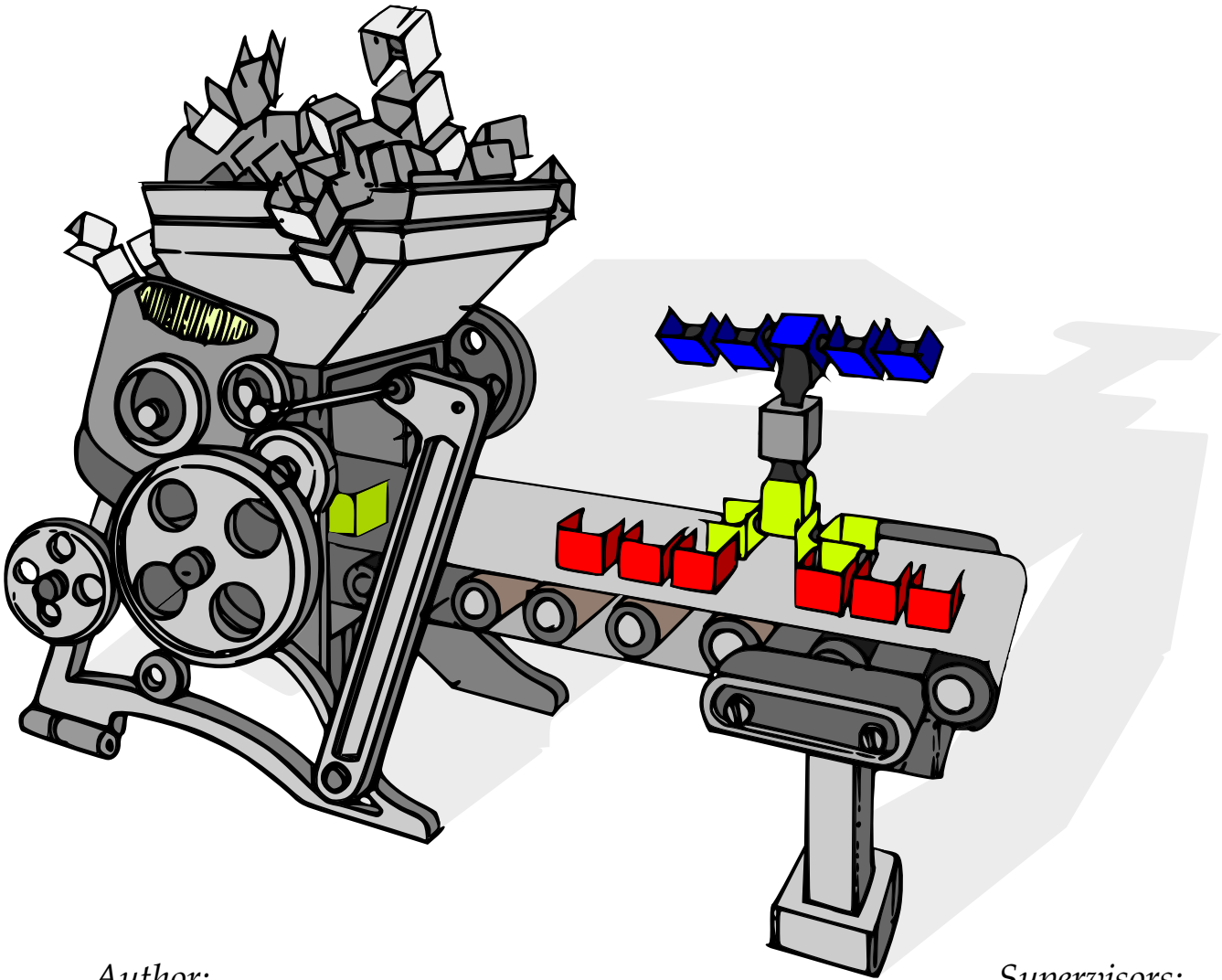


DOCTORAL THESIS

The Watchmaker's Guide to Artificial Life:

On the Role of Death, Modularity and Physicality in
Evolutionary Robotics



Author:
Frank VEENSTRA


Supervisors:
Kasper STØY
Sebastian RISI

*A thesis submitted in fulfilment of the requirements
for the degree of Doctor of Philosophy*

in the

Robotics Evolution and Art Laboratory

IT UNIVERSITY OF COPENHAGEN

REAL 
Robotics, Evolution and Art Lab

Copyright © 2018 by Frank Veenstra
All rights reserved.

Dedicated to my family

Surely if a machine is able to reproduce another machine systematically, we may say that it has a reproductive system. What is a reproductive system, if it be not a system for reproduction? And how few of the machines are there which have not been produced systematically by other machines? But it is man that makes them do so. Yes; but it is not insects that make many of the plants reproductive, and would not whole families of plants die out if their fertilization was not affected by a class of agents utterly foreign to themselves? Does anyone say that the red clover has no reproductive system because the humble bee (and the humble bee only) must aid and abet it before it can reproduce? No one. The humble is part of the reproductive system of the clover. Each one of ourselves has sprung from minute animalcules whose entity was entirely distinct from our own, and which acted after their kind with no thought or heed of what we might think about it. These little creatures are part of our own reproductive system; then why not we part of that of the machines?

– Samuel Butler, Erewhon

IT UNIVERSITY OF COPENHAGEN

Abstract

Computer Science

Doctor of Philosophy

The Watchmaker's Guide to Artificial Life

by Frank VEENSTRA

English Abstract

Optimizing robots through the implementation of evolutionary processes plays a key role in evolutionary robotics and artificial life. Challenges that arise in the evolutionary optimization of robots are usually related to an algorithm's compromise between trying new solutions and improving previously found solutions (the exploration vs. exploitation trade-off), and whether to express genomic information directly or generatively (genotype to phenotype mapping). An additional challenge is that there is a discrepancy between simulation and reality (reality gap) where robots 'evolved' in simulation environments function differently when transferred to the real world.

The exploration vs. exploitation trade-off is addressed through describing and experimenting with the inclusion of biologically inspired intrinsic mortality and how this affects the evolvability of populations. The results contribute to our understanding of the relationship between intrinsic mortality and mutation rate. The results further indicate how it can be utilized to develop algorithms that can outperform state-of-the-art algorithms.

This thesis continues by addressing the challenge of mapping the genotype to the phenotype through investigating the influence of generative encodings on the evolution of simulated modular robots. It is investigated how an L-System as a generative encoding can lead to the formation of plant-inspired virtual creatures and shows that movement for tracking a moving light source is not an emerging phenotypic trait. Afterwards, this generative encoding is shown to better evolve modular robots for locomotion compared to a direct encoding. The addition of real-world solar panel

modules demonstrates how modular robots can be evolved toward energy autonomy.

The final part of the thesis describes the evolution of embodiment and control of physical robots. As part of this, an automated process for assembly and disassembly of modular robots is demonstrated, which can be used to evaluate evolved individuals in the real world. A viable method for implementing evolution directly is demonstrated through evolving the behavior of a knifefish-inspired physical soft robot. Both approaches represent strategies for addressing the reality gap.

The experimental results of this thesis contribute to the understanding of biological phenomena and elucidate how improvements can be made to existing methods in evolutionary robotics. It shows that we can utilize concepts from evolutionary biology to advance our understanding of evolutionary dynamics, encodings and physical designs that are valuable for the automated design of robots.

Danish Abstract

Optimering af robotter gennem implementering af evolutionære processer spiller en central rolle i evolutionær robotik og kunstigt liv. Udfordringer der opstår i evolutionær optimering af robotter er normalt relateret til en algoritmes kompromis mellem at prøve nye løsninger eller forbedre tidligere fundne løsninger (søge kompromiet) og om at udtrykke genomisk information direkte eller generativt (genetisk kodning). En yderligere udfordring er, at der er en uoverensstemmelse mellem simulering og virkelighed, hvorved robotter udviklet i simuleringsmiljøer fungerer anderledes, når de overføres til virkeligheden (virkelighedsgabet).

Udforskningen af søge kompromiet behandles ved at beskrive og eksperimentere med inddragelsen af biologisk inspireret dødelighed og hvordan dette påvirker systemets evolutionsevne. Resultaterne bidrager til vores forståelse af forholdet imellem dødelighed og mutationshastigheden. Resultaterne viser yderligere, hvordan dette kan udnyttes til at udvikle algoritmer der kan udkonkurrere state-of-the-art algoritmer.

Afhandlingen fortsætter med at kigge på udfordringen om genetisk kodning ved at undersøge indflydelsen af generative kodninger på udviklingen af simulerede modulære robotter. Det undersøges, hvordan et L-system, som er en generativ kodning, kan føre til dannelsen af planteinspirerede virtuelle væsner og viser, at bevægelse til sporing af en bevægende lyskilde ikke er et fremtrædende fænotypisk træk. Derefter vises denne generative kodning at være bedre til at udvikle modulære

robotter til bevægelse sammenlignet med en direkte kodning. Tilføjes af virkelige solpanelmoduler demonstrerer, hvordan modulære robotter kan udvikles mod energi autonomi.

Den afsluttende del af afhandlingen omhandler evolution af form og kontrol af fysiske robotter. Som led i dette demonstreres en automatiseret proces til samling og demontering af modulære robotter, som kan bruges til at evaluere udviklede individer i den virkelige verden. En brugbar metode til gennemførelse af direkte evolution demonstreres gennem udviklingen af en knivfisk-inspireret, fysisk, og blød robot. Begge tilgange repræsenterer strategier til løsning af udfordringen med virkelighedsgabet.

Resultaterne i afhandlingen bidrager til forståelsen af biologiske fænomener og belyser, hvordan man kan forbedre eksisterende metoder i evolutionær robotik. Afhandlingen viser, at vi kan udnytte koncepter fra evolutionær biologi for at fremme vores forståelse af evolutionær dynamik, kodninger og fysiske design, der også er værdifulde i forbindelse med automatisk design af robot.

Preface

This thesis is about evolution in the context of artificial life. Here, evolutionary computation shapes the progression of usually *in silico* systems toward user-specified goals through the artificial implementation of evolution. Since evolution has led to the many elegant adaptations observed in organisms today, the implementation of evolutionary computation in robotics is conducted with the aim to bring about artificial complexity using the same evolutionary processes underlying billions of years of adaptation. This is the paradigm of evolutionary robotics. As the resulting biological machinery, organisms, have been analogized to the complexity of watches, many have falsely attributed the emergence of organisms and species in this world to a higher deity. The many works of Richard Dawkins assert that the acclaimed watchmaker processes seen in nature are in fact the result of the blind forces of physics. Evolution, as a result of these forces, elicits the works of an apparent blind watchmaker. Since us evolutionary roboticists try to nudge the evolutionary processes to engender effective features in robots, we are the parody of the watchmaker for the creation of artificial life. We incorporate many biological features shown to be useful for the evolutionary processes exhibited in nature and contemplate how to thereby create beings of complexity rivaling nature's emergent 'design'.

The subtitle of this thesis explains its major constituents. Death refers to the experiments on the potential evolutionary advantage of programmed death, which has been heavily inspired by a guest lecture from Justin Werfel that was partly about altruistic aging. Modularity concerns the implementation of modular robots; in this case, it involves subjecting modular robot conglomerates to simulated evolution. This work builds on existing work of evolving modular robots done in collaboration with Andrés Faíña. Physicality is discussed by the application of evolutionary computation to physical robots; where Jonas Jørgenson and I have collaborated to evolve the behavior of a physical knifefish-inspired robot. Death, modularity, and physicality are discussed in the context of evolutionary robotics, toward a better understanding of (artificial) life.

Acknowledgements

I would like to thank my advisors, Kasper Støy and Sebastian Risi. Not only were you excellent supervisors, but the fun, relaxed and thought-provoking atmosphere made the PhD experience incredible. For this, I feel incredibly fortunate for having been able to pursue my academic interest with you. An incredible source of support and influence for this thesis was Andrés

Faíña; you enabled me to develop my own algorithms and robots through your advice and patience and I am incredibly grateful for it. As a whole, I would like to thank all of my friends and colleagues at the IT University of Copenhagen. This includes all of the former and current members of the Robotics Evolution and Art Lab. Many thanks to my advisor abroad, Josh Bongard and all of the former and current members of the University of Vermont's Morphology, Evolution and Cognition Laboratory that I met during my stay abroad. The experience of collaborating with you was incredibly influential on the progress of my research and I feel fortunate for all our memorable experiences.

Special thanks to my adorable girlfriend Franziska, whom I never would have met were it not for being able to take a course in Neurobiology for my PhD. Thanks, also, to all of my friends who have supported me during my PhD. In addition, thanks to my entire family, with honorable mentions to Mom, Dad, Jessie, Beppe & Opa. Without Dad, evolution in this thesis would have been brought about by 'decent' with modification, and credit for the awesome drawings on the cover and parts of this thesis goes to my sister, Jessie.

The work presented in this thesis received funding from 'Flora Robotica', a European Union's Horizon 2020 research and innovation program under FET Grant Agreement, No. 640959. Computation/simulation for Chapters 3 and 9 was supported by the Vermont Advanced Computer Core, University of Vermont. Computation/simulation for Chapters 6 and 7 was supported by the DeIC National HPC Centre, University of Southern Denmark, and the Vermont Advanced Computer Core, University of Vermont.

Special thanks to Sam Kriegman whose input and feedback shaped parts of Chapter 3. Special thanks to Rodrigo Moreno, Ceyue Liu, Chloé Metayer and Andrés Faíña for their contribution to the development of the modular robots presented in Chapters 4, 5, 6, 7 and 8. Special thanks to Jonas Jørgenson for his contributions to the work presented in Chapter 9.

