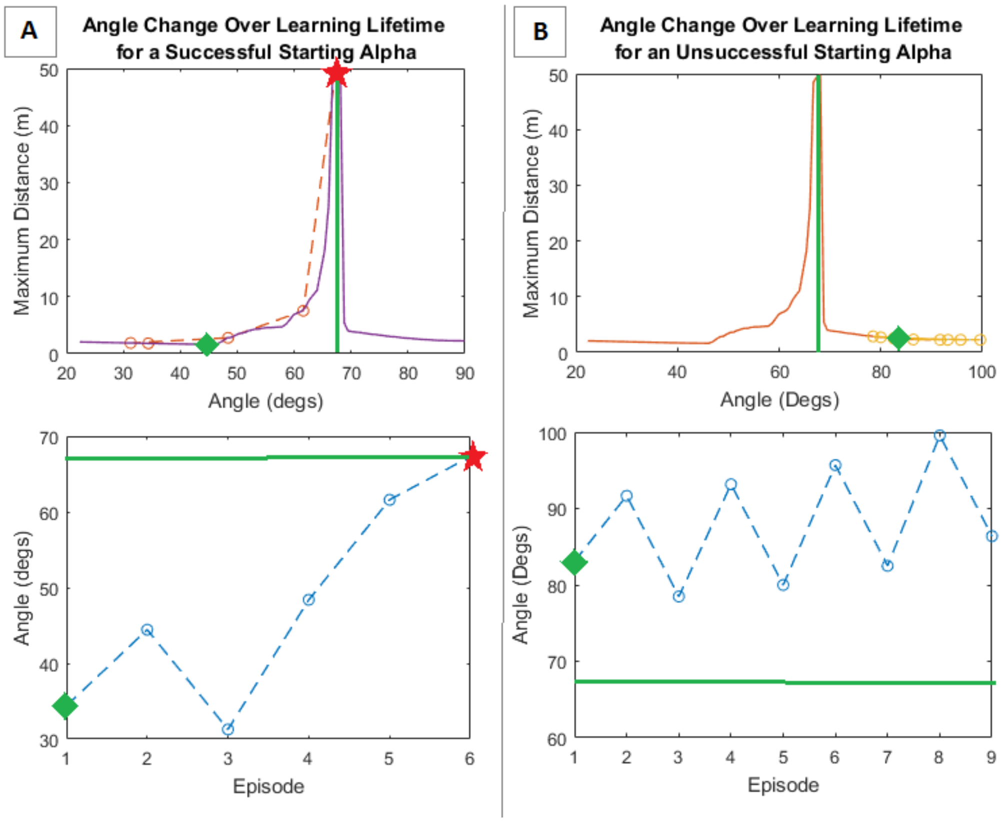


Figure 2.12: The graphs show how the top angle changing rule set attempts learn a stable solution from a variety of different starting conditions. The top figures shows how the rule adapts its angles over multiples steps with respect to the underlying cost function landscape. The bottom graphs show how the angle is changed over time. In both cases a green diamond indicates the starting angle and a red start indicates the final stable angle. A) shows results for a successful starting angle with a suitable stiffness parameter. B) shows results for an unsuccessful starting position, in this case a high angle. In both cases the green diamond indicates the start position, a red star the final position. The stiffness in both cases is kept constant at 20,000 N/m. The green lines on the bottom graphs show the angle where the model would be stable.

Table 2 shows the top 5 stiffness changing rule sets.

Rule Set	% Success Rate	x / total parameter combinations
Increase, Fixed, Fixed, 0.89, 1.00	51.00%	816/1600
Increase, Unfixed, Fixed, 0.78, 0.23	50.50%	808/1600
Increase, Fixed, Fixed, 0.56, 0.67	49.90%	799/1600
Increase, Fixed, Fixed, 0.45, 0.56	49.30%	789/1600
Increase, Fixed, Fixed, 0.67, 0.78	49.30%	789/1600

Table 2: Top 5 successful stiffness changing rule set and their corresponding success rates. These top rules can also be seen in Figure 2.13 within the red circle.

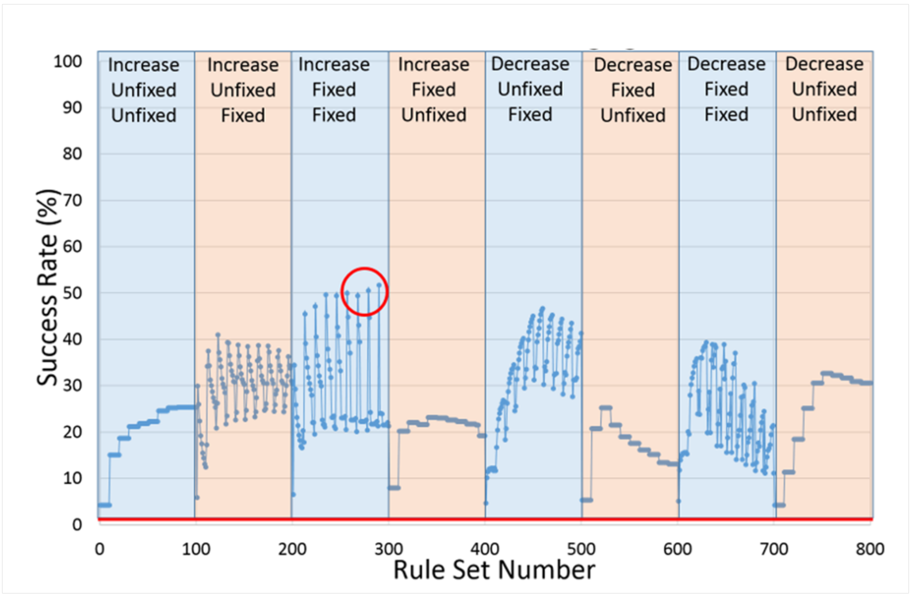


Figure 2.13: This figure shows the entire results from the stiffness changing rule set testing. In total 800 different stiffness changing rule sets were generated and systematically tested. However, only 400 of these are unique due to the redundancy of rule 5. Note that the *rule set number* is actually an number and just corresponds to the order the rule sets were tested in. Also shown in this figure is the base line percentage success rate for the basic SLIP model (this baseline is shown by the red line towards the bottom of the figure.) The red circle in this figure shows the top results for the angle changing rule set testing.

Figure 2.14 shows the starting parameter combinations for which the best stiffness changing rule set is able to find a stable solution. As with the other figures, the pink dots represent starting combinations where stable solutions could not be found. When comparing this figure to the same top angle changing rule and the non-learning SLIP model, it can be seen that in this case the regions of suitable starting combinations (that lead to stable solutions) is much bigger than for the SLIP without learning (compare Figure 2.9, success rate 3.75%), but smaller compared to the best angle adaptation rule (see Figure 2.11, success rate 78.93%). Additionally it can be seen from Figure 2.13 that all of the 800 stiffness adapting rule sets that were tested have a higher percentage success rate than the non learning SLIP model.

In a similar way to the top five angle changing rule sets, the top five stiffness changing rule sets all have the same three starting rules (Rules 1,2,3). The stiffness value is increased if the leg performs better than in the previous episode $\Delta > 0$, and decreased if $\Delta D < 0$. Both the increase and decrease of the stiffness is fixed regardless of the magnitude of ΔD.

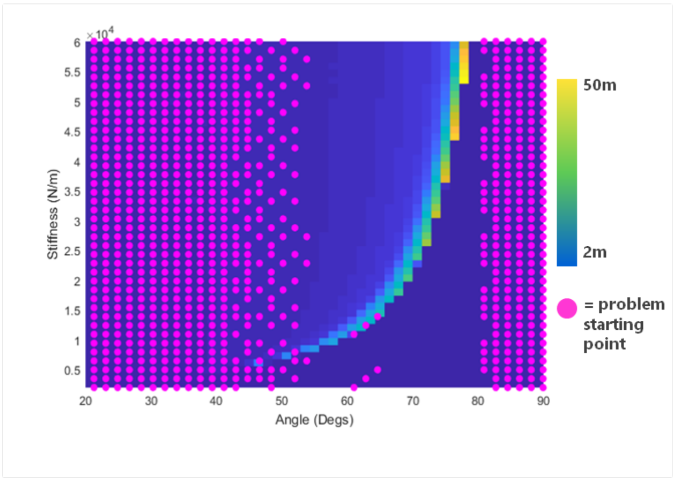


Figure 2.14: Figure showing the results from the testing of the top stiffness changing rule set. A pink dot represents a starting condition for which the top rule set was not able to find a stable solution. It can be seen that the top stiffness changing rule set was about to find stable solutions for 51.00% of the 1600 starting parameter combinations Please note that the background colours in this figure represent the distance travelled by the SLIP model in 10 seconds (one episode) at a particular starting configuration. The starting parameters for which the SLIP model travels furthest is indicated by yellow (50m), in contract the dark blue regions indicate starting combinations where the SLIP model travels the least (2m).

However, when comparing the success rates of the entire 800 stiffness / 800 attack changing rule sets there is an interesting key difference (see Figures 2.13 and 2.10). In Figure 2.10 it can be seen that rule sets beginning *[Increase, Unfixed, Fixed* outperform rule sets that have a different Rule 1,2 and 3. In contrast, when considering the entire stiffness changing rule sets there seems be various rule sets with the different first three rules (for example *[Decrease, Fixed, Fixed]*, and even *[Decrease, Unfixed, Fixed]* that are also very successful.

2.5 Combined Angle and Stiffness Changing Rule Sets

The previous two sections of this chapter show that the area of starting parameters for which the SLIP model can learn to become stable from can be expanded by allowing the model to learn from previous locomotion attempts and use this information to update **just one** of its parameters. From these results it can be seen that although the adaption of both parameters is beneficial, the region of successful starting parameters is slightly larger for the optimal attack angle adaption rule set that the optimal stiffness adaption rule set.

Given that adapting a single parameter can improve the robustness of the SLIP model, it could be speculated that adapting both parameters after each episode would improve the robustness further. Therefore, I also investigated rule sets that were able to adapt both the angle and the stiffness of the SLIP model. As mentioned above in the methods section of this chapter, these *dual adaption* rule sets are created by simply using an angle changing rule set "side by side" with a stiffness changing rule set. Both the two rule sets are kept the same, and are responsible for changing just one of the parameters. After each episode, the angle changing rules update the attack angle and the stiffness changing rules update the stiffness – this is done independently. However, because *both the attack angle and stiffness* are now updated, the distance achieved by the SLIP model in the subsequent episode is different from if just one parameter had been updated. If all possible angle changing rule sets (800 in total, 400 unique) are combined (or paired with) all possible stiffness rule sets (again 800 in total, 400 unique), this results in a possible 160,000 different rule combinations (160,000 unique). As a result it is not possible to test every single rule set through brute force. In fact, it would take 3 months of simulation time to test the entire 160,000 different rule sets for the computer setup I used.

Instead of bulk testing these dual adaption rule sets, I take a simpler approach were a small subset of these *dual adaption* rule sets are created by combining the top five angle changing rule sets with the top five stiffness changing rule sets (forming 25 rule sets in total). Although simplistic, this was considered an adequate starting point.

Indeed when considering the best dual adaption rule set combined from the best angle changing rule sets and the best stiffness changing rule, at first sight, there appears to be little benefit gained compared to just adapting the attack angle. The percentage success rate of the combined adaption rule set was 75.56%, (1209/1600). This is compared to the best angle adaptation rule alone (78.93%, 1262/1600) and the best stiffness adaptation rule (51.00%, 816/1600). The performance of this combined rule set are shown in 2.15.

Although the overall success rate of the combined rule set is lower than the top angle changing rule set, an advantage of this combination is that the the problem area shown in Figure 2.11 (by the red circle) does not show here. It appears that the addition of the stiffness adaption means that these starting combinations are now successful. However, on the other hand, a disadvantage of this combination is the region highlighted by the black circle in Figure 2.15, which seems to emerge because of the combined change of the two parameters. Specifically here, the stiffness and angle change seem to work against each other. The changes they introduce have opposite effects resulting in a zig-zag pattern in a direction that moves perpendicu-

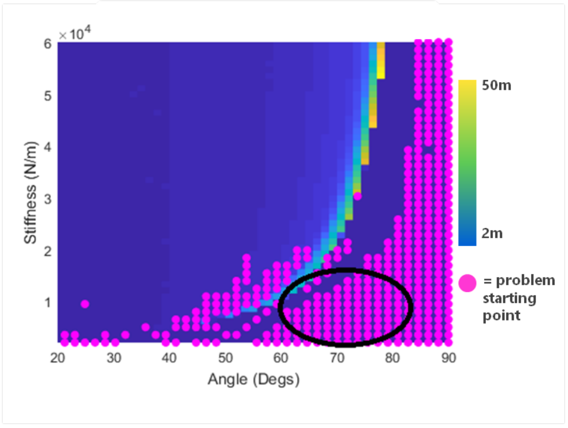


Figure 2.15: Performance when the best stiffness and angle changing rule sets are combined, i.e., *[Increase, Unfixed, Fixed, 0.67, 0.23 (angle) Increase, Fixed, Fixed, 0.89, 1 (stiffness)]*. A pink dot represents a starting condition for which a stable solution can not be found. It can be seen from the figure that this combined rule set is able to find stable solutions for 75.56% of the starting parameters. The black circle highlights an interesting area of unexpected problem starting points. See text for discussion. Please note that the background colours in this figure represent the distance travelled by the SLIP model in 10 seconds (one episode) at a particular starting configuration. The starting parameters for which the SLIP model travels furthest is indicated by yellow (50m), in contract the dark blue regions indicate starting combinations where the SLIP model travels the least (2m).

larly away from the area of stability. The summary of the top five combinations, out of the additional 25 duel adaption rule sets that were tested, are shown in the table below. Interestingly, they all use the best angle rule set, but use inferior stiffness rule sets to achieve higher success rates.

Angle Rule Rank	Stiffness Rule Rank	Success Rate
1	4	76.31%
1	3	76.31%
1	2	76.25%
1	5	75.81%
1	1	75.56%

Table 3: Table showing top 5 of the 25 different rule set combinations tested. These 25 rule sets were formed by combining each of the top 5 angle changing rule sets with each of the top 5 stiffness changing rule sets. The fitness of all 25 different rule set combinations are shown in Table 4

This suggests that there may be combinations of angle changing rules and stiffness changing rules that are less successful when used on their own, but when combined are able to further expand the region of stability. Since there are two many combinations of rule sets to bulk test, using a heuristic search algorithm like an evolutionary algorithm, may yield even better results. This idea and others for further work, plus

		Angle Rule Rank				
		1	2	3	4	5
Stiffness Rule Rank	1	75.56%	69.52%	61.25%	58.21%	58.25%
	2	76.25%	73.25%	66.81%	59.21%	57.25%
	3	76.31%	74.61%	65.25%	59.65%	57.81%
	4	76.31%	74.67%	65.25%	59.65%	57.81%
	5	75.81%	70.25%	65.25%	59.21%	58.21%

Table 4: All 25 combined angle and stiffness changing rule sets. The top 5 rule sets are highlighted by red text - they are also presented for clarity in Table 3

a summary of the chapters findings, is detailed in the next section.

2.6 Chapter Conclusions

In this chapter, I have investigated using simple rule sets to adapt the stiffness and attack angle of the Spring Loaded Inverted Pendulum Model to improve its robustness. These rule sets use information from the SLIP models previous locomotion attempts to update its morphology and control parameters for the next try. This means that when using an optimal rule set, if the SLIP model is started with unstable parameters, it has the potential to learn to become stable by episodically updating it morphology and control.

Each rule set is made up individual rules that when combined uniquely converts information from the SLIP model's interaction with the environment to determine how to adapt either the attack angle or the stiffness of the spring with the aim of becoming stable. In this case, the information from the SLIP model's interaction with the environment, is a comparison between the distance the SLIP model traveled before it fell over on its previous attempt and the distance achieved on its current attempt (ΔD).

In this chapter, 1600 different rule sets were systematically tested, although note that only 800 were unique due to redundancy in one of the rules. Each rule set was tested with 1600 different starting conditions (i.e., combinations of starting attack angle and spring stiffness). The percentage of these 1600 different starting conditions for which the the rule set in question could find a stable condition was used as a the metric for determining a successful rule set.

Overall the results detailed in this chapter show that improvement, in terms of percentage success rate, can be achieved by implementing any one of these simple rule sets. The least successful rule sets, in terms of percentage success rate, are still able to find stable solutions for approximately double the amount of starting conditions than a basic SLIP model, which is unable to adapt. Additionally the top rule sets are able to increase the baseline percentage success rate from 3.75% for the basic SLIP model to 78.93% (the best change changing rule set) and 51.00% (stiffness changing rule set). It is interesting that these percentage success rate can be achieved through systematic testing of these very simple rule sets. Furthermore, these high percentage success rates are achieved simply by adapting one single parameter (i.e, either the stiffness **or** the attack angle).

These results suggest that in the case of offline, episodic learning, updating the attack angle is more successful in achieving robust locomotion than changing the stiffness. It could be that this is slightly biased by the selected starting parameters combinations that were tested. For example, if instead of using the initial attack angle range of 0-90 degrees the range was shorted to 50-90 degrees this would change the percentage success rate. This would then make the top stiffness changing rule set success rate and top angle changing rule set success rate more comparable.

As previously discussed, the percentage success rate can be increased by at least 50% by just changing one parameter. However it is likely that the percentage success rate could be increased further by adapting both the spring stiffness and the attack angle. Some initial investigation regarding the concept of dual adaption was carried out in this chapter. The top five angle changing rule sets were combined with the top five stiffness changing rule sets (to create 25 combined dual adaption rule sets) and tested to determine their percentage success rate. Through this testing it was found that the dual adaption rule sets that were generated in this way did not improve the percentage success rate when compared with the top angle changing rule sets (there was a 3% increase). However, when compared with the best stiffness changing rule set, there appears to be a benefit to using the dual adaption (i.e, an increase in approximately 15% in percentage success rate when also adapting the attack angle) It is likely however, that this approach of generating dual adaption rule sets is too simple and other dual adaption rule sets could be more successful. Another indication that this approach is too simple is that the top angle changing rule set combined with the top stiffness changing rule set actually performs worst of all the 25 combined rule sets that were tested. The best combined rule set tested is the combination of the best angle changing rule set and the 4^{th} best stiffness changing rule set. Additionally, when considering the results from the dual adaption rule sets, the top five combined rule sets all have the same angle changing part (i.e., the top angle changing rule set found from the systematic testing.) This could provide more evidence to the hypothesis that adaption of the angle is more useful in gaining stability than the stiffness.

With regards to the dual adaption rule set testing the results are somewhat inconclusive. The combination top attack angle adaption rule set and the top stiffness adaption rule set perform worse than the top attack angle rule set alone. In fact, the top attack angle adaption rule set performs better when combined with a lower ranked stiffness adaption rule set. This presents the possibility of better combinations of rule sets, then when working alone perform poorly, but when combined outperform these investigated in this chapter.

It has already been highlighted that systematically testing all possible combined rule sets is infeasible due to computational expense. However using a search heuristic such as an evolutionary algorithm has the potential to find optimal combined rule sets in a feasible amount of time. Therefore, in the next chapter I use evolutionary algorithms to explore further optimal dual adaption rule sets and investigate further the interplay of the attack angle and stiffness adaption in the offline episodic learning approach.

An interesting discussion point surrounding this chapter is the implications the results have with regards to a physical system. Whilst the chapter has some promising outcomes, all this work has been carried out in simulations and therefore it is worth considering how these results might relate to a physical system. Due to the simplicity of the rule sets implementation on a real system is feasible; the change in distance between episodes could be measured easily by an encoder. Similarly the attack angle could be easily adjusted through use of a stepper motor. However, adaption of the spring stiffness, without dismantling the robot, is likely to be more challenging, although this could be achieved with variable stiffness actuators. Additionally, the current model is simplistic and does not include, for example, damping would also affect the performance.

Something that is not investigated in this chapter is the amount of time (i.e., the number of episodes) it takes for the SLIP model to update its parameters to become stable. In simulation this is not as large a problem. However, if, as suggested in the previous paragraph, this system was to be implemented in hardware, taking many episodes to learn to be stable would be highly energy inefficient. It would also results in a large number of required failures, which could result in damage to the robot.

Another limitation of this work is that it uses an offline approach. This has some advantages; for instance if the robot was to undergo some damage which caused its mass to change, the system would be able to re-adapt its stiffness and attack angle to combat this change. As previously discussed this offline episode approach allows the model to update parameters from unstable to stable ones through learning from past experience.

However, because the approach is offline, the robot would have to fail (fall over) and this could damage the robot further which in turn could be costly. This would also mean that every time the environment changed the robot would have to undergo a number of failures before regaining stable locomotion. Therefore, an approach where the robot could adapt stride to stride would be useful.

As well as exploration into using evolutionary algorithms to evolve optimal dual adaption rules in the offline episodic methodology, in the next chapter I also consider how to implement these rules to allow for online learning. The rule sets are altered so that they allow the parameter adaption to take place between stride; removing the requirement for failure. These online rule sets are evolved to find those that have the highest percentage success rate, but also take the shortest amount of time to find stability. Furthermore, in the next chapter, how these online rule sets are able to cope with sudden changes to the environment, such as downwards steps, is also investigated.

3 Evolving optimal adaption rule sets for robust locomotion in the SLIP Model

In the previous chapter I presented my work on using rule sets to adapt the stiffness and morphology of a simple Spring Loaded Inverted Pendulum model in order to expand the number of starting conditions that the model was able to find stability from. The adaption of either the spring stiffness or the attack angle took place episodically, i.e., the SLIP model would first attempt locomotion with one set of parameters (spring stiffness and attack angle). If the model failed (falls over) the parameters are updated according to the rule set in use and locomotion is reattempted. Using this offline, episodic approach, the SLIP model is able to learn from previous experience in order to become stable.

In total 1600 different rule sets were systematically determined to an optimal solution; 800 attack angle adapting rule sets and 800 stiffness adapting rule sets. The best rule set was one that adapted the attack angle; it increased the number of starting parameters, from which the model could learn to become stable, from 48 to 1218 (out of 1600). This is compared to the basic SLIP model that does not have the capacity to learn and therefore adapt its parameters.

In the previous chapter the focus was on adapting either the stiffness or the attack angle. Whilst some testing on dual stiffness and attack angle adaption rule sets was carried out, these were simply generated by combining the best stiffness and angle changing rule sets. Using this methods, the tested dual adaption rule sets were only able to increase on the stability area by 48 from the best angle changing rule sets.

Thus, in this chapter I explore whether using a heuristic search algorithm, in this case an evolutionary algorithm, would be able to find a dual adaption rule set able to increase the area of stability further.

There are two main parts to this chapter. Firstly, as discussed, I use evolutionary algorithms to evolve adaption rule sets that adapt both parameters in the hope of exploiting the tight interplay between the control parameter and soft morphology of the body to improve further the robustness of the SLIP model. As in the previous chapter these extended rule sets adapt the parameters (i.e., the attack angle and spring stiffness) between episodes. The difference is in this chapter is that now both parameters updated simultaneously.

Additionally, in the previous chapter, only one criteria was selected to determine the success of a genome; the percentage success rate. This was the percentage of tested starting parameter combinations for which the rule set was able to find a stable solution. As discussed in the previous section, this fitness criteria has some limitations. For example, it might take a larger number of episodes for the rule set to find a stable solution. Whilst this doesn't present a large problem in simulation, if this methodology was transferred to a physical system, taking a large number of episodes to stabilize would be highly inefficient. It would also require a large number of failures which could cause damage to the robot. Therefore, in this chapter, a second fitness criteria is also introduced which optimises for a reduced number of episodes required until the model reaches stability.

In the second part of the chapter, the rule sets are adapted to allow for online learning. A limitation of the work in the previous chapter (i.e., the offline approach) was that failure was required between parameter updates. The episodic approach also would not be capable of dealing with sudden changes in environment, such as decreases in ground level. Therefore I also alter the rule sets so that they are able to update the SLIP model parameters between strides, instead of after failure. Using this "online" approach investigation of environmental transitions are also carried out.

3.1 Offline Learning

In this first part of the chapter, offline learning, i.e. learning happens between two episodes is investigated. In this offline learning section, the adaption of the stiffness and attack angle occurs episodically, as in the previous chapter. However, in this chapter **both** parameters are updated simultaneously and an evolutionary algorithm is used to search for optimal rule sets.

The simulation of the SLIP model in this chapter is the same as in the previous chapter. Remember that if the SLIP model has the correct combination of attack angle (the angle the spring forms with the ground on touch down, α) and spring stiffness (k) then the model will be stable. It will be able to transfer between phases without loss of energy and without falling over. It is also able to withstand small perturbations to the system such as changes in ground level, i.e. it is self stabilizing. However, if the changes in ground level are too large the model will fail.

In the case of the offline learning, the SLIP model attempts locomotion on a flat horizontal landscape with a starting attack angle (α) and starting stiffness (k). As previously, each locomotion attempt is referred to as episode, an episode either ends when the model fails or when 10 seconds has expired, at this point the model was considered stable. As before, the distance the SLIP model is able to travel in one episode is compared with the distance travelled in the previous episode. This change in distance (ΔD) is used as a key measurement in the offline rule sets. These offline rule sets dictate how much the stiffness (k) or attack angle (α) should be changed. For a complete description of the SLIP model simulation please refer to the previous chapter.

As previously discussed the key difference between the offline episodic learning in this chapter and the previous is that both parameters (attack angle α and stiffness k) are now updated simultaneously. The way the parameters were updated was dictated by a rule set formed of 5 different individual rules, one rule set for changing the attack angle and one for changing the stiffness of the spring. To create a rule set capable of updating both parameters, two “5 rule” rule sets can be combined to form a “10 rule” rule set. Figure 3.1, shows a simple explanations of the individual rules used to make up a “10 rule” rule set capable of adapting both parameters. Note that these are the same individual rules used in the previous chapter, but this time combined to form a dual-adaption rule set.

Using the possible value options (detailed in Figure 3.1) for each individual rule there are 640,000 different possible combined rule sets. This would take too long to systematically test all of these rule sets so a search algorithm is required.

In this chapter, I use an evolutionary algorithm to determine optimal dual adaption rule sets. Details of the evolutionary algorithm are described in the following section.

Generic Rule Set			
Rule Number	Description	Value Options	Notes
1	If $\Delta D>0$ should the attack angle increase or decrease?	Increase/Decrease	If $\Delta D<0$ the angle will change in the opposite way to that specified
2	When it increases, should the angle increase by a fixed amount?	Fixed/Unfixed	If rules 2 or 3 are “fixed”, this amount is determined by rules 4/5. If rules 2 or 3 are “unfixed”, the amount is determined by the equation: $angle\ change = \frac{\mu}{2+\Delta D}$ Where μ is given by rule 4.
3	When it decreases, should the angle decrease by a fixed amount?	Fixed/Unfixed	
4	How much should the angle be increased by?	0.01-0.99	
5	How much should the stiffness be decreased by?	0.01-0.99	If rule 3 is “unfixed”, rule 5 is redundant
6	If $\Delta D>0$ should the attack stiffness increase or decrease?	Increase/Decrease	If $\Delta D<0$ the angle will change in the opposite way to that specified
7	When it increases, should the stiffness increase by a fixed amount?	Fixed/Unfixed	If rules 7 or 8 are “fixed”, this amount is determined by rules 9/10. If rules 7 or 8 are “unfixed”, the amount is determined by the equation: $stiffness\ change = \frac{\mu}{2+\Delta D} \times 10000$ Where μ is given by rule 9.
8	When it decreases, should the stiffness decrease by a fixed amount?	Fixed/Unfixed	
9	How much should the stiffness be increased by?	0.01-0.99	If “fixed”, rules 9 and 10 are multiplied by 10,000
10	How much should the stiffness be decreased by?	0.01-0.99	If rule 8 is “unfixed”, rule 10 is redundant

Figure 3.1: Individual rules that are encoded and combined to build a rule set. The top 5 grey rules show those relating to angle adaption. The bottom rules shown in blue relate to stiffness adaption. A complete rule set uniquely describes how the SLIP model should use the information from each episode to change the stiffness (k) and attack angle (α) of the SLIP model based on the change in distance achieved in consecutive episodes (ΔD).

3.2 Evolutionary Algorithm

This section details the key characteristics of the algorithm and the two fitness criteria used to evolve optimal duel adaption rule sets. Ten different individual rules form the genome (also known as the rule set), see Figure 3.1. An initial population of 50 genomes were formed from 10 randomly initialized rules. Specifically, rules 1,2,3,6,7 and 8 were randomly assigned one of the two values. In the case of rules 1 and 6 values were either ’Increase’ or ’Decrease’. To encode this ’Increase’ was represented as a ’1’ and ’Decrease’ as a ’0’. In the case of rules 2,3,7 and 8 these values were ’Fixed’ (encoded as ’1’) or ’Unfixed’ (encoded as ’0’). For the non-binary rules (rules 4,5,9,and 10) a random value was selected between 0.01 and 0.99 and directly encoded as thus.

Two different fitness criteria were chosen and tested. This allowed exploration as to which selection criterion would produce better results overall.

The first criterion was the *percentage success rate*. This is the same criteria that was used in the previous chapter. As a reminder, for this criteria, 1,600 different starting combinations of attack angle (40 values equally spaced between 20-90 deg) and stiffness (40 values equally spaced between 2000 - 80,000) are tested. This region was selected as it fully encompassed the “J” curve region of stability detailed in the work by Blinkhan et al. [7] and as shown in Figure 3.2. The *percentage success rate* of a rule set is defined as the percentage of starting combinations, within the selected range, for which the SLIP model could find stability when following that particular rule set.

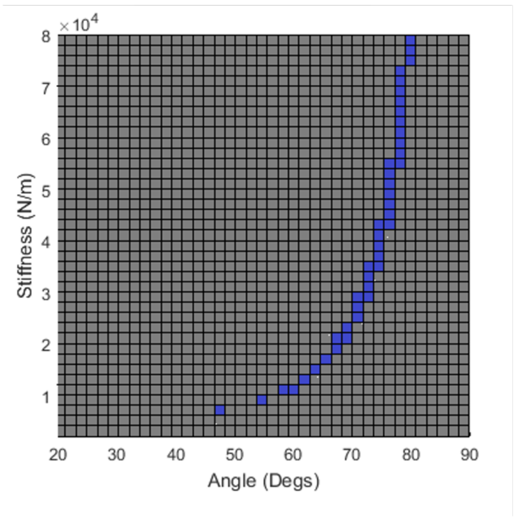


Figure 3.2: Figure showing the stability regions of a SLIP model. The blue “J-Figure” shows the starting combinations for which the basic SLIP model is stable. This figure also shows the regions of all starting combinations that were tested to determine the success of a rule set/genome

As previously discussed, in this chapter, I also use a second criterion to determine the success of a rule set: the *mean recovery time (offline learning) / time to stability (online learning)*. These criteria determine how long the SLIP model takes, starting at a particular parameter combination, to stop adapting its parameters and maintain a stable solution. For offline learning this is the number of episodes required to reach stability. The *mean recovery time* is the average *individual* recovery times for all the tested starting parameter combinations. Therefore, the lower the *mean recovery* *time* the more successful the rule set it determined to be.

For **offline learning** the evolutionary algorithm was run six times; three times using the *percentage success rate* as the fitness, and three times using the *mean recovery time*. For each different run a new initial randomised population was used.

Each time the evolutionary algorithm was run for 100 generations — each new generation was formed as following. The top 5 best rule sets from the previous generation were automatically passed onto the next (i.e. *elitism*). The next 15 rule sets of the new generation were produced by two-point crossover, as demonstrated in Figure 3.3 between randomly selected parents from the 50% most successful genomes from the previous generation. The next 15 rules sets were generated by mutating the non-binary rules (rules 4,5,9 and 10). Note that in this instance the binary rules are unchanged, initial testing determined enough variation in the binary rules was created by crossover. This was done by randomly selecting a new value between 0.01 and 0.99 for these rules, see Figure 3.4. The final 15 rule sets were randomly generated.

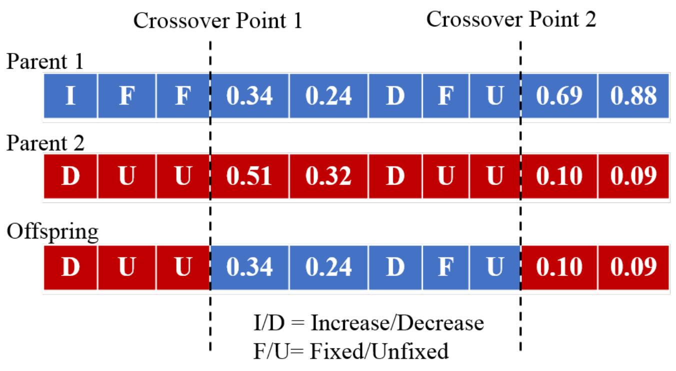


Figure 3.3: This figure shows two example genomes that are used to create an offspring via multi-point crossover, parent 1 is shown in blue, parent 2 is shown in red. The figure also shows the two crossover points. The offspring is therefore created by the combining the first three rules from parent 1, with rules 4,5,6,7,8 from parents 1 and finally rules 9 and 10 from parent 2.

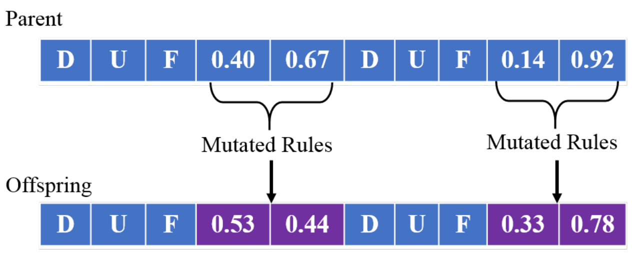
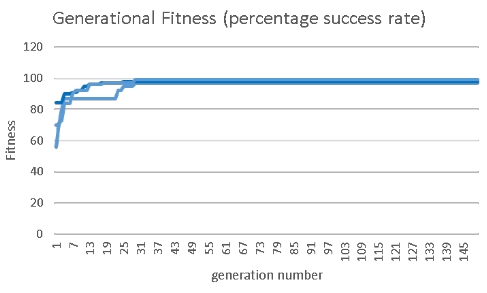

Parent
D U F 0.40 0.67 D U F 0.14 0.92
Mutated Rules
Mutated Rules
Offspring
D U F 0.53 0.44 D U F 0.33 0.78

Figure 3.4: Diagram showing how an offspring genome is created via mutation from a single parents. The parent is shown in blue and the mutated genes in the offspring are shown in purple. Note that only the non-binary rules, i.e. rules 4,5,9 and 10 are mutated

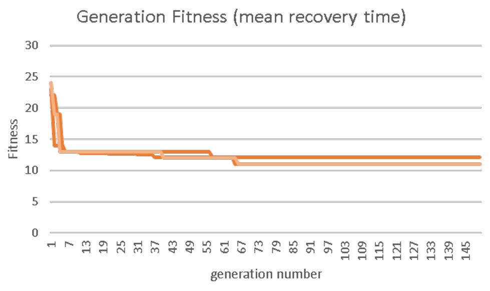
Note that the design and choice of the parameters could be considered a limitation of the research presented. The design does not match a "standard evolutionary algorithm", moreover, the amount of effort put into optimising the parameters for the algorithm was limited. This was selected simply as a preliminary design; some of the implications of this are discussed in the results section.

3.3 Offline Results

In this section, the results for offline learning are discussed. Figure 3.5 shows the fitness of the top genomes for each evolution run after each generation both both fitness criteria used. Table 5 shows the most successful rule sets for each of the 6 different optimization runs (i.e. the three repeated runs optimized for percentage success rate and the three repeated runs optimized for mean recovery time) along with their corresponding percentage success rate and mean recovery time. Remember, as a comparison a SLIP model that has no learning capabilities has a percentage success rate of 3% (compare Figure 3.2). For a comparative number, the "mean time to recovery" of a basic SLIP model is 97 episodes. This is calculated as follows. When the basic SLIP model is tested with a stable starting parameter combination it has a time to recovery of 0 episodes, i.e., it takes zero episodes to find stability because it is already stable. This occurs for 3% of the tested starting parameter combinations. For the other 97% starting points, the basic SLIP model is not able to find stability at all as it lacks the ability to update its parameters. Therefore, for these points, the basic SLIP model can be thought of as having a maximum score of 100 episodes until stability, although in actuality this number of episodes would be infinity. The average the time to recovery for all starting combinations for a non-learning SLIP model is therefore 97 episodes.



(a)



(b)

Figure 3.5: Sub-figures a and b show the fitness of each of the six generation runs. A) shows the first three runs that were evolved using the percentage success rate as a fitness metric. B) shows the second set of three runs that were evolved using the mean time to recovery as the fitness metric.