Contents

Abstract	v
Preface	viii
1 Introduction	1
1.1 Challenges	3
1.2 Structure and Contributions	5
1.3 Evolutionary Robotics	7
1.4 Scope and Context	10
I Evolutionary Dynamics	13
2 Evolution	14
2.1 Fundamentals of Evolutionary Theory	16
2.1.1 Performance Measure	17
2.2 The Genome and the Sequence Space	18
2.2.1 Selection	22
2.3 Development of the Phenotype	23
2.4 Evolutionary Game Theory	27
2.5 Evolvability	29
2.6 Evolutionary Computation	34
2.6.1 Binary Approach	37
2.6.2 Continuous Single Objective Functions	40
2.7 Spatial Models	42
2.8 Concluding Remarks	48
3 Evolution and Longevity	49
3.1 Theory on the Evolution of Senescence	50
3.1.1 Summary of Theories on Senescence	51
3.1.2 Programmed vs. Non-programmed	55
3.1.3 Why Mortality Promotes Evolvability	56
3.2 Mortality and Mutation Rate Shape Evolvability	58

3.2.1	Mortality in the Steady State Genetic Algorithm . . .	59
3.2.2	Spatial Model	63
3.2.3	Comparison to Other Algorithms	70
	Age Fitness Pareto Optimization	70
	Other Conventional Implementations	73
3.3	Changing the Domain	74
3.4	Development and Evolvability	78
3.5	Discussion	82
	3.5.1 Theoretical Implications	84
	3.5.2 Hypothetical Explanation for the Benefit of Mortality	87
	3.5.3 Application to Evolutionary Robotics	88
3.6	Conclusion	88
 II Evolving Modular Robots		91
4	Evolutionary Robotics Platform	92
4.1	Neuroevolution	94
4.2	Overview of the Evolutionary Robotics Plugin	98
	4.2.1 Evolutionary Algorithm	99
	4.2.2 Morphology	100
	4.2.3 Control	102
4.3	Current State and Functionalities	103
4.4	Real Modules	105
4.5	Concluding Remarks	109
5	Evolving Plant Morphologies with L-Systems	110
5.1	Introduction	112
	5.1.1 Phytomorphogenesis	114
	5.1.2 Simulated models	115
5.2	Approach	116
	5.2.1 Genetic Algorithm	116
	5.2.2 L-system	118
5.3	Results	120
5.4	Discussion	122
5.5	Conclusion	126
6	Comparing Encodings for Evolving Locomotion in Modular Robots	127
6.1	Introduction	128
6.2	Methodology	130

6.2.1	Common Elements for Both Platforms	131
6.2.2	Direct Encoding	133
6.2.3	Generative Encoding	134
6.3	Results	135
6.3.1	Performance Analysis	136
6.3.2	Phenotypes	138
6.4	Discussion	140
6.5	Conclusion	145
7	Toward Energy Autonomy Through Evolving Plant-Inspired Modular Robots	146
7.1	Introduction	147
7.2	Material and Methods	149
7.2.1	Modules	150
7.2.2	Connectivity	155
7.2.3	Generative Encoding	155
7.2.4	Evolutionary Algorithm	157
Neural Network	158
Fitness function	158
7.2.5	Experimental Setup	159
7.3	Results	160
7.3.1	Phenotypes Evolved in Different Environments	160
7.3.2	Movement in Energy-Harvesting Modular Robots	161
7.4	Discussion	163
7.4.1	Light Tracking	165
7.4.2	Challenges	167
7.4.3	Multi-Robot Systems	170
7.5	Conclusion	172
III	Evolving Physical Robots	173
8	Evolutionary Approaches in Physical Robots	174
8.1	Evolving Physical Robots	175
8.2	Automated Assembly of Modular Robots	177
8.2.1	Methods and Environment	177
8.2.2	Prospects on the Automated Assembly of Modular Robots	179
8.3	Concluding Remarks	182

9	Evolution in a Knifefish-inspired Soft Robot	183
9.1	Introduction	184
9.1.1	Evolution of Soft Robots	185
9.1.2	Knifefish-inspired Swimming Robots	186
9.2	Methodology	188
9.2.1	Mechanical Design of the Robot	188
9.2.2	Evolutionary Experiments	190
	Encoding	191
	Evolutionary algorithm	192
	Controller system	193
	Experiments	194
9.2.3	Comparing Behaviors of the Robot with Actual Knifefish	194
9.3	Results	195
9.3.1	Performance Analysis	195
9.3.2	Phenotypic Analysis	198
9.4	Discussion	199
9.5	Conclusion	203
10	Discussion	204
10.1	Evolutionary Dynamics of Intrinsic Mortality	205
10.2	Evolving Modular Robots	206
10.3	Evolution of Physical Systems	208
10.4	Concluding Remarks	210
A	Full Spatial Model Pseudocode	211
B	UML Evolutionary Robotics Plugin	213

List of Figures

1.1	Biomorphs	9
1.2	Evolving Virtual Creatures	10
1.3	Overview of fundamental steps in evolutionary robotics	11
2.1	Fitness landscape with a peak and a hill	21
2.2	Denticle formation influenced by hedgehog	25
2.3	Configuration of the eye of cephalopods and vertebrates	31
2.4	Basic procedure of a generational evolutionary algorithm	35
2.5	Fitness landscape of one-max function	38
2.6	Fitness landscape of the adjusted H-IFF function	41
2.7	Explanation of the H-IFF function	41
2.8	The Rastrigin and Schwefel function	42
2.9	Number of prey and predators when varying the primary energy production	46
2.10	Stable limits in predator-prey models	47
2.11	Biomass of simulated plants, rabbits and foxes	47
3.1	Survivor curves	52
3.2	Theories of aging plotted as an advantage and a disadvantage over time	55
3.3	Fitness landscape with one peak and a hill based on the sequence space	58
3.4	Relationship between the mutation rate and mortality rate	61
3.5	Evolutionary progress for different mortality rates	62
3.6	Relationship between mutation rate and mortality rate on 128-bit H-IFF	64
3.7	Relationship between mutation rate and mortality rate on 32-bit H-IFF	64
3.8	Optimal mutation and mortality rate on 32-bit Chuang f1	65
3.9	Illustration of the spatial model	67
3.10	Speed of solving H-IFF for the spatial model	69
3.11	Individual runs on the spatial model	70

3.12	Optimal mutation rate as a function of terminal age	71
3.13	Age-Fitness Pareto Optimization	72
3.14	Single evolutionary runs comparing AFPO and H-IFF	74
3.15	200 evolutionary runs are shown for both AFPO and mortality rate	75
3.16	Diversity on H-IFF using various genetic algorithms	76
3.17	Evolutionary progression on 64-bit H-IFF of generational Genetic algorithms vs. the SSGA implementing mortality	77
3.18	Comparison of evolutionary algorithms on the Schwefel function	78
3.19	Correlating developmental stages with finding the global maximum	82
3.20	Single evolutionary runs in the developing populations	83
3.21	Venn diagram depicting the relationship between reproductive rate, mortality, and mutation rate	85
4.1	Approaches to controlling a modular robot	96
4.2	Overview of the evolutionary robotics plugin	97
4.3	Illustration of the interface designed for the evolutionary robotics plugin	101
4.4	Example of growth in the simulator	102
4.5	Two modular robots controlled by decentralized neural networks	103
4.6	Module connection site	105
4.7	Interface Connector	106
4.8	Collection of Modules	107
4.9	Modular robots connected to V-REP via the plugin	108
5.1	Recursion and modularity in plants	111
5.2	Evolved Phytomorphologies	112
5.3	Movement of a light source with respect to robot	118
5.4	Three illustrations of the implemented L-system	119
5.5	Evolutionary progression of static and actuated phytomorphologies	120
5.6	The phenotype of the best evolved individual	122
5.7	Illustration of an evolutionary progression	123
5.8	Evolution of movement?	124
5.9	Generated phytomorphologies	125
6.1	Illustration of simulated and real modules	131
6.2	Representation of the direct encoding	134

6.3	Representation of the generative encoding	136
6.4	Evolutionary progression of individual runs	137
6.5	Evolutionary progressions of the direct and generative encoding	138
6.6	Phenotypes obtained through the direct encoding	141
6.7	Phenotypes acquired through the generative encoding	142
6.8	Illustration of evolved phenotypes	144
6.9	Prematurely converged phenotype	144
7.1	The modules used in the simulated and physical robot	151
7.2	Schematic diagram of the modular robot containing solar panels	154
7.3	Block diagram of the modular robotics platform	156
7.4	Different simulated environments	161
7.5	Different evolved phenotypes	162
7.6	Evolved phenotypes in different environments	163
7.7	Various evolved phenotypes in the environment with the moving the light source	164
7.8	Evolutionary runs conducted in the different environments	165
7.9	The effects of an additional energy cost	166
7.10	Incorporation of the flower module	167
7.11	Simulated and real modular robot behavior	171
8.1	Evolutionary robotics approaches to acquire a behavior	175
8.2	Assembly and disassembly process of modules	179
8.3	Sequence of automatically assembling modular robots	180
9.1	The black ghost knifefish	184
9.2	Experimental setup for evolving the knifefish	187
9.3	CAD design of the robotic knifefish	189
9.4	Cross section of the fish design	190
9.5	Angular deflection of the fin	191
9.6	Change in angle of fin rays	192
9.7	Evolutionary progressions of 5 runs	195
9.8	The best-evolved wave patterns in 5 distinct evolutionary runs	196
9.9	Performance difference between the best-evolved sinusoidal and Fourier series individuals	198
9.10	Evolved wave patterns	199
9.11	The evolutionary progression of a single Fourier Series run with a servo malfunction occurring around generation 10	202
B.1	The UML diagram of the Evolutionary Robotics Plugin	214

List of Tables

2.1	Example of the score of a 16-bit HIFF genome	40
3.1	Number of times the optimal solution was found in the SSGA	60
3.2	Number of optimal solutions for 32-bit H-IFF on a spatial model	68
9.1	Specific wavelengths and travel speeds of behaviors	197

Chapter 1

Introduction

It is interesting to contemplate an entangled bank, clothed with many plants of many kinds with birds singing on the bushes, with various insects flitting about, and with worms crawling through the damp earth, and to reflect that these elaborately constructed forms, so different from each other, and dependent on each other in so complex a manner, have all been produced by laws acting around us. These laws, taken in the largest sense, being Growth and Reproduction; Inheritance which is almost implied by reproduction; Variability from the indirect and direct action of the external conditions of life, and from use and disuse; a Ratio of Increase so high as to lead to a Struggle for Life, and as a consequence to Natural Selection, entailing Divergence of Character and the Extinction of less-improved forms. Thus, from the war of nature, from famine to death, the most exalted object which we are capable of conceiving, namely, the production of the higher animals, directly follows. There is grandeur in this view of life, with its several powers, having been originally breathed into a few forms or into one; and that, whilst this planet has gone cycling on according to the fixed law of gravity, from simple a beginning endless forms most beautiful and most wonderful have been, and are being, evolved.

– Charles Darwin, On the Origin of Species

Evolution is the ultimate explanation for life as we know it. The process of descent with modification is not only fundamental to our understanding of nature, but its merits are useful in computational models as well. Its utilization is valuable for creating smart algorithms or robots, which enables us to better understand evolution and explain phenomena that we observe in nature. Artificial evolution, defined under the umbrella term *evolutionary computation*, encompasses the algorithms mimicking biological evolution. It sprouted from the many implementations in computational models that were inspired by evolution. In evolutionary robotics, evolutionary computation is implemented as the strategy for

the automated design of intelligent robots (Harvey et al., 1992; Nolfi et al., 2000). It has subsequently been demonstrated that research in this paradigm can result in the creation of unique control systems and body plans for simulated and physical robots.

If we could rerun the evolutionary process of life on earth, the emerging entities would likely not resemble humans. Another branch of life could well have dominated and outcompeted the ancestral branch humans sprouted from. If the meteorite that killed the dinosaurs missed Earth, preventing dinosaurs from going extinct, an intelligent dinosaur species might be roaming the planet instead of us humans. This aspect is also true for artificial systems; rerunning evolutionary robotics simulations will almost never yield similar phenotypes, but rather will find unique solutions for an objective, resulting in a creative optimization search process.

Using evolution as a tool can thereby also lead to the emergence of complex robots by simply providing the fundamental building blocks. The evolutionary process in turn generates efficient machines. As is shown in this thesis, this can be done through evolving simulated robots, as well as directly implementing evolutionary computation on robots. In evolutionary robotics, we can select any stage of a robot's ancestral branch and run evolution from this point onward, designating the robot to conform to a variety of objectives, environments, resources, body plans, and control systems. Hence, an artificial evolutionary process could lead to life, but, most likely, not as we know it (Eiben, 2014).

Many people believe in a deity and use this as an argument against evolution. A classic argument used as proof for the existence of a deity is the analogy of this deity to a watchmaker. The theologian William Paley explained that finding a watch on the floor necessitates the inference that it must have been designed and made by a watchmaker (Paley, 1802). Organisms seemingly contain design elements that are complex, just like a watch, as opposed to, say, a stone. Therefore, organisms must have a 'designer' as well. This idea was formulated before Darwin's *On the Origin of Species*. Dawkins, 1986 has argued that such complex natural design is simply an emergent property of evolution through descent with modification—where “if it can be said to play the role of watchmaker in nature, it is the blind watchmaker” (Dawkins, 1986). In evolutionary robotics, the blind process of evolution is itself implemented to design robots. Hence, us evolutionary roboticists being watchmakers, having come into existence through a blind watchmaker, try to make watches using a blind watchmaker's approach.

Evolutionary computation is a branch of artificial intelligence, where contemporary research in artificial intelligence has proven its prominence

and applicability. Strategies like deep neural networks and Monte-Carlo Tree Search have especially demonstrated to produce smart machines as in the case of the recent defeat of the world's best go player (Silver et al., 2017). Despite the practical value, most research in the field of AI in general is concerned with the creation of arbitrary notions of thought processes and reasonings that are heavily inspired by how humans act and think, or by how machines can be made to act rationally (Russell et al., 2010). It is therefore also highly bio-inspired and usually takes an anthropomorphic perspective. Robotics and AI not only focus on this mimicry, but research also entails cognitive and motivational autonomy by investigating memory, imagination, mental life, planning, thinking, dreaming, or hallucinating. Although humans can serve as useful models for inspiration in artificial intelligence, it could also be viewed as an over-complicated system that developed its advanced cognitive processes based on *maximum parsimony*. Maximum parsimony means that through natural evolution, the number of state changes to find a solution is minimized. Humans happen to be the solution thus far in this context.

Many of the physiological traits and structures seen in humans might have served a completely different function in common ancestors. Thus, the evolved functionalities we see today might thus not be the most efficient ones to implement in an artificial system. Our intelligence is surely not modeled with the sole aim of acquiring it; rather, it enables the survival and reproduction of our genes. For example, there are behavioral and structural commonalities between ourselves and neighboring species that sprouted from a common ancestor. If we dissect the human body—say we split up every compartment of the brain and try to make an artificial module for the specific task allocated to the corresponding part of the brain—we might find more efficient solutions to engender these functional feats without being chained to our ancestral precursive solutions. Therefore, in contrast to the anthropomorphic modeling of an artificial system inspired by hominids, in evolutionary robotics we can reshape the entire collection of functionalities and evolve them without relying on the process of mimicking them. Evolutionary robotics thus allows us to evolve a general bio-inspired artificial system potentially as a precursor to new branches of intelligent systems and artificial life.

1.1 Challenges

Evolutionary computation is often implemented as an objective-based problem solver, similar to existing AI techniques. The power of

evolutionary computation is manifold, including its ability to optimize discontinuous, non-differentiable functions. The correct parameter set is, however, highly dependent on the *fitness landscape*. Simple hills can be climbed through local search, whereas landscapes that are more difficult necessitate more global search or diversification to find a desired optimum. This is a problem in evolutionary computation called the *exploration vs. exploitation* trade-off. The capability of traversing the fitness landscapes efficiently through adjusting parameters that favor either exploration or exploitation, is thus of great importance to evolutionary computation. The capacity for traversing a fitness landscape and thereby finding better solutions is called *evolvability*—the capacity to evolve. Evolvability in evolutionary algorithms is discussed mainly in [Chapter 3](#), where it is promoted by implementing an indiscriminate intrinsic mortality factor.

The attributes seen in organisms that have led to the emergence of life forms are also of interest in the field of evolutionary robotics. One of these important attributes that changes an organism during its lifetime is called development. Equivalent to development in evolutionary robotics are the processes that change an agent during its lifetime and, of specific interest to this thesis, the mapping from genotype to phenotype. The *genotype to phenotype mapping* is another challenge in evolutionary robotics, which is discussed in the context of evolving modular robots, mainly in [Part II](#).

One final major challenge in evolutionary robotics is the *reality gap*, or the discrepancy between simulated and real-world robots. In evolutionary robotics, the simulator usually evolves the behavior in a three-dimensional model of the robot that can be transferred to the real-world robot after an adequate solution has been found. However, transferring the evolved robot from the simulation to the real world always presents a performance or reality gap (Jakobi et al., 1995). Contemporary research has not yet created simulators that would allow morphologically complex evolved robots to be transferred to the real world and expecting similar results. However, certain abstractions can reduce this reality gap, as is discussed briefly in [Chapters 6](#) and [7](#). An alternative approach bypassing the simulator altogether is discussed in [Chapter 9](#). In this chapter, evolutionary computation is implemented directly on a physical soft robot. This is an efficient approach since a soft robot is difficult to simulate and control in the real world.

1.2 Structure and Contributions

This thesis conveys ideas and experiments regarding the process of shaping artificial robotic systems without relying on engineered solutions or detailed features derived from natural systems. The thesis is structured into three general parts that address the topics, as the title implies: Evolutionary Dynamics (**Part I**), Evolving Modular Robots (**Part II**) and Evolution of Physical Robots (**Part III**). Though the parts are not strict distinctions, the thesis is structured from theory to practice, every subsequent chapter being more set in reality than the former.

In the first part, I discuss various concepts of natural systems related to evolution and how these theoretical concepts can be implemented in computational models. In **Chapter 2**, a general overview is given of high-level concepts related to natural and artificial evolution. From here, experiments on the evolutionary dynamics of genetic algorithms and spatial models are performed in **Chapter 3**. This chapter specifically investigates the theoretical concept of how senescence (aging) could be an evolutionary advantageous trait in natural and artificial systems, senescence being simulated by an intrinsic mortality parameter. This chapter also describes the first major scientific contribution supported by a paper published at the Artificial Life conference of 2018 (Veenstra et al., 2018b). The chapter includes many details and additional experiments that were excluded from the submitted paper. The hypothesis that is being addressed in this chapter is:

Hypothesis 1 *Intrinsic mortality benefits the evolvability of a population.*

The results of this chapter indicate that there is a tight correlation between the mutation rate and mortality rate that influences evolvability in a specific manner. Experiments on various landscapes indicate that indiscriminate mortality can enhance the performance of genetic algorithms on benchmark tests and in spatial models. These experiments show that factors influencing intrinsic mortality are actually an evolutionary advantage for an evolving population.

The second part of this thesis addresses the main body of work done on Evolving Modular Robots. In **Chapter 4**, an overview is given of the evolutionary robotics simulator that was developed and used for the experiments discussed in the subsequent chapters. The three chapters that follow are based on three publications on the evolution of modular robots. The first publication (Veenstra et al., 2016) regarded evolving plant-inspired virtual creatures for function and aesthetics. It discussed whether evolved

artificial plants would likely evolve movement in a given environment (**Chapter 5**). This is formulated by the second hypothesis of the thesis:

Hypothesis 2 *Actuation in evolving phytomorphologies is beneficial for optimizing light absorption.*

Chapter 5 illustrates that in fact movement, without any added cost for movement, did not produce individuals better suited for tracking a light source. It could therefore be that movement is simply not as useful to plants as being able to create many leaves.

Chapter 6 veers the evolutionary robotics approach from optimizing for light absorption to evolving locomotion in modular robots (Veenstra et al., 2017a). The chapter discusses two approaches to generating the modular robots and how this generation affects the evolutionary search process. It therefore addresses the genotype to phenotype mapping. One approach consists of a direct encoding and the other of a generative encoding, similar to the one implemented in **Chapter 5**. This leads to the third hypothesis of this thesis:

Hypothesis 3 *A generative encoding increases the efficiency of evolving modular robots compared to a direct encoding.*

The results indicate that a generative encoding is indeed the better encoding strategy for evolving modular robots. Moreover, the video that was produced as supplementary material subsequently won the virtual creatures contest held at GECCO 2017.¹

The real physical modules that were developed were used to evaluate how to design energy-autonomous robots, as discussed in **Chapter 7** (Veenstra et al., 2017b). Although genuine energy autonomy was too difficult to achieve given the modules that were used, the main contribution of this chapter is its description of a modular robotic system in which energy harvesting modules can be integrated. This approach might be most valuable for multi-robot systems, where some robots specialize in energy harvesting while higher-order robots could utilize the energy gathered by these primary energy-producing robots. The main hypothesis addressed in this chapter is:

Hypothesis 4 *Energy autonomy in modular robots can emerge from implementing solar panel modules.*

¹The video on Evolving Modular Robots can be found here: <https://www.youtube.com/watch?v=HCDftic1AdA>

Results show that this implementation of evolution in modular robots provides a stepping-stone toward developing legitimate energy autonomy in modular robots. Moreover, it demonstrates an application in which the reality gap is negligible.

The final part of this thesis discusses applying evolution in physical systems. It opens with a rough outline of approaches to evolving physical robots and describes a way to automatically assemble modular robots in [Chapter 8](#). [Chapter 9](#) is a standalone experiment, in which a knifefish-inspired, soft, swimming robot was created (Veenstra et al., 2018a). Its body is quite complex and would be tedious to simulate. Therefore, a form of evolution bypassing a simulator environment was used. Covariance Matrix Adaptation Evolution Strategy (CMA-ES) was the algorithm implemented to evolve the undulating swimming behavior of the robot. This experiment led to final hypothesis of this thesis:

Hypothesis 5 *Evolutionary computation finds better solutions for controlling a designed soft swimming robot compared to manually encoded behaviors.*

The results of the evolutionary experiments indicate that CMA-ES is a viable method for optimizing the behavior of morphologically complex robots outperforming the manually encoded behavior.

This thesis concludes with a brief discussion of the implemented work and its prospects in [Chapter 10](#). The general aim of the thesis is to convey how and why evolutionary robotics can lead to the acquisition of functional robots. This aim considers both the biological and artificial processes related to evolution. The important common concepts in both natural and artificial evolution—such as selection pressure, genotype to phenotype mapping, evolvability, and models for evolutionary computation—are therefore described in [Chapter 2](#). This chapter contains information essential for understanding concepts that are described later in the thesis, but may not be a requirement for the reader knowledgeable in these subjects.

1.3 Evolutionary Robotics

In evolutionary robotics, we are free to model any potentially useful traits of morphologies and controllers that can be fed into an evolutionary algorithm as building blocks. This modeling, through using modular robots, forms the core of this thesis, as robots can adopt certain traits that are useful for an objective and incrementally improve themselves by altering the genetic information that encodes for their phenotype.

Any traits and features we provide as building blocks could be viewed as constituents of an artificial primordial soup. It contains all of the ingredients that are useful or available to enable the emergence of the robots. Complex robots can thus spontaneously form through the evolutionary process by supplying functional precursors, building rules and control mechanisms. The composition of our artificial primordial soup entails a vast mixture of applicable theories, concepts, and algorithms that evolutionary robotics covers, while the fundamental mechanisms giving rise to our robots include reproduction and development. Reproduction conveys how genetic information is propagated to the next generation, whereas development describes the procedure of acquiring a phenotype from this genetic information. Development is thus the mapping of the genotype to the phenotype. In addition, development can convey plasticity and learning in individuals during their lifetime.

Many of the mentioned challenges have already been considered, and a popularized example implementing a simple genotype, an encoder, and a user-guided evolutionary process (interactive evolution) can be seen in Richard Dawkins's blind watchmaker program (Figure 1.1; Dawkins 1988).

Dawkins created a genotype containing 16 variables that were in turn interpreted by a *Lindenmayer-system* (L-system; Lindenmayer 1968a) to create biomorphs—objects resembling living organism—based on simple iterative rewriting rules. Dawkins emphasized the essential role of developmental genes that encode for *recursion* expressed in the phenotype of animals. Recursively expressed genes can lead to segmentation in cells, tissues, and overall body plans. The resulting segmentation is a phenomenon called serial homology, or similarity of structures repeated along the body axis.

The genotype to phenotype mapping dictates the encoding that is used to generate an entity from a string of information. In some cases, this string of information is encoded as a generative (or indirect) encoding, in which bits and pieces of the code are reused in a recursive process to form the resulting phenotype of an individual. In contrast, in a direct encoding, every parameter of the phenotype is defined. A fractal or grammar can lead to incredibly complex patterns in the form of a drawing based on just a few functions or rewriting rules. In this case, the simplicity of the genotype and the complex emergent phenotype is an incredible result of a developmental encoding. This type of encoding is also present in nature and can be functionally utilized in artificial systems.

Karl Sims went a step further than Dawkins and implemented a similar strategy of encoding the genotype to phenotype map for evolving three-dimensional virtual creatures (Sims, 1994b). Subjecting the genomes that

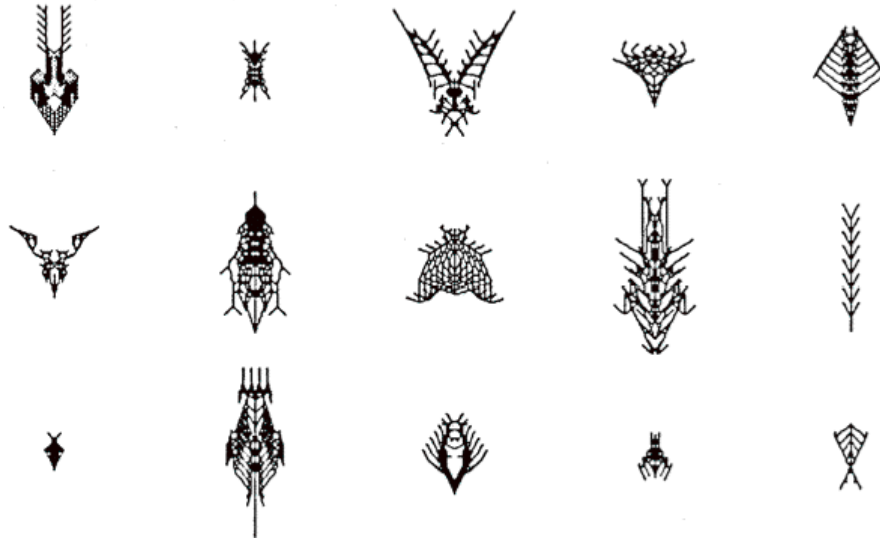


FIGURE 1.1: **Biomorphs.** Different biomorph phenotypes evolved through interactive evolution. Taken from Dawkins, 1988.

encode for the phenotypes to the evolutionary process (Figure 1.2a), many different types of phenotypes emerged with the objective of swimming or walking (Figure 1.2b). This is the first example of evolutionary computation being used to create simulated, physics-based virtual creatures. Though at the time a computer cluster was necessary to evolve the creatures, it has become more and more computationally viable to evolve these types of virtual creatures on a personal computer.