It can be seen from Table 5, that the optimal solutions found when using "mean time to recovery" as the selection criteria outperform those using "percentage success rate". Not only do they have a better mean recovery time, but also a better

	Trial Number					
	Optimized for Percentage Success Rate			**Optimized for Mean Recovery Time**		
Rule No	1	2	3	4	5	6
1	Decrease	Increase	Increase	Increase	Increase	Increase
2	Unfixed	Unfixed	Unfixed	Unfixed	Unfixed	Unfixed
3	Fixed	Fixed	Fixed	Fixed	Fixed	Fixed
4	0.01	0.98	0.82	0.57	0.7	0.53
5	0.04	0.37	0.32	0.21	0.27	0.2
6	Increase	Increase	Decrease	Decrease	Decrease	Decrease
7	Fixed	Fixed	Fixed	Unfixed	Unfixed	Fixed
8	Unfixed	Fixed	Fixed	Fixed	Fixed	Fixed
9	0.59	0.93	0.03	0.05	0.14	0.01
10	–	0.01	0.86	0.9	0.79	0.94
% Success Rate	**99**	**97**	**98**	**97**	**99**	**98**
Mean Recovery Time	**24**	**12**	**14**	**11**	**12**	**11**
STD Recovery Time	**7**	**6**	**6**	**6**	**6**	**6**

Table 5: This table shows each of the optimal genomes founds in the six evolution trials for **offline learning**. The first three rule sets were found when optimising for percentage success rate. The final three rule sets were found when optimising for mean recovery time. Colour coding is used to highlight the differences in rule set. The binary rules are represented by either blue or brown text whilst the non-binary rules are between red (very low values) or green (very high values).

percentage success rate. Therefore this indicates that the mean recovery time is a better fitness metric, i.e. it evolves better performing genomes **in both fitness criteria**, than optimizing for percentage success rate. Additionally, when considering the individual rules in each of the top genomes, there are strong similarities in those that were evolved for mean recovery time. All three rule sets have the same first three rules – "Increase", "Unfixed" and "Fixed", and similar values for all non binary rules (i.e. rules 4,5,9 and 10). In contrast the individual rules in the top rule sets optimized for percentage success rate show less correlation.

When comparing all of the top 6 best rule sets it can be seen that they all outperform the non-learning SLIP model and are able to find stability for almost all starting combinations in our exploration region. This shows an increase in robustness from the non-learning SLIP model. Although there are some cases (i.e. starting configurations) where these top rule sets are unable to find stable solution, these are all at the limits of the testing values (i.e. at a starting angle of 90 degrees). This is not a large problem as 90 degrees is equivalent to a completely horizontal leg, which is quite unnatural.

If these dual adaption rule sets are compared with those explored in the previous chapter, it can be seen that the using the evolutionary algorithm search did in fact yield much better solutions that just combining the two individual (angle and stiffness changing rule sets). Similarly the work in this chapter shows that using the dual adaption method, rather than just adapting one parameter can also improve the robustness of the SLIP model.

For the wide majority (over 98%) of the tested starting parameter combination these optimal rules sets are able to update the parameters in order to find stability. Furthermore, evolving for mean time to recovery ensures that stability is found time-efficiently.

However, it is also important to consider what implications the un-optimised evolutionary algorithm parameters may have on the results. As previously discussed, the evolutionary algorithm was designed quickly, with little thought given to the hyper-parameters. Furthermore, it does not confirm to what is considered to be a standard evolutionary algorithm [29]. In Figure 3.5 it can be seen that the algorithms converge almost immediately. This indicates that the evolutionary algorithm isn't as successful as it could be with a better design. In the case of this offline learning approach, the results, especially the percentage success rates are very high. Therefore, the weakness of the evolutionary algorithm isn't considered as much of a problem as it could be in other experiments.

Additionally these algorithms were only run 6 times (only 3 repeats for each experiment). However, for both experiments the standard deviations between the optimised values (percentage success rate for the first experiment, mean recovery time for the second) are low ($\mu <= 1$). Whilst lack of repeated experiments is a limitation of this work, given the low standard deviation this is more acceptable.

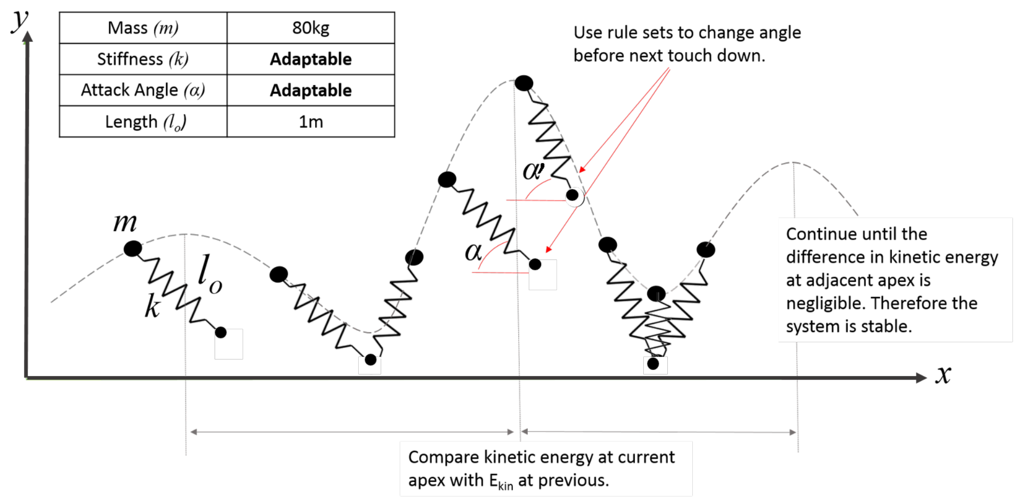
The limitation of requiring failure to learn still remains. Also, as discussed, the offline learning method does not allow for sudden changes in the environment, such as decreases in ground level.

In the next section, I investigate how these offline rule sets can be translated to online rule sets. These online rule sets allow adaption of the parameters between strides, rather than after failure.

3.4 Online learning

As discussed in the introduction to this chapter there is a limitation to the offline learning approach – it requires failure in order to learn to become stable. Whilst this isn't necessarily a problem in the case of simulation, if this work is implemented on a real robot, failure could be very costly. Therefore, I extended the offline rule sets previously discussed earlier in this chapter (Section 3.1) (and also in the chapter before, Chapter 2) to allow for online learning. In contrast to the previously used offline learning, in this online approach the SLIP model is not required to fail; instead adaption of the attack angle (α) and spring stiffness (k) occurs just before touch down of the leg. Indeed, if the leg does fall over this terminates the learning for that particular starting parameter. Given that in this online learning no episodes occur, the rule sets are no longer able to use ΔD as one of the key parameters.

Therefore, the online rules are based around the exchange of kinetic energy between each stride. This is based on the observation that in a completely stable SLIP model the kinetic energy at each apex of the flight phase should remain constant; therefore, if there is a change in kinetic energy, this could be an indication of instability. Note however, that since this is a lossless system the overall energy would not change - instead the respective amounts of kinetic and potential energy would change. Therefore it should also be noted that the same results are likely to be achieved by using apex height or horizontal velocity as the key parameter, instead of kinetic energy. The kinetic energy is calculated, by measuring the horizontal velocity at the apex (the vertical velocity is zero at this point) at each stride as shown in Figure 3.6. Before touch down the difference in kinetic energy between consecutive flight apexes ($\Delta KE_(i) = KE_{(i)} - KE_{(i-1)}$) is considered and this information is used by the rule sets to determine how the parameters (i.e. α and k) should be updated.



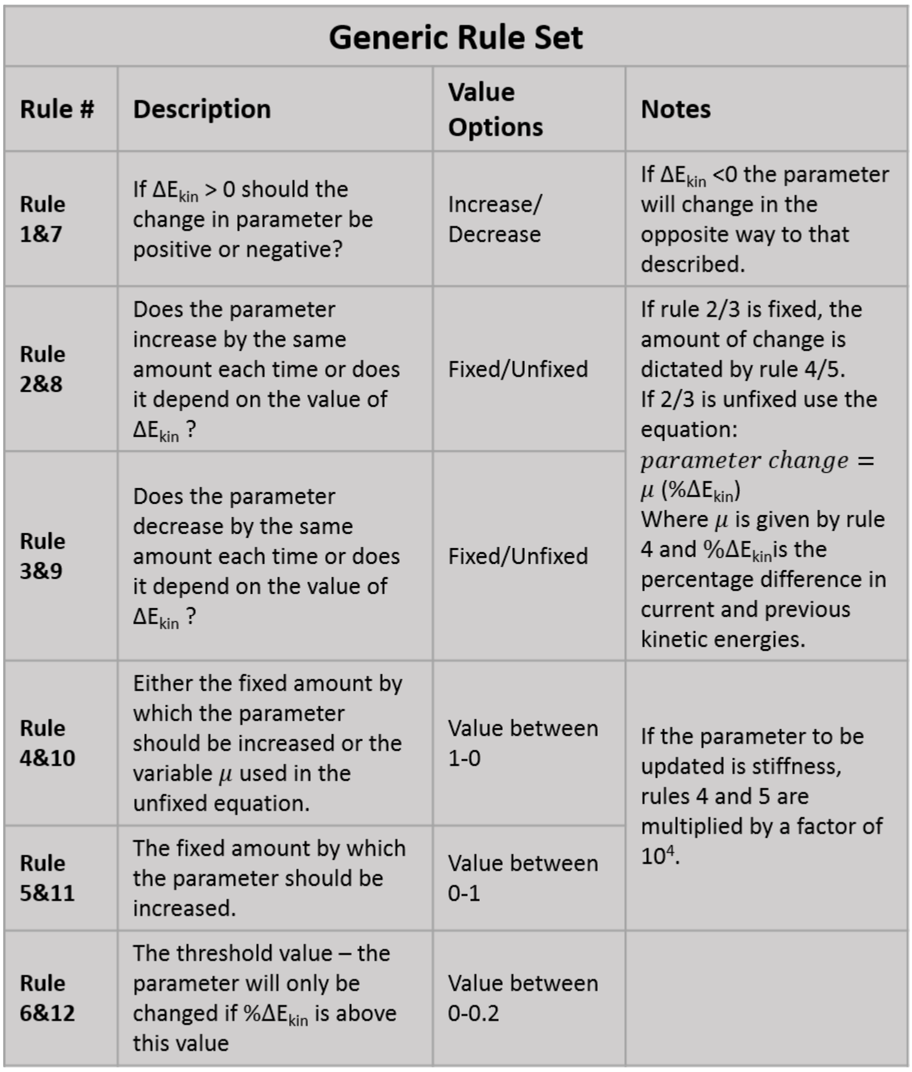
Mass (m)	80kg
Stiffness (k)	**Adaptable**
Attack Angle (α)	**Adaptable**
Length (l_o)	1m

y
x
m
k
l_o
α
α'
Use rule sets to change angle before next touch down.
Continue until the difference in kinetic energy at adjacent apex is negligible. Therefore the system is stable.
Compare kinetic energy at current apex with E_{kin} at previous.

Figure 3.6: Diagram shows how the basic SLIP model is adapted to incorporate online learning. Instead of the SLIP model being allowed to fail and the attack angle and stiffness being adjusted upon reattempt, these parameters are instead adjusted before the touch down depending on the rule set being followed. Instead of change in distance between episodes the rule sets are now based on changed in kinetic energy ΔKE between consecutive apex of the flight phases.

The change in key measurement parameter (i.e. the change from using ΔD to ΔKE) is not the only change required from offline learning to an online learning approach; the rules need to be changed too. Table 3.7 shows the new rules updated for the online learning approach.

The main difference between these online rules and the offline rules, is the change in rules 2,3,8 and 9. Instead of using change in distance between episodes (ΔD), online rules use the change in kinetic energy (ΔKE) between consecutive flight apex. However, in addition, two new rules (numbers 6 and 12) have been added. These are referred to as the *threshold rules.* Any change to the system, i.e. adaption of stiffness or attack angle, will only occur if the percentage change of the kinetic energy (ΔKE) is higher than this threshold. This rule was added to avoid the



Generic Rule Set

Rule #	Description	Value Options	Notes
Rule 1&7	If $\Delta E_{kin} > 0$ should the change in parameter be positive or negative?	Increase/ Decrease	If ΔE_{kin} <0 the parameter will change in the opposite way to that described.
Rule 2&8	Does the parameter increase by the same amount each time or does it depend on the value of ΔE_{kin} ?	Fixed/Unfixed	If rule 2/3 is fixed, the amount of change is dictated by rule 4/5. If 2/3 is unfixed use the equation: $parameter\ change = \mu\ (\%\Delta E_{kin})$ Where μ is given by rule 4 and $\%\Delta E_{kin}$ is the percentage difference in current and previous kinetic energies.
Rule 3&9	Does the parameter decrease by the same amount each time or does it depend on the value of ΔE_{kin} ?	Fixed/Unfixed	
Rule 4&10	Either the fixed amount by which the parameter should be increased or the variable μ used in the unfixed equation.	Value between 1-0	If the parameter to be updated is stiffness, rules 4 and 5 are multiplied by a factor of 10^4.
Rule 5&11	The fixed amount by which the parameter should be increased.	Value between 0-1	
Rule 6&12	The threshold value – the parameter will only be changed if $\%\Delta E_{kin}$ is above this value	Value between 0-0.2	

Figure 3.7: Table shows the online learning rules. The rule sets are similar to the those used in the offline learning although slightly adapted to allow for online learning. The first 6 rule sets relate to the adaption of the attack angle, the second 6 about the adaption of the stiffness. Further details about the online rules can be seen in the main text.

scenario where the SLIP model is started with, or has reached, stable parameters, but still tries to update. Without this threshold value, the SLIP model would be constantly updating itself from one set of stable parameters to another which would not be energy efficient. The maximum threshold value for both rules 6 and 12 was selected to be 0.02. This was chosen after initial testing that showed a high deterioration in success with thresholds higher than this value.

The same evolutionary algorithm was used for online learning as in offline learning, i.e. the same number of generations, initial population and crossover/mutation rates. However, as in the offline testing the *mean recovery time* proved to be the better fitness metric in this case *percentage success rate* was not used.

The definition of the *mean recovery time* was updated from the one used in the offline rule set testing, which used number of episodes to stability to determine rule set success. As the concept of episodes no longer exists in online learning, instead the mean number of seconds it took for a rule set to find a stable solution (i.e., for ΔKE to be below the specified threshold) for all tested starting parameters was used.

To clarify further, an example of a successful online rule set is shown in Figure 3.8. In this case the threshold was a percentage change in kinetic energy of 1% (rather than the 0.2 actually used in testing) for clarity. Initially the change in kinetic energy varies from stride to stride, as a result the parameters are updated. This continues until approximately 31 seconds (shown by the red line) at which the parameters now stable. The variation of kinetic energy between strides is below the threshold value (see the red circle). The attack angle (α) and stiffness (k) are no longer changed. This 31 seconds is the *time to stability* for this rule set at this starting parameter combination.

3.5 Online Results

In this section, the results from **online learning** are presented. As before with the offline rule sets, the evolutionary algorithm was run three times with three different randomly initialized starting populations using the fitness criteria of **mean time to stability**. The best rule sets from each of these three evolution set runs are show in Table 6. Whilst these rule sets were evolved for *mean time to stability*, the *percentage success rate* was also calculated to allow for comparison, but was not included in the algorithm as a fitness metric. Additionally, Figure 3.9 shows each of the starting parameters for which a stable solution could be found for each of the best rule sets. In Figure 3.9 the different colours represent the time to stability. The darker the colour, the quicker it reached stability. Areas of grey represent areas where the starting combination of parameters could not find stability with a given rule set.

It can be seen that the top three rule sets all alter the region of stability in the same

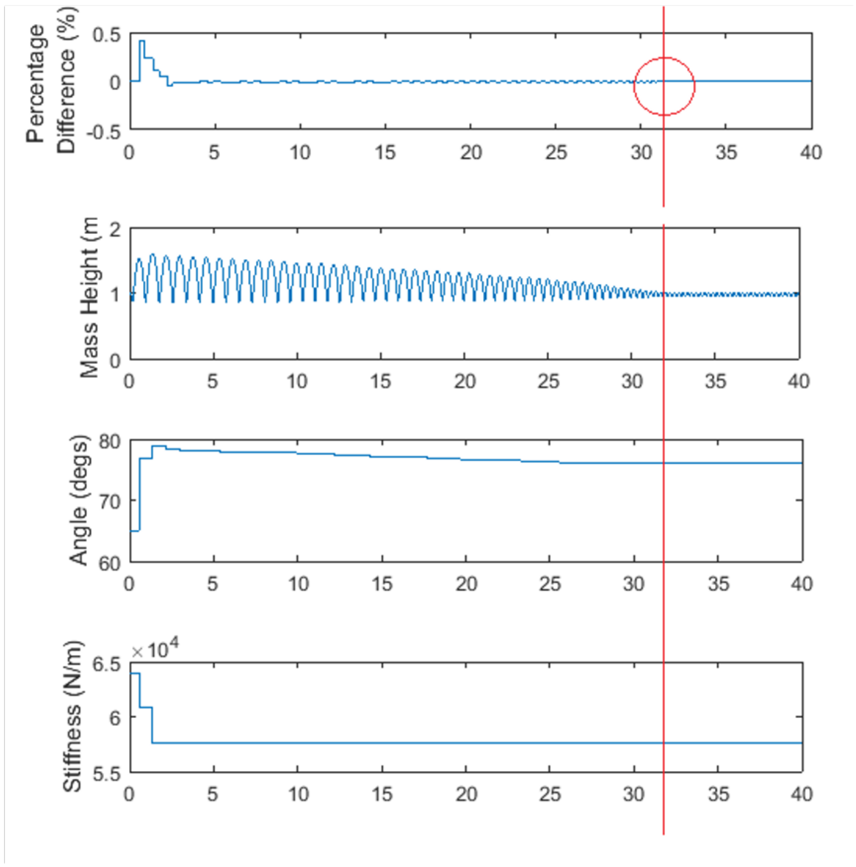
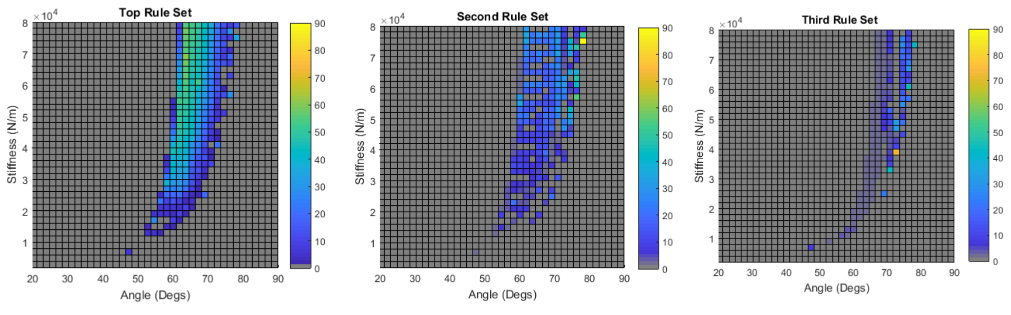


Figure 3.8: Showing time to stability for an online learning SLIP model with stable starting parameters. The top graph shows the percentage difference between consecutive flight apex. Once this percentage difference is below the threshold (in this case 0.01%) the angle and stiffness become fixed. This can be seen in the 3rd and 4th graphs in this figure which show the change in stiffness and angle over time. Also shown in the second graph is the height of the point mass during the learning process – here one can see when the model is in flight phase and when it is in stance phase.



(a) (b) (c)

Figure 3.9: Sub-figures a,b and c show the results from each of the online evolution runs. A grey square represents a starting parameter combination where stability could not be found. The lighter the square the longer the model takes to find stability at that starting parameter.The rule sets for each of these results are shown in Table 6. For comparison to SLIP model without the ability to adapt, refer to Figure 3.2

	Top Rule Sets		
Set Number	Top Rule Set	Second Rule Set	Third Rule Set
Rule 1	Decrease	Decrease	Decrease
Rule 2	Unfixed	Unfixed	Unfixed
Rule 3	Unfixed	Unfixed	Unfixed
Rule 4	0.2	0.41	0.35
Rule 5	0.23	0.24	0.56
Rule 6	0.014	0.025	0.009
Rule 7	Decrease	Decrease	Increase
Rule 8	Fixed	Fixed	Unfixed
Rule 9	Fixed	Fixed	Unfixed
Rule 10	0.32	0.54	0.52
Rule 11	0.49	0.2	0.83
Rule 12	0.008	0.03	0.019
Mean Time to Stability	83	82	85
% Success Rate	20%	20%	15%
STD Time to stability	30	30	31

Table 6: Table showing the results from the three online evolution runs. Each run was started with a random initial population and evolved to minimise the mean time to stability. The percentage success rate of each of the top genomes was calculated after the evolution. Note that for a basic SLIP model (i.e., one without the capability to learn) has a mean time to stability of 94 seconds and a percentage success rate of 3.75% (rounded to 4%). Also included the standard deviation of the time to stability

way, i.e. through expanding the region to the left hand side of the curve. None of the found rule sets are able to expand the region of stability for higher angle values (i.e. the right hand side of the curve). It can also been seen that these rule sets reach stability in a shorter time if the angle is low, compared with starting parameter configurations that are close to the standard “J” curve.

It should also be noted that all three top rule sets have very similar angle changing rules. All the rule sets aim to change the angle opposite to the change in kinetic energy by amounts proportional to the change. Rules 4,5,6 also have similar values across all three top rule sets. Similarly, the stiffness changing rule also has good consistency between the top two results. However, in the third rule set the first three rules are the opposite to the other top rule sets. This could explain why the mean time to stability for this third rule set is slightly longer and the percentage success rate is lower.

Additionally note that the evolutionary algorithm was only repeated three times, i.e., with three different random starting populations. Although this is a limitation, more repeated runs would yield more statistically convincing results, the standard deviation of the three repeats is small ($\mu = 1.57$). Given this is the case, three repeated runs is more acceptable.

3.6 Investigating Instability Regions

To understand the dynamics of the best evolved rule sets better, the unsuccessful regions were investigated. These regions are shown in grey in Figure 3.9 and are where the model could not find a stable solution. These regions include particularly low/high angles and the regions of instability in the number 2 rule set were investigated (for example, see Figure 3.10). In the case of starting parameters with combinations of high angles and low stiffness, failure before any adaption takes place — i.e. it falls over on the first touch down and therefore does not have time to adapt to be stable. In contrast, starting parameter combinations with low angles, fail on second touch down. Although in theory there is enough time for the model to adapt, there is such a large percentage difference in kinetic energy between the two apexes that the angle is also changed by such a large amount that is outside the range of acceptable attack angles. Therefore, the system fails. The changes in attack angle and stiffness for the unstable grey areas (Figure 3.9c) are shown in Figure 3.8. Three different unstable starting configurations, shown by the different coloured diamonds were explored by using the number two rule set. Three different starting configurations were selected to test whether there were any patterns in how the parameters were updated between the three different positions. Interestingly, when following this rule set, the pattern of parameter changes are the same for all three. Initially the attack angle is increased to an angle just higher than within the stable range and the stiffness is kept relatively similar. Then the stiffness in each case is gradually decreased, whilst the attack angle oscillates between attack angle values that are just larger than the region of stability shown in Figure 3.9a. Eventually the model simply fails.

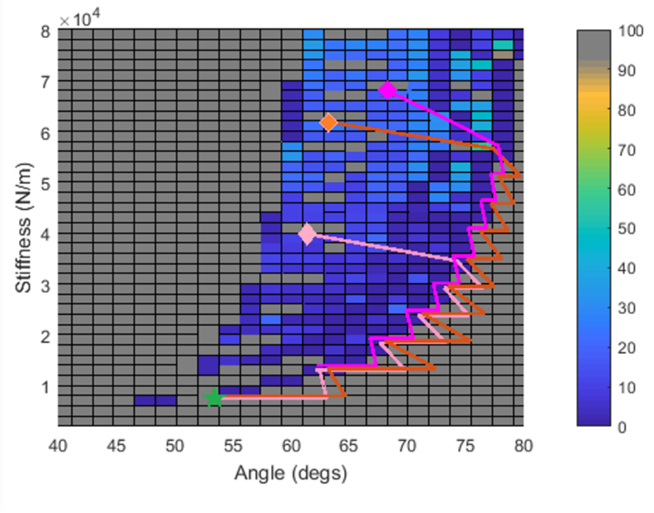
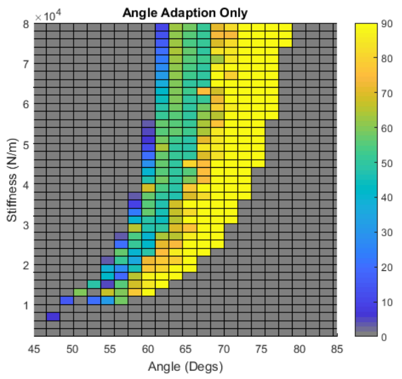


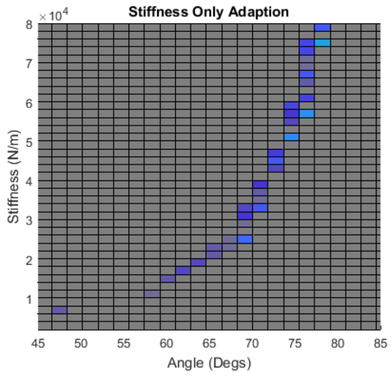
Figure 3.10: Plot showing how the attack angle and stiffness change, whilst using rule 2, during the online learning process for three different unstable starting conditions (the gray areas). The coloured diamonds show the starting parameter configurations and the green star shows the final configuration. It is at this configuration that the SLIP model has fallen over (i.e., $y < 0$).

For the top rule set (Table 6) the two parts of the rule set (i.e., the first 6 angle changing rules and 6 stiffness changing rules) were tested separately. This was to investigate how the interplay between the attack angle changing and the stiffness changing affected the overall success of the rule set. First, only the angle was allowed to adapt, following the first 6 rules from the top rule set. The stiffness remained constant and fixed throughout this simulation. Secondly, it was the attack angle that remained fixed and the stiffness was allowed to adapt follow the last 6 rules from the top rule set. Figure 3.11 shows the two mean time to stability plots for these individually tested rules. Interestingly, it can be seen that the angle adaption rule maintains the high percentage success rate seen in the top combined rule, i.e. it is still able to find stability for a wide range of starting parameters. The stiffness adaption is much less successful when on its own. However, it appears that the stiffness adaption is needed for the model to settle quickly. When only the angle is adapting the *mean time to stability* is much higher, in some cases, whilst not falling over, not finding stability at all. The change in kinetic energy never falls below the percentage threshold. This finding is consistent with the other two top rule sets (i.e., both the other top rule sets maintain a high percentage success rate when only adapting the attack angle but show a much higher mean time to stability then when also adapting stiffness).

This is explored further in Figure 3.12 showing the top rule set. On the top graph the stiffness adaption part is not included. The model is started in three different starting configurations. For all configurations, initially the angle changes a lot, but then the changes are small until stability is reached. However, on the bottom graph



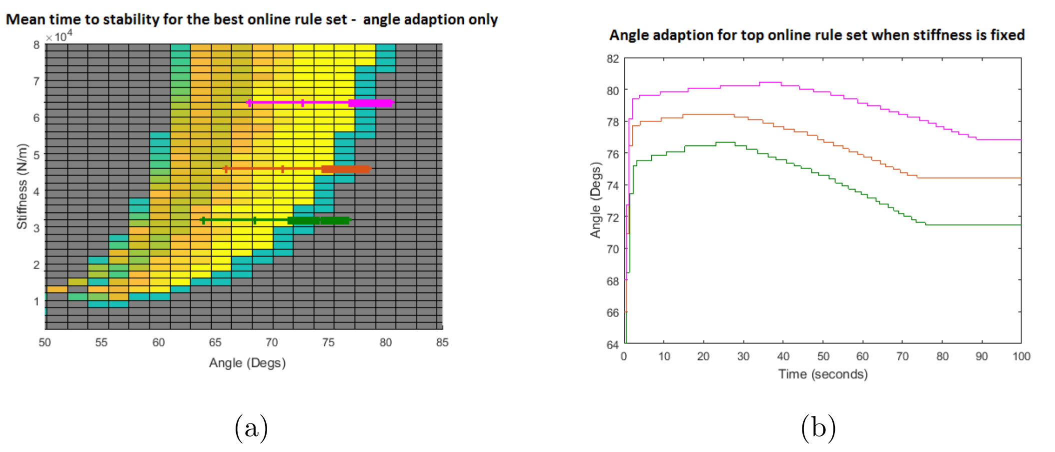
(a) This graph shows the behaviour of the model following the first six rules of the best online adaption rule set, i.e., only the angle was allowed to change and the stiffness remains fixed and constant.



(b) This graph shows the behaviour of the model following the second six rules of the best online adaption rule set, i.e., only the stiffness was allowed to change and the angle remains fixed and constant.

Figure 3.11: Diagram showing the *mean time to stability* plots for the two individual parts of the top rule set, i.e. just adapting the angle and just adapting the stiffness. As before a grey square represents a starting parameter combination for which a stable solution could not be found. For the coloured squares the lighter the colour the longer the model takes to find a stable solution.

the stiffness adaption is included - this small change in stiffness allows the model to find stability much quicker.



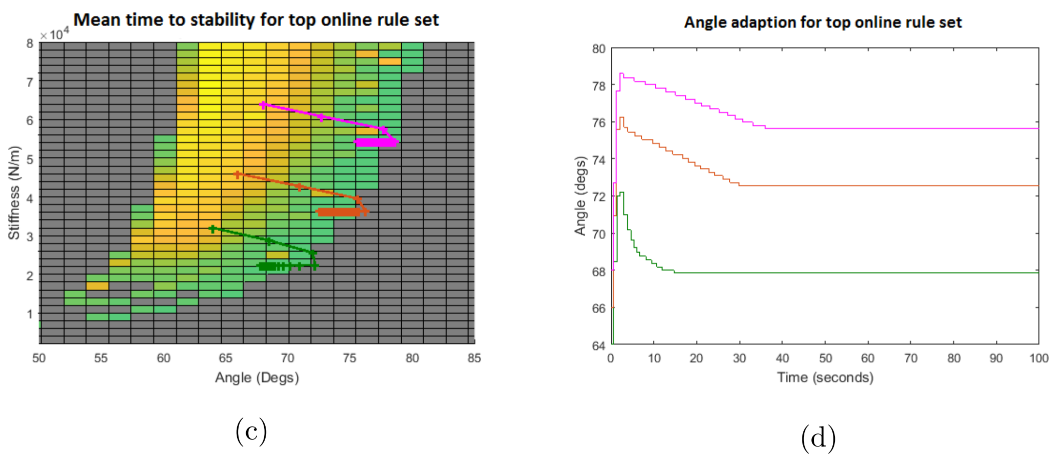


Figure 3.12: This figure shows how adapting both stiffness and attack angle affects the time to stability for the top online rule set (see Table 6). The top two graphs (a and b) show the how the model adapts when the stiffness is fixed for three different starting parameter combinations. The bottom two graphs (c and d) show the same starting parameters combinations following the same rule set but this time the stiffness is also able to adapt.

3.7 Investigating Environmental Change

To further investigate how robust these learning rule sets are, they were tested in a further challenge. The model is now required to maintain stability when then ground level is decreased (via a step).

Figure 3.13 illustrates how online learning rather than offline learning is necessary to deal with changes in environment. In Figure 3.13a the top offline learning method (see Figure 5), tested at a single starting configuration (26000N/m, 50 degs, selected as it was in the center of the stable online region), attempts locomotion in the new "stepped" environment. In this case the ground level is decreased by 2.5m (x2.5

greater than the resting length of the spring of 1m). In Figure 3.13b the same landscape is attempted by the top online learning rule set (Table 6), using the same starting parameters. In the case of the online rule set this stability is found almost immediately after approximately 3 seconds. In contrast the offline rule set takes a larger number of episodes, where it fails after each, in order to update its parameters so that they are stable. Whilst the online learning method is able to immediately react to the decrease in ground level and adapt its parameters without losing stability, the offline learning method the SLIP model fails almost instantly when it reaches the step. It is reset with updated parameters and another attempt is made, but it always fails when reaching this step. It can be seen from the top graph (Figure 3.13) that offline learning has numerous attempts at overcoming the step, shown in different colours which represents different episodes, but given it is reset to the start ($x = 0m$) each time it is unable to maintain stability for the distance. In addition to this, it has failed many times.

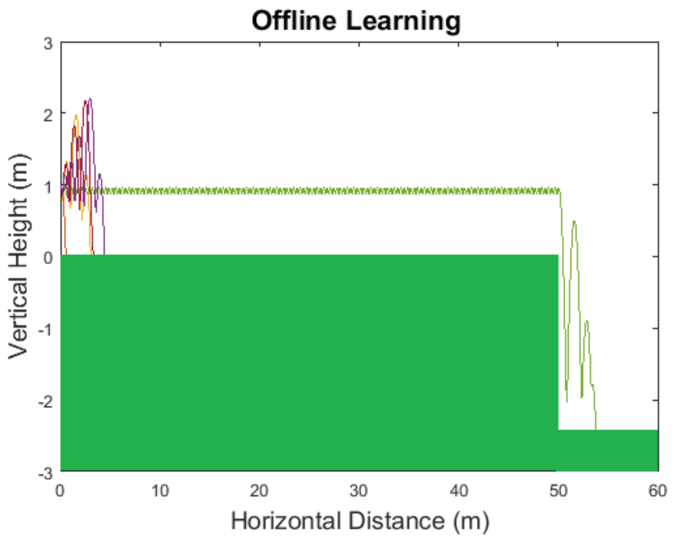
Note that when the mass moves down the step there is a loss of potential energy; the change in energy now causes the mass to not only travel further between strides, but also reach a higher height during the flight phase.

Whilst Figure 3.13 demonstrates the potential advantages of using the online rule set to maintain stability when encountering a change in ground level, it does not fully explore how robust these rule sets are. For example, what is the maximum drop in ground level for which the rule set can maintain stability.

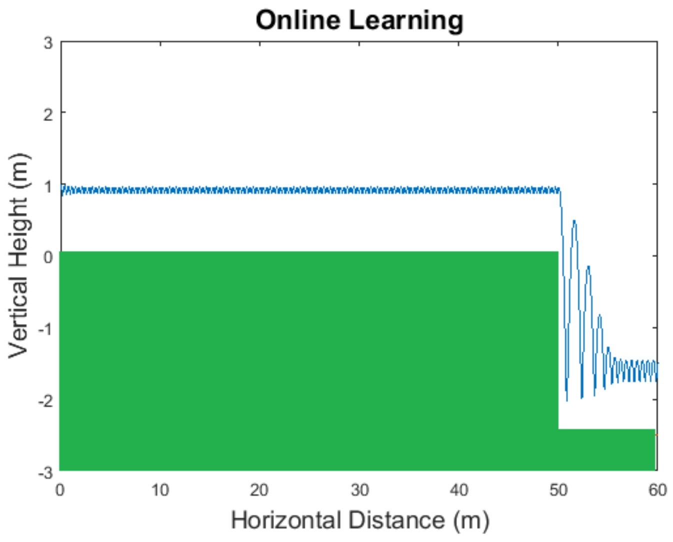
Therefore, to explore this aspect further, the success of the top three **online** rule sets were tested across a variety of step heights. As before each rule set was tested with 1600 different starting conditions and success was measured in terms of percentage success rate and mean time to stability. However in this case, the step height was introduced half way through the simulation landscape.

Figure 3.14 symbolises how the percentage success rate changes with step height for a basic SLIP model, one following the first evolved rule set and one following the 3rd best rule set. Rule set two was not included in the graph due to its similarity with rule set one. The downward step height was increased until the percentage success rate of the model was zero. Also tested was a small increase in ground level. Along the x-axis is the percentage success rate of the starting combinations for which the two rule sets (grey and orange) and the basic un-adapting SLIP model (blue). Along the y-axis is the step height that the SLIP model needs to overcome. It can be seen that for the non-adapting SLIP model that at a step height of zero, 3.75% of the starting combinations are stable. This percentage success rate decreases as downward step height increases, until the percentage success rate reaches zero at a step height of approximately -1m. In contrast, for the best rule set (shown in orange) the percentage success rate at zero step height is 20%. This decreases almost linearly with increase of downward step height until it becomes almost zero at a step height of -14m.

Figure 3.14 shows that a basic SLIP model without the capability to adapt can



(a) The offline learning approach dealing with the change in environment, i.e., a decrease in ground level height of 2.5m. Each of the different colours represent a different episode, all starting from the same position but with different stiffness and attack angle values. The model fails many times, including when it encounters the change in ground level. In this instance the staring attack angle is set to 40 degrees and the starting stiffness was set to 26000N/m



(b) This graph shows when the SLIP model uses the online learning approach to attempt to maintain stability when encountering the change in ground level. In one attempt the online learning is able to overcome this downwards step and maintain stability throughout the simulation. The starting attack angle was set to 40 degrees and the stiffness was set to 26000N/m.