The typical approach taken in evolutionary robotics is similar to Dawkins's and Sims's approach. Evolutionary computation takes care of automatically adjusting a population of individuals through gathering performance measures from resulting phenotypes. Considering the loop of reproduction and evaluation that is essential for the correct implementation of an evolutionary algorithm, a minimal schematic representation of the important aspects in the evolutionary robotics approach is illustrated in Figure 1.3. The agent in this case covers the pathway from a genotype to a phenotype through development, the agent in biology being an organism, and in the artificial context of this thesis, a robot. However, in an evolutionary setting, the agents themselves are not optimized, but rather their gene pool is. This can mean that many small robots might function just as well as a large, complex, robot or vice versa.

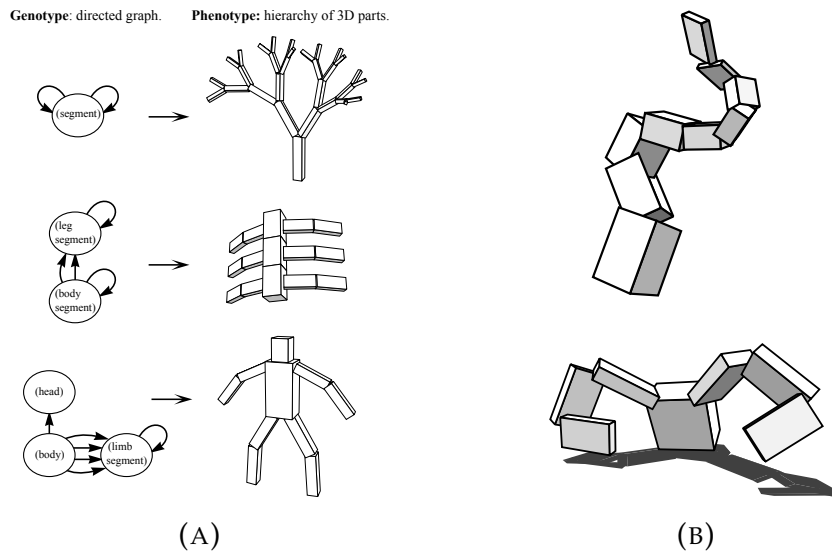


FIGURE 1.2: **Evolving Virtual Creatures.** The genotype to phenotype mapping (a) and the two evolved phenotypes (b) from Sims's work (Sims, 1994b). The two evolved behaviors were optimized for swimming (top) and walking (bottom).

1.4 Scope and Context

In contrast to conventional approaches in artificial intelligence, understanding intelligence in evolutionary systems may be of lesser importance. Instead of an intelligent thought process being evolved, other characteristics in the morphology of robots could emerge, bypassing intelligent control while still adhering to an objective (Pfeifer et al., 2006). More abstract forms of intelligence, or emergent collective intelligence, could as well suffice for artificial systems. Evolutionary robotics is therefore more concerned with obtaining behavior adhering to specific objectives or self-reproduction, rather than intelligence. This is also true for natural systems. If a morphological change would be more evolutionarily advantageous than intelligence, this would be selected for.

In ecosystems, there is usually an *arms race* within and between populations that drives evolutionary progression, the arms race being an emergent conflict between organisms for gathering finite resources. In artificial systems, the agent's goal could similarly be set by more ultimate aims, such as energy acquisition. This goal should, after all, be a requirement for reproduction. For a robotic ecosystem, primary energy producers might simply be intelligently arranged solar panels from which

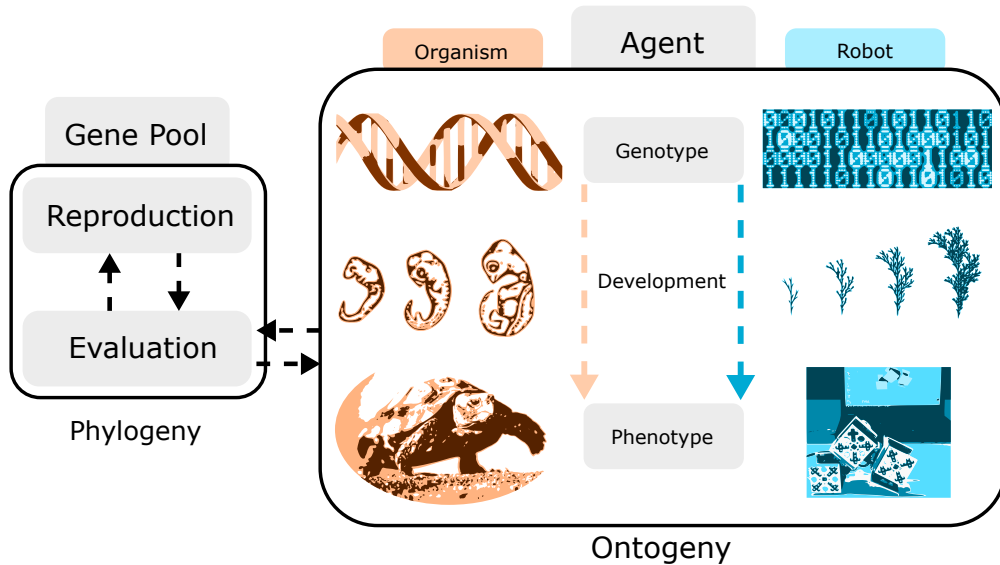


FIGURE 1.3: **Overview of fundamental steps in evolutionary robotics.** The adaptations in individuals can be considered on two time-scales: ontogeny (during an individual’s lifetime) and phylogeny (throughout evolutionary history). The genomes of agents develop into phenotypes of individuals, resulting in the organisms or robots. Based on the performance of these individual phenotypes, the genotype of the individuals reproduce. Imperfections in the replication process result in evolution.

higher-order robots can gather their energy. In addition, more proximate goals, such as the control of movement, can be set. Locomotion and energy acquisition are intertwined behaviors in nature that can be isolated as single objectives in artificial evolution, which has been done in [Part II](#).

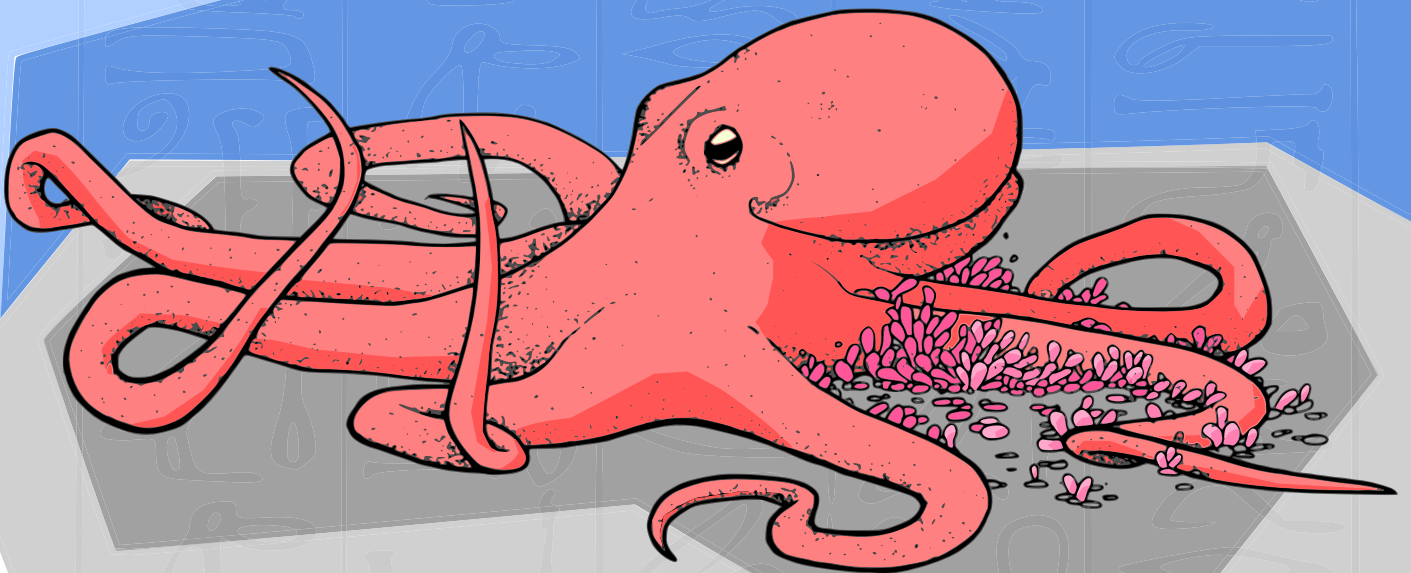
Though the type of optimization algorithm to be used can sometimes be problem specific, evolution can often be implemented as a general strategy. Parameter tuning of the algorithm can in turn be used to adapt the algorithm’s functionality to different problems. This makes an evolutionary algorithm a versatile optimization strategy. Many improvements have been made on evolutionary algorithms to increase their performance. These improvements usually address one or more of the mentioned major challenges in evolutionary robotics: *exploration vs. exploitation*, *genotype to phenotype mapping*, and the *reality gap*. From a problem-solving perspective, these challenges are mainly be addressed in the various experiments of this thesis. In addition, these problems can sometimes be related to equivalent

issues in biology. Hence, some aspects are taken into consideration from a biological perspective.

The purpose of this thesis is to elucidate various aspects of evolution in artificial systems. It will in part clarify why evolution works, how additional concepts from evolutionary biology might actually increase the efficiency of existing optimization strategies (Chapter 3), and why these concepts are important when designing new robotic systems (Chapters 4, 5, 6, and 7). It is striving for a future where artificial life can emerge through an automated process that transcends the current limitations of the autonomy and technology of artificial systems, though conceptually the hardware of the eventual system is elusive. Various strategies easing the construction of a potentially variable robotic system have been implemented. Throughout this thesis, some main intrinsic problems related to the field of evolutionary robotics are addressed, with the predominant experiments focusing on a specific type of robots called *modular robots*. This is because modular robots are reconfigurable, making it easy to not only implement different simulated robots, but also to adjust the hardware configuration on demand. Using the modular robotics approach presented in this thesis would allow any researcher, in theory, to design multiple modules and automatically evolve their composition and behavior, for either locomotion or energy acquisition, using the presented robotic platform. Simulations are used to elucidate many aspects of robot design and biology with the aid of strategies such as neural networks, evolutionary computation, modular robotics, and a range of different objectives that are discussed in the majority of this thesis.

Part I

Evolutionary Dynamics



Chapter 2

Evolution

Different sorts of survival machine appear very varied on the outside and in their internal organs. An octopus is nothing like a mouse, and both are quite different from an oak tree. Yet in their fundamental chemistry they are rather uniform, and, in particular, the replicators that they bear, the genes, are basically the same kind of molecule in all of us—from bacteria to elephants. We are all survival machines for the same kind of replicator—molecules called DNA—but there are many different ways of making a living in the world, and the replicators have built a vast range of machines to exploit them. A monkey is a machine that preserves genes up trees, a fish is a machine that preserves genes in the water; there is even a small worm that preserves genes in German beer mats. DNA works in mysterious ways.

– Richard Dawkins, *The Selfish Gene*

The origin of species—or the “mystery of mysteries, the first appearance of new beings on this Earth” as Darwin wrote in his diary (Darwin, 1845)—has long been apocryphal until the emergence of Darwinian evolutionary theory. As of today, it has been proven in many different instances that natural selection and descent with modification are the main factors promoting adaptive change in populations of organisms. The self-adapting principles of evolution have in turn been adopted as a metaheuristic optimization strategy in the paradigm of evolutionary computation. The applicability of evolutionary computation as a general problem-solver has been demonstrated in engineering for optimizing satellite antennas (Lohn et al., 2003) to bioinformatics for the prediction of RNA structures (Van Batenburg et al., 1995). Evolution is thereby not solely a theory, but also a useful applicable optimization strategy.

Natural evolution does not have a predefined goal; it is a blind process in which the only criteria are survival and reproduction (Dawkins, 1978). However, we can still view the process itself as a stochastic problem

solver, since populations are continuously adapting to their environments. These adaptations in specific traits are selected for if they increase an organism's survival and reproductive success, with some traits being adaptive during an organism's lifetime (*acquired*), whereas others are inherently encoded in their genome (*innate*; Bolhuis et al. 2005). For example, body plans are mostly defined by innate factors, while muscle mass is influenced by mechanical stress and is thus in part an acquired trait. Genes not only shape the individual organism, but their phenotypic expression also shapes the environment that can assist in their replication, a phenomenon more commonly known as the *extended phenotype* (Dawkins, 1978). The extended phenotype can change the *ecological niche* across time; therefore, it encompasses cultural evolution as well (Cagnoni et al., 2014). This is the case in ants and bees building hives, or monkeys and crows using tools for capturing insects. The organisms that interact with the environment are only the machinery of the expressed information encoded in the genome. This machinery promotes complex behaviors for locomotion, flying, digging, thinking, and even humanity. The resulting machinery would not exist without the genome, and the genome could not replicate without the machinery. All of these traits having emerged from information-containing, reproducing entities subject to evolution by natural selection.

Approach

Tinbergen (1963) has demonstrated that too often researchers described animal behavior too vaguely, and descriptions between researchers were not consistent. He proposed four different approaches to analyzing behavior, using two proximate questions on how and two ultimate questions on why behavior arises (Bolhuis et al., 2005). The two proximate questions encompass the causation and development of the behavior of the organism. Causation covers how the mechanisms of the underlying structure, or machinery, of the organism function. Development describes the conditions and factors important for the development of specific structures of the organism that are responsible for behavior. The two ultimate questions are used to explain the evolution and survival value of behavior. Evolution involves the processes that have altered the behavior across generations, while the survival value of the organism investigates why the behavior is used by the organism.

In this chapter we consider Tinbergen's reasoning and begin discussing two proximate descriptions of the genome (Section 2.2) and development (Section 2.3) with respect to the fitness landscape (Subsection 2.1.1).

The ultimate questions of survival and evolution itself are considered by discussing Evolvability (Section 2.5). This background information motivates how and why we can implement evolutionary computation (Section 2.6) and how spatial models can relate results from evolutionary computation to biology (Section 2.7). To start off, the next section will boil down the fundamentals of evolution.

2.1 Fundamentals of Evolutionary Theory

Natural selection through descent with modification is the primary driving force allowing for evolution in natural systems. This sole mechanism resulted in the grandeur of immensely diverse and unique creatures living on this planet today. Selection shapes the trajectory towards certain features present in the population, while variation through modification of the *gene pool*—the set of all genes in a given population—allows for the emergence of novel phenotypic traits. These novel traits are, for example, driven by mutation, diploidy (chromosomes exist in pairs), and crossover that promote diversity and variation in most organisms.

Even though evolution works, organisms are still riddled with imperfections that are difficult to improve through evolution. A famous example in mammals is the laryngeal nerve connecting the brain to the larynx. This nerve takes a route from the brain, around the aorta, and then to the larynx, instead of taking a shortcut by directly connecting the brain to the larynx. Though this might not be an issue for most animals, a giraffe's neck can grow over 2 meters long, meaning that signals from the brain to the larynx of the giraffe can take significantly more time than they would when directly connected. Moreover, a direct connection would also reduce the material, or number of cells, needed to make the connection. This denotes the importance of evolution working incrementally, where some changes in the design of organisms would simply require leaps that might not be easily achievable from ancestral precursors. Evolutionary adjustments are not sudden, but rather are incremental steps towards better-adapted populations, species, or genes. These incremental adjustments to a population contribute to the noticeable, yet gradual, genotypic and phenotypic change that can be seen throughout various phylogenetic trees. The incremental change is both the power and the limitation of evolution and evolutionary computation.

For evolution to occur, there needs to be variation in the population. In addition, for variation to emerge, there must be genetic change. Mutations will allow offspring to be genetically different from their parent(s). These

mutations can occur in the gametes of the organism during its lifetime and be passed on to its offspring. Mutations are mainly caused by an inaccurate duplication of a genome during cell division, but they can also be caused by other factors such as mutagens. Moreover, if chance has it, a mutation can lead to a new organism that will be better able to survive and reproduce in its environment. Since selection works in favor of the better-adapted individual, there is a higher probability that this individual is *selected* for reproduction. Thus, evolution is simply an incremental process whereby a population adapts to the conditions of a dynamic environment over time.

2.1.1 Performance Measure

The evolutionary optimization process in evolutionary computation usually needs a measure of performance to acquire adequate functionality. This measure of performance is usually denoted by a numerical value called the *fitness* value. This performance measure can be a number, or multiple numbers in the case of multi-objective approaches, directly translating to the fitness value of an agent. In biology, a *personal fitness* (Hamilton, 1964) value is commonly ascribed to the number of viable offspring an individual is able to produce. Another measure, the *inclusive fitness* (Hamilton, 1964) value, of an individual includes altruistic factors that would enable the genes of an individual to propagate to the next generation. An individual's inclusive fitness is, for example, increased when it is able to give its sibling, and thereby half of its own genes, more viable offspring at the cost of producing offspring itself. However, as noted by Dawkins (1988), the term inclusive fitness can be a cryptic measure, since it is difficult to postulate how single organisms will affect the population over time. The emergent properties of multiple individuals could be measured considering the entire gene pool of a population, since the behavior of a population is an aggregate, rather than merely the sum of the actions of agents (Holland, 2012).

The fitness values that are given are an indirect result of an organism's ability to survive and reproduce in its environment. In evolutionary robotics, fitness values are usually ascribed to the performance of robots on specific tasks. Thereby the fitness value is usually not related to the number of viable offspring an individual has produced, but rather the other way around. A fitness value is given as a performance measure for an individual in a particular environment, and this performance value will subsequently determine the number of viable offspring an individual is able to produce based on a selection operator. Ultimately, a fitness value is inferred from the genotype and is thus an indirect mapping from genotype to fitness, with

phenotype in-between. An exception to this approach can be seen in work done trying to embody the evolutionary process itself, an approach called ‘embodied evolution’ as described by Watson et al. (1999) and Eiben (2014). In this approach, robots do not have a fitness value, but evolution is an emergent phenomenon through allowing robots to reproduce based on the interactions they have with other robots in an environment.

Considering different possible genomes, naturally and artificially, a *fitness landscape* can usually be formed to map the performance of a gene or genome. This landscape can depict the reproductive rate of specific amino acid sequences of proteins, as in Wright (1932) and Nowak (2006), though the term in evolutionary computation usually conveys the performance of different genotypes. However, producing many low-performing offspring compared to producing a few high-performing offspring indicates that the measure of reproductive rate might not be a good general performance measure of a system, be it biological or artificial. Producing less, but better-fit offspring as compared to other individuals in the population should be considered. Therefore, not only reproduction, but also survival, is important.

In evolutionary computation, an individual can be considered an evaluation instance. If an evolutionary algorithm could implement strategies to find a sufficient solution that requires less individuals to be evaluated, it would perform more efficiently. Reproduction is therefore not a logical measure of fitness, but rather a type of cost to an evolutionary algorithm. Hence, the number of viable offspring or the reproductive rate is not interesting in evolutionary computation when compared to other performance measures.

2.2 The Genome and the Sequence Space

Genes cause the emerging phenotypes in nature, and therefore connect evolutionary computation to biologically equivalent genetic representations. Genomes, in biology, are the four-letter sequences of deoxyribonucleic acid (DNA) that forms the lexicon for information storage in life as we know it. Deoxyribonucleic acid and ribonucleic acid (RNA) form the blueprint for all biological life and enable the flow of genetic information through heredity, DNA being the double-stranded variant of nucleotides and RNA the single-stranded form. These strands are composed of the nucleotides adenine (A), cytosine (C), guanine (G), and thymine (T) in DNA, whereas thymine is replaced with uracil (U) in RNA. These chains of DNA and RNA can encode information that is eventually translated into

proteins, which form the building blocks of life such as enzymes, hormones, antibodies, etc. The sequence of nucleotides that contains the information for life are the genes, which are functional sequences that encode for proteins consisting of chains of typically up to 20 different types of amino acids. Every amino acid in the protein is derived from a triplet of subsequent nucleotides. Since the genetic code can potentially account for 64 amino acids, the fact that there are only 20 amino acids present in most organisms makes the genetic code redundant. Included in genes are signals for proteins such as start and stop codons, eventually allowing for the transcription and translation of the gene. The combined expression of genes can form complex gene regulatory networks that include the promotion and inhibition of other specific genes.

Mutations cause the DNA strand in mammals to change, the main cause of mutation being errors during replication of the DNA. Various types of mutations can either be detrimental, neutral, or, in very rare instances, beneficial. It can occur through the insertion or deletion of specific parts of the genome, or even the duplication of an entire genome or chromosome. These random duplication events have been important for speciation. For example, two rounds of whole-genome duplication events in vertebrates were necessary before mammals could emerge (Reece et al., 2010). Although events of gene duplication are rare, they have been crucial for the generation of new functional proteins, as is likely to have happened for red and green color cones of mammalian vision that spurred from a duplication event in ancestral species (Dehal et al., 2005).

The Sequence Space

As first described by John Maynard Smith, the sequence space is a way to describe the possible protein chains that can result from a fixed number of amino acids. Every solution on this sequence space, in turn, may have a representative performance value that can be related to the reproductive success of the corresponding gene. When a gene, and thereby the resulting protein, is better adapted, it allows for more copies of itself to be passed on to the next generation. A fitness landscape can thereby be created based on the reproduction rate of the genomic sequences (Wright, 1932; Eigen et al., 1977). Considering the potential combinations of amino acid chains that result from the triplets of base pairs, the sequence space is immense. For example, since there are 20 possible amino acids, the sequence space of a protein is 20^L , where L is the number of amino acids of a protein. It is therefore impossible to evaluate the entire sequence space of most proteins, and as a result, the mapping of this entire sequence space on a fitness

landscape is difficult. However, based on the functionality of a protein when mutated, the neighborhood of different amino acid sequences can be mapped to create a local fitness landscape.

A collection of similar genomic sequences is generally termed a *quasispecies*. Not to be confused with biological definitions of a species, the quasispecies refers to a similar set of genomic sequences that are influenced by the mutation-selection process. The similarity of sequences can be determined by their Hamming distance, which is a measure of the minimum number of substitutions required to change one string into the other. Therefore, there is an arbitrary distinction between a collection of genomic sequences that can be mapped by genetic distance between two individuals. The distinction is still useful since it can explain how the fitness values of a quasispecies can vary over time. This usually results in an adaptation of the quasispecies to local or global peaks in the fitness landscape that vary in their stability depending on the environment.

The mutation rate in a quasispecies determines this stability and the traversability of the population on the fitness landscape. Higher mutation rates allow for more variation, and thus higher traversability of the fitness landscape, though they also make certain regions in the landscape unstable. As illustrated in [Figure 2.1](#), a mutation rate (u) being above or below certain mutation rate thresholds (u_1 & u_2) will determine the stability of the population in a part of the fitness landscape (Nowak, 2006). In a robust, stable equilibrium, the resulting quasispecies does not consist of solely the fittest genomes, but rather a distribution of genomes around the stable basin of attraction in the fitness landscape. These principles of stable sequence spaces in relation to the mutation rate can shape evolutionary progressions, which are discussed throughout this thesis.

In evolutionary computation, the genes are defined as mutable parameters of a system. As has been introduced in genetic algorithms (Holland, 1975), a binary string can be used to represent the genome and sequence space of an agent *in silico*. This is the simplest analogy to a DNA string, which is also a digital, as opposed to an analogue, information storage mechanism. Since the binary genome is represented by bits, the only difference is that DNA is equivalent to a quaternary numeral system. Similar to biological measures of quasispecies and sequence spaces, a binary genome can be directly mapped to form a fitness landscape, where the fitness of genomes can be defined by specific functions that are discussed in [Section 2.6](#).

There are many factors in nature influencing the genetic information present in a species. Mendel first described gene alternatives, known as alleles. The *genotype* contains a set of alleles, while the *phenotype* is the

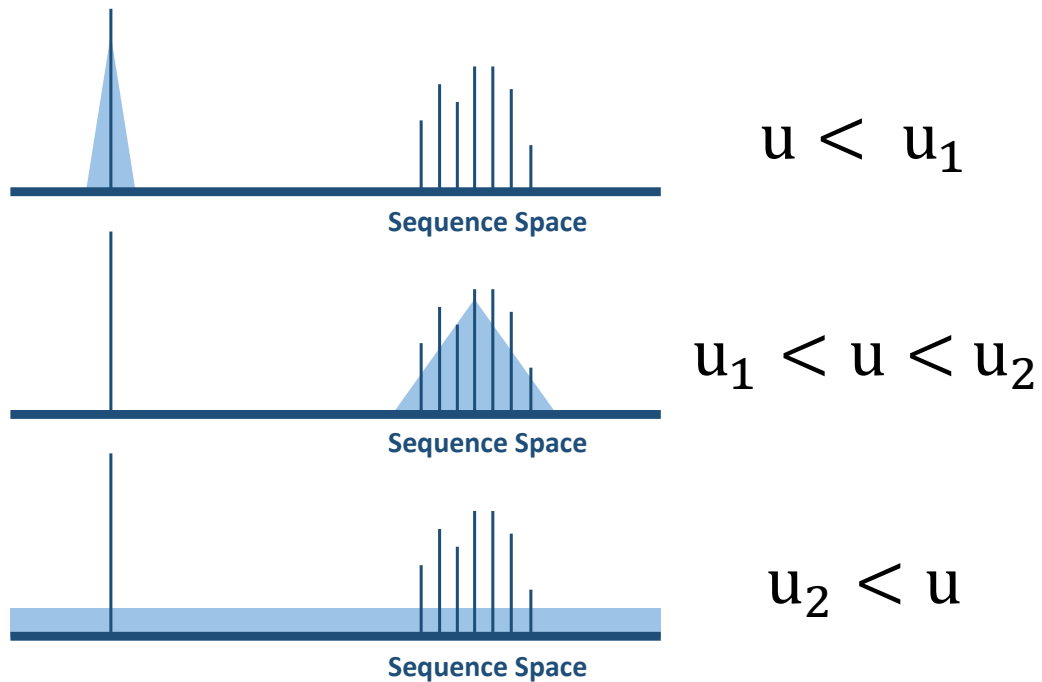


FIGURE 2.1: **Fitness landscape with a peak and a hill.** u represents a mutation rate, u_1 a critical value and u_2 an error threshold. u 's value determines the region that the quasispecies occupies as represented by the blue area. Taken from Nowak (2006)

exhibited characteristic of these alleles in the organisms (Reece et al., 2010). In diploid organisms, where individuals have two copies of the same gene, segregation of two alleles from the genome of an individual lead to many potential combinations of alleles in offspring. The random arrangement of gene alleles in the offspring alters their fitness compared to their parents, changing their phenotype. This combination accounts for some of the sustained genetic variation within a population, since there can be several combinations of various alleles in an individual. However, considering only Mendelian mechanisms, and considering only one environmental niche, the population would inevitably reside in a *zero-evolvability state* in which no new genetic information enters the gene pool. Especially when considering haploid organisms, the emergence of new genetic information necessitates change in the genetic code. This change is not solely promoted through mutations; it is also promoted by (imperfect) crossover events that allow for the recombination of genomes. Moreover, factors like gene duplication events have been important for the emergence of functional *paralogue* genes, genes that share a common ancestor.