Figure 3.13: Comparison of offline and online learning approaches when faced with a change in ground level

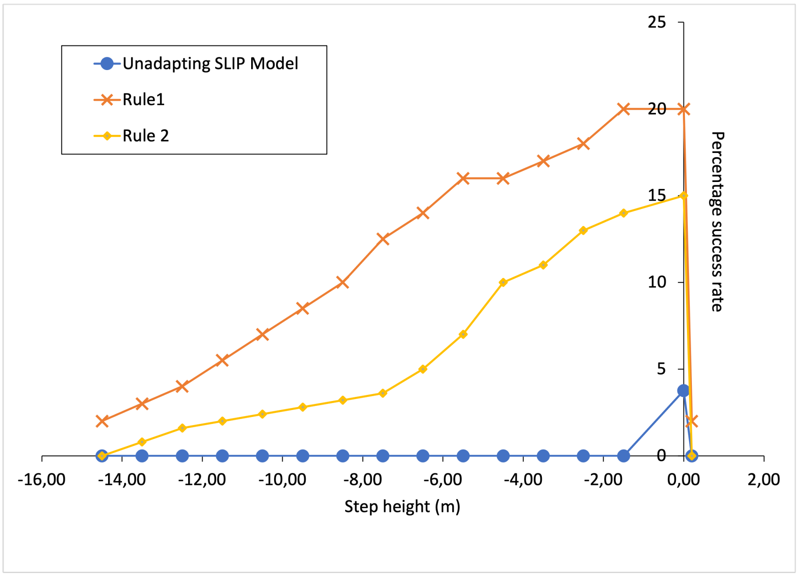


Figure 3.14: Graph showing how the percentage success rate varies with step height for two online update rules and the basic SLIP model (one that is not able to adapt).

only withstand a very small change in environment. At a step size of only -1m the percentage success rate of the basic SLIP model reduces to zero, i.e., there is no starting parameter combination for which the basic SLIP model can maintain stability when the ground level decreases by 0.5m. This basic SLIP model data is shown by the blue line in Figure 3.14. When following the optimal online rule set, the model is able to withstand a downwards step height of up to 2.5m without any changes to the percentage success rate when compared with a level ground (i.e. a step change of zero). This is shown by the orange line in Figure 3.14. The top online rule set is that at some starting conditions is it also able to withstand a decreasing step of up to 14m. It is also able to withstand increases in step heights up to 0.3m. Figure 3.14 also shows that the percentage success rate of the both the top and second best online rule set decreases as the step height increases. It is interesting to see that the shape of the successful starting combinations change as the ground level change increases. Therefore, stable starting parameters for the different step levels are shown in Figure 3.15. From Figure 3.15 it can be seen that as the change in ground level increases (i.e., the downwards step size increases) the amount of stable starting parameters decreases and favours higher stiffness values. However, for a given step height, if starting at a lower stiffness it is possible to quickly find a stable solution (i.e. it will need less time to become stable). A point of interest that arose from the successful starting combination plot (Figure 3.15) is shown in Figure 3.16. The SLIP model is started at a successful combination of attack angle and stiffness. How these parameters then change between strides are are shown in red. First the model settles in the curve of the J-Figure of the basic SLIP model (see Figure 3.2). This is as expected, the SLIP model is becoming stable before any change in ground level. However, when the ground level changes the angle changes again, this time settling in a very different area, shown by the red star. Although it is speculated that this could be due to the increase in energy in the system due to the step and exploration of this could be part of future work.

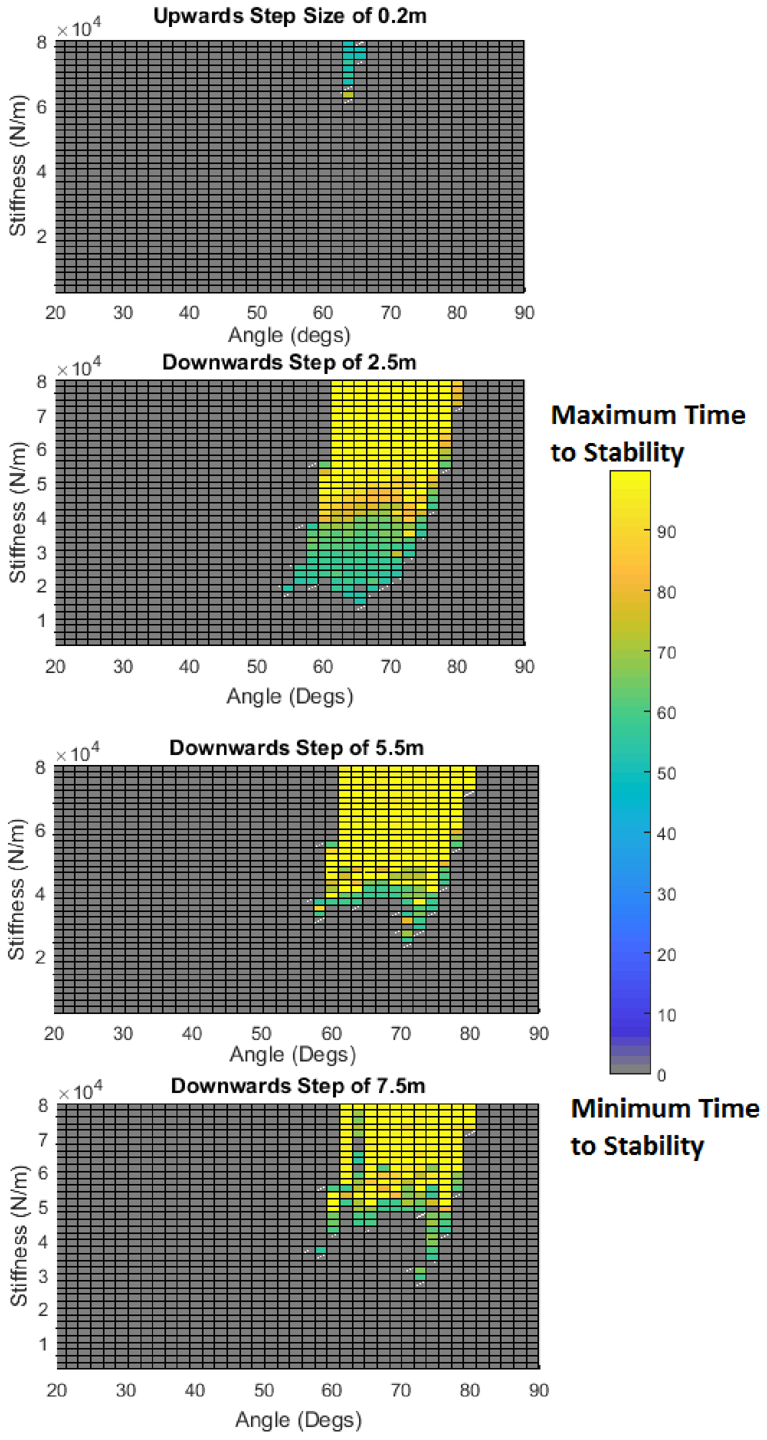


Figure 3.15: Graph showing how the mean stability time varies with step height for the top online update rules. Note that the increase in settling time (when compared to Figure 3.9) is due to the addition of the step at 50 seconds).

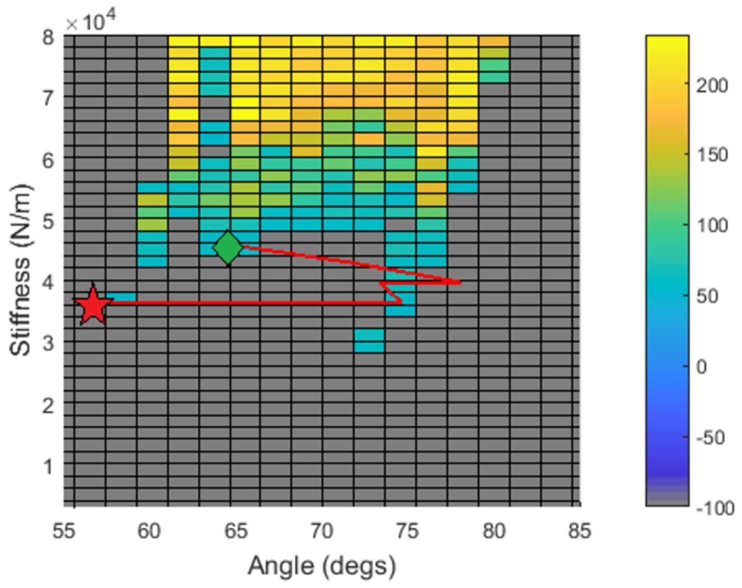


Figure 3.16: Graph showing how the attack angle/stiffness change at a particular starting configuration, when the model encounters a down step of 7.5m. The green diamond shows the starting position and the red stars shows the finishing configuration.

3.8 Chapter Conclusions

The results in this chapter build upon those presented in the prior chapter relating to the offline learning of the SLIP model. In this chapter, I use evolutionary algorithms to evolve rule sets capable of simultaneously adapting the spring stiffness and attack angle in order to find stability. In the offline learning approach the SLIP model is allowed to fall over (fail) and the adaption takes place before a locomotion reattempt. In the prior chapter, rule sets that are capable of adapting either the attack angle or stiffness were systematically tested for success. Then to test dual adaption the best individual rule sets were combined. This approach yielded combined rule sets that were able to find stability for a percentage success rate of 79%. In this chapter, however, when I used an evolutionary algorithm, I found an optimal dual adaption rule set that was able to find stability for 99% of the starting combinations. As discussed in the previous chapter being able to find stability from a wide range of starting parameters is useful, especially if these methods were to be transferred to a physical robot. For example, if the robot experience wear during its use, it would be able to re-adapt its attack angle or stiffness to find stability. However, there are other ways that may also be useful to test the success of a rule set. In this chapter, I explore using mean time to recovery (offline)/ mean time to stability (online) as a fitness metric. By just measuring the success of a rule set by the percentage success rate a situation could arise where the attack angle or stiffness was being adapted for a large number of episodes before finding stability. Not only would this require a robot to fail many times, which could be a problem for a physical system, it would also require a vast amount of energy – also a potential problem. Optimising for mean time to recovery/stability instead reduces these potential issues.

Interestingly, this chapter shows that when evolving for mean recovery time/mean time to stability, the optimal rule sets obtained also have larger percentage success rates (equivalent to the when percentage success rate is used as the fitness metric). This could suggest that the fitness landscape when evolving for percentage success rate is easy to navigate. It is relatively easy to find successful rule sets. This can be seen by the percentage success rate generational fitness graphs (Figure 3.5) which show a quick convergence. This hypothesis is reinforced by the fact that the top offline rule set (when evolved for percentage success rate) all have a different genetic structure, i.e., the individual rules that make up the optimal rule sets are dissimilar. In contrast, it would appear that there are fewer areas of optimal solutions in the *mean time to recovery* fitness landscape. In this case the optimal rule sets all have a very similar genetic structure and the evolutionary algorithms take a larger time to converge when compared to when using the *percentage success rate* as the fitness metric. On the other hand it could be because the *mean time to recovery* is indirectly affected by the percentage success rate. For example, if a rule set has a low percentage success rate it means that many of the starting combinations are not able to achieve stability and therefore are given maximum recovery time (100 seconds). However, if more starting parameter combinations are able to find stability there is a strong possibility that they are able to find this stability in a shorter amount of time than 100 seconds, reducing the *mean time to recovery*, i.e., there is a correlation between the *percentage success rate* and the *mean time to recovery*.

A problem outlined in the discussion in the previous chapter, i.e., the requirement to fail before stability can be found, is also addressed in this chapter. The offline rule sets are adapted to allow for online learning which are also evolved using an evolutionary algorithm. This means that, when following these online rule sets the SLIP model is able to adapt to changes to the environment without the need to fall over and restart. The optimal online rule sets from the testing are able to increase the percentage success rate by approximately 15%. Additionally, at some starting parameters the SLIP model is also able to withstand changes in ground levels of up to 14m meters (step decrease).

An interesting observation made in this chapter is how the adaption of the two parameters affect the behaviour of the SLIP model. When the stiffness is fixed, the top online rule set is still able to find stable conditions for the same amount of starting parameters as when both parameters as adapted. However, the mean time to stability is very long without the stiffness adaption, i.e., the stiffness adaption is necessary for a speedy recovery. Although not specifically tested for, there are suggestions of this finding also in the results of the offline testing. When only evolving for percentage success rate the stiffness rule sets are less important – this could be why there is little similarity between the stiffness rules of the top sets and good similarity between the angle rules. However, when evolving for mean time to stability, adaption of the stiffness is more important.

Although I do adapt both the morphology (i.e., the stiffness) and control (attack angle) at the same time in these experiments, they are still kept relatively separate. In the introduction section, I described how the closely linked morphology and control are, and the need for a suitable morphology for a particular control system. By changing the attack angle and stiffness together expands the percentage of starting combinations for which the model can become stable. I initially expected that dual adaption may take longer to find a solution that only adapting one parameter. Both parameters need to be optimised, but changing one parameter will have a strong influence on the other. However, based on these experiments it appears that adapting the stiffness somewhat independently but at the same time actually makes find the solution quicker.

As previously discussed throughout the chapter, a limitation of the results presented is the weak design of the evolutionary algorithms. This does not appear to be a problem for the offline learning approach, where especially the percentage success rate is high (only a few starting combinations at the extremes of the parameter ranges are not able to find stability.) Given this success from the offline learning experiments, the evolutionary algorithm was not updated for the online experiments. It is therefore likely that using a better performing algorithm would yield even lower mean time to stability values.

Additionally, the choice of kinetic energy as a key parameter is perhaps a limitation in this work. Although using kinetic energy has been shown to work, and achieve the aim of expanding the region of stability, there may be better key parameters to use. For example, it is likely that equally good results could be achieved by using horizontal velocity or apex height. Kinetic energy was selected, somewhat arbitrarily, early on in the experiment design but because it yield good results, careful consideration as to its benefits over other parameters was not carried out. Furthermore, although not a problem in simulation, other parameters such a horizontal velocity may be easier to measure if required to be implemented in a real robot.

Another limitation of the online SLIP model is that it is only able to withstand a decrease in ground level. When the SLIP decreases in height the system converts potential energy. This increase in energy appears to cause the SLIP to increase the height reached during the flight phase (see Figure 3.13). In contrast, the SLIP model would require energy in order to increase its vertical position, i.e., if the ground level was to increase. This may imply that to be able to withstand increases in ground level the height of the pendulum during flight phase would have to decrease to allow for this raise in ground level. There is only so much the height during the flight phase could be decreased by and the system currently has no other way to combat the decrease in energy. Therefore, to be able to maintain stability when the ground level increases the SLIP model would require a way of adding energy to the system.

Another interesting way to extend this model further is to consider adding damping to the system and allowing this parameter to be adapted also. Alternatively, the model could use a non-linear spring where the parameters of the equation describing the spring could be adapted. These additions to the system could expand the region of stability even further.

Additionally, evolving offline and online rule sets together could increase performance even further. Initially the SLIP model would rely on the online learning approach; using this approach the model would be able to counteract sudden changes in the environment such as the downwards steps. However, if the perturbations became too large the offline rule sets could adapt the parameters back into a region where the online rules were sufficient. This would combine the benefits of both approaches – the large percentage success rate from the offline approach whilst reducing the need for failure.

In conclusion, the results from this chapter and the previous show the benefits of evolving the **way** of adapting both control and morphological parameters in the SLIP model, instead of optimizing fixed parameters. Using both offline and online approaches the SLIP model has been shown to have increased robustness, compared with a system that has no ability to adapt.

An important finding from these chapters is that is a benefit to either only adapting the morphology of the system, or adapting in combination with the control. As discussed in the introduction section, traditionally, if further robustness is required it is the control system that is changed. In these two chapters it has been shown that for offline learning, adapting the only morphology of the SLIP model can increase the robustness as well. Also, in the case of online learning, adaption of the spring stiffness is crucial for a low mean time to stability.

However, the SLIP model is simple and theoretical. Therefore, in the next chapter I investigate how well the concept of evolving methods of morphological adaption works in a more complex system. Additionally, I use this framework to further investigation how a robot can learn from past experience to become successful in a wider range of situations and tasks.

4 Evolving morphological adaption mechanisms for in voxel based soft robots

4.1 Introduction

In the previous two chapters, the advantages of evolutionary algorithms to evolve simple rules which dictate how a Spring Loaded Inverted Pendulum should adapt its morphology (spring stiffness) and control system (attack angle) were investigated. Using both offline and online adaption the optimal rule sets were able to expand the range of tested starting parameters for which the SLIP model could learn to become stable from (to 1520 out of 1600 of the for the offline learning and 320 out of 1600 for the online learning, from the baseline value of 48 out of 1600). Furthermore, using the online learning approach the optimal rule set was able to adapt the SLIP model to overcome large changes in the environment, such as sudden decreases in ground level.

For the next part of the thesis, I tested how a similar methodology, i.e., evolution of adaption rules can be employed, on a less abstract, more complex robot. Moreover, in this chapter I aim to find a set of adaption rules that allows a robot to adapt to a variety of distinct environments. A key aim of this thesis is to explore the possibility of creating robots capable of adapting their morphology to new environments. Ideally the robot could be introduced to one environment and follow a single set of adaption rules to change their morphology to suit that environment. Then, if the environment were to change, the robot could follow the same adaption rules to reconfigure itself to the new environment. The two morphologies would not necessarily be the same, the adaption rules would rely on feedback from the environment and given the environments are different the feedback would also be different. However, crucially the underlying adaption rule which the robot follows in both environments would be the same. Whilst in the previous chapter the concept of adapting to an environment was somewhat explored – I investigated how the SLIP would could adapt to decreases in ground level – in this chapter there is a stronger focus on environmentally transferable adaption rules which generate morphological change in different environments. Therefore, whereas in the SLIP model chapters an optimal rule set one was able to learn stability, based on interaction with the environment, for the widest range of starting conditions, here an optimal adaption system is one that can successfully adapt the same initial robot to locomote the fastest in a number of different environments.

In this chapter the robot has an initial morphology made up of a number of individual soft modules (i.e., voxels) that are arranged into a cube. This is more complex as in the previous chapter the SLIP model was formed of only one linear spring, whereas now each voxel has its own stiffness. More detail into the initial morphology of the robot is described in the methodology section (Section 4.2. The robot has a set of voxels that are able to expand and contract which provides the actuation required for the whole robot to move, these are located at the center of the robot. The rest of the voxels (i.e., the outer voxels) are passive and therefore do not expand and contract. Similar to the offline approach described in the second chapter, the robot interacts with the environment over a number of episodes. After each episode the robot is adapted based on the physical feedback from these interactions and the overall adaption rules and thus the stiffness of the individual voxels are changed. If the stiffness of the voxel is changed below a certain threshold that voxel is removed. Thus, the morphological adaptions takes the form of sculpting. Note that, similar to the previous chapter, it is the adaption rules that are evolved rather than the morphology specific to the environment. Thus the aim of this chapter is to find an optimal, transferable adaption rule which use the robots interaction with the environment to create successful morphologies in a diverse range of environment.

The structure of this chapter is as follows: first presented is the methodology used to evolve the adaption control system, i.e., the way the robot uses the information from its interaction with the environment to change the properties of each voxel and when to remove the voxel entirely. This section includes the design of the initial robot, the simulation environments and the basis of the overall adaption control system and the evolutionary algorithm. Optimal neural networks are first evolved in individual environments, they are then tested in other environments (for which they were not evolved in) to determine transferability. These results are then compared to when the neural networks are evolved in three unique environments. Also included in this chapter is a comparison of adapting just the stiffness of the robot compared with removing voxels. Following this is an exploration into how the neural networks converge, and also whether there is any benefit to evolving first in one environment and then transferring to a new one. Finally, a section on using the evolved neural networks for damage recovery concludes this chapter.

4.2 Methodology

The starting structure of the initial robot was made from 216 modules (or voxels) arranged into a cube with the dimensions 6x6x6 voxels, as shown in Figure 4.1. Each module has a volume of 0.125m^3 and a starting stiffness of 5,000N/m. The density of each module was set to 1200kg/m^3.

To simulate the robot, the soft-body physics simulator Voxelyze was used [44]. In Voxelyze, a module takes the form of a 3D cube called a voxel; adjacent voxels are connected together by simulated beams. VoxCad creates actuation within these soft robots by employing a sinusoidally varying global control signal (termed temperature by the Voxelyze software). Active voxels expand and contract in phase with this sinusoidal temperature variation – additionally a phase shift can be added to an individual voxel.

Using this voxel expansion mechanism, a simple locomotion control signal was applied to the center of the robot, as shown in Figure 4.1. Specifically, a central sinusoidal control signal with a period of $T = 0.25$ seconds was applied to a set of

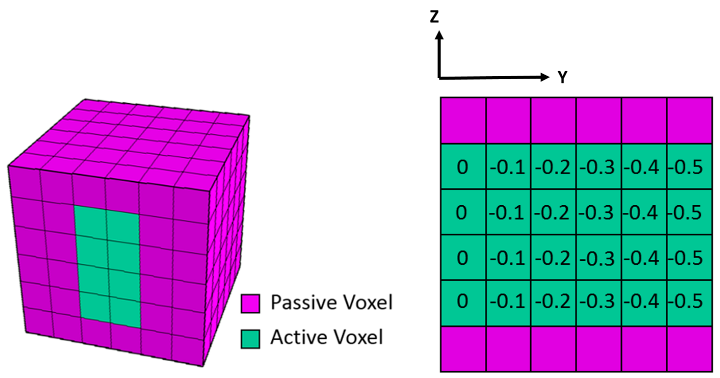


Figure 4.1: The initial morphology of the robot. The inner green voxels are able to expand/contract by 20% percent in time with the control signal (sinusoidal, with a period of 0.25 seconds). There is also a phase shift along the y-axis of the robot. The phase shift increases with each voxel, starting at 0 and ending at -0.5 rads.

specific, so-called active voxels allowing them to expand and contract by 20% (green voxels in Figure 4.1). In addition, an increasing phase shift from front to back (from 0 to -0.5 rads along the y-axis) has been added to break symmetry and to achieve locomotion (see Figure 4.1). Note that this control is fixed throughout the simulation. It's not part of the optimisation process. Also note that the magenta-colored voxels in Figure 4.1 are passive and, hence, are not driven by the central control signal.

It should also be noted that the size of the robot was selected to be large enough that there would be a variation in environmental feedback in between voxels, but small enough so that simulation times were a manageable length.

As previously discussed the adaption process in this chapter takes place episodically, i.e., the robot interacts with its environment for a number of fixed time periods (episodes, each episode lasts for 28 control signal periods) and adaption takes place between episodes. A diagram showing the overall adaption process of the robot and the method of adaption is shown in Figure 4.2. Whilst the fitness of the robot is taken to be the distance travelled in the final episode, it is still required to interact with the environment during every episode, not just the final one. This means that here the interaction with the environment and the corresponding feedback is key to morphological adaption.

Each episode lasted for 28 control cycle seconds. This episode length was chosen as it allowed for a number of voxel expansion cycles whilst still maintaining a short enough run time for efficient optimization. It also meant that the robot had a sufficient time to interact with the environment. The feedback from the environment was the kinetic energy KE_i^j of each individual voxel i which was recorded throughout each episode (j). Although other parameters were considered to characterise the robot's interaction with the environment, such as pressure or voxel strain, kinetic energy was selected not only because of its success in the previous chapter regarding online learning in the SLIP model but also because of its success in preliminary experiments. The kinetic energy of the *ith* voxel is calculated by the formula $KE_i = \frac{1}{2}m_i v_i^2$, where m_i is the mass and v_i is the resultant velocity of the ith voxel. The kinetic energy of each voxel is dependant not only on its own properties (mass) but also its environment, i.e., how constrained it is by other voxels and features of the external environment like the ground. For example, a voxel that is highly constrained will not be able to move as easily, its velocity and therefore its kinetic energy is likely to be low. Upon reflection, the choice of kinetic energy as the key parameter may have not been optimal. Indeed, in his work, Kriegman [62] uses stress/pressure as his key parameters. Therefore a potential limitation of this work is that I chose to use kinetic energy without rigorous scientific investigation. More discussion related to this is included in the conclusion section of this chapter.

In this chapter, the adaption rules, that dictate how the robot should change its morphology, is in the form of a simple neural network. The output of the neural network is how much the stiffness of a voxel should be changed. As it is the adaption rule that is optimised through the evolutionary algorithm, it is the weights of the neural network that are evolved (not the robot morphology). It is important to note that the **same neural network** (i.e., the same adaption rules) is applied to each passive voxel so that each voxel is updated every episode. The two inputs into the neural network are the kinetic energy for the voxel in question and the change in distance the entire robot traveled between the current jth and previous $(j-1)$th episode, ΔD (as shown in Figure 4.3). More specifically the average kinetic energy of the individual voxel over an episode and the average kinetic energy across the whole robot, also over the episode, is calculated. The difference in these two values is used as the input to the neural network. Therefore, each voxel at every episode receives both local information specific to itself (the kinetic energy) and global information regarding the entire robot (the distance travelled and kinetic energy of the entire robot). Using the global information, as well as the local information, as an input to the neural network allows the robot to adapt its morphology to changes locally depending on whether or not the previous adaption had been successful. To clarify, if after episode one, the robot adapts in a way that causes it to travel less far in episode two ΔD would be negative. Whereas if it travels further in episode two than in episode one ΔD would be positive.
As previously mentioned if the stiffness of a voxel is changed below a given threshold that voxel is removed. This threshold was set to 1kN/m after initial testing (the initial stiffness is 5kN/m). Initially, it was anticipated that the threshold would be zero, i.e., the stiffness could be continued to be reduced until it became 0N/m and the voxel “disappeared”. However, preliminary testing of this method showed that the Voxelyze software encountered fatal errors when simulating stiffness below 1kN/m. Therefore, this value was included as a threshold value, i.e., voxels were removed when decreased below 1kN/m. It should be noted that given the structure of the neural network “regrowth” could be possible. A voxel that has been removed has a kinetic energy of zero. However, there is the potential that if the global input (i.e., the absolute change in distance) was high the change in stiffness to the “empty space” could be enough to increase the voxel stiffness to above the threshold again.

However it was observed that this regrowth is never occurred.

The adaption process continues for 15 episodes; this is termed a lifetime. The length of lifetime was selected after initial testing as it allowed enough morphological adaption to occur whilst still keeping computational running costs low. As previously discussed the overall performance of the genetically encoded rules (i.e., the fitness) is the distance travelled in the final episode, i.e. $D_{j=15}$.

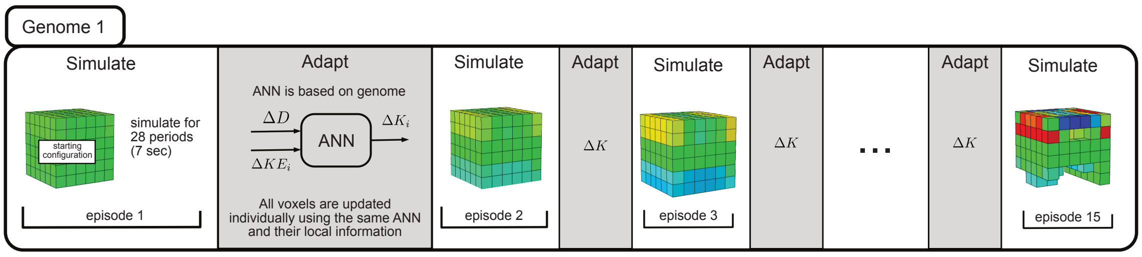


Figure 4.2: This figure shows the basic method of how a suitable morphology is sculpted out of the original cube, using an evolved sculpting adaptation system over time. After each episode j the total distance travelled D_j is recorded as well as the individual average kinetic energy KE_i^j in each voxel i. The change in stiffness of each voxel is calculated by inputting the above values *for each voxel* into the overall adaptation system (see ANN in Figure 4.3). Note that in this figure the different colours represent the different stiffness values of each voxel (red is more stiff and blue less stiff).

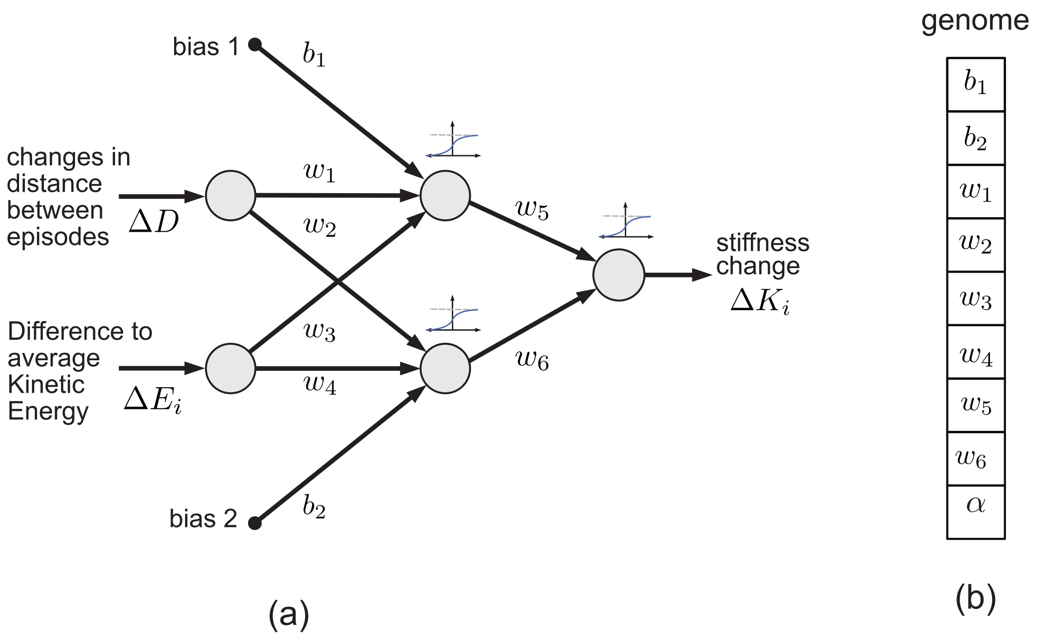


Figure 4.3: This image shows the neural network that implements the sculpting adaptation system which is responsible for how the robot adapts between episodes. The weights of the neural network, as well as the two bias values, are all optimized through the use of an evolutionary algorithm.

In the past few years, methods of evolving robot morphologies, has mainly utilised computational pattern producing networks (CPPNs). Based on neural networks, traditionally CPPNs determine a robots morphology by considering each co-ordinate point in the robots solution space (x,y,z) and outputting whether or not there is part of the body there. The activation function at each node is typically different and thus the output of the CPPN is symmetrical, complex and unique. However, in this thesis I chose instead to differentiate from the CPPN approaches. The design of my neural network was kept as simple as possible, with one hidden layer and a tanh function applied at each node. For each input node a bias was also added. The bias values were introduced so that, in the case of the two input values being zero, a non zero value would be outputted from the network. Although it has achieved the criteria of being simple, the design of the neural network could still be much improved. For example the addition of two bias values is not necessary; the same effect could be achieved with a single bias value. However, the neural network was designed early on in experiments, and as shown below, was able to yield good results. Therefore, it was not changed, although I anticipate that a more complex ANN or perhaps a CPPN approach would yield more complex and better performing results.

An initial population of 30 different randomised genomes was formed. As discussed before, per genome, the robot was first simulated as a complete cube of 216 voxels. Each voxel had a starting stiffness of 5,000N/m and the inner voxels were able to expand and contract, as described above, see Figure 4.1. After one episode (simulated 28 control cycles) the stiffness of each voxel was updated according to the neural network that was encoded in the genome. This was continued for 15 episodes, after which the distance in the final episode $D_{j=15}$ was recorded and used as a fitness measurement for the current genome. This process was repeated for all of the 30 randomised genomes in the initial population. Once the distance reached in the final episode was recorded for the entire population, the population was sorted accordingly to their performance, i.e., how far they have travelled in the final episode. A new population for the next generation was formed in the following way. The best two genomes in the current generation were kept in the population as they were and formed the first two genomes of the next generation. The next 18 genomes were formed by mutating successful genomes from the current generation. Specifically, a genome that was ranked in the top 50% was selected at random and four out of the total parameters of the genome were mutated, i.e. randomly changed to a different value. The final 10 of the new generation were created from completely new, randomised genes. This was continued for 150 generations. The parameters of the evolutionary algorithm were selected after initial testing and were deemed to be appropriate (large enough to allow convergence, but short enough to not be too computationally expensive.)

As mentioned in the introduction section, the aim was to find an optimal way of adaptation that works in a wide range of environments. It should be the case that a single adaption rule would be transferable between environments. Each rule neural network was based on feedback from the environment (in the form of kinetic energy), therefore it was unlikely but not infeasible that an adaption rule evolved in a single environment would also work in others.

Therefore, initially adaption in three different environments was investigated, the fitness of which was the distance travelled in the final episode in that single environment. The three environments were selected to be distinct, but also range in difficulty. Upon selection it was hypothesised that the Environment A (a flat horizontal plane) would be the simplest landscape and Environment B (an uphill slope of 15 degrees) a step up in complexity. The hypothesised most complex environment was selected to again be a slope of 15 degrees but this time the robot rotated so the line of actuation pointed across the slope and it was this direction that the robot was required to locomote. This environment (Environment C) was selected to investigate how the neural network would be affected by asymmetry either side of its central active voxel region. Figure 4.4 shows diagrams of all three environments.

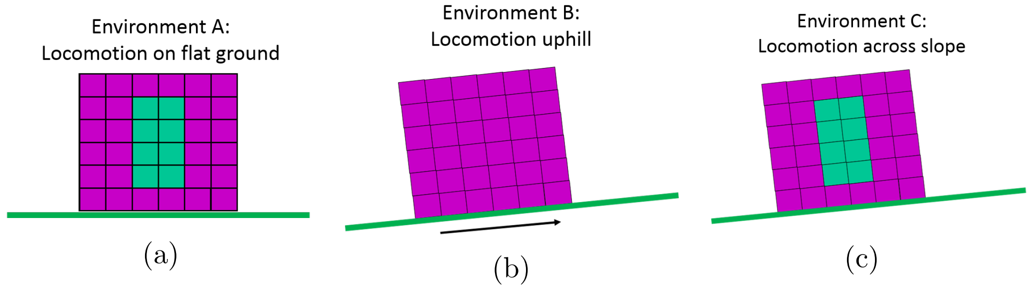


Figure 4.4: Figure showing the three investigated environments a) In Environment A the robot locomotes on a flat landscape. b) In Environment B the robot is required to travel up a slope of 15 degrees. c) In Environment C the slope remains at 15 degrees but now the robot is required to locomote across the slope .

At this point it is important to describe in more detail the fitness function for the robot. Initially the fitness was the distance traveled from a starting point in any direction. Whilst this is adequate for Environment A, this fitness function is too simple for Environment C. Preliminary testing in Environment C showed a well performing solution was for the robot to rotate and locomote downhill rather than across the slope. Therefore, the fitness function was updated to distance travelled in direction of actuation and candidates which moved in a different direction were penalised. An example of this is shown in Figure 4.5. This "direction in line of actuation" fitness was used in all three environment for consistency.

For each environment the evolutionary algorithm was run 10 times with different randomized initial populations.

4.3 Results

As discussed in the methodology section, for each environment the evolutionary algorithm was run 10 times with different randomized initial populations. Figure 4.6 shows the average generational fitness (average over the 10 evolution runs) for each of the three environments. To clarify, the fitness from the most successful genomes from each of the repeated evolution runs were averaged and plotted against generation number for each of the three different environments. From this graph it

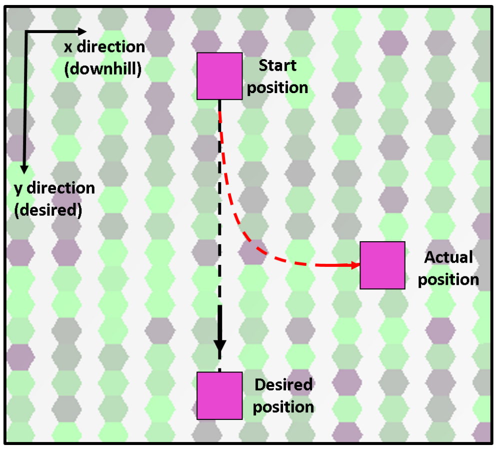


Figure 4.5: Figure showing a potential, undesirable behaviour that was witnessed in preliminary testing for robots attempting locomotion in Environment C

can be seen that the evolutionary algorithm converges quickest in Environment A, closely followed by Environment B (the point of convergence is marked by a line for each environment). However, it takes longer to converge in Environment C.

This in itself is interesting as when selecting the environments it was hypothesised that Environment A was the most simple and Environment C the most challenging for the robot to successfully locomote. This evolutionary fitness graph appears to confirm this.

Figure 4.9 shows the top genomes from each of the Environment A evolution runs and their corresponding final sculpted morphology. In the case of the neural nets (i.e., the genomes) red indicates a negative weighting whereas green indicates a positive weighting. For the final morphology red indicates a very stiff voxel, whereas blue indicates a soft voxel. For Environment A, the majority of the final morphologies have a very similar overall body shape – they have a stiff top front part and stiff bottom back part. The front bottom voxels have been removed giving the illusion of “hind legs”. However, given the similarities in the final morphologies, surprisingly, there is a significant amount of variation between the individual weights of the top genomes used to sculpt these morphologies. The final morphologies in this environment are very successful; an unsculpted robot is only able to locomote slightly backwards (-0.61 voxels) whereas these the top genomes sculpt robots able to travel forward by 54 voxels in a single episode. Additionally, Figure 4.10 shows the locomotion pattern of a typical final morphology, moving in Environment A.