Genetic diversity through mutation operators that change the genes of artificial agents shapes the resulting potential phenotypes and the robustness of solutions found by evolutionary computation. Although quasispecies can be defined by measuring genetic difference in a population, small population sizes in evolutionary computation usually lead to the convergence of a population to a specific area on the fitness landscape. The relationship between the variation and the ability of a population of agents to traverse the fitness landscape is important to consider in evolutionary computation, more so than other natural factors that enable the evolution of natural systems.

2.2.1 Selection

Selection acts as the driving factor for which region in the fitness landscape will be occupied by the quasispecies. Variation is necessary for the population to traverse the fitness landscape and enable a quasispecies to climb local hills or explore new regions in the sequence space. Selection pressure can sometimes be negligible for specific areas in the sequence space, and this can result in genetic drift. In this case, the genetic information is changes randomly across generational time. This random genetic change can result in *neutral evolution*, whereby change in the genomic sequence does not affect the fitness. Genetic drift can also occur in functional sequences when selection pressure is low or the mutation rate is high. An advantage of this type of drift is that a gene can transform into a new gene or find a unique sequence in the fitness landscape that could not be reached under conditions of high selection pressure. Though beneficial mutations are rare, a new, more suitable gene can potentially be found due to drift. Genetic drift can play an especially important role in the development of paralogous genes when gene duplication events occur.

For selection to work, at least two different types of genotypes with different phenotypes need to be present in the gene pool of a specific population. When considering that these two quasispecies reproduce and die in a specific environment and have an exponential growth rate, the quasispecies that contains most individuals in an environment will normally take over the entire population. This is true when the fitness values of both quasispecies are the same and lead to a concept called *survival of the first* (Nowak, 2006). However, when the quasispecies have different fitnesses, there will be a selective pressure towards the *survival of the fittest*. Alternatively, if both populations have a sub-linear growth rate, one quasispecies will never outcompete the other, leading to an equilibrium known as the *survival of all*. Selection thus shapes the frequency of the genes

in the gene pool but does not facilitate evolution on its own. It changes the frequency of genotypes based on the phenotypic fitness of the individuals and the number of individuals in a population. Since a variety of plants and animals have emerged via the process of breeding through artificial selection, evolutionary computation can be viewed as an equivalent, in which breeding is done through setting objectives. Selection operators usually determine the selection pressure on individuals in a population and, together with variations in the genes, this shapes the evolutionary trajectory of a population.

2.3 Development of the Phenotype

The phenotype entails the observable characteristics of an organism (Wolpert et al., 2007). The genotype is inherited by the parent(s), whereas the phenotype is the expression of the genes that can be influenced by the environment (Nolfi et al., 2000). An important distinction is that the environment shapes the eventual phenotype of an organism. The process of how the genotype translates into the phenotype is called *development*. The relationship between the genotype and phenotype is a mapping shaped by the environment and the encoding of the genes themselves. In evolutionary computation, this relationship between the genotype and phenotype is usually called the *genotype to phenotype mapping*, where the genes of the genotype determine the generative program for the development of the agent. This genotype to phenotype mapping is important because it can greatly influence the evolutionary progression in an artificial system. The gestalt of all of the genes, proteins, and cells is what ultimately defines the complexity of an individual. With pleiotropic genes, one gene can influence multiple, seemingly unrelated phenotypic traits that are expressed in a multitude of cells; we usually cannot attribute isolated functionalities to specific proteins.

Development in humans is brought about by the intricate network of only around 20,000 functional genes that shape the entire phenotype of an individual. Considering the number of cells and cell types present in a single human, this means that many proteins are reused in different organs and tissues. This implies that the trillions of cells present in a human exhibit *recursion* and *redundancy*. Moreover, organs and specific components of organs tend to isolate parts of their functionality, which is often called *modularity* in computer science and robotics (Stoy et al., 2010). Both recursion and modularity are important concepts that can greatly affect the evolutionary progression of artificial systems.

In multicellular organisms, an organism develops from a single cell, usually from a fertilized egg cell which contains a fused set of genes from two haploid parent cells. The egg cell, in turn, gives rise to a multiplicity of cell types. In this developmental process, many commonalities varying from within species to within kingdoms of animals can be seen. For example, all of the animals classified in the subphylum vertebrate contain a developmental process that generates three germ layers that can already be determined in the blastula stage, an initial developmental stage of an organism containing a spherical layer of cells. Moreover, many substances can act as a *morphogen*, as first described by Turing (1952), whose graded distribution across cells varies and is involved in pattern formation in organisms (Christian, 2012). A morphogen is thus an important substance encoded in genes that influences the patterned expression of a variety of traits, including germ layers (See [Text Box 2.1](#) for an example). The germ layers sprout from pole cells that are derived from the asymmetric distribution of proteins and granules across the cytoplasm of cells. Any disturbances that change the functionality of these morphogens result in catastrophic changes in the development of the organisms.

Morphogens influence the genotype to phenotype mapping in organisms, determining, for example, the patterned expression of traits across the developing organism. Computationally, having proteins that divide an organism into expressed patterns is relevant for the emergence of complex phenotypic traits in animals, and can therefore also be considered in the realm of evolutionary computation, as has been done by Dawkins (1988) and Sims (1994a). The process of artificially developing the morphology of artificial systems is largely described by the paradigm of *morphogenetic engineering* (Doursat et al., 2013), which tries to encompass the self-organization of a complex system for shaping emerging phenotypic traits. The change of the evolutionary progression brought about by implementing development is therefore useful. This approach being derived from evolutionary developmental biology is informally known as *evo-devo*.

There are, moreover, a variety of examples of development that allow phenotypic change. Heterochrony, or the developmental timing that influences the size and shape of morphological structures, is another example of a trait that is adjustable and changes the scale of resulting phenotypes (De Beer, 1940). Neural plasticity determines the change of the neurons and synapses during an individual's lifetime. Additionally, there are factors such as regeneration and metamorphosis that also drastically change an individual's layout during its lifetime.

Not all solutions are limited to an organism's ancestral history, and many similar homologue phenotypes have emerged from different species.

Text Box 2.1: Hedgehog

An example of an important gene that critically influences the development in organisms is called *hedgehog*, which also acts as a morphogen. Hedgehog, with its homologues and paralogues across species, is another example of a conserved and robust gene that plays a crucial role in the segmentation of parts of the body. In *Drosophila* it determines, along with other genes, the segmentation process of parasegment boundaries (Wolpert et al., 2007). Implementing a *null-mutation* results in the loss of this segmentation, as can be seen in the expression of denticles in *Drosophila* larva (Figure 2.2). In other organisms, null-mutations in sonic hedgehog, a paralogue of hedgehog, has a different effect in mammals. Here it has, for example, been shown to be related to the development of the neural tube and somite and foregut patterning (Varjosalo et al., 2008).

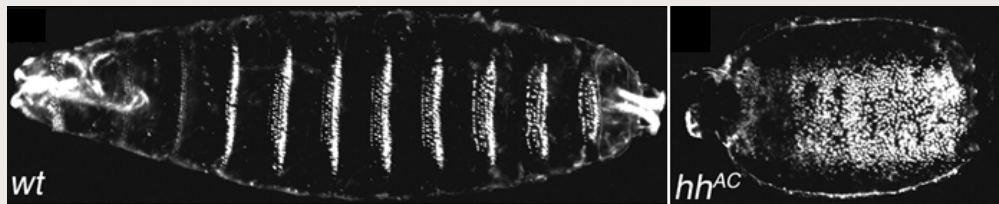


FIGURE 2.2: **Denticle formation influenced by hedgehog.** Ventral view of denticles of a wild type (left) and hedgehog knock-out (right) larval cuticle of a *Drosophila*. In the knockout a 'lawn' of denticles without clear polarity is shown, whereas the wildtype depicts the normal eight belts of denticles. Adjusted from Desbordes (2003).

Flight, for example, is not solely evolved in birds, but it has also evolved in mammals (i.e. bats). This could be considered as a form of non-bijective mapping, many different genotypes can be translated into the same phenotype. On the other hand, since many developmental mechanisms are highly conserved, entire lineages of organisms can also be stuck in local sub-optimal solutions. It could therefore be postulated that another developmental strategy would be even more advantageous for developing organisms—by, for example, having four initial germ layers instead of

three. The stepping stones required towards this fourth germ layer might, however, be too convoluted for mammals that have already evolved efficient behaviors using only three germ layers. If we could do a hard reboot of the evolution of life, we do not know whether organisms that develop using four germ layers could emerge. More distally expressed genes that might solely shape a single characteristic are likely more prone to change if they do not affect early developmental mechanisms, hence we do not see the developmental process change much on a phylogenetic timescale. Moreover, since germ layers and morphogens are usually highly conserved in the genome, it can be seen that developmental stages recapitulate evolutionary stages (Mayr, 1994). Darwin already noted that ancestral history could be derived from looking at the developmental stages of an animal (Darwin, 1872).

The genotype to phenotype mapping, in addition to the genetic operators, are important for the success of an evolutionary system (Floreano et al., 2008). This mapping largely determines how well the specific implementation of an evolutionary algorithm can traverse the fitness landscape. Additionally, acquired behavior can also greatly influence the evolutionary trajectory of a system, and this influence is formally known as the Baldwin effect (Baldwin, 1896). A classic example of how learning, a form of development resulting in acquired behavior, can improve the performance in an evolutionary context can be seen in Hinton et al. (1987), where learning has been shown to guide evolution where seemingly no evolutionary paths led. In this simple example, Hinton and Nowlan demonstrated that a learning population (based on binary representations of individuals) is more efficient, since learning grants additional adaptability of individuals during their lifetime and not only on a phylogenetic timescale. However, this work has been critiqued since, from a computational perspective, the individuals in Hinton et al. (1987) were granted more evaluations and thus also explored more of the search space (Santos et al., 2015). The concept of how learning over a lifetime can discover certain aspects that make an individual fitter could still pose a benefit, especially in environments that require continuous adaptation. In a static environment, the previously acquired trait could lead to a genetic determination whereby the trait will be incorporated genetically (Smith, 1987). Moreover, a fixed developmental *sweep* can enable individuals in a population to discover and efficiently express specific phenotypic traits (Kriegman et al., 2017). These traits might at first be expressed only shortly during an individual's lifetime, but can be selected for, which in turn expands the time individuals spend in optimal developmental stages.

Imagine a *needle in the haystack* problem in a robotics simulator, where

there is only one efficient gene with a value we denote by the integer 42. Accepting that an individual agent can be simulated for 10 iterations before its fitness value is measured, one can either test one variable during a robot's lifetime, or as in the experiments of Kriegman et al. (2017), decide to evaluate multiple numbers between two points in the lifetime of the agent. Optimizing static genomes to optimize on this landscape would then be equivalent to random search. However, by stating for example that during the 10 iterations the agent will change its phenotypically expressed value by adding one number each iteration, it is able to explore 10 values instead of one. This sweep thus makes it more probable that the number 42 will be found in the evolutionary simulator. However, when the developmental alternative finds 42, it will only exhibit this optimal state one-tenth of the time of the non-developing agent. Being able to evolve the length of expression of the gene in this hypothetical example can thus lead to initial sweeps of the sequence space, followed by the encapsulation of the specific efficient timed expression. The sweep thus allows for an increased search breadth while allowing for subsequent encapsulation of beneficial traits.

We can view the development of agents as making a solution sweep of across multiple phenotypes, learning behavior, and genotype to phenotype mappings. A developing agent that exhibits these aspects might therefore be better able to find specific solutions in the search space. Although learning is not discussed in this thesis, the mapping and development as a phenotypic sweep are important considerations and are discussed in various chapters of this thesis.

2.4 Evolutionary Game Theory

The dynamic interactions of populations in their artificial and natural environments are essential for understanding evolutionary dynamics. The struggle for life is immensely influenced by the competition between individuals and between species, and can be illustrated by simplifying behaviors or phenotypes in games and contests. Contests in ecosystems can be viewed as analogues to rock-paper-scissors style games, where certain traits of individuals can have advantages with trade-offs. Though the actual dynamics are much more complex, experimenting with variable phenotypes can elucidate potential evolutionary trajectories and determine overall stable or unstable phenotypes. Long vs. short-term adaptive traits can lead to various equilibria, basins of attractions, oscillations etc. (Hofbauer et al., 2003).

Perhaps the most influential mathematical example of population dynamics has been given by Vito Volterra and Alfred Lotka. The mathematical model of predators and prey displays the interactions between predators and preys over time in elegant equations (Equation 2.1) for prey x and predator y . This has been essential in understanding and depicting the stability and permanence of populations in a given environment.

$$\begin{aligned}\dot{x} &= x(a - by) \\ \dot{y} &= y(-c + dx)\end{aligned}\tag{2.1}$$

Simply put, x represents a population of prey and y the population of predators, with \dot{x} representing the change in population x , a the reproduction rate of prey, and b the death rate of prey x that is influenced by y . In turn, \dot{y} encompasses the change in predators over time, with c being the death rate of predators and d the reproduction rate that is based on the prey population x . These equations can thus depict a change in predator and prey frequencies over time, and usually lead to a stable limit cycle where prey and predator populations sequentially exhibit stable oscillations.