In Figure 4.11 the successes of the top genomes from the environment A evolutionary algorithm are shown in blue (the white star further indicates that it is environment A for which these genomes were evolved). Also shown in this figure is the performance of these genomes in the other two environments (i.e., the environments not experienced during evolution). This was done to investigate how well the genomes evolved in one environment transferred to another. It was initially expected that whilst it is feasible for a genome specially evolved for one environment would be

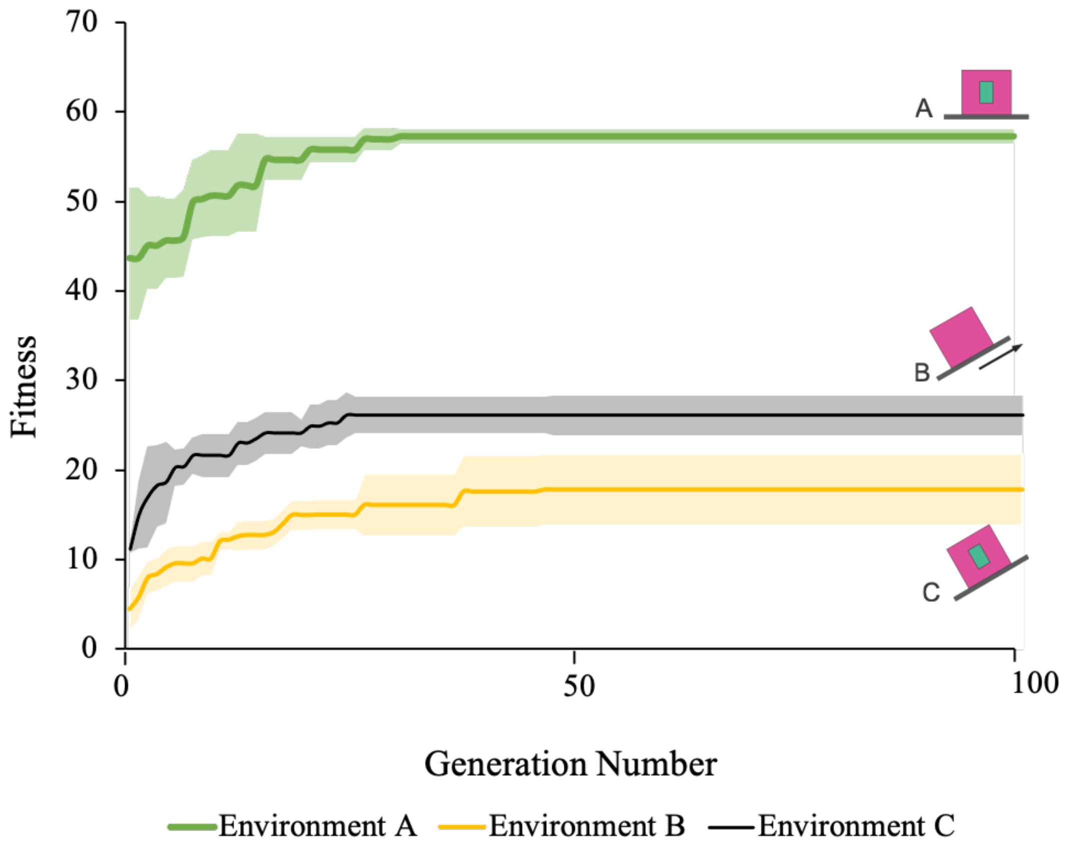


Figure 4.6: This graph shows a plot of the fitness for the average of the best genomes, per generation, for each of the three environments. To clarify, the fitness of the best genome for each of the 10 evolution runs in **Environment A** are averaged and plotted at each generation in **green**. Note that the fitness is the distance travelled in the final episode. Similarly the fitness of the best genomes for each of the 10 evolution runs in **Environment B** are averaged and plotted in **grey** and **Environment C** in **orange**. For each of the three different environments the point at which evolution converges is marked by a line. Also note that although the evolutionary algorithms were run for 150 generations, for readability only 100 generations are shown on the graph. Shown with 95% confidence interval

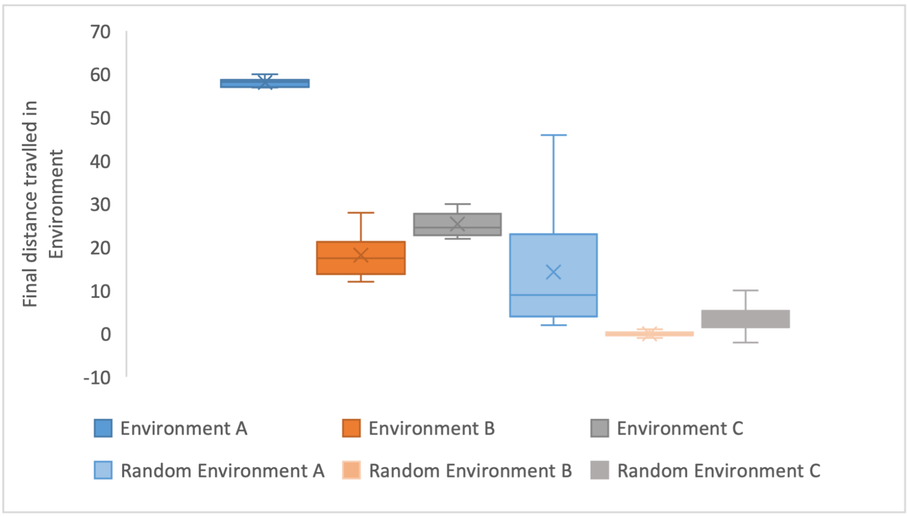


Figure 4.7: Box and whisker plot comparing evolution in the three environments. Plots of the champions of the three evolutions are compared with the same number of random genomes in each environment. The plots show that in each environment the neural networks develop morphologies that outperform those developed from unoptimised ANNs.

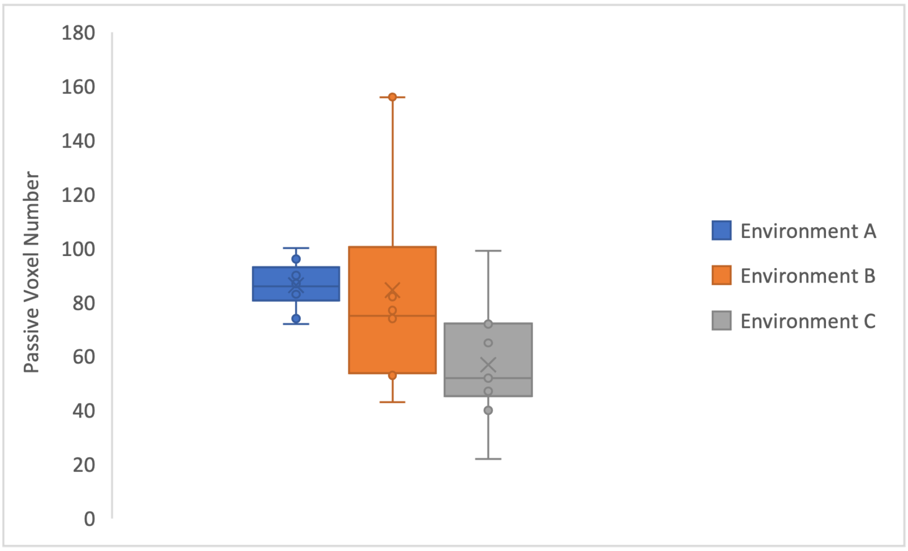
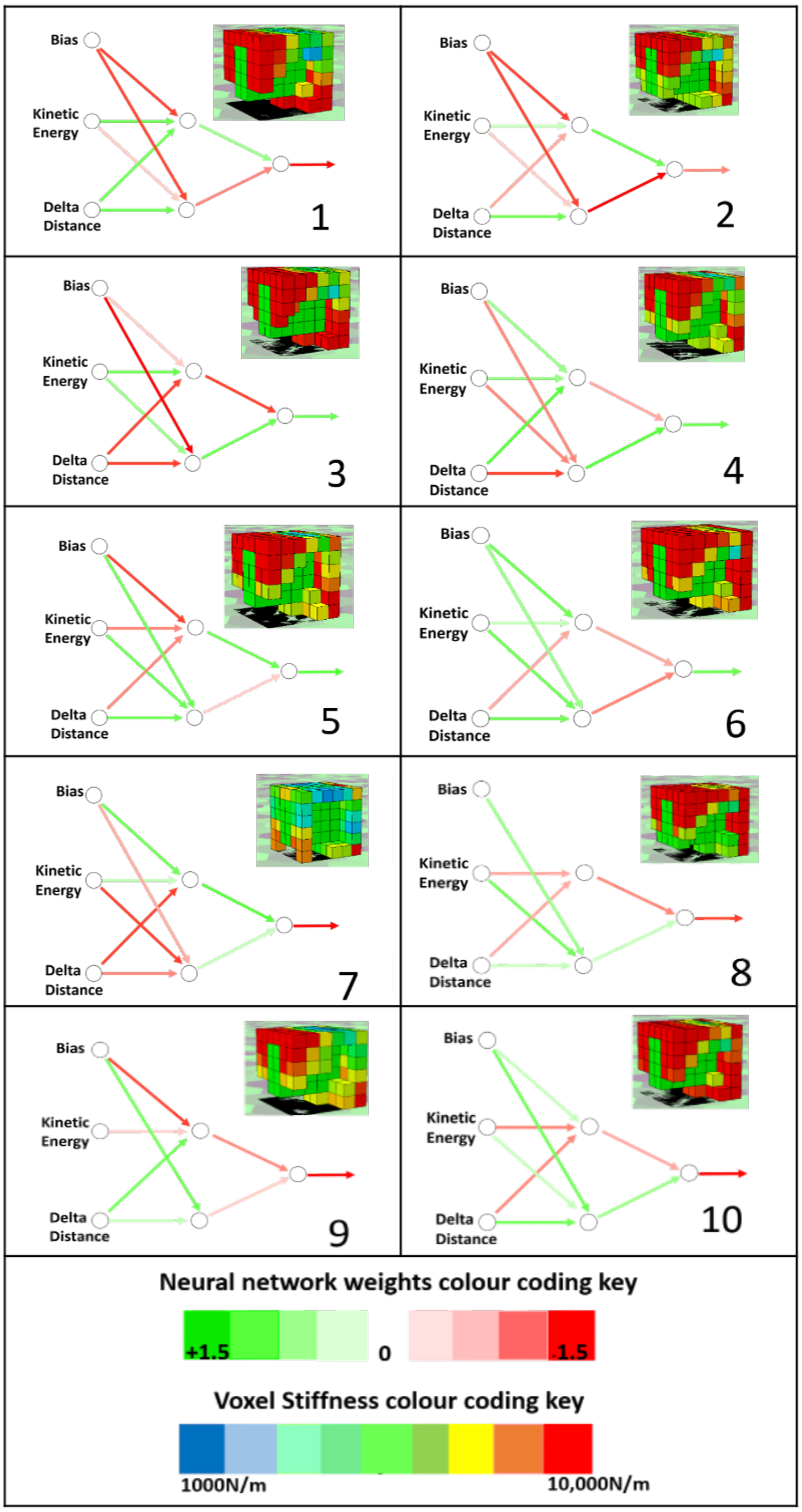


Figure 4.8: Box and whisker plot comparing the amount of passive voxels removed depending on environment. The plot indicates that significantly more voxels are removed in environment A than environment B. Additionally, more voxels are also removed in environment B than environment C. T-tests (alpha = 0.05) were conducted to confirm this resulting in p = 0.002 (environment A/C) and p = 0.03 (environment B/C)



Bias
Kinetic Energy
Delta Distance
1
2
3
4
5
6
7
8
9
10
Neural network weights colour coding key
+1.5
0
-1.5
Voxel Stiffness colour coding key
1000N/m
10,000N/m

Figure 4.9: This figure shows the top ten neural networks evolved for environment A, i.e., the top genomes from each of the evolutionary runs. Also shown is the final morphology sculpted when using the neural network. In the case of the neural nets red indicates a strong negative weighting whereas green indicates a strong positive weighting. For the final morphology red indicates a very stiff voxel whereas blue indicates a soft voxel.

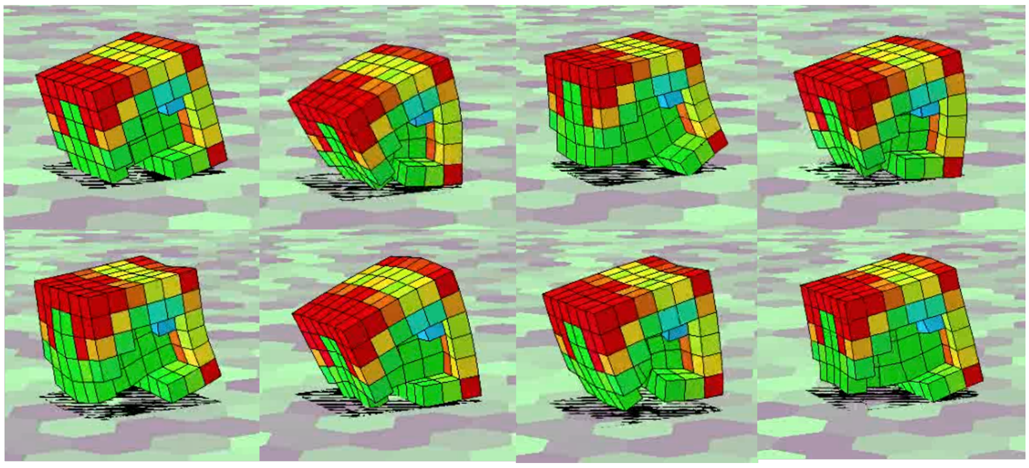


Figure 4.10: Figure showing a typical final morphology (in this case sculpted from genome 1) locomoting in Environment A. See https://youtu.be/Mzlt-HRCAao for video

able to sculpt successful morphologies in another it is very unlikely. Additionally, if it was successful the resulting morphology would perform worse than if the genome had also been evolved specifically for this second environment. In these "test" cases the initial robot is simulated in environment B, but it uses the neural networks (the genomes) evolved in environment A to dictate how it should adapt. Since the kinetic energy input to the neural network is different for each environment a different final morphology is sculpted. If the morphology created is successful in this environment, i.e., one that the neural network was not evolved in, the neural network would be transferable. A genome was considered to be successful if its fitness (i.e., its speed) is greater than zero. Therefore, it is travelling in the direction of the actuation and not turning or going backwards. As previously discussed it was anticipated that if a genome was successful (i.e., it produced a morphology capable of some locomotion) the performance would be poor as the genome had not been specifically evolved for this new environment. From this figure (Figure 4.11) it can clearly be seen that the only environment that these genomes are successful in is the one they experienced during evolution, i.e., only the blue bars symbolising performance in Environment A are positive. In Environment B and Environment C the final morphologies either fall down hill (environment B) or turn and locomote downhill (environment C). This is why the fitness in these environments are shown to be negative.

Additionally, an investigation into the frequency of the passive voxels final stiffness, when developed in environment A was carried out. This is shown in figure 4.12. For this environment, the final morphologies contain mostly stiffer voxels.

The same process for Environment B is now shown in Figures 4.13 and 4.14. Note that an un-optimised robot with the same stiffness all over moves downhill (i.e., the opposite direction) a distance of -3.2 voxels. In Figure 4.13 the top genomes, i.e., those that produced the best performing final morphologies in Environment B are shown, along with the morphologies that they produce. Figure 4.14 shows how these top genomes perform in the other two environments for which they were not

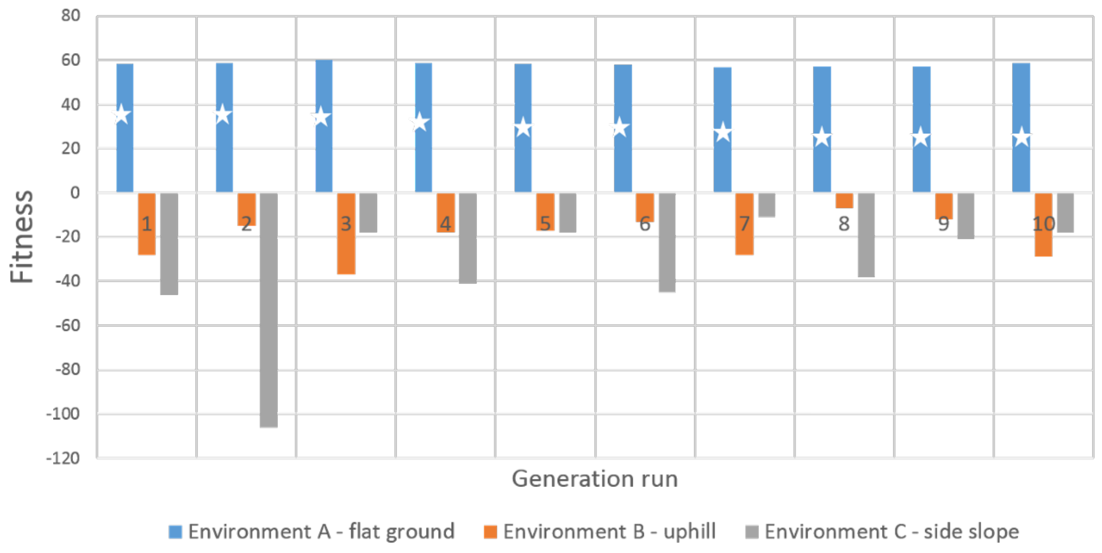


Figure 4.11: This figure shows the ten champion neural networks from each of the 10 evolution runs evolved for environment A, simulated in all three environments. To clarify these genomes are the results from the top evolution runs evolved for environment A. As a reminder the genome is the weight of the neural network. These neural networks are then used to sculpt morphologies, starting with the same cubed robot, in the other two environments. The white star is used to indicate that it is this environment for which the genome was originally evolved. Note that for this environment, an un-optimised robot has a fitness of -0.61. Therefore it moves slightly in the wrong direction.

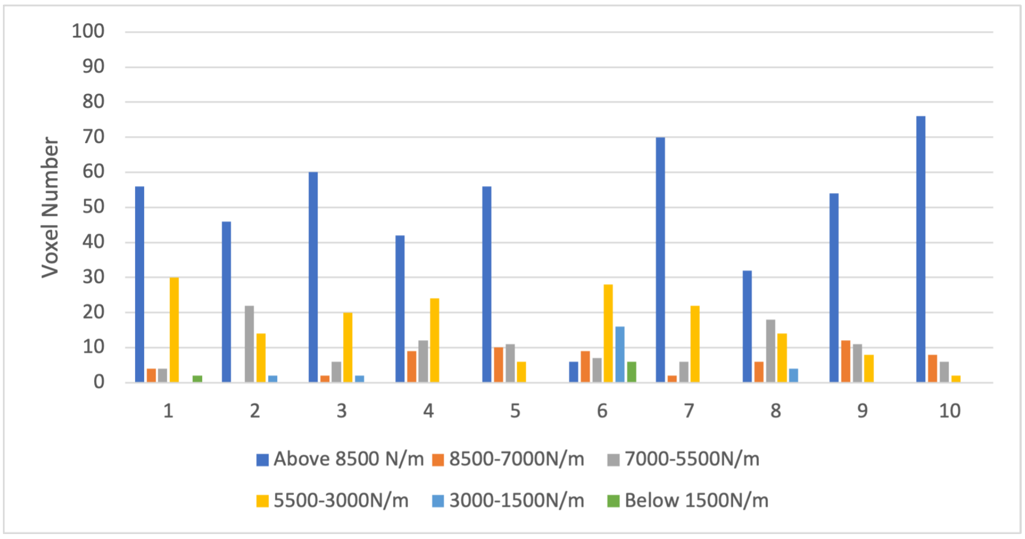


Figure 4.12: Frequency plot of voxel stiffness in final morphologies, developed in environment A. In this environment there is a strong bias towards stiffer voxels than softer for the majority of final morphologies

evolved in. Figures 4.13 shows that in this environment there is more variation in the final sculpted morphologies. In some cases (1,2,3 and 9) the final morphologies are quite similar to those from Figure 4.13 with a stiff front and the bottom middle voxels removed. An example of the way these type of morphologies locomote is shown in Figure 4.16. Note that at the start of the simulation these morphologies balance on the back voxels as a result of their interaction with the slope. When the activation of the inner voxels begins the robot falls over onto its "back" and continues locomotion from this position as shown. In other cases, such as 5,6,8 and 10 the overall morphologies contain much softer voxels (see Figure 4.15 for how these final morphologies locomote. In two examples 4 and 7 almost all of the voxels are removed bar the top front and back. It is hypothesised that these passive voxels are required to balance the robot as when all the voxels are removed the robot falls over shortly after the beginning of the episode. Similarly to the genomes evolved for environment A, the individual weights of the neural nets are different for each of the top genomes. For example, it is clear to see that genomes 4 and 7 produce the same final morphology, however the weights of the neural network are very different.

Figure 4.14 show how the genomes which have been evolved in environment B perform in not only environment B (oranges bars with white stars) but also environments A and C. Remember that these genomes have only experienced environment B during evolution. As with Figure 4.11, when these genomes are simulated in environment B they perform very well - this is not surprising since this is the environment they were optimised for. However, with two exceptions (genomes 4 and 7), these genomes are also able to sculpt successful morphologies in environment A. This is interesting as these genomes had not experienced this environment during evolution. Additionally there is one genome (genome 1, highlighted by the red bar) that is also able to sculpt a successful morphology for environment C.

Additionally, Figure 4.17 shows a frequency plot of the voxel stiffness for the final morphologies evolved in environment B. In this environment there are more soft voxels than stiffer (i.e., more blue/green voxels than red/orange). Indeed, when a t-test (alpha =0.05) was carried out comparing number of red (above 8500N/m stiffness) voxels, there were significantly more red voxels in the morphologies in environment A than environment C ($p = 0.0001$).

Finally Figures 4.18 and 4.19, show the neural networks, the final sculpted morphologies and the performance of these genomes in each of the three environment for the genomes optimised for environment C. Note that an un-optimised simple robot where all the voxels have the same stiffness has a fitness of -1.2 voxels. These genomes all sculpt successful morphologies for the environments they were optimised for (environment C), but also for environment A. When compared to the final sculpted morphologies for environments A and B there is much more variation in this environment (although as before there is little similarly in the neural network weights). One similarity is that all these final morphologies do show is an asymmetry. This was expected as there is clear asymmetry in the environment i.e., the slope and therefore in the feedback. The voxels on the right hand side of the robot (those further downhill) are either removed or are much softer than those uphill.

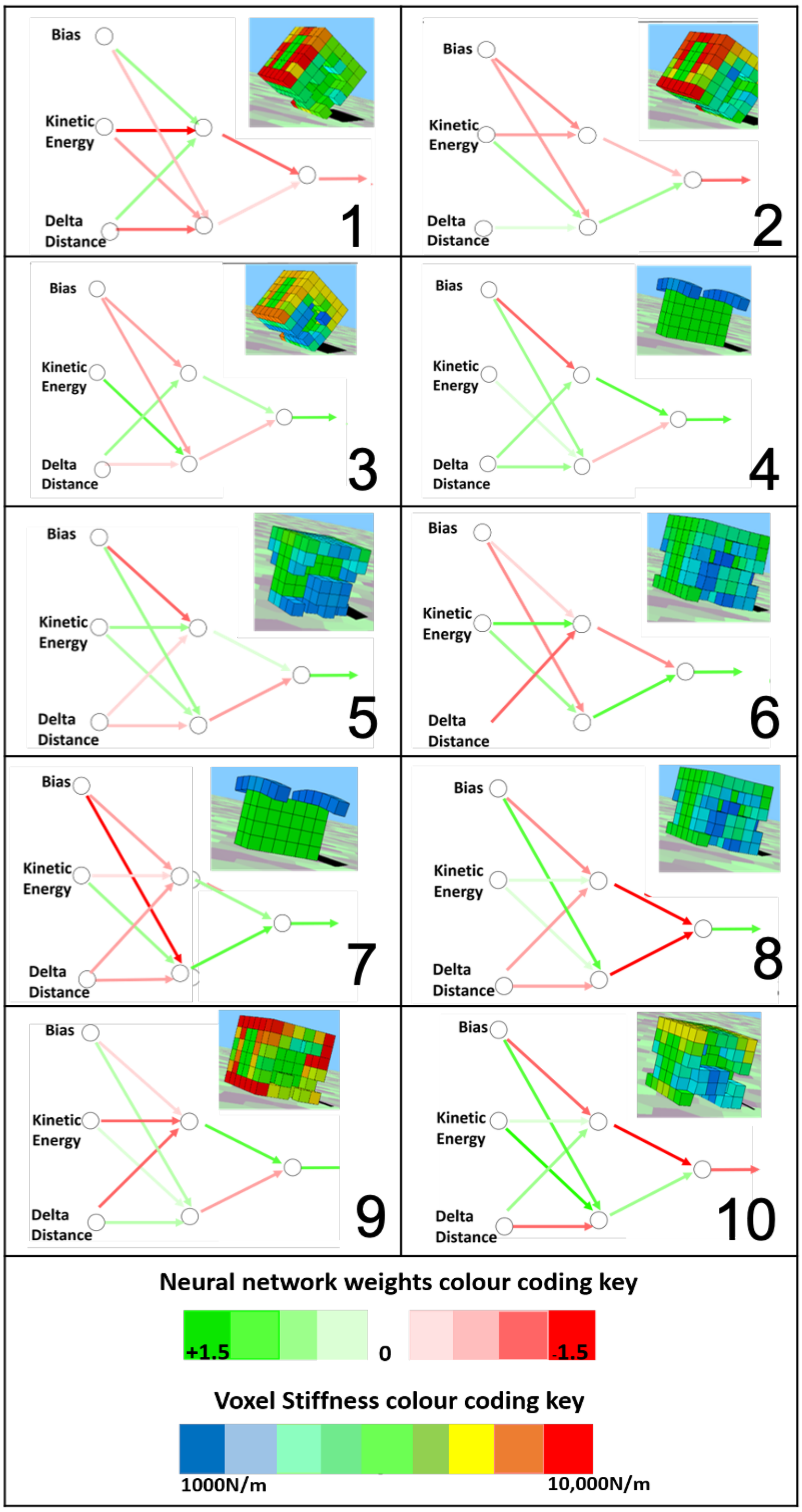

Bias
Kinetic Energy
Delta Distance
1
2
3
4
5
6
7
8
9
10
Neural network weights colour coding key
+1.5
0
-1.5
Voxel Stiffness colour coding key
1000N/m
10,000N/m

Figure 4.13: This figure shows the ten champion neural networks from each of the 10 evolution runs for environment B, i.e., the top genomes from each of the evolutionary runs. Also shown is the final morphology sculpted when using the neural network.In the case of the neural nets red indicates a strong negative weighting whereas green indicates a strong positive weighting. For the final morphology red indicates a very stiff voxel whereas blue indicates a soft voxel.

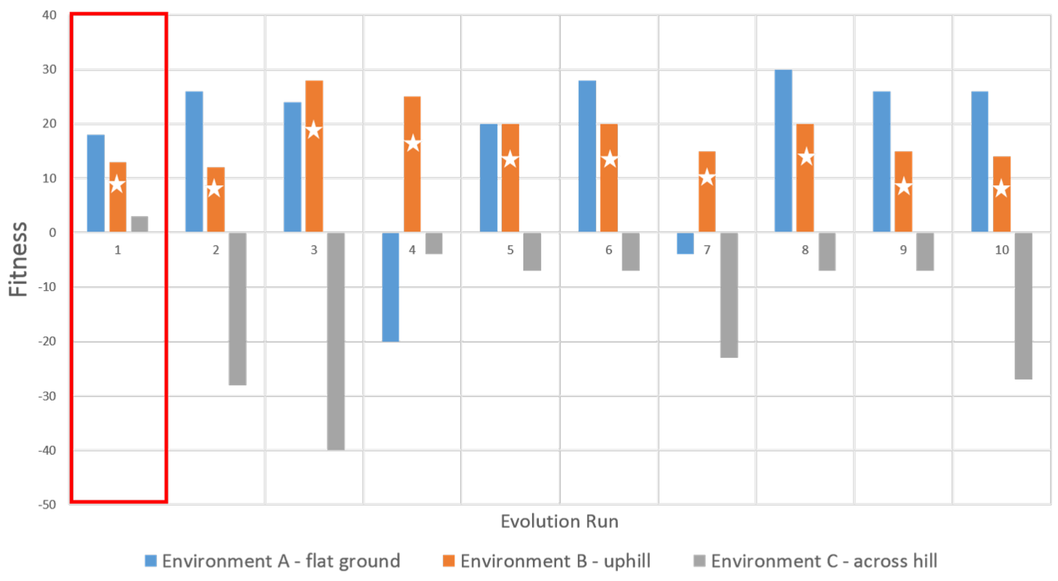


Figure 4.14: This figure shows the ten champion neural networks from each of the 10 evolution runs for Environment B, simulated in all three environments. To clarify these genomes are the results from the top evolution runs evolved for environment B. As a reminder the genome is the weight of the neural network. These neural networks are then used to sculpt morphologies, starting with the same cubed robot, in the other two environments. The white star is used to indicate that it is this environment for which the genome was originally evolved. As a result all the genomes are able to sculpt successful morphologies in environment B as this is the environment they were evolved for. Note that an up-optimised robot locomotion in this environment has a fitness of -3.2, i.e., it moves downhill.

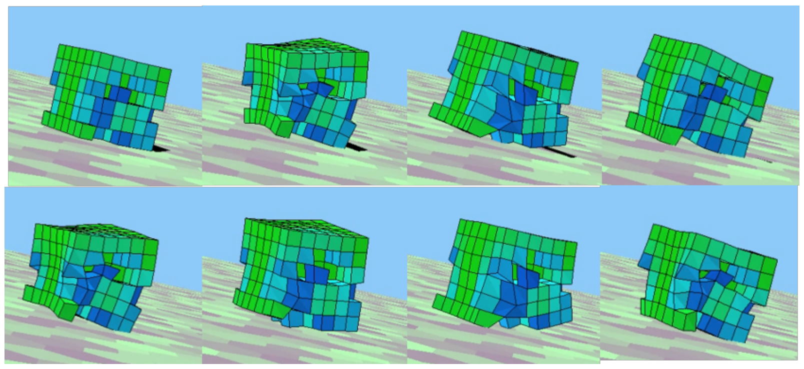


Figure 4.15: This figure shows one of the typical locomotion patterns for the type of final morphologies sculpted by genomes such as genome 6 in environment B. See https://youtu.be/TjNg701Un9o for video

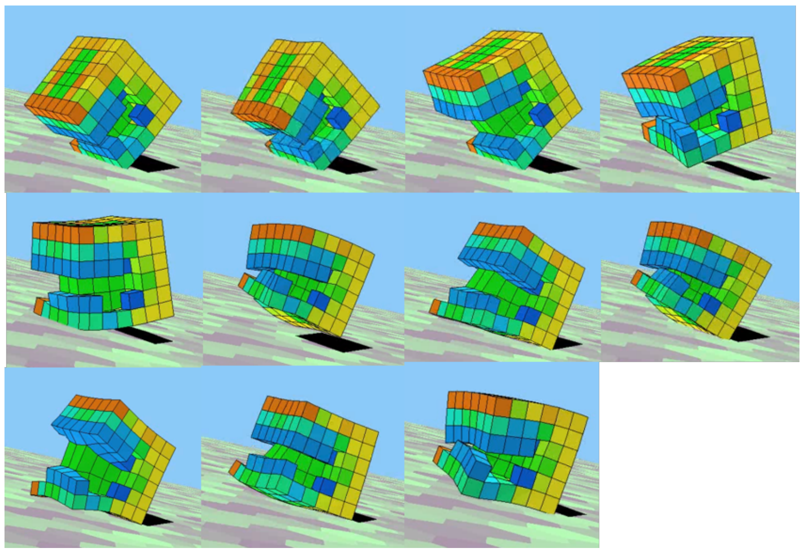


Figure 4.16: This figure shows one of the typical locomotion patterns for the type of final morphologies sculpted by genomes such as genome 1 in environment B. https://youtu.be/TjNg701Un9o for video

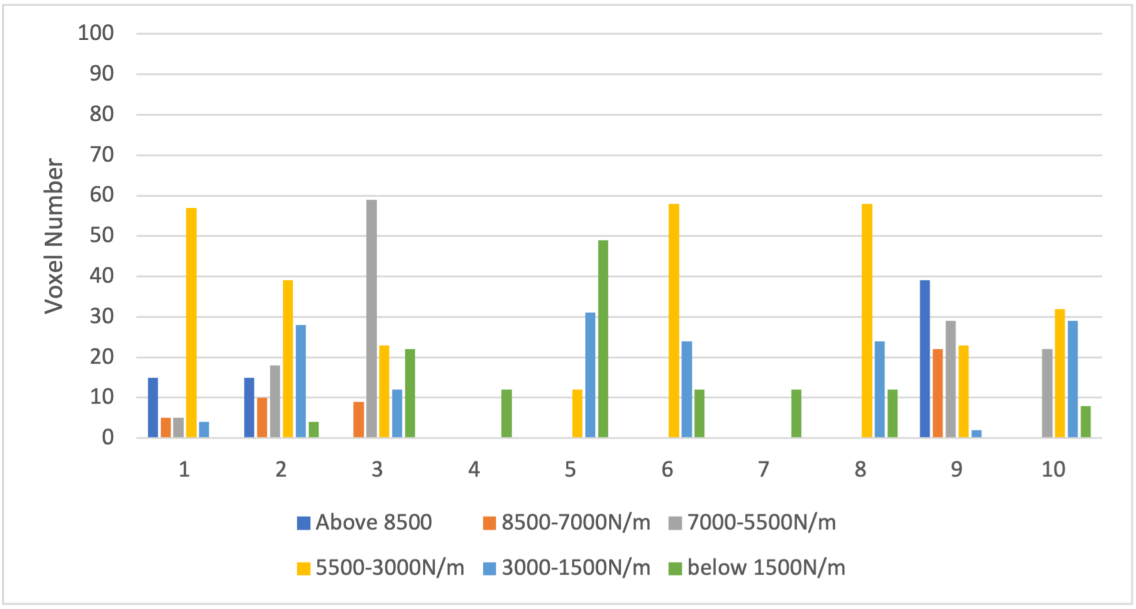


Figure 4.17: Frequency plot for voxel stiffness in final morphologies developed in environment B. In this environment there are slightly more soft passive voxels than stiffer ones.

Additionally, Figure 4.20 shows how a typically morphology evolved in environment C locomotes and Figure 4.21 shows the frequencies of voxel stiffness for the final morphologies developed in environment C.

Note that if just the final morphologies sculpted in environment C were then tested in environment A, unable to adapt, they would be not successful. The asymmetrical morphology on flat ground would cause the robot to move in circles; thus the distance travelled from the starting point would be low. However, because it is the way that the robot adapts to an environment that is evolved, when tested in environment A the robot is able to sculpt a symmetrical morphology, even when following a sculpting adaption system optimised in the asymmetric environment C.

When comparing the final morphologies developed in each of the three environments there are some interesting observations. Firstly, those developed in environments A and C tend to utilize stiffer voxels (i.e., reds and oranges) than those developed in environment B (blues). Why this is the case is not inherently clear, it could be speculated that the softer more compliant voxels are somehow useful in overcoming the gravitational challenges of the uphill slope. However, this is simply initial speculation and perhaps further research into why environment B seems to favour softer voxels is required. Secondly, when comparing the amount of remaining passive voxels after sculpting, considerably more passive voxels remain in environment C than those developed in the other environments. Here, t-tests were carried out comparing the amount of remaining passive voxels and it was found that significantly more voxels are removed in environment A and B compared with C (p= 0.002 for environments A and C) and (p=0.03 for environments B and C). Here, it is likely that the disparity is caused by the complex asymmetry of the environment C as instead of losing voxels on both sides as is the case in the other two environment, the morphologies in environment C only remove voxels on one side to achieve asymmetry.

Another interesting observation is the variety of sculpted morphologies produced from the evolution in environment C. This would appear to agree with the results from Bongard and Auerbach [5] where it has been shown that the more complex the environment the robot is evolved in the more complex the morphology. In [5] they directly evolve one fixed morphology for different environments whereas here I evolve the methodology of adaption. However, there does seem to be a parallel. This would indicate that the side slope (environment C) is the most complex environment tested (as originally hypothesised), closely followed by environment B (uphill) and then environment A (flat landscape). However, as previously discussed, this theory is not supported by Figure 4.6 where the evolutionary algorithm converges first in environment C. The most simple environment does appear to be environment A. Not only is there little variation in final sculpted morphologies, the evolutionary algorithm converges quickly. Furthermore the majority of the genomes that are evolved in different environment are able to sculpt successful morphologies in environment A without having experienced this environment during evolution.

However, the observation that there is more diversity in the morphologies developed

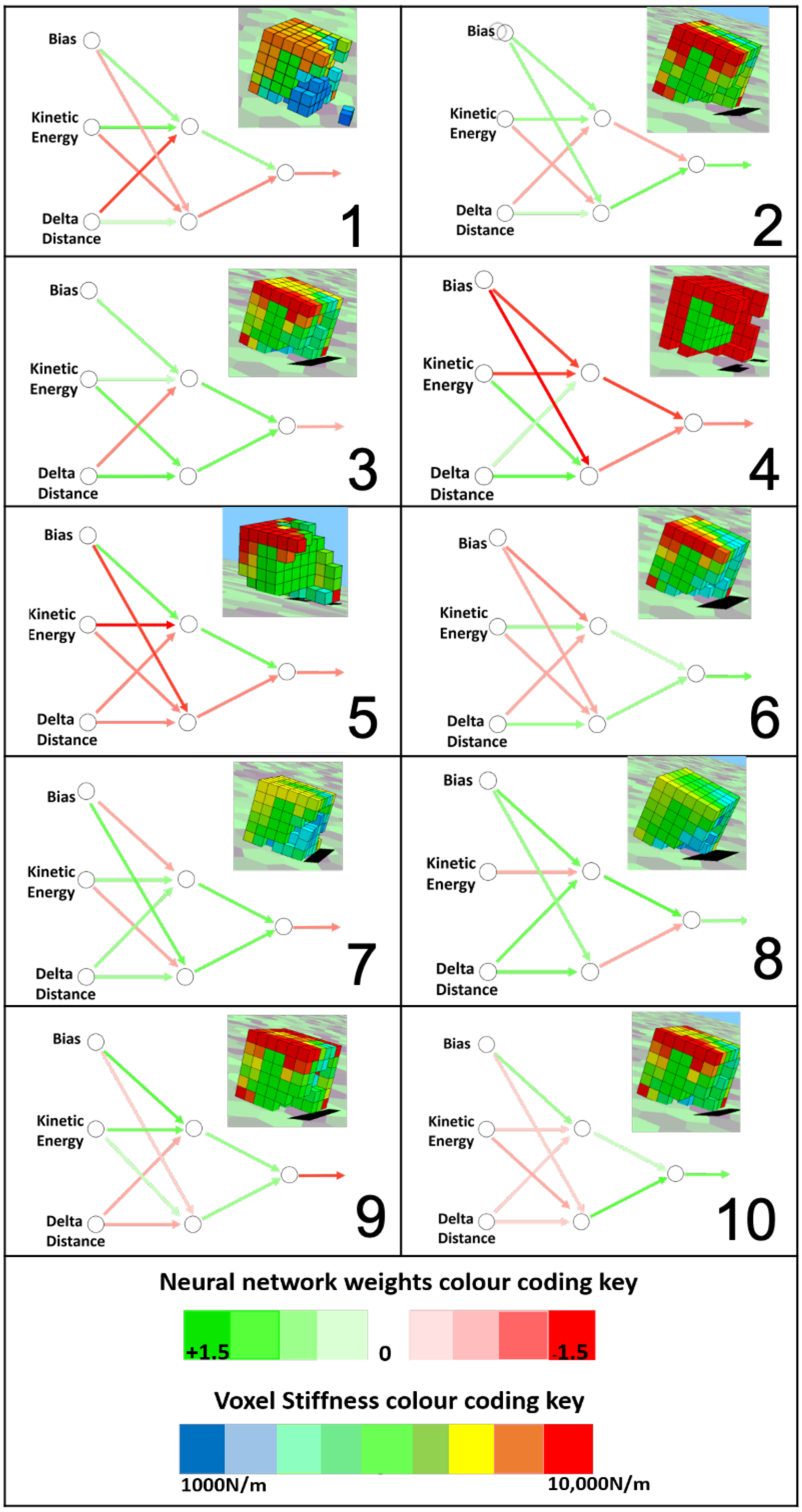

Bias
Kinetic Energy
Delta Distance
1
2
3
4
5
6
7
8
9
10
Neural network weights colour coding key
+1.5
0
-1.5
Voxel Stiffness colour coding key
1000N/m
10,000N/m

Figure 4.18: This figure shows the top ten neural networks evolved for environment C, i.e., the top genomes from each of the evolutionary runs. Also shown is the final morphology sculpted when using the neural network. In the case of the neural nets red indicates a strong negative weighting whereas green indicates a strong positive weighting. For the final morphology red indicates a very stiff voxel whereas blue indicates a soft voxel.

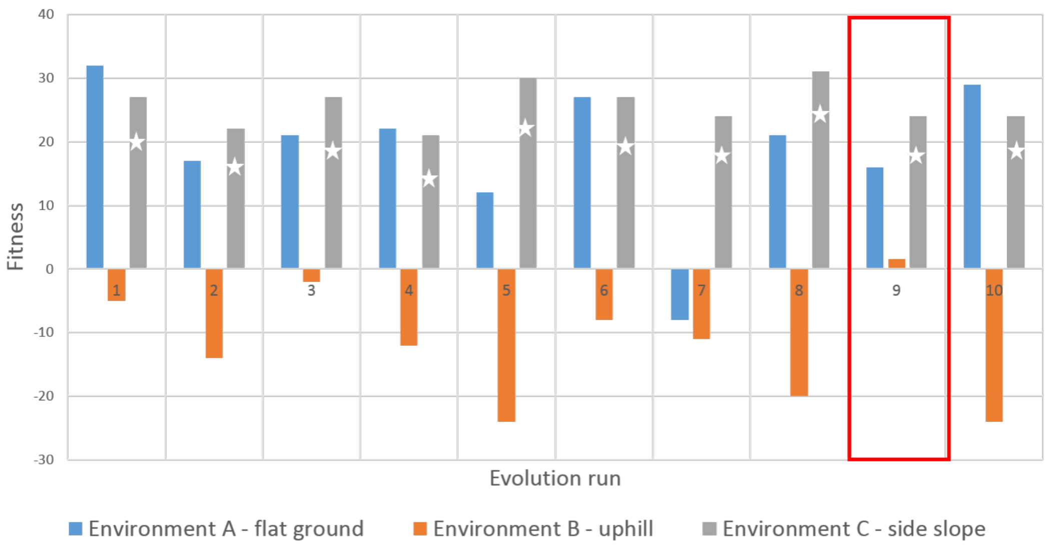


Figure 4.19: This figure shows the top ten neural networks evolved for environment C, simulated in all three environments. To clarify these genomes are the results from the top evolution runs evolved for environment C. As a reminder the genome is the weight of the neural network. These neural networks are then used to sculpt morphologies, starting with the same cubed robot, in the other two environments. The white star is used to indicate that it is this environment for which the genome was originally evolved. As a result all the genomes are able to sculpt successful morphologies in environment C as this is the environment they were evolved for. Note that in this environment and un-optimised morphology locomotes -1.2 voxels.

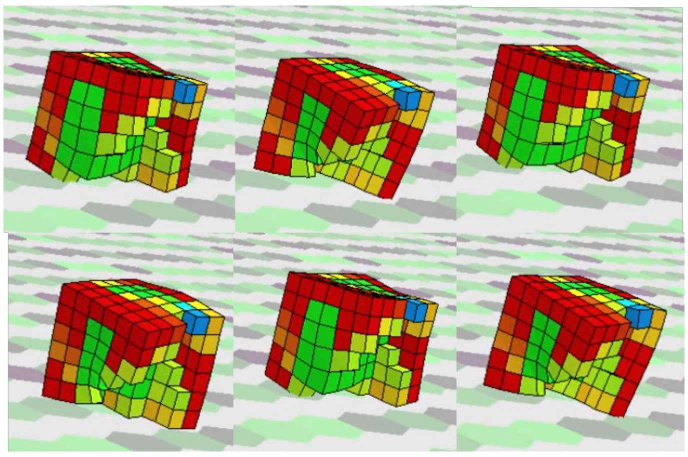


Figure 4.20: Figure showing how a typical morphology, evolved for Environment C, locomotes. See https://youtu.be/5f27h6tlMko for video

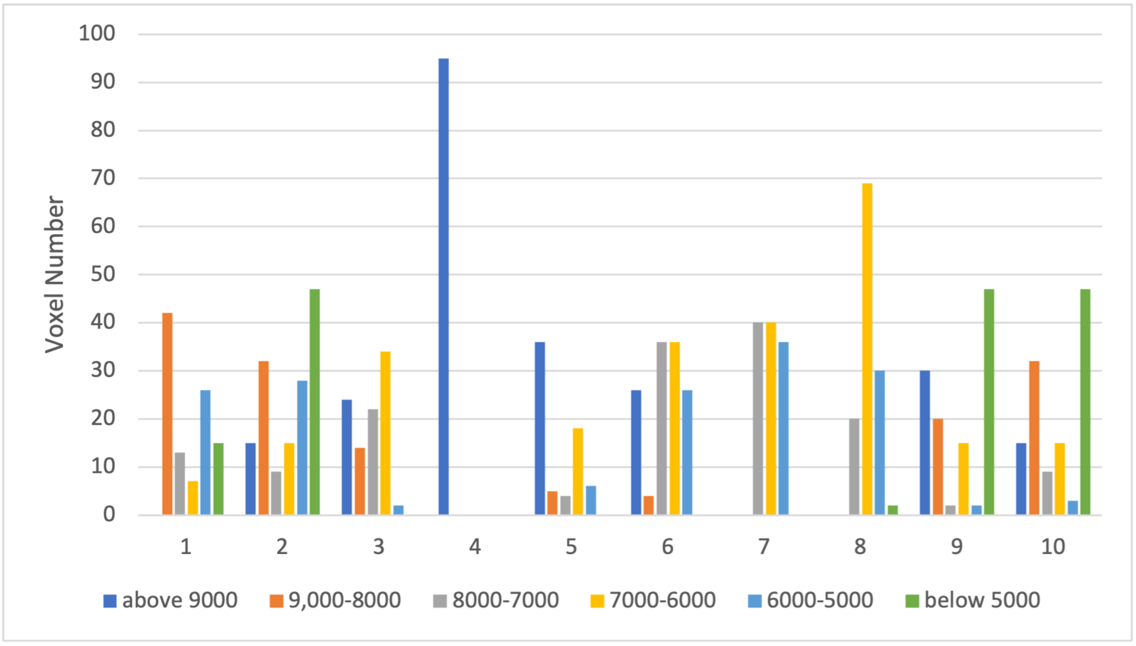


Figure 4.21: Frequency diagram of voxel stiffness in final morphologies developed in environment C.

in environment C than the other environments is simply by eye. A quantitative investigation into the diversity has not been carried out. Although not included in this thesis, a possible way of determining diversity in the final morphologies would be to consider those which had common voxel locations. For example, when comparing two different morphologies, for each possible voxel location, if both robots had a voxel present they would gain a score of zero. Similarly, if both robots did not have a voxel at a particular location then they would also gain a score of zero. However, if one robot had a voxel at a particular location and the other didn't, they would increase their comparative score by one. Once all possible locations had been considered, the higher the cumulative score the more "different" the two morphologies would be.

Another simpler approach to compare robot diversity is just to consider how many passive voxels remain after development, and this is something that was carried out earlier in the chapter (see figure 4.8). In this case it could be suggested that environment B has yielded the most diverse robots as it has the largest range of number of removed voxels (followed closely by environment C). Perhaps as expected, environment A has a small range of number of removed voxels suggesting a lack of diversity in this environment.

Out of the 30 genomes presented so far (the top ten from each environment) two genomes worked in all three environments, i.e., they were capable of sculpting successful morphologies in environments for which they were not optimised. Whilst is it interesting that these two genomes are transferable it is worth noting that even though all the genomes are based on feedback from the robots interaction with the environment, evolving in one environment would be expected to find optimal adap-

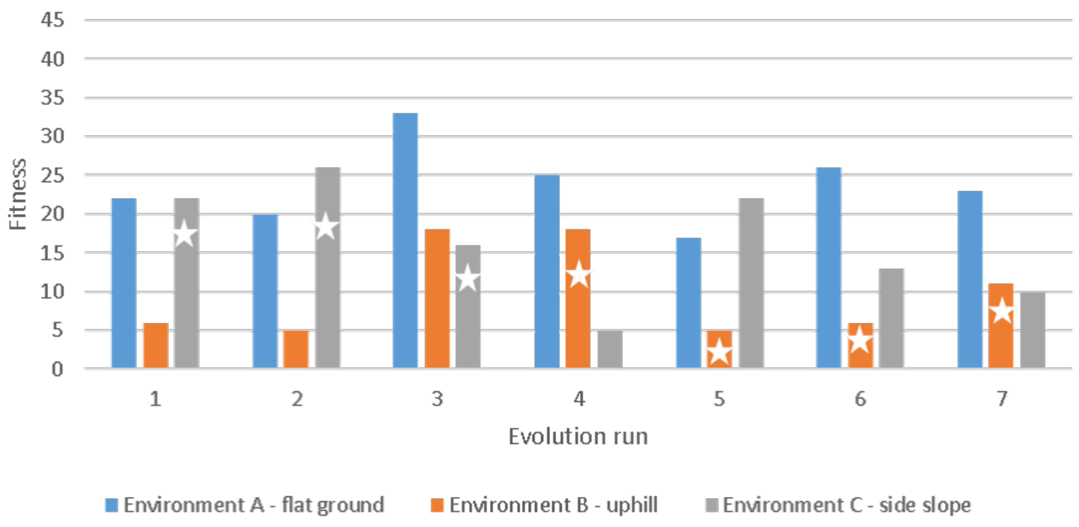


Figure 4.22: Figure shows the only genomes from each of the three evolutionary algorithms carried out in a single environment that are able to sculpt successful morphologies in all three environments. The white star symbolises which of the environments the genome was originally evolved for.

tion rules specific to that particular environment. Just because the rule set is based on the environmental feedback this isn't enough to guarantee that it is transferable to new environments. The 30 genomes that have been tested are the elite - the ones that perform the highest in each particular environment. It could be that these high performing genomes are too specialised, whereas those that are less successful in one environment are more transferable between tasks. Thus the top 50 genomes from each environment (150 genomes in total) were tested in the other two environments to test this hypothesis. Here, the "best" genomes were those that were able to sculpt successful morphologies in each of the three environments.

Figure 4.22 shows the genomes from this testing that were able to sculpt successful morphologies for all three environments, as shown there were only 7 out of 150 genomes that were transferable between environments. The white star symbolises which of the environments the genome was originally evolved for. Note that none of these transferable genomes were originally evolved in environment A (flat ground). Instead these transferable genomes all come from Environment B and Environment C .The share between these two environments is quite equal (3 from Environment C, 4 from Environment B).

These 7 transferable genomes are investigated further. Figures 4.23, 4.24 and 4.25 shows how these 7 genomes sculpt the initial robot over the 15 episodes of a lifetime.

In addition to Figures 4.23, 4.24 and 4.25, Figures 4.26, 4.27 and 4.28 show how the distance travelled in each episode (the fitness) of each of the transferable genomes varies over the lifetime of the robot for each of the different environments.

From Figures 4.26, 4.27 and 4.28 it can be seen that in each environment, for

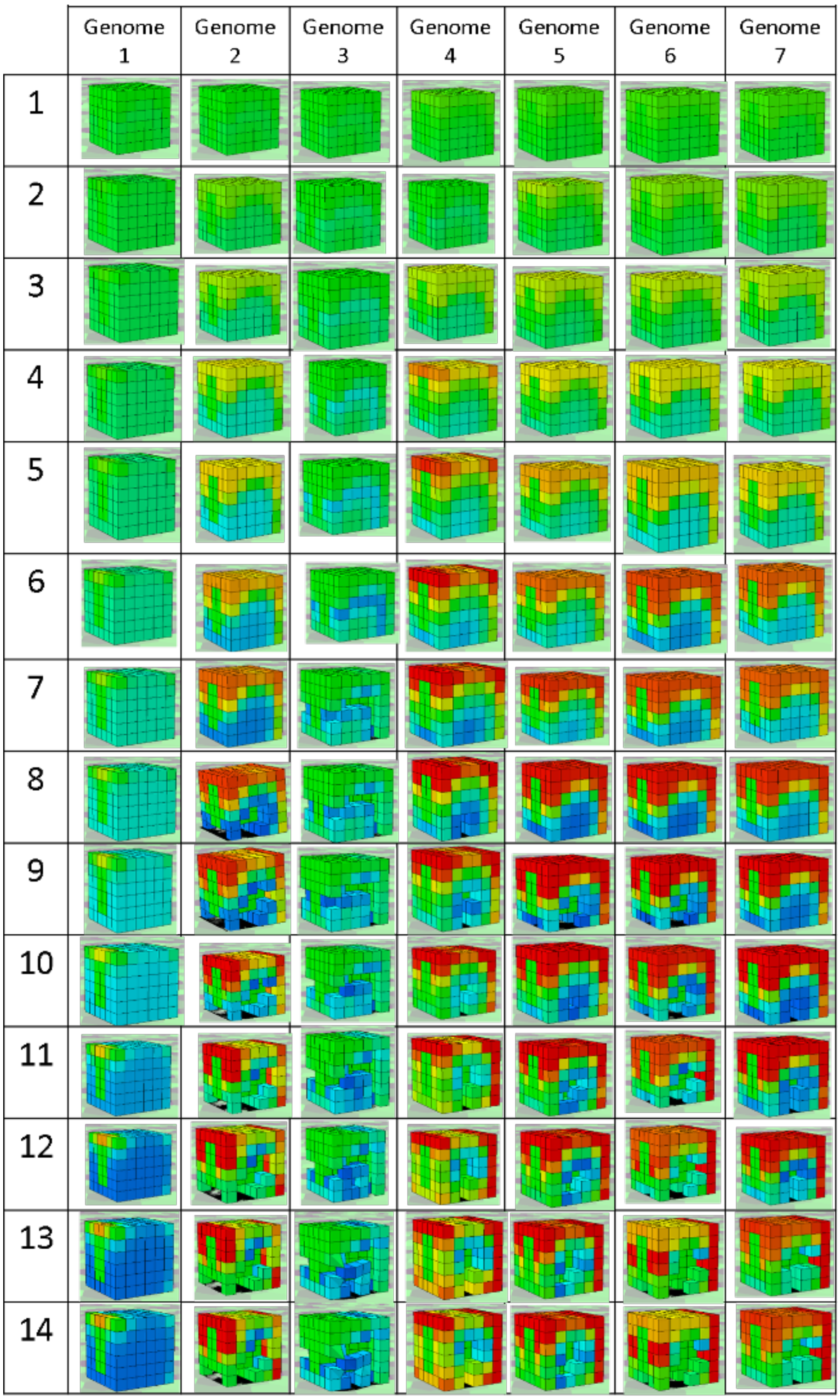


Figure 4.23: Figure shows how the 7 "transferable" genomes sculpt the initial robot over the 14 episodes in environment A - the flat ground.

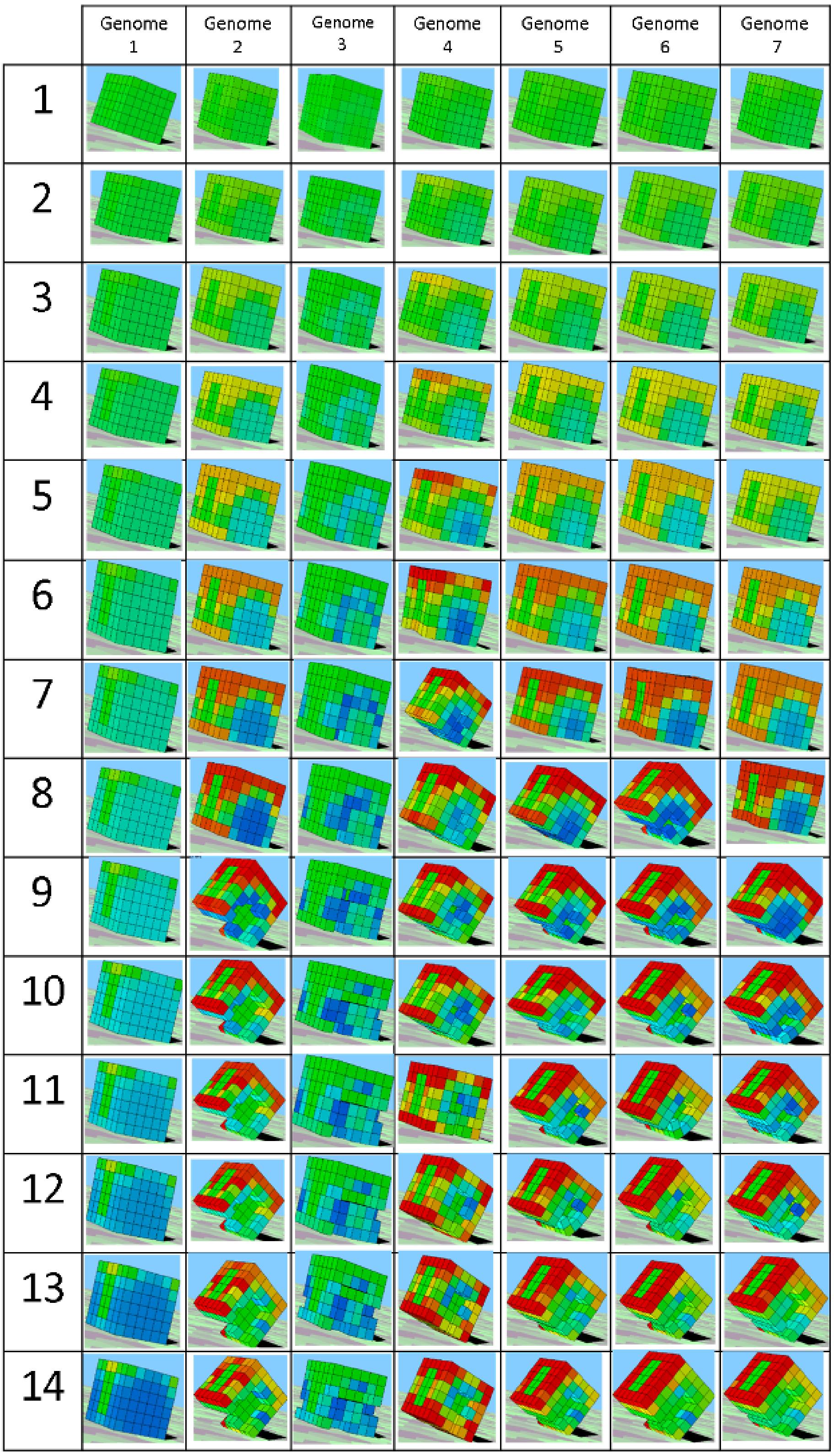


Figure 4.24: Figure shows how the 7 “transferable” genomes sculpt the initial robot over the 14 episodes in environment B - uphill.

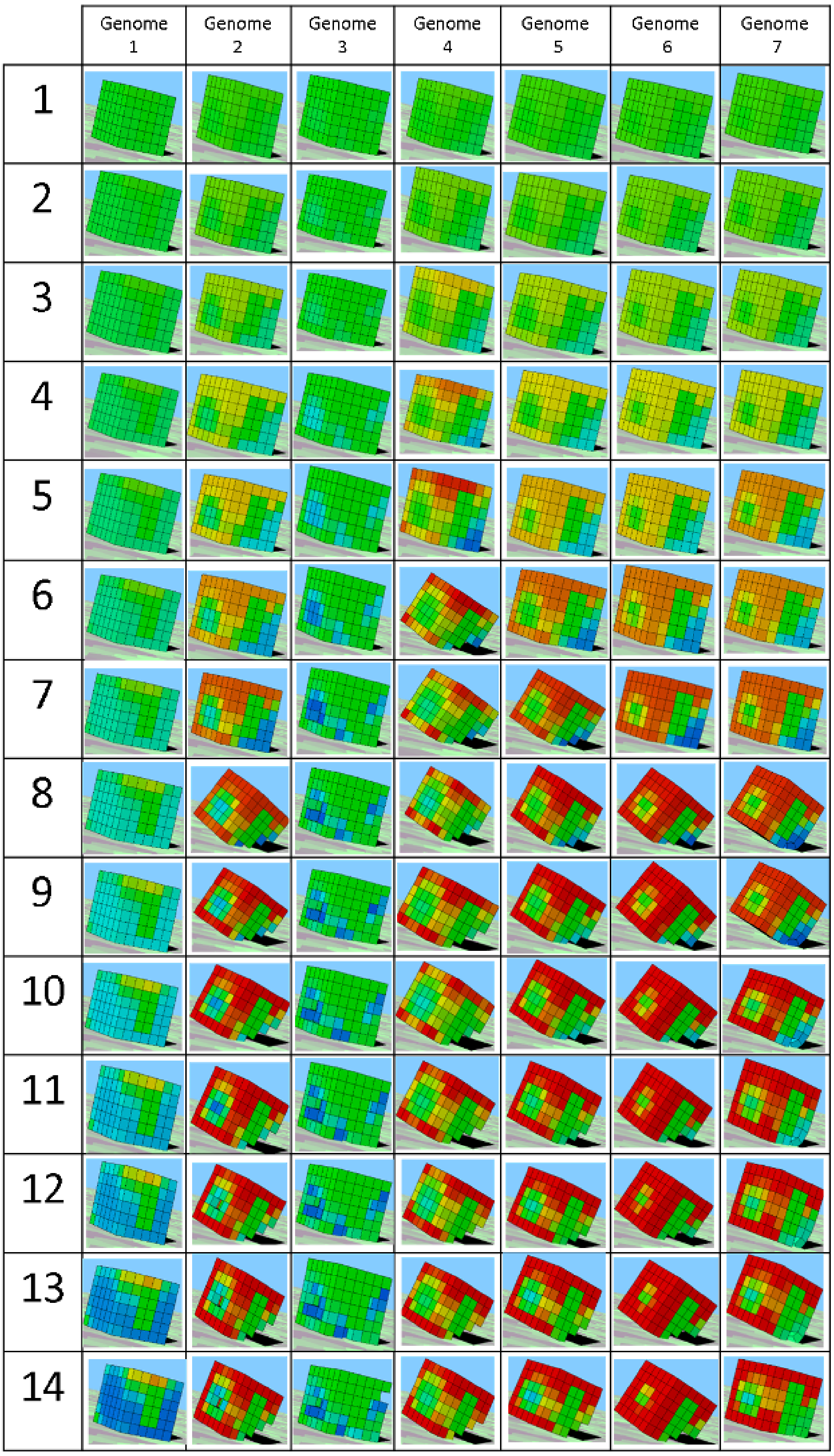


Figure 4.25: Figure shows how the 7 “transferable” genomes sculpt the initial robot over the 14 episodes in environment C - across the slope.

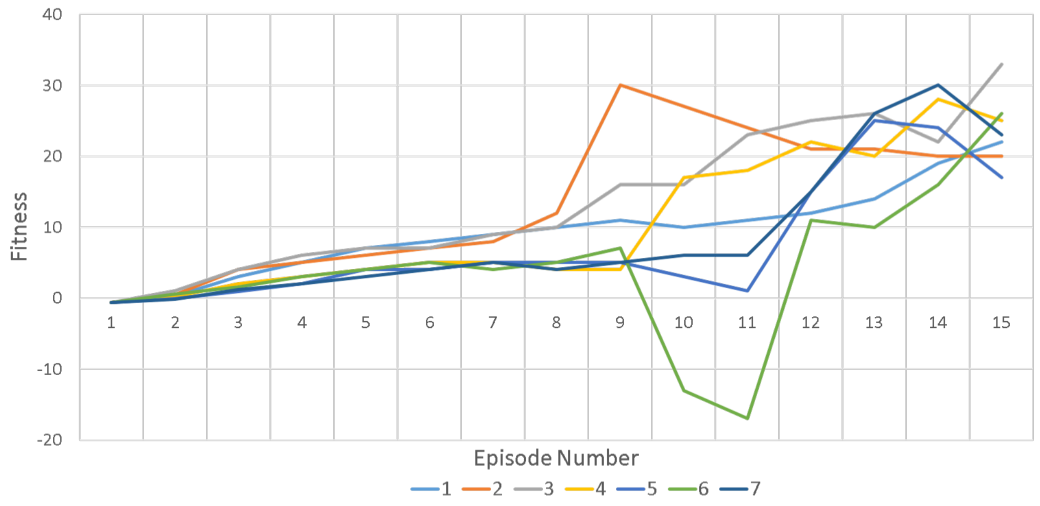


Figure 4.26: Graph showing how the distance travelled in each episode (the fitness) of each of the transferable genomes varies over the lifetime of the robot for Environment A.

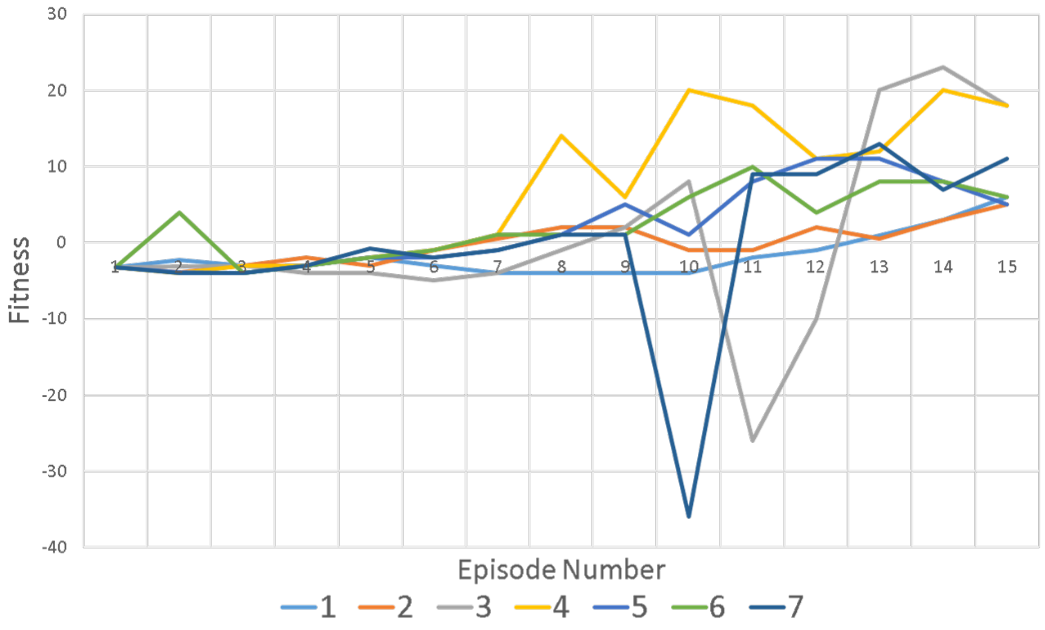


Figure 4.27: Graph showing how the distance travelled in each episode (the fitness) of each of the transferable genomes varies over the lifetime of the robot for Environment B.

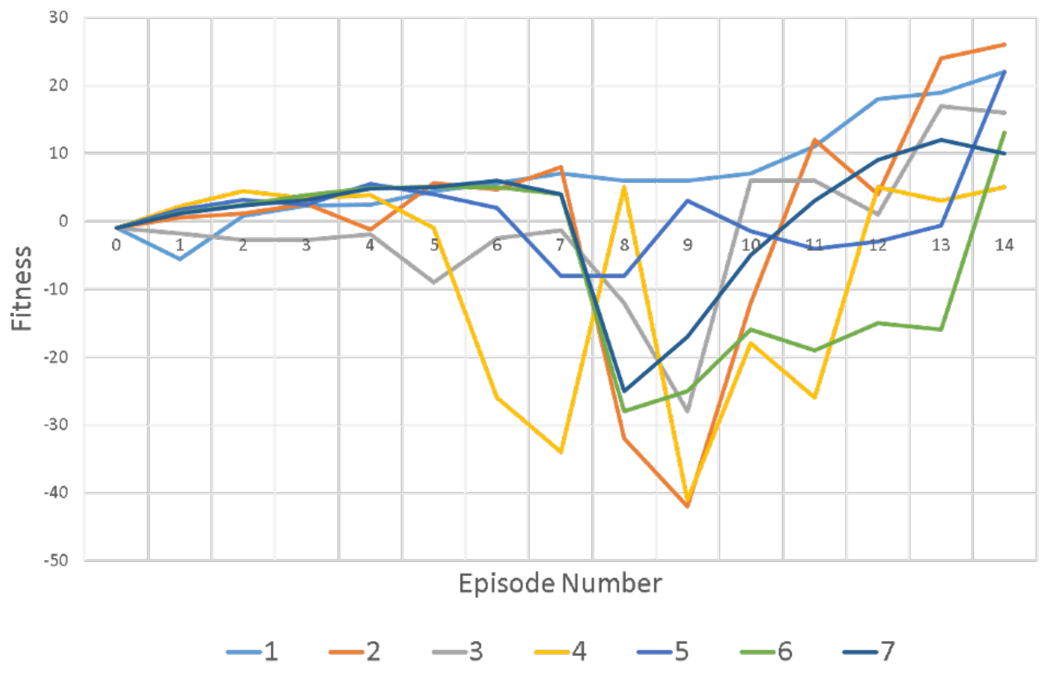


Figure 4.28: Graph showing how the distance travelled in each episode (the fitness) of each of the transferable genomes varies over the lifetime of the robot for Environment C.

the first few episodes the distance travelled per episode increases gradually. When considering the corresponding sculpting figures it can be seen that in the early episodes no voxels are removed. Instead only the stiffness of the voxels change. Firstly, this shows that actually the performance of the robot can be improved just by altering the stiffness distribution. At approximately episode 8 (in some genomes this occurs earlier, some later) the stiffness of some voxels is decreased to the point where the voxels are removed. In the majority of cases this temporarily results in a decrease in performance, especially prevalent in the more complex environments B and C. The reason that this drop in performance is not as obvious in environment A could be due to the simplicity of environment A – there is no slope to fall down so any unbalance in the system would only cause a smaller loss in performance, rather than falling down the slope.

A possible reason as to why the such a large drop in performance is suddenly observed could be due to the threshold for removal. Although not tested, it could be speculated that if this threshold value was set much lower, for example to zero, then changes to the performance over the lifetime of the robot would be less pronounced. However, as previously discussed, this was not tested due to errors in the simulation software.

After these voxels have been removed the distance travelled per episode starts to increase again. However very few other voxels are removed. Instead the increase in performance is once again down to the change in stiffness. The voxels, especially those surrounding the removed voxels are stiffened (i.e., they become harder).

In some cases a “zig-zagging” effect can be observed, where after the initial drop in performance the fitness increases in the next episode then decreases in the one after before increasing again. It is hypothesised that this effect is caused by the either the low resolution of voxels (216) or too large a scaling factor (alpha – figure 4.3). The robot responds too much to a change in performance and over compensates by adjusting the stiffness too much. As a result this change in stiffness causes the robots performance to change once more, again causing a large change in stiffness. If the scaling factor was reduced too much the stiffness change would be less and therefore the performance would not be as affected. Similarly if the resolution of the robot was greater, i.e., there were more voxels, the large stiffness change of one voxel would not affect the global performance of the robot as much and therefore it is likely that the zig-zagging effect would be reduced.

4.4 Further Evolution in Multiple Environments

A further area to explore is evolution of transferable genomes in all three environments. Therefore, in this section the genomes were evolved in all three environments and the fitness was the weighted (with different weights depending on the environment) average distance travelled in the final episodes across all three environments. The average was weighted to ensure that there is no bias towards one environment during the evolution – for example the final morphologies in Environment A are capable of locomoting over 5 times the distance of the morphologies in the other two environments. Therefore the weighted average ensures that the evolutionary algorithm doesn’t just find solutions that perform well in environment A.

The weighting system used to determine the overall fitness is given by the equation below:

$Overallfitness = D_A/5 + D_B + D_C/3$

Where D_A is the distance travelled in the final episode n Environment A, D_B is the distance travelled in the last episode in Environment B and D_C is the distance travelled in the last episode in Environment C. These weighting were based on how far the most successful sculpted robot traveled in each environment when the genomes were evolved in that single environment.

The evolutionary algorithm was run ten times (i.e., with ten different randomised starting populations). Each population had 30 genomes and the new generations were formed as previously. However, this time the evolutionary algorithm was run for 200 generations as preliminary testing showed that evolution in the three environments took longer to converge than in a single environment.

Figure 4.29 shows the final morphologies created used the champion genome from each of the ten evolution runs in all three environments. It can be seen that once again with the increase in task complexity (i.e., the requirement to successfully locomote in all three environments) there is also an increase in morphological diversity. Figure 4.30 shows the neural networks (the genomes) used to sculpt the initial robot into its final morphology. As with the previous tests there seems to be very little similarly between these genomes, although in this instance this could be less surprising given the amount of diversity in the final morphologies produced.

Figure 4.31 shows the average generational fitness for each of the evolutionary algorithms tested in one environment compared with the evolutionary algorithm tested in all the environments. Note that the fitness for single environments is not strictly comparable to the fitness for the multi-environment evolution as this fitness is a weighted average of all three environments. However, what this graph does clearly show is that when evolving in three environments, the evolutionary algorithm takes over twice as long to converge as when it is only optimising for one environment. Also note that the crosses at the end of each data series show the point at which the evolutionary algorithm was terminated.