The main insight from this model is that predators and prey in a given environment oscillate periodically. It has many types of extensions where the equations can, for example, be adjusted to describe competition and cooperation between species or for more than two populations. However, the reliance on analyzing population dynamics mathematically can be complex compared to simulating populations dynamics in a computer simulation. In computer simulations, one can test multiple species that can change and evolve over time in a dynamic environment. An example of such computer simulations are agent-based models, or finite populations. These simulations can be used to confirm mathematical concepts and test new hypotheses to determine how selection and evolvability influence evolutionary models.

To implement evolutionary games in spatial models, stable equilibria of various strategies within populations can be easily simulated. For example, depending on the sole neighborhood analysis of cellular automata, different *Prisoner's Dilemma* strategies can form equilibria in populations, resulting in various emergent spatial phenomena. If defecting has a large pay-off in a population of cooperating individuals, then defecting is a behavior adhering to an environmental niche. This niche is, however, dynamic since the benefits of defecting can be low if a population is filled with defectors and high in populations only containing cooperators. Defectors

and cooperators can be defined with one value for a cell in a spatial model. To expand this type of evolutionary system, each cell could instead contain an entire genome that can elucidate more about the evolutionary dynamics of the evolutionary trajectory. This is done in [Chapter 3](#) and described in [Section 2.7](#), where each cell in a spatial model is given a binary genome that is evaluated by a specific fitness function.

2.5 Evolvability

One of the often-used definitions to explain how well a species is able to adapt on a phylogenetic time scale is *evolvability*—the capacity to evolve. Evolvability can mean the efficiency of finding solutions that make a gene, an organism, or a population better adapted to their environment. Evolvability greatly depends on the specific fitness landscape or fitness functions for a given population. A major problem that limits the evolvability of a population is the need for traversing regions of less-fit solutions to acquire a better-fit solution in the long run. In this case, an adaptive solution for a niche or environment requires a population to traverse a local valley on the fitness landscape to acquire the better-fit solution. Accepting a given mutation rate, or variability rate, in the population, this valley is difficult to cross when there is selection pressure in the opposing direction. However, since it is unknown how convoluted the fitness landscape is for higher-order organisms like mammals, a common mechanism that could allow for the traversability of these valleys would be beneficial for a species, since this prevents it from getting stuck in a local optimum and enables it to find better or unique solutions. Hence, here we find overlap between the problem-solving nature of evolutionary algorithms and the evolutionary advantage of species having a greater evolvability than others.

In the scientific literature, there doesn't seem to be a clear consensus on the term *evolvability*, and it differs in meaning from researcher to researcher. Smith (1970) mentions that for a protein to be able to evolve into another protein, it needs to traverse the sequence space of that protein without going through non-functional regions. Smith describes this process by giving an example of changing the word 'WORD' to the word 'GENE', the words being an analogy to a protein sequence. In the optimal scenario, considering single mutation steps, the change of the word WORD to GENE needs to follow a specific set of mutations that do not allow any intermediate steps to be non-words. For example, an optimal

evolutionary path for changing WORD to GENE would be: WORD-WORE-GORE-GONE-GENE. In this sequence of mutations, the word has not gone through non-functional regions in which the word would not be considered a word. This transition has moreover taken the least possible steps, and therefore obeys *maximum parsimony*. A problem arises when the transition words are lesser-fit variants of the word WORD that necessitates the evolutionary progression to traverse a local valley in the fitness landscape. In this scenario, if we consider the word GENE to be the optimal solution, this change requires the population to traverse regions of the landscape with lesser selective pressure due to the decreased performance. In the long run, finding the word GENE might enable the population to outcompete another population that has not *found* the word GENE yet, illustrating the importance of evolvability.

Although the term ‘evolvability’ is not mentioned in the original paper by (Smith, 1970), the term has been attributed to his example as explained in Haubold et al. (2006). However, as mentioned, the definition of evolvability is somewhat loosely used in the literature. It is important to clarify our meaning and argue why we did not use any other term. One of the early definitions of evolvability in computer science was derived by Altenberg (1994), who equated it to the variation in offspring produced by a parent population. A follow-up paper by (Wagner, 1996) indicated that evolvability is “the genome’s ability to produce adaptive variants when acted upon by the genetic system”. Similar to Smith’s illustration of the protein space, *adaptive* variation indicates that the variation has a chance to produce a better-fit organism. In Wagner (2008), the term is still used to describe the ability of the *system* to produce evolvable mutations, as it has also been used in Floreano et al. (2008). However, despite the more nuanced use of the word ‘system’, evolvability is still being measured in Wagner (2008) by “how easily a blind random walk starting from a given phenotype can find a pre-defined but otherwise arbitrary ‘target’ phenotype”. Robustness and evolvability were set as properties of one individual genotype (or phenotype, as in the case of their experiment; Wagner 2008).

A similar way of defining evolvability by Lehman (2012), who derived it from Wagner (1996), defines an individual to be more evolvable when, after mutating several copies of the individual, it can produce more phenotypically varied offspring than other individuals can. One of the ways of measuring evolvability, as mentioned in Lehman (2012), is that an individual can be said to be more evolvable if, after mutating the offspring, the individual is able to produce more varied, and perhaps more fit, offspring. This measurement also takes the sole individual into account as

Text Box 2.2: Locked in Imperfection

The human eye contains photoreceptors on the back of the eyeball that measure light. There is, however, one main issue with the organization of the photoreceptors since they are located behind a layer of ganglion cells (Bear et al., 2016). It could be that for some yet unknown reason having the layer of ganglion cells in front of the light receptors might be somehow beneficial, but considering the literature, the consensus is that the layer is in front of the light receptors due to evolutionary precursors. There are, however, animals, like cephalopods (Budelmann, 1995), that have evolved their photoreceptors to be in front of this layer of neurons (Figure 2.3). Intuitively, this seems to be the better option and this example illustrates how evolution can be limited by ancestral precursors. Since the eye is a highly conserved structure across species, it must be difficult to alter something in the existing phenotypes to change the position of photoreceptors to the front of the layer of cells. This might even necessitate a few generations of animals that need to cross less functional areas, where they may be virtually blind. Mammals that can traverse this landscape to enable the proper organization of cells and photoreceptors might have a distinct benefit, improving their eyesight significantly, and leading to outcompeting mammals that have not made this shift. Therefore, considering eyesight as the sole performance measure, being able to find this proper organization quicker would make a population more evolvable.

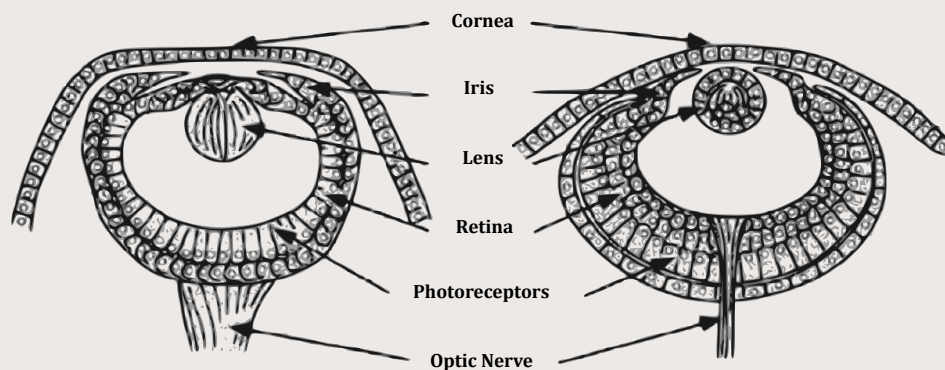


FIGURE 2.3: Configuration of the eye of cephalopods (left) and vertebrates (right). Adjusted from Novella (2008).

a measure of evolvability. However, having a varied population and not a

high mutation rate can still lead to the population being able to find diverse solutions and fit individuals. Therefore, I stress that it is the entire gene pool as a whole that contributes to evolvability. Basing a measure of evolvability on an individual should thus be similar to how Hamilton (1964) describes inclusive fitness—taking not just the individual into account, but also the population and the environment. Like inclusive fitness, evolvability being measured from the individual level would be more accurately described as *personal evolvability*. Using Hamilton’s lexicon, a better term that would allow for a distinction would thus be *inclusive evolvability*, though it can be hard to envision how an individual contributes to the evolvability of the entire gene pool. The selection operator in this case can greatly influence the existing variation in a population. Although Lehman’s and Wagner’s measurement can explain the transition of proteins explained by Smith (1970), the main disadvantage of these interpretations is that time and the rest of the population are not included in the measurement. A population could produce offspring that do not greatly vary. Measuring the change in diversity over 10 generations might indicate that this population has become more fit and more diverse than a more evolvable population as described by Wagner and Lehman. The key concept is that if a population or gene pool is better able to traverse a fitness landscape to find solutions that are more efficient than the ones in existence, it is more evolvable.

Others, such as Valiant (2009), have used the term evolvability more esoterically by equating evolvability to learning, as defined in Valiant (1984), but with the inclusion of, for example, a polynomial set of hypotheses and the toleration of a decrease in performance. Reisinger et al. (2005) and König et al. (2009) have used the term evolvability more arbitrarily and define it in terms of the genotype to phenotype mapping being able to create phenotypes that are more favorable. Reisinger et al. (2005) mentions that evolvability is “an adaptive organization of the genotype-phenotype mapping such that the search operators can produce more favorable phenotypes”. However, regardless of the genotype to phenotype mapping, a population with individuals having no mutation or crossover probability will never create a better phenotype. Therefore, though the genotype to phenotype mapping can enhance the evolvability of a system by, for example, decreasing the genome length, it is solely a property of evolution. The same genotype to phenotype mapping can produce different variations and improvements depending on the parameters, such as the mutation rate, that are being set in an evolutionary algorithm.

Natural evolution based on maximum parsimony is still prone to *premature convergence*, in which a population stagnates in a local optimum,

locking itself in local regions of the search space (Text Box 2.2). Considering benchmark tests for evolutionary algorithms, when global optima are known, the most evolvable population would be able to find the global optima quicker than others would. We can say that finding the global optima is only possible through a change in the genotype; hence, evolvability is directly determined by the gene pool of a population. In this thesis I therefore define evolvability as:

Evolvability The ability of a *population* to create *adaptive* genetic diversity across *generations*.¹

Hereby, the entire population, or gene pool, is considered as opposed to the individual, adaptive genetic diversity requires the genetic variation to eventually produce fitter individuals, and across generations implies that evolvability can be measured after a many generations, in contrast to after one generation.

Evolvability, despite its several meanings, is important for an evolutionary algorithm to properly be able to traverse the fitness landscape. There are many factors in evolutionary algorithms that will allow a population to traverse the fitness landscape more efficiently. Two of the factors discussed in Chapter 3 are mortality and development. Other factors that influence the evolvability of the system in evolutionary algorithms are the genotype to phenotype mapping, the genome length, mutation rate, and selection operators. Depending on the landscape, these factors should be taken into account for quickly acquiring decently performing algorithms.

The *robustness* of the system is directly related to the evolvability. In evolutionary computation, features like elitism, low mutation rates, and smaller genomes allow for greater robustness of the system, since there is a higher probability that the genotypic equivalent of the fittest individual is also in the next generation. In the scenario in which the best possible solution has already been found in a population, it is therefore beneficial to keep these individuals in the population, favoring a robust population over a changing one. However, since evolutionary algorithms can archive all genomes of all individuals that have been simulated, robustness is of lesser importance to evolutionary computation as compared to evolvability.

¹To clarify how evolvability is interpreted in the remainder of this thesis, a definition of evolvability has been added after the acceptance of the official thesis manuscript.

2.6 Evolutionary Computation

As mentioned previously, evolutionary computation is the umbrella term unifying all sub-fields related to evolutionary algorithms (De Jong, 2006). The first popularized form of evolutionary computation was by Bremermann (1962). From this initial approach, the subareas of evolutionary programming (Fogel et al., 1966), genetic algorithms (Holland, 1975) and evolution strategies (Rechenberg, 1973) emerged. Nowadays, it is common that the terms are not strictly differentiated, since there has been much crossbreeding in these paradigms. In this thesis, the term evolutionary algorithm specifies the algorithmic implementation of evolution, but *genetic algorithm* is still used in the experiments described in Chapter 3 done with binary genomes, as it has been used classically. However, an acronym for a steady-state genetic algorithm is SSGA, and this acronym does not always refer to a binary genomic implementation of evolutionary algorithms. The different techniques have been widely used as optimization tools without specifically attempting to model biology (Bongard, 2003). Many of the different types of evolutionary algorithms contain subtle differences. There are, for example, many approaches to encoding an evolutionary algorithm, but only a few aspects roughly shape a minimal evolutionary algorithm (Figure 2.4). An evolutionary algorithm is a population-based optimization strategy structured with an iterative procedure. An evolutionary algorithm starts with an initialization step, where genomes of a specific population size are usually randomly initialized. After the initialization step the algorithm loops through a few steps until a termination point has been reached. The phenotypes of the population of genomes are evaluated to acquire fitness values for every genome. From here on, the evolutionary algorithm loops through the following steps:

1. Based on these fitness values, a selection operator is used to select parents that can create offspring for the next generation.
2. In turn, the offspring are produced, and their genome is changed based on mutation / crossover operators.
3. The fitness of the offspring is subsequently determined based on their phenotype.
4. A replacement operator determines how offspring integrated into the existing population.