Figure 4.32 compares the success of the genomes evolved in all three environments with the transferable genomes evolved in single environment. The top graph shows the transferable genomes from the previous tests but the axis is scaled to allow direct comparison with the bottom graph which shows the results from the evolution in all three environments. On the top graph the star symbolises the environment for which the genome was originally evolved for. It can be seen from the two graphs that, unsurprisingly, those evolved in all three environments do perform better than those that are evolved in a single environment. To quantify this increase the total distance travelled in each environment, for each genome, was summed and averaged across the two approaches. To clarify the total success of the 7 transferable genomes through single environment evolution were averaged and compared to the average total distance of the 10 found through evolution in all three environments. The average total distance in the transferable genomes was 49 voxels, compared with an average of 59 voxels for the genomes evolved in the three environments. However, as previously stated evolving for success in 3 environments does take longer to find an optimal solution than evolution in one environment.

Interestingly, despite the implementation of the average weighting there is one genome that has been evolved in all three environments but actually doesn't work in Environment C. This can be seen in lower graph in Figure 4.32 where genomes 4 is highlighted by a red bar. This is because the genome has performed so well in Environment B, which has the highest average weighting this it is able to counteract the low score (negative) score in Environment C and still have a high overall fitness.

In Figure 4.33 the distance travelled by the robots, generated by the top 10 genomes from evolution in all three environments, is shown for each of the three environments. Here, there are a number of interesting differences between these graphs and those produced for the 7 "transferable" genomes previously investigated. In the case of environment A there is a lot more "zig-zagging" behaviour in the early episodes of the genomes evolved for all three environments, whereas in Environment B there

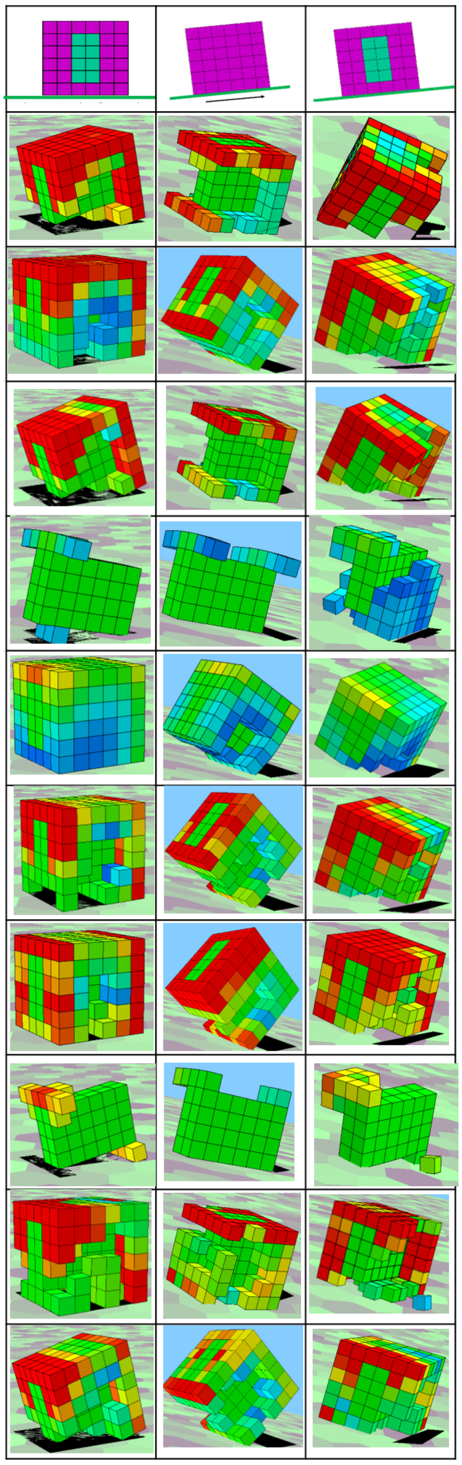


Figure 4.29: Figure shows the final morphology created by the best genomes from each of the evolution runs where the fitness was dependant on all three environments. Note that as before a red voxel represents a high stiffness, whereas a blue voxel represents low stiffness. The corresponding fitness of each of these morphologies in each environment are shown in Figure 4.32 and the neural networks used to develop the morphologies in 4.30

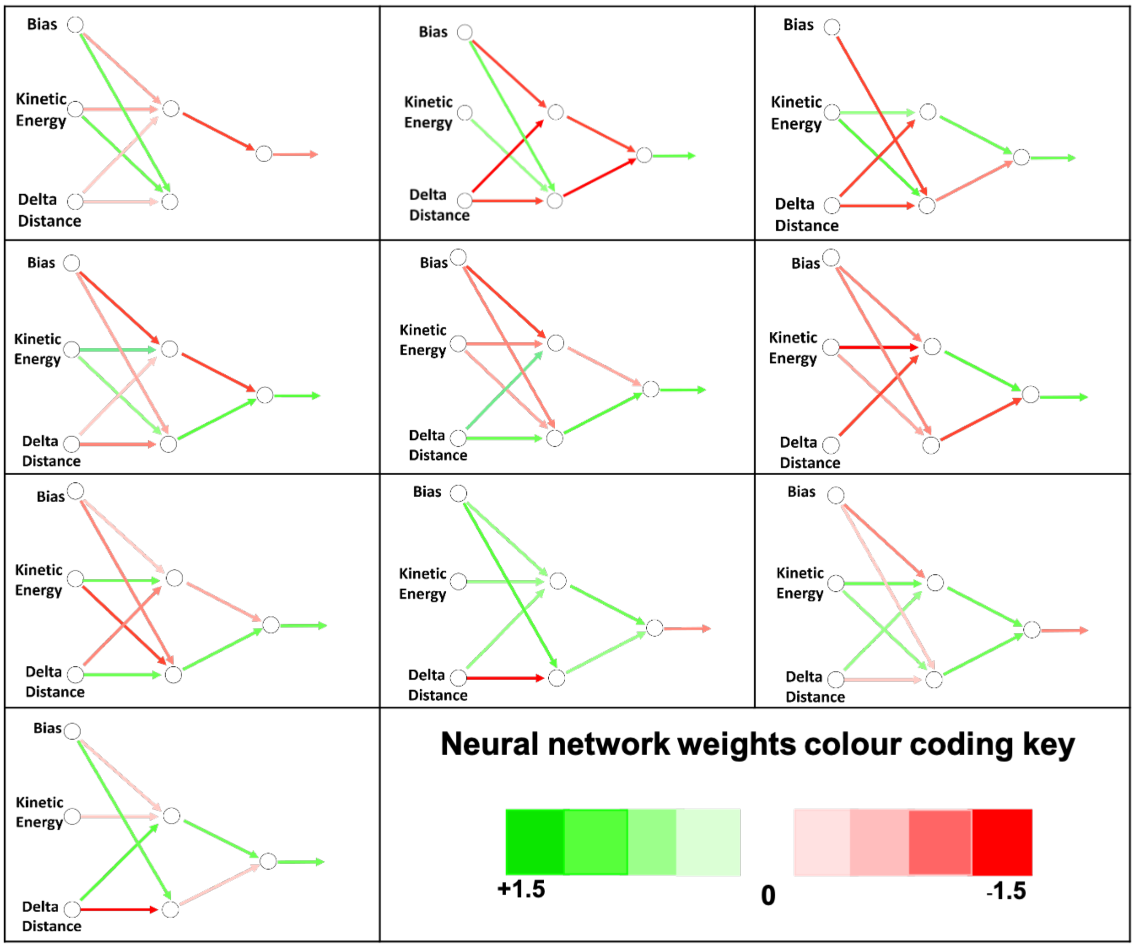


Figure 4.30: Figure shows the top 10 best performing genomes (neural networks) from each of the evolution runs where the fitness was reliant on success in all three environments. Here the strength of the synaptic weight is represented by the colour of the connection; a strong negative synaptic weight is a bright red, whereas a strong positive weight is a bright green. The morpologies produced by these neural networks are shown in 4.29 and the fitness shown in Figure 4.32

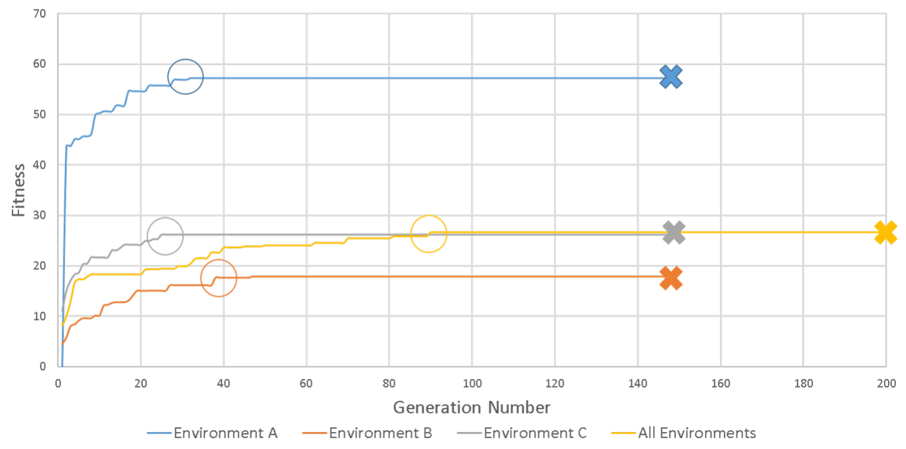
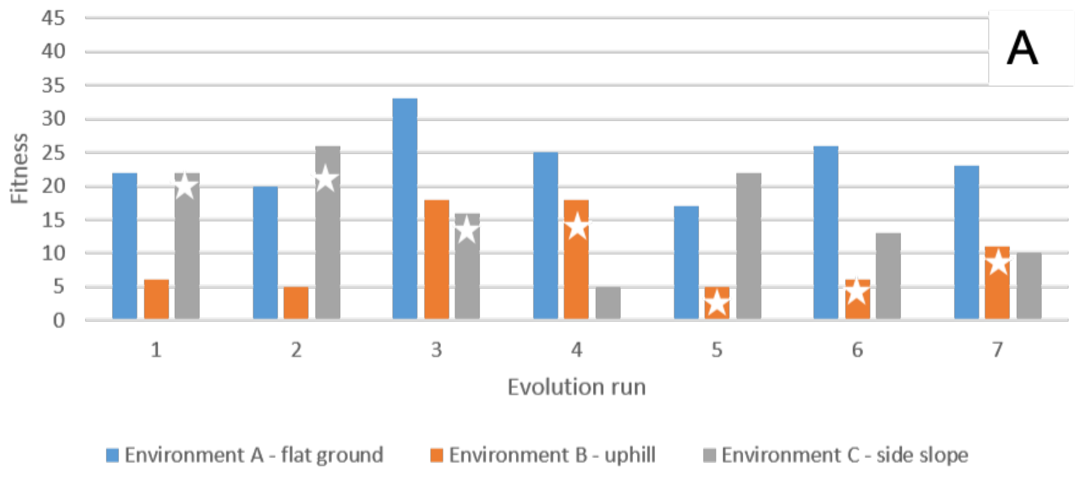


Figure 4.31: Figure showing the average generational fitness for each of the evolutionary algorithms tested in one environment compared with the evolutionary algorithm tested in all the environments. Note that the fitness for single environments is not strictly comparable to the fitness for the multi-environment evolution as this fitness is a weighted average of all three environments.

is very little zig-zagging and instead the performance steadily, but slowly, increases until the final episodes (with the exception of genome 4). Similarly in environment C there is much less zigg-zagging during the starting episodes that in the previous 7 genomes that were evolved for a single environment. In these cases the majority of the adaption is done by stiffness change only and it is not until the final few episodes (10+) that voxels are removed. The stiffness of the remaining voxels then continue to change to stabilise the robot.

To test the transferability of the top 10 genomes further, they were tested in 4 new environments for which they were not evolved. These environments were; 1) a downhill slope of 15 degree, 2) a 2D bumpy environment, 3) a 3D bumpy environment and 4) a "rolling hills" environment, as shown in Figure 4.34. These environments were selected to present new and diverse landscapes to the robot. Both the 2D and 3D bump environments were chosen so there was a different initial ground reaction force for the robot. Note that the difference between the 2D and 3D bump environment. In the 2D environment the ground height only varies in the y direction, the direction in which the robot is required to locomote. In the 3D environment the ground height varies in both the x and y directions, which added asymmetry and therefore added complexity to the environment. The rolling hills environment was selected as an increase in challenge for the robot. Initially the robot is on a small hill, directed downhill in the positive x direction. As the robot locomotes in a straight line the slope of the hill changes so that it now goes downhill in the negative x direction. These environments were also selected so that the height of the "bump" was never more than twice the length of a voxel as this was considered too unnecessarily challenging for the robot.

Figures 4.35, 4.36, 4.37 and 4.38 show the success of the top 10 genomes that were evolved for Environments A, B and C in the new environments, D, E, F and G. Also



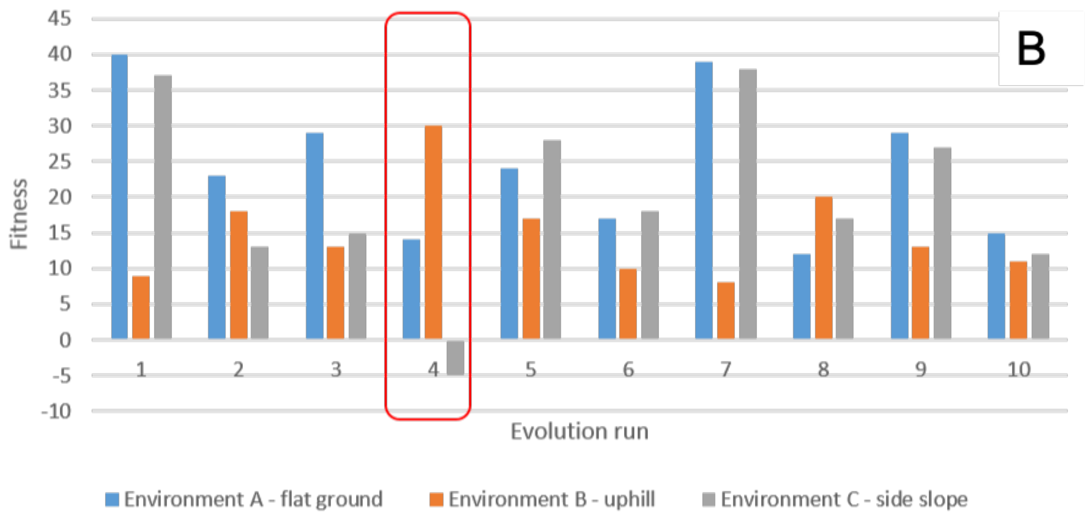
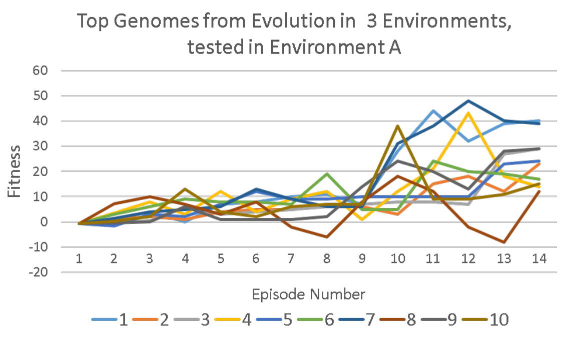
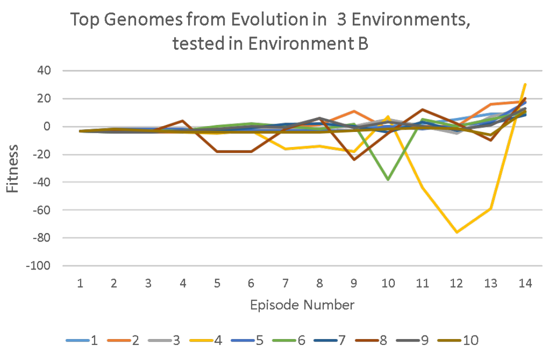


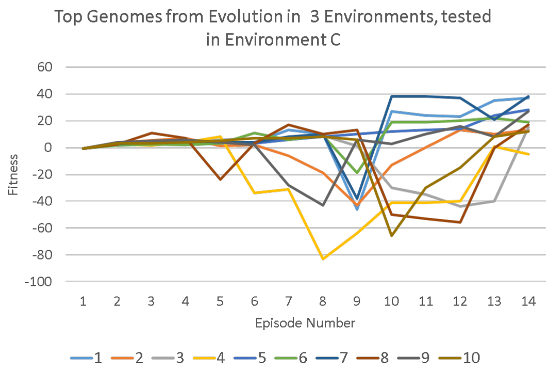
Figure 4.32: Figure comparing the success of the genomes evolved in all three environments with the transferable genomes evolved in single environment. The top graph (A) shows the transferable genomes from the previous tests but the axis is scaled to allow direct comparison with the bottom graph which shows the results from the evolution in all three environments. On the top graph the star symbolises the environment for which the genome was originally evolved for. The bottom graph (B) shows the genomes evolved for all three environments. The red bar highlights an instance where the genomes was not successful for all three environments – it is unable to locomote in the direction of actuation in Environment C and thus has a negative fitness.



(a)



(b)



(c)

Figure 4.33: Three figures showing how the far the robots, generated by the top 10 genomes from evolution in all three environments, travelled in each environment (i.e., the fitness) varies per episode. A) shows this variation for environment A (flat ground), B) shows it for Environment B (uphill) and c) for Environment C (side slope). Note the change in axis scale in (b).

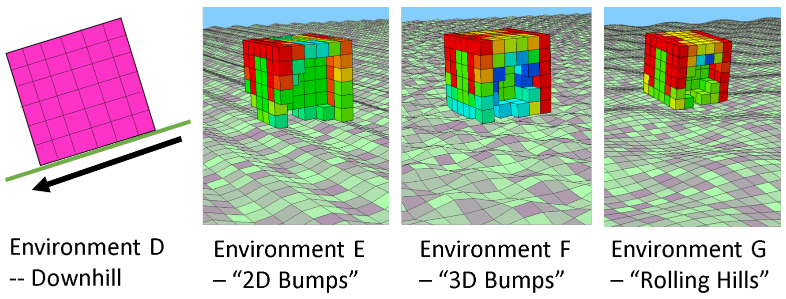

Environment D -- Downhill
Environment E – "2D Bumps"
Environment F – "3D Bumps"
Environment G – "Rolling Hills"

Figure 4.34: Figure showing the extra four environments for which the genomes were tested to ascertain further transferablity. Note that the robots included in these diagrams are not necessarily the most successful in these environments are were selected randomly to show scale.

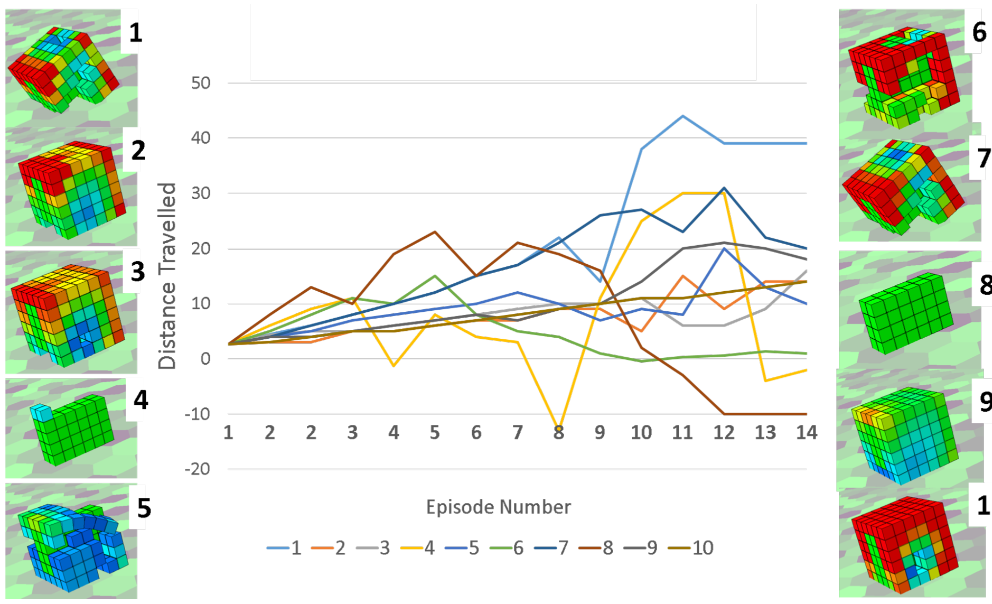


Figure 4.35: This figure shows the success of the top 10 genomes that were evolved for Environments A, B and C in the new environments D– downhill. Also shown in this figure is the final morphologies sculpted for this environment. Note that in this environment, the un-optimised morphology has a fitness of -3.2.

shown in these figures are the final morphologies sculpted.

Figure 4.39 shows a box an whisker plot comparing the key statistic from each of the environments. However, the plot shows that there is no obvious difference between the environments for which the genomes were evolved, compared with those for which it was not. For comparison, the fitness of the un-optimised starting morphologies, where the stiffness of each voxel is the same (5000N/m) for each environment are shown in table 7.

The majority of the genomes that were previously evolved only in Environments A, B and C were also able to find successful solutions in the later environments, Environments D,E,F,G. There are some exceptions; for example, genome 4 strug-

Environment	**Fitness of Starting (un-optimised) morphology**
A - Flat ground	-0.6
B - Uphill	-3.2
C - Side Slope	-1.2
D - Downhill	3.2
E - 2D Bumps	-3.5
F - 3D Bumps	-1.9
G - Rolling Hills	-5.8

Table 7: For comparison, the fitness of the un-optimised starting morphologies, where the stiffness of each voxel is the same (5000N/m) for each environment are shown here.

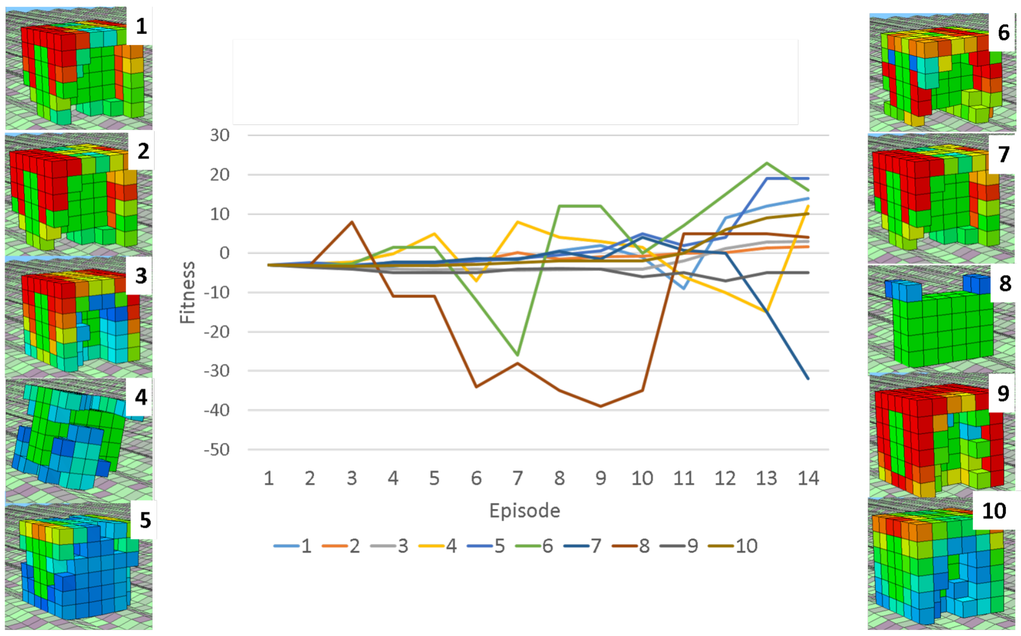


Figure 4.36: This figure shows the success of the top 10 genomes that were evolved for Environments A, B and C in the new environments E– 2D bump. Also shown in this figure is the final morphologies sculpted for this environment. Note that in this environment the un-optimised robot has a fitness of -3.5

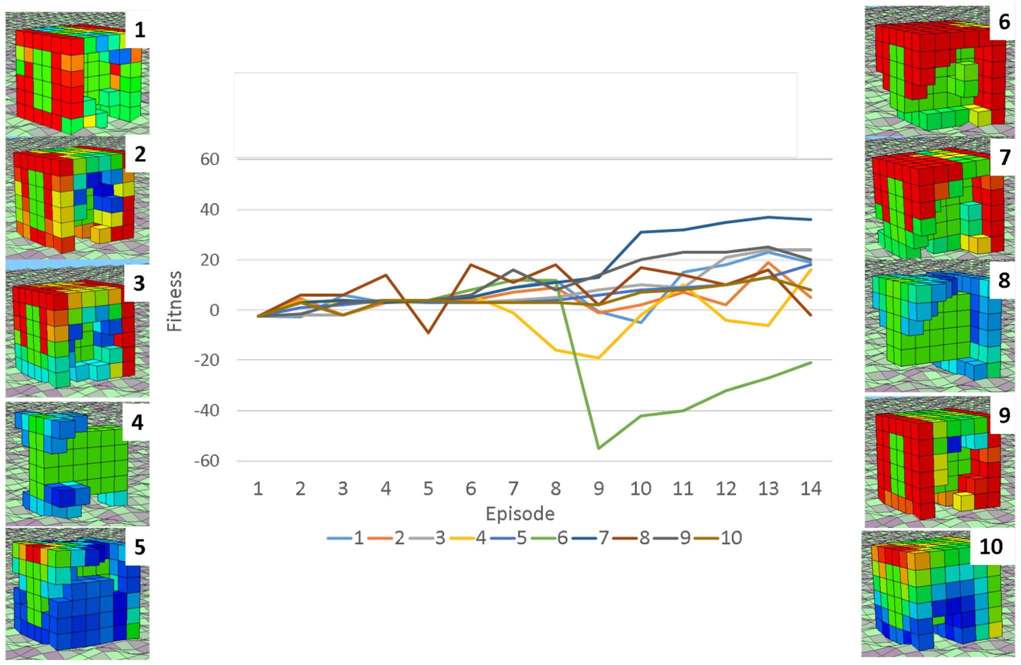


Figure 4.37: This figure shows the success of the top 10 genomes that were evolved for Environments A, B and C in the new environments F– 3D bump. Also shown in this figure is the final morphologies sculpted for this environment. Note that in this environment an un-optimised morphology has a fitness of -1.9.

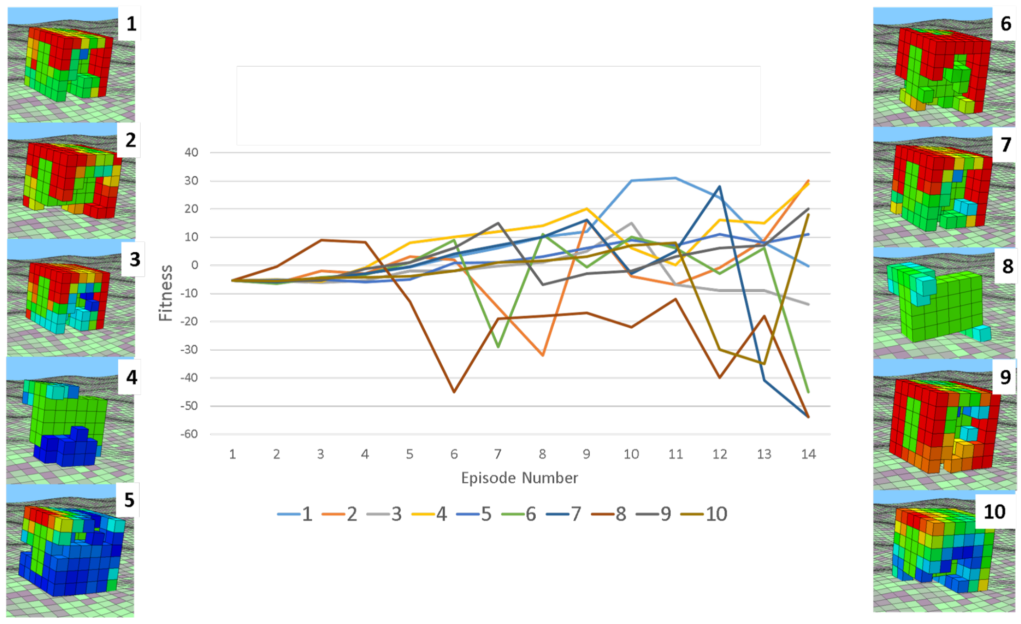


Figure 4.38: This figure shows the success of the top 10 genomes that were evolved for Environments A, B and C in the new environments G- Rolling Hills. Also shown in this figure is the final morphologies sculpted for this environment. Note that in this environment the un-optimised robot has a fitness of -5.8.

gles to find successful morphologies in the downhill environment. Remember that genomes 4 did not work in environment C. However, at least 50% of these genomes are successful in all these new environments.

Interestingly, it was initially assumed that the downhill environment would be easy for the genomes to sculpt successful robots. It was expected that the final morphologies would roll or bounce downhill in an uncontrolled manner. However, this does not appear to be the case. For three of the ten genomes the final morphology either falls onto its side (genomes 4 and 8) or turns and walks across the slope, and is thus penalised. Over the lifetime of the robot the success at each episode zig-zags from being high to low in a similar way the other environments. This zig-zagging is not necessarily negative, but it could be reduced by lowering the value of α or increasing the resolution of the voxels as discussed area in this chapter. The exception of this is for the genome 10, which only changes its stiffness and does not remove any voxels.

In the 2D bump environment the majority of the genomes were able to sculpt successful morphologies, despite not being evolved for this environment, i.e., they are transferable. For genome 9, the front of the robot is too stiff, and as a result the robot gets stuck between the the bumps, i.e., in the valley. In the case of the genomes 9, which is also unable to sculpt a successful morphology, it could be argued that this is simply due to the fitness metric used. In this the final morphology travels a long distance from its starting point, however, it is not in a straight line in the direction of actuation. As a reminder the fitness metric penalises the robot if it does not move in the direction of actuation. However, it could be argued that in this bumpy environment, genomes 7 is still successful as the final morphology does

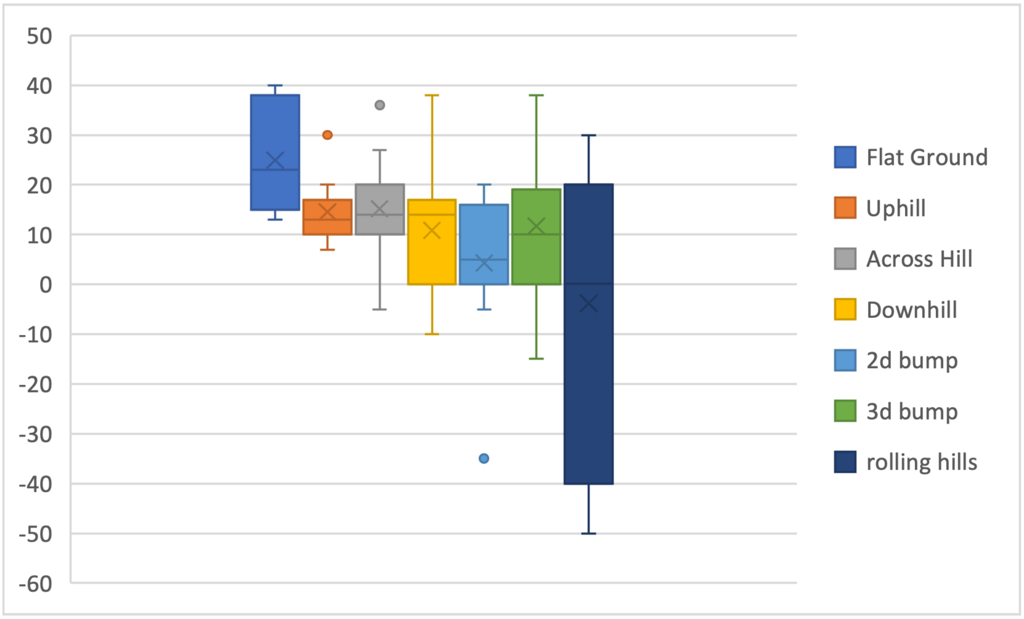


Figure 4.39: Box and whisker plot comparing the environments for which the robot were not evolved in, with the Environments A,B, and C. The plot shows that there are no conclusive findings.

not fall over or get stuck in the bumps and therefore the fitness metric may not the appropriate in this environment. This fitness metric may also be the cause of the "failure" of genome 6 in environment F (the 3D bump environment). When observing the fitness graph (Figure 4.37) it can be seen that the final fitness for genome 6 is very negative, i.e., it appear to perform very poorly. However, when observing the final morphology it is able to locomote far from its initial starting point, it moves in a straight line at an angle to the direction of actuation – which is why it has such a negative fitness. In terms of the current fitness metric is has failed, however in this situation whether or not the genomes has actually "failed" to locomote is questionable.

Another interesting observation of the results from the 2D bump environment is that for the majority of the genome only the stiffness changes for the first 10 episodes (i.e, for a longer number of episodes than the other environments). It is hypothesised that this is because of the lack of change in distance between episodes. It was observed that initially the robots get stuck in the first valley and therefore the change in distance between episodes, and only one of the inputs to the neural network is close to 0. It is not until the stiffness distribution of the robot has changed by a large amount that the robot is able to overcome this valley; at this point the neural network receives a stronger input and the voxels start to be removed.

The final environment, the rolling hills, was selected to test how the gnomes dealt with a non-homogeneous environment. The previous environments have all been homogeneous, i.e., however far the robot travels the ground reaction force will be the same. In the case of the rolling hills as the robot travels in a straight line this changes – the hills were designed so that there is a phase/wavelength mismatch between the environment and the robot control. Therefore, this environment can be used to tests how these genomes cope with a non-homogeneous environment. Note that if the phase/wavelength of the environment was a multiple to that of the robot, a result more akin to the 2D/3D bumps may occur. Given that it is particularly challenging for the robot to regrow voxels, it could be that the voxels that are removed at an early episode, where the slope goes downhill in one direction, are required when the direction of the slope changes.

There are some genomes, such as genomes 2 and 3, that are able to cope with this non-homogeneous environment, despite the fact that they did not experience this type of environment during evolution. Although perhaps further transferability could be gained by making it easier to grow voxels over the lifetime of the robot it is shown that a final morphology can be sculpted to deal with this environment.

Just because these genomes are transferable between these 7 environments does not mean that they would be successful in every possible environment. For instance, the size of the ground perturbations in Environments E,F and G are relatively small and it is unlikely that if the perturbations became too large the robot would still be successful. However, these results suggest that the evolved genomes have good adaptability to new environments.

4.5 Stiffness Adaption vs Voxel Removal

An observation from the results in the previous section shows that a gain in performance can be achieved from just adapting the stiffness of the voxels. Additionally, when a voxel is first removed the success of the robot is initially reduced. The remaining voxels then re-adapt to increase the success of the robot once more. This in fact results in a robot that is more successful than before the removal of the voxel.

To test this observation further, the top 10 genomes from the previous experiments, i.e., those from the evolution in Environments A, B and C, underwent a further investigation. In each environment the adaption was re-run but without any voxel removal. The stiffness of the voxel was capped at the threshold, so that the stiffness remains the same, rather than the voxel being removed.

Figure 4.40 shows the results of these experiments. In this figure the episodic fitness of the top 10 genomes (that were evolved in Environments A, B and C) are shown. Note that these are the same graphs as shown in the previous section but are repeated here for ease of comparison (i.e., Figures 4.26,4.27, 4.28). On the right hand side is the episode fitness for the same set of genomes, but here only the stiffness were allowed to adapt and no voxels were removed. It is easy to see that, in all cases, adapting the stiffness of the robot results in worse performing final morphologies than when the system is allowed to remove voxels. In fact many of the genomes completely converge on a final morpholgy that were less than those using voxel removal. However, it can also be observed that no episodes produce values that were highly negative and there is little zig-zagging between high and low values over the course of a lifetime. This could be an advantage if the robot needs to completely avoid an area outside its path and is destroyed if it deviates from its path too much. For instance, although the final morphologies generated through voxel removal would get along the straight path quicker. However, if they would need the previous episodes to learn to sculpt this morphology and the robot would have been destroyed in an earlier episode. However, as previously discussed, this zig-zagging effect observed when voxels are removed could be reduced by increasing the granularity of the robots, i.e., larger robots with more voxels. Then, when a single voxel is removed, the overall effect on the robot is reduced.

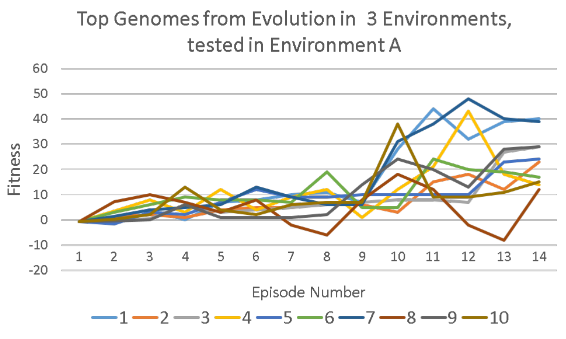
Also, as previously discussed, a lower 'threshold' value was assigned to the voxels, i.e., if the stiffness of a voxel was to be changed to below this threshold, instead the stiffness would remain fixed. This threshold value was set in response to an observed error in the simulation software; where the software would fail if trying to simulate voxels with very low stiffness. It is feasible that if this threshold was set much lower, just adapting the stiffness of the robot would be able to yield a better performance than the one reported here.

4.6 Investigating Different Starting Populations

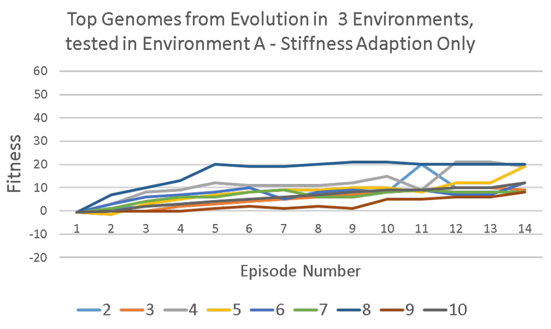
In this section, the idea of transfer of knowledge between evolution in environments was investigated. The previous results indicated that there might be environments that are more complex than other – for example, Environment A the flat ground environment appear to be very simple. It should be easy for the evolutionary algorithm to find a successful morphology. Furthermore, for the majority of cases genomes evolved in Environments B and C (environments that could be considered more complex) are usually transferable to Environment A.

In this section, I investigated whether there were any benefits from first evolving genomes for one environment and then using the best genomes as a starting population for another environment. For instance it could be hypothesised that solutions that were first evolved in a simple environment could then accelerate evolution in a more complex environment and perhaps result in a final better performing solution. Alternatively, given that solutions evolved in Environments B and C already work to some extent in Environment A, this could also result in better performing solutions in Environment A.

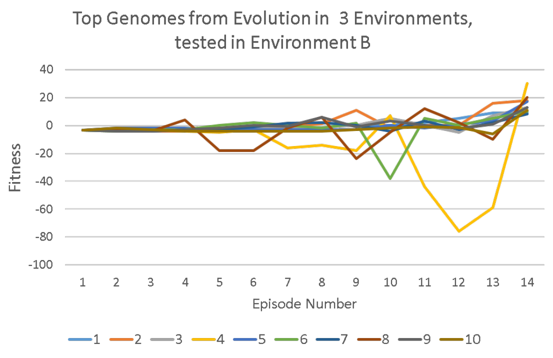
Therefore, the top genomes found at the start of this chapter, i.e., those that were evolved in the individual environments, were used as a starting population for the other two environments. To clarify the top genomes from each of the evolution runs in environment A were used as a starting population of 10 genomes for evolution in Environment B. These 10 genomes were then tested in Environment B, ranked and then mutated to form a new generation (population size of 40 as with the previous



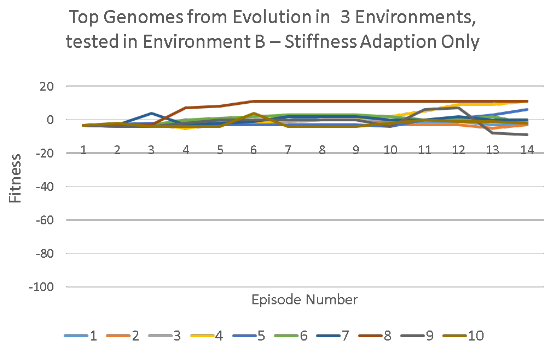
(a)



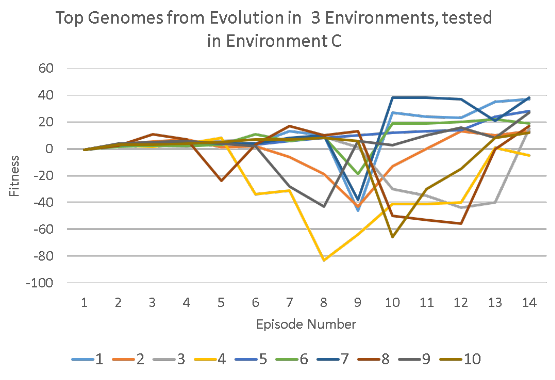
(b)



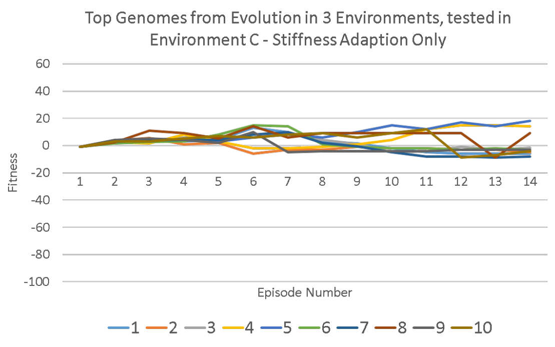
(c)



(d)



(e)



(f)

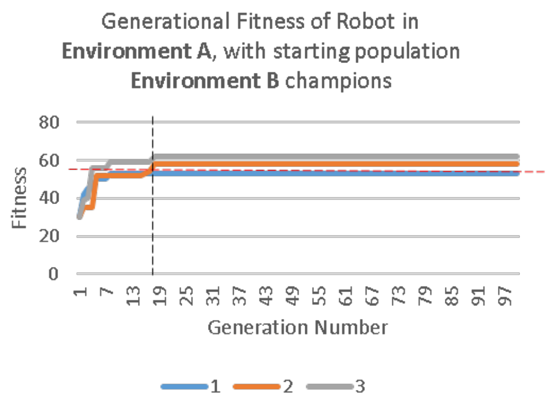
Figure 4.40: This figure shows a comparison of the top 10 genomes, evolved from testing in Environments A,B C when they are able to remove voxels and when the only adapt stiffness. Graphs a,c,e show when voxel removal is allowed and graphs b,d and f show when only the stiffness of the robot is allowed to adapt.

experiments). This evolution process then continued for 100 generations. This experiment was then repeated for Environment C. The results of these are shown in Figure 4.41, a and b. Then the top genomes from the evolution in Environment B were used as a starting population for evolution in Environment A and Environment C. The results of this evolution is shown in Figure 4.41 c and d. Finally, the top genomes from evolution in Environment C are used as a starting population for evolution in Environment A and B. These results are shown in Figure 4.41, e and f.

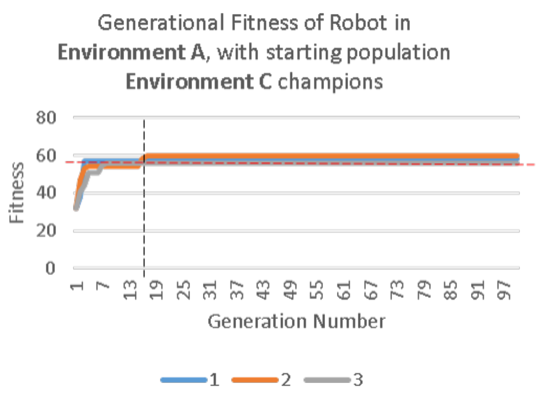
As the starting population of each of the evolution runs is the same, i.e., not randomised, the evolution runs were repeated 3 times for each testing environment. This to to allow for randomisation in the mutations when creating the new population for the next generation. Also shown on the graphs in Figure 4.41 is the mean fitness from the initial evolutions in each environment (shown by the red dotted line).

In some cases the evolutionary algorithms do converge on solutions much quicker than when they were started with randomised populations. Additionally in almost all the repeated runs the evolutionary algorithms were able to find solutions with a similar or higher fitness to the mean found previously. However, these values are not a lot higher. In some cases the amount of time taken to converge to these solutions is similar or slightly longer than the original evolution runs.

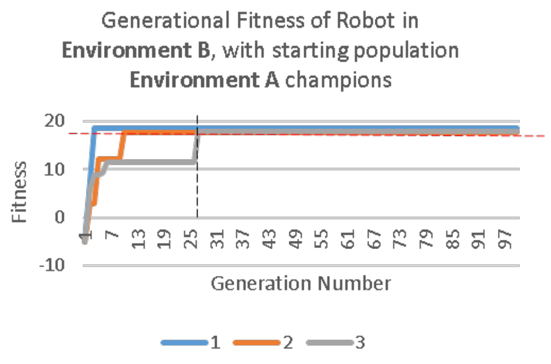
Therefore, these results show that although there is very little benefit to first evolving in one environment and then transferring the best solutions to another environment. Additionally it does not appear to matter whether or not the original environment was more or less complex than the later environment.



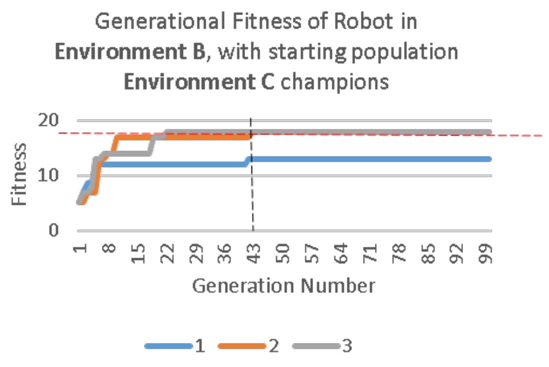
(a)



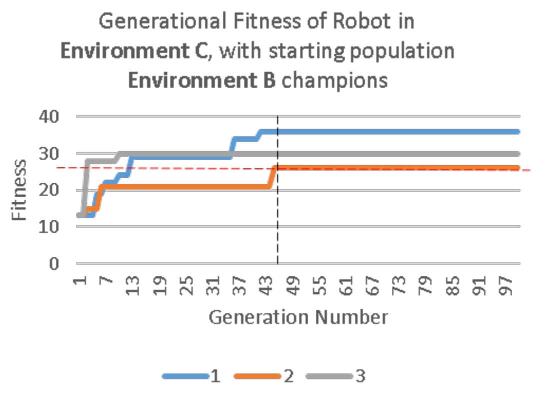
(b)



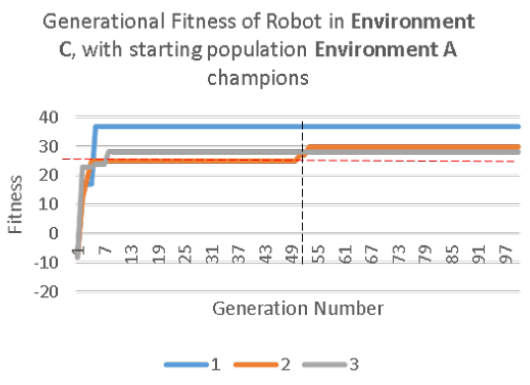
(c)



(d)



(e)



(f)

Figure 4.41: Generational fitness of the evolutionary algorithms for when the starting populations are the best genomes from other environments. Also shown in this figure is the mean fitness from the original evolution This is represented by the red dotted line.

4.7 Damage Recovery

As shown in the previous sections, the majority of the optimal genomes found follow a similar "two-step" method of adaption. First the stiffness distribution in the robot was changed; then at around episode 8 the softest voxels were removed. After these initial voxels have been removed very few others follow. Instead, the remaining voxels stiffened to stabilise the robot. This adaption method suggests that the robot may have the capability to recover from damage by readjusting its stiffness. Therefore, in this section, this hypothesis was tested. Firstly, the 7 transferable genomes, i.e., those from the first part of the chapter were tested for damage recovery. Remember, these genomes were those that were evolved in a single environment, but happened to be able to find successful solutions in the other environments. These genomes were tested to investigate whether their learning experience in a certain environment affected their damage recovery.

The method used to test the damage recovery ability of these transferable sculpting adaption systems is similar to the original sculpting methodology. The robot is first simulated in its un-optimised form, i.e., a 6x6x6 cube of voxels with a stiffness of 5000N/m. Then, using the transferable sculpting adaption systems found previously, the robot starts to adapt over a number of episodes as before.

Half way through the lifetime of the robot (episode 8), four voxels at the front left of the robot were removed, i.e., the robot was damaged. The position and amount of damaged (removed) voxels was selected as it was considered to be an acceptably large enough challenge for the robots to recover from. Episode 8 was selected as the time at which the voxels should be removed – this was the episode at which the genomes would start to remove voxels themselves. The robots were then allowed to adapt, again following the transferable rules.

It has previously discussed that many of the robots experienced a decrease in performance after any voxels have been removed – even when this removal is part of the adaption method. Therefore, I anticipated that allowing the robots to remove voxels as part of the adaption method may be too large a challenge if also required to perform damage recovery. Therefore, instead the lower stiffness value was capped. If the stiffness of a voxel was to be decreased below the capped value, whereas previously the voxel would have been removed, in these experiments the voxel remained but with a fixed lower stiffness. Therefore, the only voxels removed were those deliberately removed, i.e., the front four.

Figure 4.42 shows how each of the 7 "transferable" genomes sculpted the initial robot over the 15 episodes in environment A. In the figure it can be seen that at episode 8, four voxels at the front right of the robot were removed, i.e., the robot experiences damage. From then on each of the robots redistributed the stiffness of the other remaining voxels, without removing any more voxels themselves, in an attempt to maintain successful locomotion, i.e., to maximise the distance travelled. Figure 4.43 shows the distance travelled in the x direction at each episode. It can be seen that before episode 8 the distance travelled increases at each episode. When

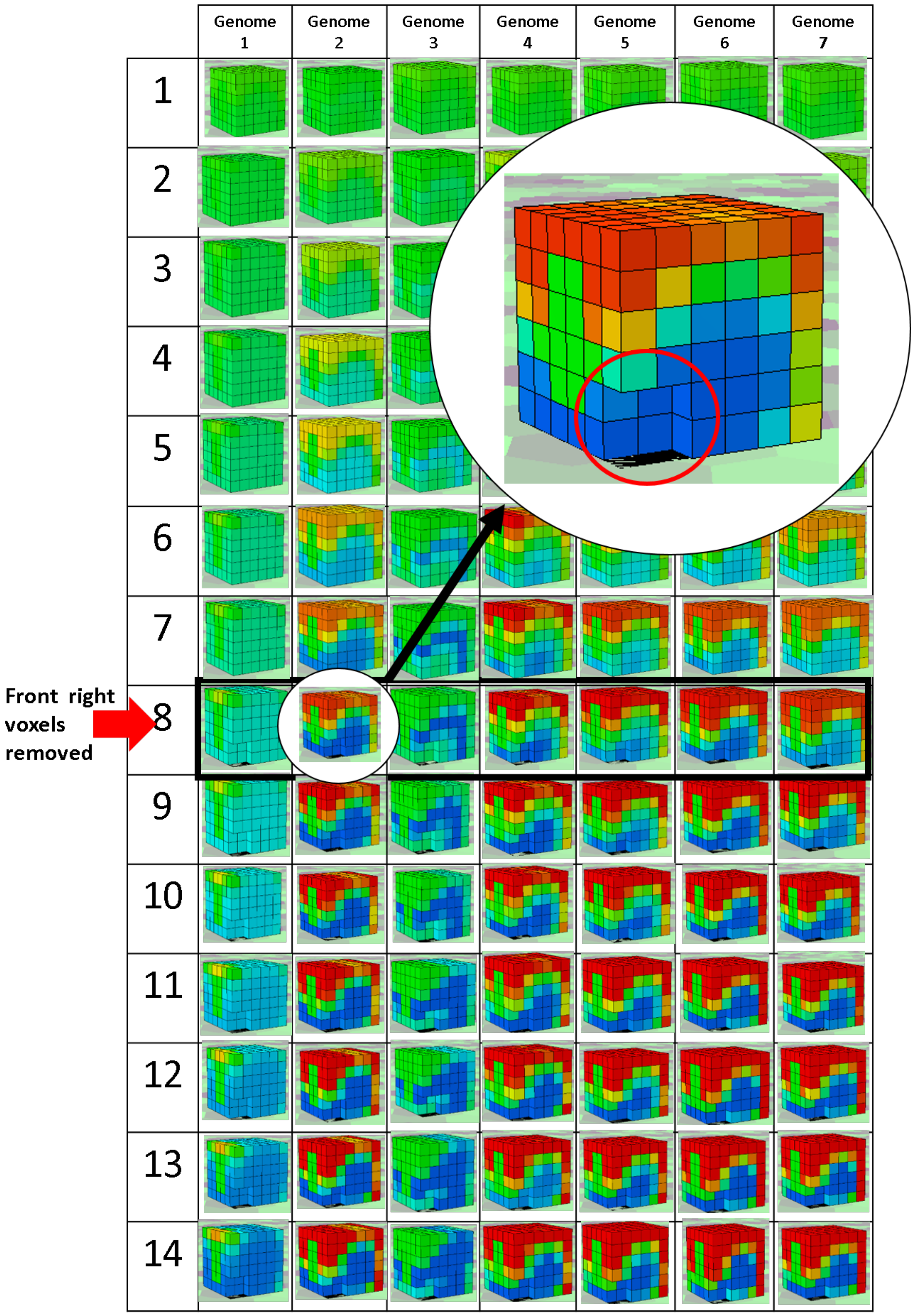


Figure 4.42: Figure shows how the 7 "transferable" genomes sculpt the initial robot over the 15 episodes in environment A - the flat ground. After episode 8 four voxels at the front right of the robot are removed. The robot then changes its stiffness in an attempt to counteract this damage. In this instance only the stiffness is changed and the robots are not allowed to remove voxels themselves.

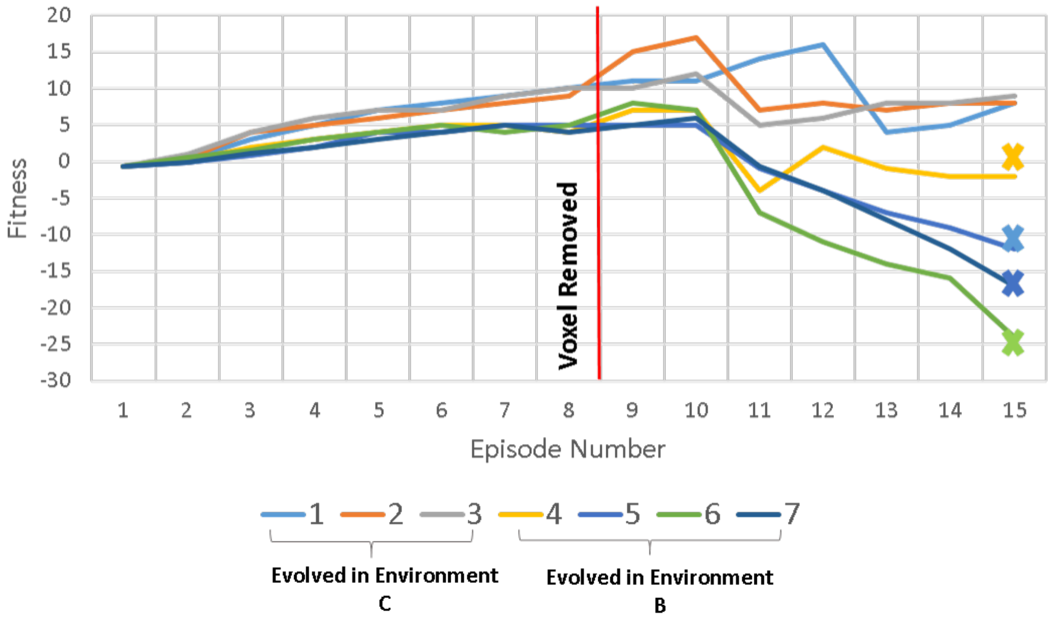


Figure 4.43: Figure shows the success of the 7 "transferable" genomes at each of the episodes in environment A. At episode 8, four voxels are removed from the front right of the robot. Note that only the stiffness of these robots are automatically changed; the robots are not allowed to remove the stiffness themselves.

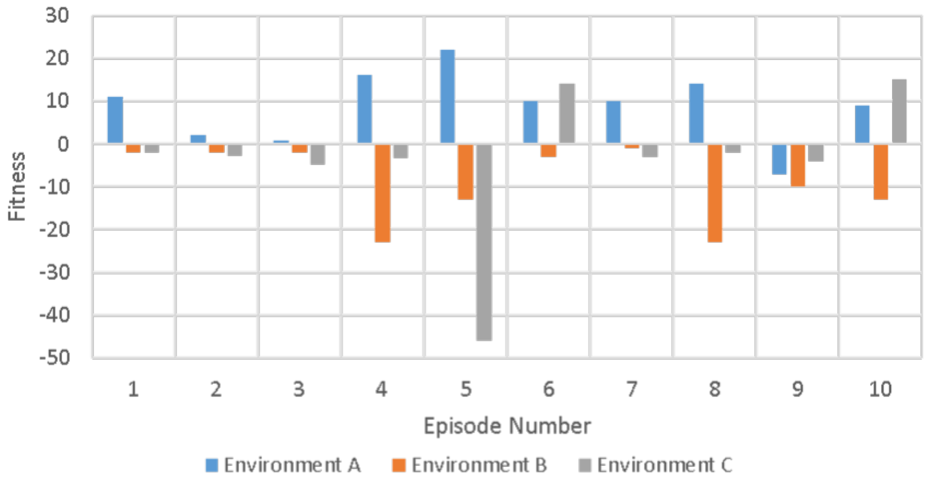


Figure 4.44: Figure shows the success of the final morphologies, developed using ten champion genomes found from evolution in three environments but also incorporates damage (voxel removal) in episode 8.

the front right voxels were removed the performance decreased in the majority of the robots. Although it is interesting to note that this decrease was delayed and did not occur for a few episodes. In the majority of cases the loss of performance was due to the fact the robot now twisted its body, when the central active voxels expand and contract, towards the area of removed voxels as it faced less resistance than the other side with the stiffer voxels. This means that the entire robot started to locomote with a bend to the right (i.e., in the direction of the removed voxels). It can be seen in the two robots formed by the genomes 2 and 3 that over the episodes following the "damage" the voxels immediately surrounding the site of removal started to stiffen up. Additionally, the back right hand voxels were also stiffened more than the left hand side which further combated the twisting affect caused by the removed voxels. Therefore, the robots following these two genomes were able to re-adapt to counteract the damage. In contrast, the robots following genomes 4-7 were unable to counteract the damage. Although they too adapt in response to the loss of voxels, this time it is the voxels on the left, opposite to the site of removal that were stiffened. This means that the active voxels had even more resistance to expansion on the left hand side so twisted even further towards the right. Therefore, over these final episodes the performance of these robots got worse (as seen in Figure 4.43). Genome 1 performed differently to the other genomes. The stability of genome 1 comes from its extremely soft body, rather than optimal body sculpting. It is interesting to note that all the genomes that are successful in combating "damage" are those that are initially evolved in Environment C, which itself is asymmetric (similar to the area of voxel removal). If it is the case that the genomes must have experienced an asymmetric environment then it would expected that all the genomes from the next sets of testing (i.e, those evolved in all three environments) should have the ability to recover from this damage. Therefore, to test this hypothesis, the 10 champion genomes from each evolution run (where the genomes were evolved in all three environments) underwent damage recovery testing. That is they were allowed to only adapt their stiffness and at episode 8 four voxels at the front left of the robot were removed. Additionally, as before, these genomes were also tested for damage recovery in Environments B and C. Figure 4.44 shows the results of this testing.

Given that the criteria for a "successful" genomes is one where the final morphology has a positive fitness, 9 out of 10 of the champion genomes were successful in Environment A. However, it should be noted that in some cases the fitness is very small. Additionally in the other two environments very few of the genomes were successful at all (only two in environment C). Therefore it can be hypothesised that the second two environments are too challenging for the genomes to recovery from damage in.

Finally, for a comparison the 10 champion genomes were again tested for damage recovery in Environment A. However, this time the lowest stiffness value was not capped and they were allowed to remove voxels themselves (in addition to the 4 voxels being removed at episode 8). For simplicity, only Environment A was tested. The final morphologies produced are shown in Figure 4.45. Also shown in Figure 4.46 is the performance of the final morphology when the robot was not allowed to

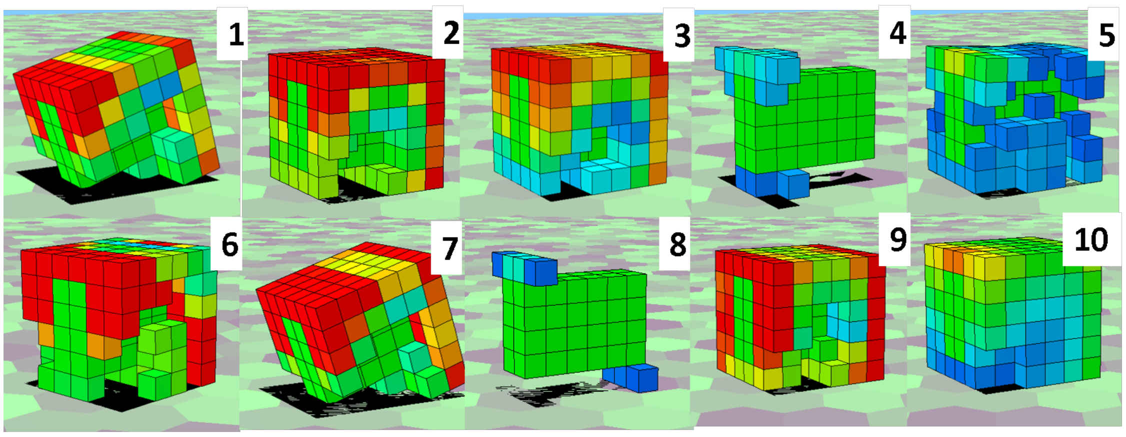


Figure 4.45: Figure shows the final morphology of the 10 champion genomes that were damaged at episode 8, but were also allowed to remove voxels themselves (instead of having their lower stiffness capped).

remove voxels itself.

From Figure 4.46 it can be seen that 7 out of 10 of the champion genomes were able to recover from damage by removing voxels as well as adjusting their stiffness. However, this could be misleading as in some cases (for example, genome 1) the voxels removed as a result of "damage", were going to be removed anyway by the algorithm at around episode 8.

The results from this section are somewhat inconclusive. Whilst the tested genomes show some ability to recover from damage, in general they only work in one environment (Environment A which is considered to be the most simple). Additionally, the fact that these genomes are able to recover in one environment could indicate the possibility of using a evolutionary algorithm to evolve specifically damage resistant genomes capable of re-adapting their morphology after voxel removal.

4.8 Chapter Conclusions

In this chapter, evolving neural networks that use feedback from the environment (in the form of kinetic energy) to adapt the stiffness of a voxel based robot (and therefore sculpt a final morphology) was explored. The results from this chapter show that a single neural network is capable of sculpting successful morphologies in a number of discrete environments, including some for which it was not evolved for.

Also investigated in this chapter was whether or not there was a benefit to first evolve adaption in one environment and then transfer the knowledge (i.e., using the optimal genomes as a starting population) to a new environment. The results show that there may be a slight benefit to using this approach.

Additionally, using the evolved adaption rules to enable damage recovery was inves-

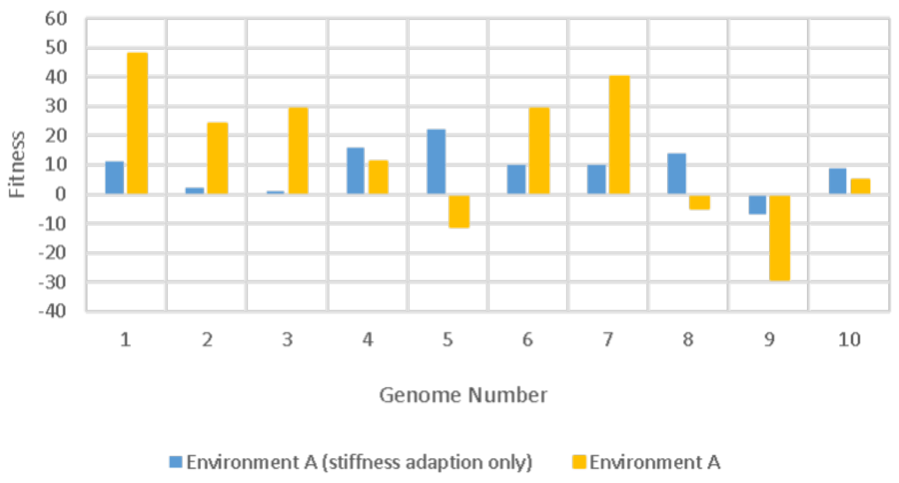


Figure 4.46: Figure shows the performance of the final morphologies created from each of the 10 champion genomes. The blue data show the scenario where the stiffness was caped at a lowest value -i.e., the robot was not allowed to remove its voxels as part of its adaption methodology. The yellow data shows when the robot was allowed to remove voxels as part of the adaption.

tigated. The results show that the evolved genomes are able to adapt the robot to recover from damage in Environment A.

Whilst the results from this chapter show good potential, there are a number of considerations for future work. These are discussed below.

One extension to this research would be to consider adding voxels as well as removing them. This two way adaption (adding and removing body parts) has the potential to create robots that are capable of even more sophisticated adaption. This could be achieved by adding an additional rule to the existing infrastructure in which if the stiffness of a voxel exceeded a certain threshold extra voxels, surrounding the “too stiff” voxel in question, should be added. In the next episode these voxels would then adapt their stiffness and have the potential to be removed if their stiffness became to low. This mechanism for adding voxels to the robot may also be beneficial in the damage scenario, as it would give the potential for the robot to simply regrow the removed voxel. Although it is interesting to note that recovery of the robot can be achieved simply by re-configuring the stiffness of the remaining voxels it would be interesting whether the ability **regain** voxels would increase the speed and success of recovery.

Although potentially adding the ability to “grow” voxels as well as remove them would not required any changes to the system infrastructure, particularly the neural network, another potential limitation of the work presented here is the simplicity of the sculpting adaption system. Currently the neural network, which forms the sculpting adaption system, has only one hidden layer and no recurring nodes, although the role of the recurring nodes is fulfilled by the “change in distance” input to the neural network . It is therefore possible that more robust adaption mechanisms could be produced if a more complicated neural network, perhaps with more hidden layers, was used. Another option for adding complexity to the neural network without adding extra hidden layers could be to consider the number of inputs. In this work each voxel takes into account its own kinetic energy and only indirectly considers the other parts of the robot through the use of the "difference to average kinetic energy." As a reminder, the average kinetic for all the voxels during an episode is calculated and then the input to the neural network is the difference between this average value and kinetic energy specific to the voxel in question. Whilst it has been shown that this yields good results another interesting approach would be to take inspiration from cellular automata and input the kinetic energy of the voxel neighbours also into the neural network. The approach of using cellular automata to create virtual creatures, in general, has been previous explored, e.g. by [16, 81, 18]. In these previous works one single rule is applied to each and every cell in the robot and this rule dictates how the specific cell should alter its state depending on the state of its neighbours. Whilst these studies have been successful in generating diverse and well performing virtual creatures, only the **state** of the surrounding neighbours is considered, rather than the neighbours interaction with the environment. An extension to the work presented in this thesis would be to add the kinetic energy of the voxels neighbours into the neural network as additional inputs to explore whether this would result in more robust rule sets.

In this chapter, the sculpting methodology has only been tested on one initial morphology with a single locomotion control system. Remember that in this chapter the control system is the active voxels in the centre of the robot with a phase shift applied front to back. As future work it would be interesting to study how the effect these parameters have on the sculpting adaption system and the final morphology produced. Is it possible to find a single adaption system that is capable of sculpting success morphologies no matter the locomotion control system or starting morphology? Additionally, the robots are all started with the same fixed orientation. It would be interesting if this starting orientation was also altered as it is my expectation that this would result in an increase in final morphology diversity.

An interesting hypothesis, that has arisen from these findings and could be investigated in future work, is does this removal during development of a morphology evolve better performing robots? If only the final morphology of the robot was evolved, it could be hypothesised that an evolutionary algorithm may get stuck in local optima and not be able to navigate through the area of poorer performing robots to access the better solutions. Does evolution need damage during a developmental stage to yield better performing robots? However, as previously discussed, the requirement of damage during the development phase could only be necessary if using a threshold value for voxel removal. This is a interesting hypothesis and should be considered for future work.

One area that has shown promise in this chapter is the exploration into damage recovery. Despite not being evolved for it, these robots are able adapt their morphology to recover from damage. However, this has only been shown to work in one environment and with one scenario. Therefore, more research into directly evolving damage recovery should be considered.

A potential limitation of the work presented in this chapter is the episodic learning approach. Although this approach has been shown to be powerful in creating adaption methods capable of sculpting successful morphologies for different starting environments. The sculpted robots are capable of locomoting large distances so long as the environment they are in does not suddenly change. For example, if the robot was initially sculpted in Environment A, and then the environment suddenly changed to Environment B, it is not currently clear whether or not the robot would be able to adapt to this changed environment. It may be successful using an offline approach, i.e., episodic learning. However, it may require an online learning approach. An online learning approach could utilize a neural network with recurring nodes (as suggested earlier in this section) and update the stiffness of the voxels either homogeneously or after a fixed number of time steps.

As discussed earlier in this chapter, there are some limitation to the work presented here. For example, the use of kinetic energy for the key parameter was chosen somewhat arbitrarily. Although the results in this chapter have shown kinetic energy to produce robots capable of adapting over their lifetime to a variety of environments, using a variable such as pressure would be more consistent to existing literature (e.g., Kriegman et al. [62, 61]). Additionally pressure, or simply velocity, may be easier to measure if the method presented in this thesis were to be explored in real life (rather than simulation).

Another limitation is the design choices of the evolutionary algorithm. Using a better designed EA may also yield better, more diverse results.

In conclusion, the main aim of this chapter was to use evolutionary algorithms to find optimal method of adaption that allow a robot to change its morphology to achieve success in a wide variety of environments.

5 Conclusions and Future Work

In this chapter, I will discuss the findings from the entire thesis and how these relate to the current trends in the relevant research area of developmental robotics. I will also discuss potential future areas of study, based on the research questions that have been raised as a result of the work presented.

As first presented in the introduction section, the key aims and objectives of this thesis were:

- To investigate the use of evolutionary algorithms to evolve methods of adapting the morphology of compliant robots in order to achieve robust locomotion.
- To carry out the investigation of evolving methods of adaption in two contexts. Firstly, using the theoretical Spring Loaded Inverted Pendulum model as a "base robot". Secondly, using a more complex voxel-based robot as a platform for adaption.
- To determine how transferable, and therefore robust, the evolved methods of adaption are in terms of performance in different starting environments and different starting morpologies

I believe that throughout this thesis, the above aims and objectives have been met. In chapters two and three, I investigated how the adaption of the morphology and control of the theoretical Spring Loaded Inverted Pendulum model could improve its overall robustness. In the first chapter, I devised a number of rule sets that changed either the spring stiffness (the morphology) or the attack angle (the control) of a Spring Loaded Inverted Pendulum model based on interactions with the environment. Remember that if the combination of morphological and control parameters in the SLIP model are within a particular range, i.e., the J-Figure, the model will be stable and be able to locomote indefinitely without falling over. In the first set of experiments, the interaction with the environment was simply defined as the distance that the SLIP model was able to travel with the current set of morphological and control parameters (i.e., the combination of the spring stiffness and attack angle) before the pendulum fell over. The adaption took place episodically, that is the SLIP model was allowed to fail, was reset, parameters updated, and locomotion was attempted again. Each rule set uniquely dictated how the SLIP model should adapt its parameters, based on the comparison between distance travelled in the current episode with the distance travelled in the previous, in order to find stability.

In Chapter 2, the adaption rule sets were systematically tested to find the optimal way of adapting the morphology and control of the robot separately, i.e., the optimal rule sets found adapted either the spring stiffness or the attack angle, but initially not both. The results from this chapter show that just the adaption of one parameter can expand the amount of combinations for which the SLIP model could become stable.

Crucially, the adaption via the proposed rule set methodology allowed the robot to learn to become stable when started with an unstable combination of parameters.

The research in the Chapter 3 also related to the adaption of the SLIP model. In the first part of Chapter 3, the episodic adaption was considered further – this time both the stiffness and the attack angle were allowed to change simultaneously after each episode and an evolutionary algorithm was used to determine the optimal "combined" rule set. Allowing both sets of parameters to adapt meant that the amount of starting parameter combinations for this model could become stable increased even more. For the tested starting combinations over 98% were now able to be adapted so that the SLIP model became stable (compared with 3% when no adaption was present in the SLIP model, 78% when only the attack angle was adapted and 51% when only adapting the stiffness). The presented adaption via rule set methodology has the potential to increase the robustness of robots based on the SLIP model.

Whilst previously in literature, there have been a numerous of examples of work extending the region of stability of the SLIP model, e.g., through extra leg segments [98, 99], there were previously no models in place that allow the SLIP model to adapt itself to become stable if started with unsuitable morpholgical and control parameters. Whilst this is not necessarily a problem – a robot could be designed with an appropriate spring and joint mechanism, over time the lack of ability for the robot to adapt could become a problem. Mechanical parts such as springs and joints are subject to wear and tear, for example, over time the stiffness of a spring could decrease. A robot, based on the SLIP model, without the ability to adapt, would now fail as the combination of its parameters are not within the stable range. However, if an adaption system such as that described in this thesis was implemented, the attack angle could be adapted and stability could be regained. The investigations carried out in this thesis are entirely in simulation and the above damage scenario is simplistic, however, it is hoped that the methodology I have presented could be implemented on a real robot in the future, or used as a starting point, to increase robustness in robots based on the SLIP model.

Also in Chapter 3, the case of online adaption was explored. Whilst the benefits of the offline adaption (i.e., episodic, where the robot is allowed to fail and adaption takes place after failure) have been discussed above, there are some limitation of this approach. The main one is the requirement of failure – this could cause further damage to any physical robot which has the potential to be catastrophic. Therefore, the offline rule sets were adapted to allow for online learning, where the SLIP model instead adapts its parameters between strides in order to find and maintain stability. Whilst this online method is not able to find stability for as large a number of starting parameter combinations as the offline approach, my results show that it is able to cope with some environmental changes without failure. In addition, using optimal rule sets the SLIP model is able to overcome large downward steps in ground level without falling over by adapting both its stiffness and attack angle.

In these two chapters, investigation of optimising methods of adaption for the theoretical Spring Loaded Inverted Pendulum model has been carried out, achieving part of the second aim of this thesis. Furthermore, in Chapter 3, I used evolutionary algorithms to evolve these methods of adaption, completing the first aim of the thesis. In both chapters I test how transferable these rule sets are, by considering their performance over a large range of starting robot configurations. In Chapter 3 I also investigate how these rule sets are able to deal with environmental changes (aim 3).

In Chapter 4, the concepts presented in the first half were explored in a new context. The SLIP model is theoretical and simplistic, so the ideas of adaption were explored on a more complex system - a modular (voxel) based robot. This part helps to achieve the second key thesis aim. Also, the concept of adapting to different environments was explored further achieving the third thesis aim. In this part of the thesis I wanted to further determine whether or not it was possible to evolve a method of adaption that would allow a robot to change its morphology, based on its interaction with a particular environment. I also wanted to explore the potential to adapt to a wider range of new environments. Additionally, instead of using "rule sets" as a method of adaption, how the robot changes its morphology was dictated by a simple neural network. It is the weights of the neural network that were evolved via the evolutionary algorithm.

Chapter 4 shows that is it indeed possible to evolve single adaption methods (i.e., a single neural network), which alters the morphology of a robot based on its interaction with the environment, to produce final morphologies, in a range of environments. This morphological adaption is achieved by adapting the stiffness of each voxel until the stiffness is altered below a specified threshold and the voxel is removed. The robot starts out as a cube of 6x6x6 voxels and a final successful morphology is therefore sculpted out of this initial robot. The final morphology sculpted in each environment is different, however the adaption method crucially is the same. Not only does a single neural network sculpt successful morphologies for all three environments for which it was evolved it, the found genomes show transferability. That is they are able to sculpt successful morphologies for environments that they did not experience during optimisation.

Also in Chapter 4, I investigated the benefits of first evolving adaption systems in one environment and using the results from this evolution as a stepping stone (i.e, a starting population) for evolution in the next. I also investigated the benefits of removing voxels, versus just adapting the stiffness and whether the current evolved adaption methods were able to be used for damage recovery.

When considering how the results from this thesis relate to currently published literature, I argue that it most closely aligns with the work published by Kreigman et al. [62, 61] which related to postnatal development of the stiffness of voxels in a soft voxel based robot. In Kreigman's work the stiffness of each voxel is linearly changed over the lifetime of the depending on the pressure/stress of the voxel. It is one of the few examples of where robots have changed their morphology after "birth" based on environmental feedback. However, the morphological change of the robot is limited to stiffness change rather than overall body shape change, and the key aim of the research is to improve robot locomotion in a single environment, rather than to generate adaption between environments. Whereas, in my work presented in this thesis, I evolve the way the robot interprets environmental feedback from *different* environments through use of adaption rules/neural networks. It the first part of the thesis, like Kreigman, I only adapt the stiffness of the robot morphology, albeit on a simpler robot that Kreigman used. However, in the later chapter, I advance this area of research further by allowing the robot to remove voxels and thus change its shape.

In the later chapter regarding voxel based robots, the robots adapt to different environments by, in some cases, removing up to 40% of their voxels. This large amount of shape change is similar to origami robots which are also about to change to different body shapes after initialisation, e.g., Miyashita et al. [75], Kotikian et al. [59] etc. However, these origami robots usually snap between a few hand designed morphologies which the human has determined optimal for the environment or task. Whereas, in the work presented in this thesis, evolution is used to determine the way the robot should use information to adapt (develop) to new environments. This way a single evolution adaption system has the possibility to develop the initial robot into a wide variety of different morphologies.

In both parts of this thesis the potential to use morphology as a way to recover from potential damage is considered. In the case of the SLIP model this is based around the idea that if the robot was damaged in a way that altered the attack angle, the optimal rule set would be able to adapt the stiffness of the spring to regain stability. In the second section this is investigated in more detail with the removal of voxels and the observation that in some cases the robots are able to stiffen the remaining voxels to combat the removal, i.e., the damage.

However, both these cases present an argument for adapting the morphology to combat damage. In the SLIP model, they were able to re-adapt to become successful without the need to change their control system; all the re-adaption takes place in its morphology. Whilst in the case of the voxel based robot this adaption only works for Environment A, note that the adaption systems were not evolved to deal with damage recovery. This was an expected outcome of the evolution in different environment. Therefore, I would argue that further damage recovery would be possible if this was used as the fitness metric for the evolutionary algorithm.

The idea of using morphological adaption for damage recovery in robotics is only starting to be investigated. Kriegman et al. [64] showed recently how deforming the shape of a damaged robot can recover performance better than adapting the control system. This is one of the few works that consider adaptive morphology for damage recovery. Instead researchers (e.g. [80, 10, 2, 11]) have mostly focused on adapting the control system in response to damage to the morphology. Being able to adapt the morphology of a robot in response to damage would be a powerful tool in creating robots that, for example, are capable of remote working, without the need for over complicated control systems.

The preliminary results from this thesis showcase the potential of using morphological adaption as an alternative, or in addition, to adapting the control system in order to recover from damage.

The results from Chapter 4, showcase the benefits of using evolutionary algorithms to evolve optimal methods of adaption. When considering the investigations regarding the SLIP model, the fitness landscape relatively simple. The areas surround the optimal stable parameter combinations show a strong fitness gradient, i.e. the closer to the stable region the further the model will travel before it falls over. Whilst the evolutionary algorithm clearly works well in order to find optimal rule sets, it could be speculated that due to the obvious fitness gradient a method such as gradient ascent would also yield good results. However, the results from the adaption of the more complex voxel based robot suggest that a gradient based method would not be as successful. During the first few episodes the voxel based robot only increases its stiffness and the performance of the robot also increases more or less linearly, as though it is following a clear fitness gradient. However, loss of voxels causes the performance of the robot to decrease before it increases further to a final better performance than if no voxels had been removed at all. Therefore, it could be speculated that a learning metric such as gradient descent (or other gradient-based learning algorithms, namely backpropagation that are typically used for training neural networks) would work up until the point at which the voxels were removed. They would then loose the fitness gradient which may cause them not continue in the same direction in the fitness landscape which would have enabled them to find the better solution (i.e., with the voxels removed). Evolutionary algorithms do not rely on gradients within the fitness landscape and therefore are able to generate adaption methodologies capable of traversing over areas of the landscape where the fitness is low.

Therefore to summarize, the key finding/contributions of this thesis are:

- Implementation of a framework to use evolutionary algorithms to find optimal rule sets capable of adapting the control and morphological parameters of the theoretical spring loaded inverted pendulum model. Separate rule sets were found for both an offline learning and online learning approaches. The offline approach increased the percentage of starting configurations for which the stability could be reached from 3.75% to 98.00%. For the online approach the percentage is increased to 20.00%.
- Using an optimal online rule set, at some starting configurations, the SLIP model is able to contact decreases in ground level of up to 14m.
- Implementation of a framework to use evolutionary algorithms to find optimal sculpting adaption systems. Using a single optimal sculpting adaption system the same starting robot is capable of successfully adapting to a number of unique environments.

Whilst good results have been obtained throughout the thesis, as discussed a limitation of this research is the design choice of the evolutionary algorithms. After reflection, I believe that these design choices could be improved and more suitable parameters and cross-over selection methods may have yielded better results. This is also perhaps the case for the choice of kinetic energy as they key parameter throughout the thesis. I acknowledge that the choice of kinetic energy was somewhat arbitrary (although it has produced good results in both the SLIP model and voxel based robot.)

Finally, as discussed in the introduction, in machine learning there is the theory of "no free lunch". This is also true for the work presented here. In the final chapter (regarding the voxel based robots), when the robots are evolved for a single environment they perform better in that environment than when evolved for multiple. The more adaptive the robot becomes, the less specialised and successful and performing single tasks. Additionally, whilst the voxel based robot shows good transferablity between environments for which it was not optimised, there are environments it is not successful in. I anticipate it is therefore likely that even if evolved for thousands of environments, there would always be some for which the robot would not be successful in. However, the results from this thesis do show that by adapting/developing the morphology, the robot can be successful is a wide range of environments – many more than if the morphology was fixed.

5.1 Further Work

In this section I discuss possible areas of future work that have arisen as a result of this thesis work.

The first area of possible future work is investigation into simultaneous adaption of a robots control and morphology. In the second and third chapters the interplay between adaption of the control system (in this case the attack angle) and adaption of the morphology (spring stiffness) was investigated. However in Chapter 4, i.e., relating to the voxel based robot, only the morphology was adapted. A further extension would be to consider adaption of the *control* and *morphology* simultaneously – using a similar approach to that used to simultaneously adapt the attack angle and spring stiffness of the SLIP model. The adaption of both the morphology and control parameters showed an improvement in robustness when compared to adapting just one of the parameters. Therefore, it could be hypothesised that also allowing adaption of the control in the voxel based robots could yield an even more robust adaption methodology capable of locomotion in a wider range of environment. However adaption of control and morphology both via evolution or development is highly challenging, as discussed in the introduction of this thesis. The same theory of embodiment that stresses the importance of optimal morphology also implies why co-evolution is such a challenge [15] as slightly changing the morphology during evolution can have a large detrimental effect on the control system. Whilst some researchers such as Lipson et al. (2016)[68], Cheney et al. (2018) [15] and Stensby et al. (2021) [112] have explored ways to combat this, such as preserving the morphology of the robot for a number of generations whilst still evolving the controller, co-evolution or co-development remains an open challenge. Whilst in my thesis some level of co-adaption of control and morpholophy is achieved, this occurs in the simplistic SLIP model and the two parts of the system are kept relatively separate. In the more complex voxel based robot, I anticipate development of the control and morphology would be a much greater challenge. Yet even the more complex voxel robots explored in the later chapter are still much simpler than animals and humans, due to their limited fixed control system. Whilst this thesis has shown evidence of the benefits of an developmental morphology, I believe that to truly rival the adaption observed in nature, both suitable morphology and control are required. Although challenging, I believe this is an exciting area for future research.

The second area of possible future work is a transition from simulation to real world robotics. Both parts of the thesis, i.e., regarding the SLIP model and the voxel based robot, are carried out in simulation. In both cases this has produced results that, it is hoped, will provide groundwork in the area of developmental morphological robotics. However, robots considered to be physical systems and therefore some discussion of how to transfer the knowledge gained from this thesis into the design of physical robots needs to be provided.

With regards to the work relating to the SLIP model, there are already a number of physical robots built on the area of the spring loaded inverted pendulum model (e.g. see [125] for a review). It would therefore be easy to use one of these many robots as a starting point and implement the best evolved rule sets within it. A variable stiffness actuator could be used to adjust the spring stiffness and a stepper motor the attack angle.

In the case of the voxel based robot, an obvious framework for this transfer would be the system created by Kriegman et al [63, 43, 64]. In their work the simulated voxels were replaced by physical, hollow, silicone voxels. The thicker the silicone walls the stiffer the voxel. The active voxels are able to expand and contract by altering the air pressure within the voxel. If the neural networks used in this thesis instead used voxel pressure as an input rather than voxel kinetic energy it should be feasible to incorporate pressure sensors into the physical system. After an episode, or locomotion attempt pressure data could be inputted into previously evolved (via simulation) neural network which would determine which voxels stiffen, which to soften and which to eventually remove. The “adapted” robot would then be rebuilt by hand, or in an automated way, for the next episode.

5.2 Concluding Remarks

In conclusion, this thesis discusses the use of evolutionary algorithms to evolve optimal ways of adapting the morphology of a robot in order to achieve its robustness - both to damage of the robot itself and to changes in its environment. Although a lot still needs to be done in this field, the evolutionary and developmental approaches presented in this thesis have shown potential. It is hoped that these methods will play some part in the exciting future of developmental robotics.

References

[1] I. Abraham, Z. Shen, and J. Seipel. A nonlinear leg damping model for the prediction of running forces and stability. *Journal of Computational and Nonlinear Dynamics*, 10(5), 2015.

[2] R. J. Alattas, S. Patel, and T. M. Sobh. Evolutionary modular robotics: Survey and analysis. *Journal of Intelligent & Robotic Systems*, 95(3-4):815–828, 2019.

[3] B. Andrews, B. Miller, J. Schmitt, and J. E. Clark. Running over unknown rough terrain with a one-legged planar robot. *Bioinspiration & biomimetics*, 6(2):026009, 2011.

[4] J. E. Auerbach and J. C. Bongard. On the relationship between environmental and morphological complexity in evolved robots. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, pages 521–528, 2012.

[5] J. E. Auerbach and J. C. Bongard. Environmental influence on the evolution of morphological complexity in machines. *PLoS Comput Biol*, 10(1):e1003399, 2014.

[6] Y. Bar-Cohen and C. Breazeal. Biologically inspired intelligent robots. In *Smart Structures and Materials 2003: Electroactive Polymer Actuators and Devices (EAPAD)*, volume 5051, pages 14–20. International Society for Optics and Photonics, 2003.

[7] R. Blickhan. The spring-mass model for running and hopping. *Journal of Biomechanics*, 22(11-12):1217–1227, 1989.

[8] R. Blickhan, E. Andrada, R. Müller, C. Rode, and N. Ogihara. Positioning the hip with respect to the com: Consequences for leg operation. *Journal of theoretical biology*, 382:187–197, 2015.

[9] J. Bongard. Morphological change in machines accelerates the evolution of robust behavior. *Proceedings of the National Academy of Sciences*, 108(4):1234–1239, 2011.

[10] J. C. Bongard and H. Lipson. Automated damage diagnosis and recovery for remote robotics. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, volume 4, pages 3545–3550. IEEE, 2004.

[11] J. C. Bongard and H. Lipson. Automated robot function recovery after unanticipated failure or environmental change using a minimum of hardware trials. In *Proceedings. 2004 NASA/DoD Conference on Evolvable Hardware, 2004.*, pages 169–176. IEEE, 2004.

[12] J. C. Bongard and R. Pfeifer. Repeated structure and dissociation of genotypic and phenotypic complexity in artificial ontogeny. In *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, pages 829–836. Morgan Kaufmann Publishers Inc., 2001.

[13] J. Bruce, K. Caluwaerts, A. Iscen, A. P. Sabelhaus, and V. SunSpiral. Design and evolution of a modular tensegrity robot platform. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3483–3489. IEEE, 2014.

[14] V. Cacucciolo, Y. Ansari, A. L. Shoushtari, M. Cianchetti, and C. Laschi. Adaptive locomotion on uneven terrains by means of a functional separation of time scales in the design and control of robots. In *Proc. of Adaptive Motion in Animals and Machines*, 2015.

[15] N. Cheney, J. Bongard, V. SunSpiral, and H. Lipson. Scalable co-optimization of morphology and control in embodied machines. *Journal of The Royal Society Interface*, 15(143):20170937, 2018.

[16] N. Cheney and H. Lipson. Topological evolution for embodied cellular automata. *Theoretical Computer Science*, 633:19–27, 2016.

[17] N. Cheney, R. MacCurdy, J. Clune, and H. Lipson. Unshackling evolution: Evolving soft robots with multiple materials and a powerful generative encoding. *ACM SIGEVOlution*, 7(1):11–23, 2014.

[18] N. A. Cheney. Automated design of embodied machines: Optimization algorithms for soft robot morphologies and behaviors. 2017.

[19] S. Collins, A. Ruina, R. Tedrake, and M. Wisse. Efficient bipedal robots based on passive-dynamic walkers. *Science*, 307(5712):1082–1085, 2005.

[20] S. H. Collins, M. Wisse, and A. Ruina. A three-dimensional passive-dynamic walking robot with two legs and knees. *The International Journal of Robotics Research*, 20(7):607–615, 2001.

[21] F. Corucci, M. Calisti, H. Hauser, and C. Laschi. Novelty-based evolutionary design of morphing underwater robots. In *Proceedings of the 2015 annual conference on Genetic and Evolutionary Computation*, pages 145–152, 2015.

[22] F. Corucci, N. Cheney, S. Kriegman, J. Bongard, and C. Laschi. Evolutionary developmental soft robotics as a framework to study intelligence and adaptive behavior in animals and plants. *Frontiers in Robotics and AI*, 4:34, 2017.

[23] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret. Robots that can adapt like animals. *Nature*, 521(7553):503–507, 2015.

[24] E. Del Dottore, A. Mondini, A. Sadeghi, and B. Mazzolai. Characterization of the growing from the tip as robot locomotion strategy. *Frontiers in Robotics and AI*, 6:45, 2019.

[25] F. Dellaert and R. D. Beer. A developmental model for the evolution of complete autonomous agents. In *Proceedings of the fourth international conference on simulation of adaptive behavior*, pages 393–401. MIT Press Cambridge, MA, 1996.

[26] S. Doncieux, N. Bredeche, J.-B. Mouret, and A. E. G. Eiben. Evolutionary robotics: what, why, and where to. *Frontiers in Robotics and AI*, 2:4, 2015.

[27] C. Ducros, G. Hauser, N. Mahjoubi, P. Girones, L. Boisset, A. Sorin, E. Jonquet, J. M. Falciola, and A. Benhamou. Rica: A tracked robot for sampling and radiological characterization in the nuclear field. *Journal of Field Robotics*, 34(3):583–599, 2017.

[28] P. Eggenberger. Evolving morphologies of simulated 3D organisms based on differential gene expression. In *Proceedings of the Fourth European Conference on Artificial Life*, pages 205–213, 1997.

[29] A. E. Eiben, J. E. Smith, et al. *Introduction to evolutionary computing*, volume 53. Springer, 2003.

[30] J. Englsberger, P. Kozlowski, and C. Ott. Biologically inspired dead-beat controller for bipedal running in 3d. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 989–996. IEEE, 2015.

[31] D. Floreano and F. Mondada. Automatic creation of an autonomous agent: Genetic evolution of a neural network driven robot. In *From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, pages 421–430. The MIT Press, 1994.

[32] K. A. Franklin. Shade avoidance. *New Phytologist*, 179(4):930–944, 2008.

[33] H. Geyer, A. Seyfarth, and R. Blickhan. Compliant leg behaviour explains basic dynamics of walking and running. *Proceedings of the Royal Society B: Biological Sciences*, 273(1603):2861–2867, 2006.

[34] R. M. Ghigliazza, R. Altendorfer, P. Holmes, and D. Koditschek. A simply stabilized running model. *SIAM review*, 47(3):519–549, 2005.

[35] J. D. Greer, L. H. Blumenschein, R. Alterovitz, E. W. Hawkes, and A. M. Okamura. Robust navigation of a soft growing robot by exploiting contact with the environment. *The International Journal of Robotics Research*, page 0278364920903774, 2020.

[36] M. F. Hale, M. Angus, E. Buchanan, W. Li, R. Woolley, L. K. Le Goff, M. De Carlo, J. Timmis, A. F. Winfield, E. Hart, et al. Hardware design for autonomous robot evolution. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 2140–2147. IEEE, 2020.

[37] M. F. Hale, E. Buchanan, A. F. Winfield, J. Timmis, E. Hart, A. E. Eiben, M. Angus, F. Veenstra, W. Li, R. Woolley, et al. The are robot fabricator:

How to (re) produce robots that can evolve in the real world. In *Artificial Life Conference Proceedings*, pages 95–102. MIT Press, 2019.

[38] I. Harvey, P. Husbands, and D. Cliff. Seeing the light: Artificial evolution, real vision. *From animals to animats*, 3:392–401, 1994.

[39] H. Hauser and F. Corucci. Morphosis—taking morphological computation to the next level. In *Soft Robotics: Trends, Applications and Challenges*, pages 117–122. Springer, 2017.

[40] H. Hauser, A. J. Ijspeert, R. M. Füchslin, R. Pfeifer, and W. Maass. Towards a theoretical foundation for morphological computation with compliant bodies. *Biological cybernetics*, 105(5-6):355–370, 2011.

[41] H. Hauser, A. J. Ijspeert, R. M. Füchslin, R. Pfeifer, and W. Maass. The role of feedback in morphological computation with compliant bodies. *Biological cybernetics*, 106(10):595–613, 2012.

[42] H. Hauser, K. Nakajima, and R. M. Füchslin. Morphological Computation -– The Body as a Computational Resource. In H. Hauser, R. M. Füchslin, and R. Pfeifer, editors, *E-book on Opinions and Outlooks on Morphological Computation*, chapter 20, pages 226–244. 2014.

[43] J. Hiller and H. Lipson. Automatic design and manufacture of soft robots. *IEEE Transactions on Robotics*, 28(2):457–466, 2011.

[44] J. Hiller and H. Lipson. Dynamic simulation of soft multimaterial 3d-printed objects. *Soft Robotics*, 1(1):88–101, 2014.

[45] J. D. Hiller and H. Lipson. Evolving amorphous robots. In *ALIFE*, pages 717–724, 2010.

[46] T. Howison, J. Hughes, F. Giardina, and F. Iida. Physics driven behavioural clustering of free-falling paper shapes. *PloS one*, 14(6):e0217997, 2019.

[47] T. Howison, J. Hughes, and F. Iida. Morphologically programming the interactions of v-shaped falling papers. In *Artificial Life Conference Proceedings*, pages 359–366. MIT Press, 2020.

[48] J. W. Hurst, J. E. Chestnutt, and A. A. Rizzi. An actuator with physically variable stiffness for highly dynamic legged locomotion. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, volume 5, pages 4662–4667. IEEE, 2004.

[49] F. Iida and R. Pfeifer. Sensing through body dynamics. *Robotics and Autonomous Systems*, 54(8):631–640, 2006.

[50] J. Iqbal, A. M. Tahir, R. ul Islam, et al. Robotics for nuclear power plants—challenges and future perspectives. In *2012 2nd international conference on applied robotics for the power industry (CARPI)*, pages 151–156. IEEE, 2012.

[51] A. Iscen, A. Agogino, V. SunSpiral, and K. Tumer. Controlling tensegrity robots through evolution. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pages 1293–1300, 2013.

[52] M. Joachimczak, R. Suzuki, and T. Arita. Artificial metamorphosis: evolutionary design of transforming, soft-bodied robots. *Artificial life*, 22(3):271–298, 2016.

[53] S. H. Juan and J. M. M. Tur. Tensegrity frameworks: Static analysis review. *Mechanism and Machine Theory*, 43(7):859–881, 2008.

[54] J. Y. Jun. Characterization and optimization of running with curved legs. 2011.

[55] C. D. Jung, W. J. Chung, J. S. Ahn, M. S. Kim, G. S. Shin, and S. J. Kwon. Optimal mechanism design of in-pipe cleaning robot. In *2011 IEEE International Conference on Mechatronics and Automation*, pages 1327–1332. IEEE, 2011.

[56] J. D. Karssen, M. Haberland, M. Wisse, and S. Kim. The optimal swing-leg retraction rate for running. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 4000–4006. IEEE, 2011.

[57] S. H. Klidbary, S. B. Shouraki, and S. Faraji. Finding proper configurations for modular robots by using genetic algorithm on different terrains. *Int J Mater Mech Manuf*, 1(4):360–365, 2013.

[58] M. Komosiński and A. Rotaru-Varga. Comparison of different genotype encodings for simulated three-dimensional agents. *Artificial Life*, 7(4):395–418, 2001.

[59] A. Kotikian, C. McMahan, E. C. Davidson, J. M. Muhammad, R. D. Weeks, C. Daraio, and J. A. Lewis. Untethered soft robotic matter with passive control of shape morphing and propulsion. *Sci. Robot*, 4(33):7044, 2019.

[60] P. Krcah. Evolving virtual creatures revisited. In *GECCO*, volume 7, pages 341–341, 2007.

[61] S. Kriegman, N. Cheney, and J. Bongard. How morphological development can guide evolution. *Scientific reports*, 8(1):1–10, 2018.

[62] S. Kriegman, N. Cheney, F. Corucci, and J. C. Bongard. A minimal developmental model can increase evolvability in soft robots. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 131–138, 2017.

[63] S. Kriegman, A. M. Nasab, D. Shah, H. Steele, G. Branin, M. Levin, J. Bongard, and R. Kramer-Bottiglio. Scalable sim-to-real transfer of soft robot designs. In *2020 3rd IEEE International Conference on Soft Robotics (RoboSoft)*, pages 359–366. IEEE, 2020.

[64] S. Kriegman, S. Walker, D. Shah, M. Levin, R. Kramer-Bottiglio, and J. Bongard. Automated shapeshifting for function recovery in damaged robots. *arXiv preprint arXiv:1905.09264*, 2019.

[65] C. Laschi and M. Cianchetti. Soft robotics: new perspectives for robot bodyware and control. *Frontiers in bioengineering and biotechnology*, 2:3, 2014.

[66] Z. Li and S. Bai. A novel revolute joint of variable stiffness with reconfigurability. *Mechanism and Machine Theory*, 133:720–736, 2019.

[67] E. T. Liknes and D. L. Swanson. Phenotypic flexibility of body composition associated with seasonal acclimatization in passerine birds. *Journal of Thermal Biology*, 36(6):363–370, 2011.

[68] H. Lipson, V. Sunspiral, J. Bongard, and N. Cheney. On the difficulty of co-optimizing morphology and control in evolved virtual creatures. In *Artificial Life Conference Proceedings 13*, pages 226–233. MIT Press, 2016.

[69] H. H. Lund. Co-evolving control and morphology with lego robots. In *Morpho-functional machines: the new species*, pages 59–79. Springer, 2003.

[70] H. H. Lund, J. Hallam, and W.-P. Lee. Evolving robot morphology. In *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC'97)*, pages 197–202. IEEE, 1997.

[71] D. Marbach and A. J. Ijspeert. Online optimization of modular robot locomotion. In *IEEE International Conference Mechatronics and Automation, 2005*, volume 1, pages 248–253. IEEE, 2005.

[72] T. McGeer et al. Passive dynamic walking. *I. J. Robotic Res.*, 9(2):62–82, 1990.

[73] T. A. McMahon and G. C. Cheng. The mechanics of running: how does stiffness couple with speed? *Journal of biomechanics*, 23:65–78, 1990.

[74] K. Miras, E. Ferrante, and A. Eiben. Environmental influences on evolvable robots. *PloS one*, 15(5):e0233848, 2020.

[75] S. Miyashita, S. Guitron, S. Li, and D. Rus. Robotic metamorphosis by origami exoskeletons. *Science Robotics*, 2(10):eaao4369, 2017.

[76] A. P. Moczek, S. Sultan, S. Foster, C. Ledón-Rettig, I. Dworkin, H. F. Nijhout, E. Abouheif, and D. W. Pfennig. The role of developmental plasticity in evolutionary innovation. *Proceedings of the Royal Society B: Biological Sciences*, 278(1719):2705–2713, 2011.

[77] V. C. Müller and M. Hoffmann. What is morphological computation? on how the body contributes to cognition and control. *Artificial life*, 23(1):1–24, 2017.

[78] S. Murata and H. Kurokawa. Self-reconfigurable robots. *IEEE Robotics & Automation Magazine*, 14(1):71–78, 2007.

[79] S. Murata, E. Yoshida, A. Kamimura, H. Kurokawa, K. Tomita, and S. Kokaji. M-tran: Self-reconfigurable modular robotic system. *IEEE/ASME transactions on mechatronics*, 7(4):431–441, 2002.

[80] E. Najarro and S. Risi. Meta-learning through hebbian plasticity in random networks. *arXiv preprint arXiv:2007.02686*, 2020.

[81] S. Nichele, M. B. Ose, S. Risi, and G. Tufte. Ca-neat: evolved compositional pattern producing networks for cellular automata morphogenesis and replication. *IEEE Transactions on Cognitive and Developmental Systems*, 10(3):687–700, 2017.

[82] J. Nordmoen, T. F. Nygaard, K. O. Ellefsen, and K. Glette. Evolved embodied phase coordination enables robust quadruped robot locomotion. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 133–141, 2019.

[83] T. F. Nygaard, C. P. Martin, E. Samuelsen, J. Torresen, and K. Glette. Real-world evolution adapts robot morphology and control to hardware limitations. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 125–132, 2018.

[84] P.-Y. Oudeyer and L. B. Smith. How evolution may work through curiosity-driven developmental process. *Topics in Cognitive Science*, 8(2):492–502, 2016.

[85] D. Owaki and A. Ishiguro. Mechanical dynamics that enables stable passive dynamic bipedal running-enhancing self-stability by exploiting nonlinearity in the leg elasticity. *Journal of Robotics and Mechatronics*, 19(4):374–380, 2007.

[86] C. Paul, H. Lipson, and F. J. V. Cuevas. Evolutionary form-finding of tensegrity structures. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pages 3–10, 2005.

[87] F. Peuker, C. Maufroy, and A. Seyfarth. Leg-adjustment strategies for stable running in three dimensions. *Bioinspiration & biomimetics*, 7(3):036002, 2012.

[88] R. Pfeifer, F. Iida, and J. Bongard. New robotics: Design principles for intelligent systems. *Artificial life*, 11(1-2):99–120, 2005.

[89] R. Pfeifer, F. Iida, and G. Gómez. Morphological computation for adaptive behavior and cognition. In *International Congress Series*, volume 1291, pages 22–29. Elsevier, 2006.

[90] R. Pfeifer, F. Iida, and M. Lungarella. Cognition from the bottom up: on biological inspiration, body morphology, and soft materials. *Trends in Cognitive Sciences*, 18(8):404–413, 2014.

[91] J. B. Pollack, H. Lipson, S. Ficici, P. Funes, G. Hornby, and R. A. Watson. Evolutionary techniques in physical robotics. In *International Conference on Evolvable Systems*, pages 175–186. Springer, 2000.

[92] J. B. Pollack, H. Lipson, G. Hornby, and P. Funes. Three generations of automatically designed robots. *Artificial Life*, 7(3):215–223, 2001.

[93] H. V. Quy, L. Aryananda, F. I. Sheikh, F. Casanova, and R. Pfeifer. A novel mechanism for varying stiffness via changing transmission angle. In *2011 IEEE International Conference on Robotics and Automation*, pages 5076–5081. IEEE, 2011.

[94] M. Raibert, K. Blankespoor, G. Nelson, and R. Playter. Bigdog, the rough-terrain quadruped robot. *IFAC Proceedings Volumes*, 41(2):10822–10825, 2008.

[95] T. S. Ray. Aesthetically evolved virtual pets. *Leonardo*, 34(4):313–316, 2001.

[96] J. Rieffel, D. Knox, S. Smith, and B. Trimmer. Growing and evolving soft robots. *Artificial life*, 20(1):143–162, 2014.

[97] J. Rieffel, B. Trimmer, and H. Lipson. Mechanism as mind-what tensegrities and caterpillars can teach us about soft robotics. In *ALIFE*, pages 506–512, 2008.

[98] J. Rummel, F. Iida, J. A. Smith, and A. Seyfarth. Enlarging regions of stable running with segmented legs. In *2008 IEEE International Conference on Robotics and Automation*, pages 367–372. IEEE, 2008.

[99] J. Rummel and A. Seyfarth. Stable running with segmented legs. *The International Journal of Robotics Research*, 27(8):919–934, 2008.

[100] A. Sadeghi, E. Del Dottore, A. Mondini, and B. Mazzolai. Passive morphological adaptation for obstacle avoidance in a self-growing robot produced by additive manufacturing. *Soft Robotics*, 7(1):85–94, 2020.

[101] A. Sadeghi, A. Mondini, and B. Mazzolai. Toward self-growing soft robots inspired by plant roots and based on additive manufacturing technologies. *Soft robotics*, 4(3):211–223, 2017.

[102] J. Schmitt and J. Clark. Modeling posture-dependent leg actuation in sagittal plane locomotion. *Bioinspiration & biomimetics*, 4(4):046005, 2009.

[103] J. Schmitt, J. R. Stinchcombe, M. S. Heschel, and H. Huber. The adaptive evolution of plasticity: phytochrome-mediated shade avoidance responses. *Integrative and Comparative Biology*, 43(3):459–469, 2003.

[104] A. Seyfarth, H. Geyer, M. Günther, and R. Blickhan. A movement criterion for running. *Journal of Biomechanics*, 35(5):649–655, 2002.

[105] A. Seyfarth, H. Geyer, and H. Herr. Swing-leg retraction: a simple control model for stable running. *Journal of Experimental Biology*, 206(15):2547–2555, 2003.

[106] M. A. Sharbafi, A. M. N. Rashty, C. Rode, and A. Seyfarth. Reconstruction of human swing leg motion with passive biarticular muscle models. *Human movement science*, 52:96–107, 2017.

[107] M. A. Sharbafi and A. Seyfarth. Vbla, a swing leg control approach for humans and robots. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 952–957. IEEE, 2016.

[108] Z. Shen, P. Larson, and J. Seipel. Rotary and radial forcing effects on center-of-mass locomotion dynamics. *Bioinspiration & biomimetics*, 9(3):036020, 2014.

[109] Z. Shen and J. Seipel. A fundamental mechanism of legged locomotion with hip torque and leg damping. *Bioinspiration & biomimetics*, 7(4):046010, 2012.

[110] K. Sims. Evolving 3D morphology and behavior by competition. *Artificial Life*, 1(4):353–372, 1994.

[111] K. O. Stanley, B. D. Bryant, and R. Miikkulainen. Evolving adaptive neural networks with and without adaptive synapses. In *The 2003 Congress on Evolutionary Computation, 2003. CEC'03.*, volume 4, pages 2557–2564. IEEE, 2003.

[112] E. H. Stensby, K. O. Ellefsen, and K. Glette. Co-optimising robot morphology and controller in a simulated open-ended environment. *arXiv preprint arXiv:2104.03062*, 2021.

[113] M. Taghavi, T. Helps, and J. Rossiter. Electro-ribbon actuators and electro-origami robots. *Science Robotics*, 3(25), 2018.

[114] N. L. Tagliamonte, F. Sergi, D. Accoto, G. Carpino, and E. Guglielmelli. Double actuation architectures for rendering variable impedance in compliant robots: A review. *Mechatronics*, 22(8):1187–1203, 2012.

[115] R. Tedrake, T. W. Zhang, and H. S. Seung. Learning to walk in 20 minutes. In *Proceedings of the Fourteenth Yale Workshop on Adaptive and Learning Systems*, volume 95585, pages 1939–1412. Beijing, 2005.

[116] F. Veenstra, A. Faina, S. Risi, and K. Stoy. Evolution and morphogenesis of simulated modular robots: a comparison between a direct and generative encoding. In *European Conference on the Applications of Evolutionary Computation*, pages 870–885. Springer, 2017.

[117] L. Voesenek, T. Colmer, R. Pierik, F. Millenaar, and A. Peeters. How plants cope with complete submergence. *New phytologist*, 170(2):213–226, 2006.

[118] H. Q. Vu, X. Yu, F. Iida, and R. Pfeifer. Improving energy efficiency of hopping locomotion by using a variable stiffness actuator. *IEEE/ASME transactions on mechatronics*, 21(1):472–486, 2015.

[119] H. Vu Quy, G. Ramstein, F. Casanova, L. Aryananda, M. Hoffmann, F. I. Sheikh, and H. Hauser. Gait versatility through morphological changes in a new quadruped robot. 2011.

[120] M. Wisse, A. L. Schwab, and F. C. Van Der Helm. Passive dynamic walking model with upper body. *Robotica*, 22(6):681, 2004.

[121] M. Wisse and J. Van Frankenhuyzen. Design and construction of mike; a 2-d autonomous biped based on passive dynamic walking. In *Adaptive motion of animals and machines*, pages 143–154. Springer, 2006.

[122] E. Yoshida, S. Murata, A. Kamimura, K. Tomita, H. Kurokawa, and S. Kokaji. Evolutionary motion synthesis for a modular robot using genetic algorithm. *Journal of Robotics and Mechatronics*, 15(2):227–237, 2003.

[123] H. Yu, M. Li, and H. Cai. Analysis on the performance of the slip runner with nonlinear spring leg. *Chinese Journal of Mechanical Engineering*, 26(5):892–899, 2013.

[124] Z. Zhakypov and J. Paik. Design methodology for constructing multimaterial origami robots and machines. *IEEE Transactions on Robotics*, 34(1):151–165, 2018.

[125] X. Zhou and S. Bi. A survey of bio-inspired compliant legged robot designs. *Bioinspiration & biomimetics*, 7(4):041001, 2012.

[126] M. Ziegler, F. Iida, and R. Pfeifer. Cheap" underwater locomotion: roles of morphological properties and behavioural diversity. In *International Conference on Climbing and Walking Robots, CLAWAR, Karlsruhe*, 2006.