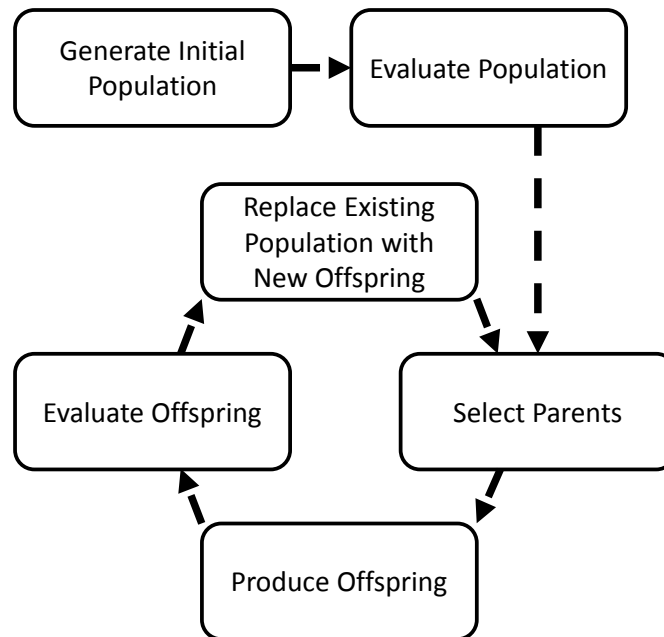


FIGURE 2.4: **Basic procedure of a generational evolutionary algorithm**

These four steps are repeated until the algorithm has reached some type of terminal requirement, which is usually set by a maximum number of generations or maximum number of evaluations.

There are two main variants of evolutionary algorithms: a generational and a steady state evolutionary algorithm. A generational evolutionary algorithm replaces the offspring of a population with the existing population entirely (Vavak et al., 1996a). A steady state evolutionary algorithm only replaces individuals in the existing population when they are outcompeted by offspring (Syswerda, 1991). Thus, the implementations vary only in their replacement operator. In a steady state implementation, instead of discarding the existing population of individuals entirely and replacing them with the offspring, the replacement operator compares the fitness of specific offspring with individuals in the existing population and only replaces an existing individual with an offspring if the fitness value is higher. A steady state algorithm can thus be compared with a population of immortal individuals that can only be replaced when they are out-competed. There are, however, various ways in which steady state algorithms can still get rid of certain ‘unwanted’ individuals, as in the deletion of the worst individuals of the population (Syswerda, 1991).

Additionally, the selection operator of evolutionary algorithms can be adjusted to give better individuals a higher chance of creating offspring

compared to less fit individuals. Selection operators can be completely random, but usually tournament selection or a sort of fitness proportionate selection (also known as a roulette wheel sampling) is used (De Jong, 2006). In addition, these techniques can also be implemented in the replacement operator, where the offspring and the existing population can be subjected to tournament or fitness proportionate selection in order to reduce the population size back to the population limit. Elitism is another commonly implemented feature in evolutionary algorithms. Elitism usually retains a certain number of the best (elite) individuals in the next generation without altering their genetic code. This generally increases the robustness of the evolutionary algorithm since the best genome is not lost. Steady state algorithms do not require an elitism operator, since the elite can only be replaced when a better individual replaces it, thus inherently promoting elites in the population.

Considering the exploration vs. exploitation trade-off, certain implementations clearly result in changing evolutionary trajectories. In essence, elitism, or the deletion of the worst individuals, usually leads to a more exploitative algorithm, while looser selective pressures lead to more exploration. Too much exploitation usually leads to a population stagnating in a local optimum, or *premature convergence* (Leung et al., 1997). To cope with premature convergence, many different techniques have been implemented to increase the diversity of individuals in a population. Classic implementations have usually considered niching (also described as speciation) methods, such as crowding and fitness sharing. In crowding, the replacement of individuals is restricted to other individuals that are similar, based on the genotypic comparison of individuals (Mahfoud, 1995). Deterministic crowding is an adaptation to crowding in which an initial step groups two parents in a population and offspring are subsequently compared to one of the parents (Mahfoud, 1995). In fitness sharing, the premise is that there is an advantage for individuals to explore different niches when another niche becomes too crowded (Holland, 1992). Here, the fitness of population elements is altered by existing fitnesses. Moreover, evolutionary algorithms can usually be tuned (before the evolutionary run) and controlled (during an evolutionary run), which can drastically change the performance of evolutionary algorithms (Eiben et al., 1999). In recent years, many new methods have been introduced to improve diversification of populations, as can be seen when using an Age-Layered Population Structure (ALPS) (Hornby, 2006; Bongard et al., 2010), novelty search (Lehman, 2012), Age Fitness Pareto Optimization (Schmidt et al., 2011) or developmental approaches (Kriegman et al., 2017), to name a few.

However, considering the existing approaches to diversification, one

should also consider that this usually helps steady state approaches to evolutionary algorithms, whereas generational algorithms tend to diversify more on their own. Although only some of the mentioned diversity-promoting methods are implemented in the experiments of this thesis, it is useful to consider these methods when discussing the specific implementation of an evolutionary algorithm and its relationship to the evolvability of a system. Since evolutionary algorithms can be adjusted in many ways, sometimes simple adjustments such as a lower selection pressure or a higher mutation rate can already make a population more diverse and potentially more evolvable, which sometimes makes it difficult to compare novel implementations to genetic algorithms that are tuned to solve a specific problem.

Computer simulations can serve as a tool for analyzing systems that are much more complex than those analyzable mathematically (De Jong, 2006). Accepting this premise, we can empirically investigate various models of natural evolution that have classically been defined mathematically, as described in Hofbauer et al. (2003) and Nowak (2006). Moreover, implementing various strategies on different benchmark problems removes the necessity for mathematical models when implementing new strategies. To investigate the performance of variants of evolutionary algorithms, fitness landscapes can be manually defined for a specific sequence space. It is therefore commonplace to use benchmarks that might consist of functions that translate the genome of an agent to a specific performance measure. These functions can be defined in many ways, though the important functions implemented as testbeds in this thesis are either *continuous single objective functions* or *binary functions*. Through this approach, a basic implementation of the evolutionary algorithm would just consist of the problem function and the evolutionary algorithm itself. Although the eventual aim of using benchmarks could be to find efficient algorithms to implement in a robotics simulator, testing new algorithms directly on a robotics simulator is of limited value since the fitness landscape is largely unknown due to the large potential search space.

2.6.1 Binary Approach

In genetic algorithms (GAs), as classically defined by Holland (1975), the genome of an individual is binary and is represented by a binary genome or a bit string. To test how well an evolutionary algorithm performs, a function can give the bit string a fitness value based on the sequence of bits in its genome. For example, the one-max function (Figure 2.5) provides each genome with a fitness value based on the number of bits with a value of

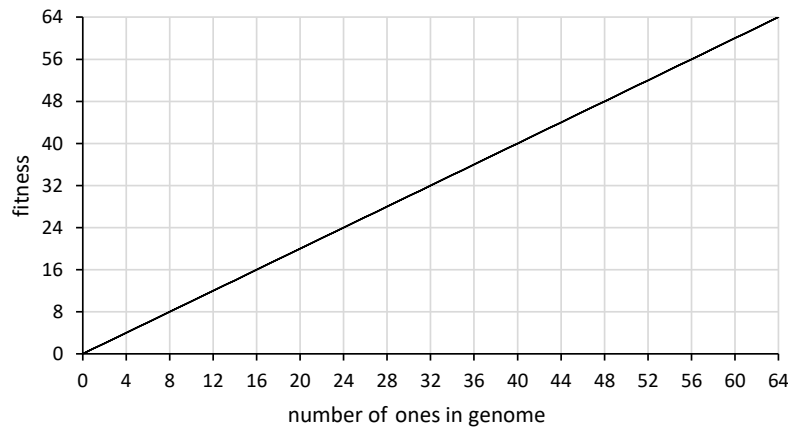


FIGURE 2.5: **Fitness landscape of one-max function.** Fitness landscape of a 64-bit sequence space. The global optimum in this landscape is located at the absolute right side of the landscape when the fitness is plotted as the function of the number of ones in the genome.

one present in the genome. Since every possible gene can directly improve itself based on the addition of a one to their genome or the replacement of a zero with a one, the one-max function/problem is one of the most basic functions on which exploitative algorithms work best. Due to the random probability of bits in these bit strings being mutated, a normal GA applied to this function quickly converges to a population of genomes that have a genome that consists of solely ones. The one-max function is hardly interesting for solving with evolutionary algorithms since, due to its linearity, it is not at all representative of robotic systems or natural systems. It is not a challenge for any optimization method.

There are a few binary fitness landscapes that are more challenging to solve and have been designed to be convoluted and deceptive such as the Chuang f1 (Chuang et al., 2010), Royal Road (Mitchell et al., 1991), or the hierarchical if-and-only-if (H-IFF) function (Watson et al., 1998). H-IFF is the landscape mainly used in this thesis. The H-IFF function creates a fractal deceptive fitness landscape and can be used to evaluate the performance of implementations of evolutionary algorithms. In H-IFF, a binary genome is evaluated based on self-similarity. One can check for self-similarity in the genome in multiple layers by initially checking the similarity of a pair of bits across the genome, continuing in the next layer by checking a pair of two bits, followed by checking a pair of four bits etc. In each layer, a fitness value can be ascribed to the self-similarity score of the genome. This score is usually derived from the number of self-similar parts in the genome and

the layer depth that is being checked.

As an example, [Table 2.1](#) illustrates how one can derive a fitness value from a 16-bit genome that results in four layers on which to check for self-similarity. Note that in the original implementation a *null* bit was possible in the genome, which resulted in an additional fifth layer. This *null* possibility has been omitted in this thesis to increase the computational efficiency and ease the visualization of genetic change over time. The omission of this *null* possibility makes the evolutionary progression easier to visualize by plotting the fitness value over the number of ones in the genome ([Figure 2.6](#)), similar to the representation of the one-max function depicted in [Figure 2.5](#). The gray area in [Figure 2.6](#) illustrates the possible fitness values an individual can achieve when having a certain number of zeros in its genome. The landscape is, however, unchanged, with as many local optima as the original implementation. The maximum fitness of an individual with either only zeros or only ones is 192, and this is the value of the global maxima on H-IFF. However, when half of the genome is composed of zeros and the other half of ones, the fitness value of that particular individual ranges somewhere between 4 and 160 depending on the specific order of the bits.

For H-IFF there are two potential global maximums regardless of the length of the genome. One global maximum contains a bit string of only ones while the other contains only zeros. In between these extremes, there are many local optima and one can generally state that there is a local optimum between each two high optima. This makes the landscape inherently fractal and deceptive. To understand how different sets of genomes correspond to fitness values based on the landscape produced by H-IFF, [Figure 2.7](#) illustrates how four genomes are located on the fitness landscape of [Figure 2.6](#) using the explanation in [Table 2.1](#). A score for self-similarity in this illustration is simply denoted by a red color. The area of the fields of the table being colored red directly translates into the fitness value. In this case, the best fit genome is a bit string of only zeros.

Considering a binary genome that consists of 64 bits, there are a total of 2^{64} total possible configurations of a genome. This large search space in turn makes it difficult for algorithms to solve binary functions. Taking a brute force approach to solving the H-IFF function would take a considerably long time, hence genetic algorithms have been employed to ease the search process. Since unknown fitness landscapes of robotics simulators might contain many local optima that may not be interesting to explore, an evolutionary algorithm that is able to traverse the landscape efficiently is essential. Though many factors influence the breadth of exploration vs. the exploitation behavior of an evolutionary algorithm,

TABLE 2.1: **Example of the score of a 16-bit HIFF genome.**
 The table shows the fitness derived from the bit string 0001-1011-1111-1111. The genome has a fitness value of 14 (out of a maximum of 32). The explanation is based on Watson et al. (1998).

row	0	0	0	1	1	0	1	1	1	1	1	1	1	1	1	tot	
1	1 · 1	1 · 0	1 · 0	1 · 1	1 · 1	1 · 1	1 · 1	1 · 1	1 · 1	1 · 1	1 · 1	1 · 1	1 · 1	1 · 1	1 · 1	6	
2	2 · 0		2 · 0		2 · 1		2 · 1		2 · 1		2 · 1		2 · 1		4		
3	4 · 0				4 · 1				4 · 1				4				
4	8 · 0								8 · 0								0

it would be ideal to have an algorithm that can be adjusted over time or even be self-adjusting to deal with premature convergence. Moreover, implementing strategies derived from population dynamics can feed back as to why we see certain features in nature arise. A large chunk of the next chapter discusses this issue in detail by showing how mortality as a bio-inspired implementation influences the efficiency of traversing the fitness landscape on H-IFF while comparing it to existing methods.

2.6.2 Continuous Single Objective Functions

In considering domains different from binary functions, we can look at continuous single objective functions. In this case, it is a variable of the genome that can be represented by one or more floating-point numbers. In these implementations, a mutation operator does not change a bit, but rather changes the value of the numbers slightly. Usually with a mutation operator based on a Gaussian distribution (De Jong, 2006). An objective function can in turn determine the fitness value of the specific gene containing a floating-point number. Based on the specific parameters implemented in the evolutionary algorithm, the efficiency at which the algorithm can achieve the maximum fitness changes. Similar to the one-max binary function, many other functions are available that have a similar effect on continuous objective functions. Some specific evolutionary algorithms are especially effective to be implemented on these types of landscapes, such as Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES; Hansen et al. 2003). In CMA-ES, the algorithm is able to predict mutations toward a specific basin of attraction. Covariance Matrix Adaptation Evolutionary Strategy is usually superior to other evolutionary algorithms on this landscape since mutations are not directed in normal evolutionary algorithms. It is also able to efficiently traverse some types

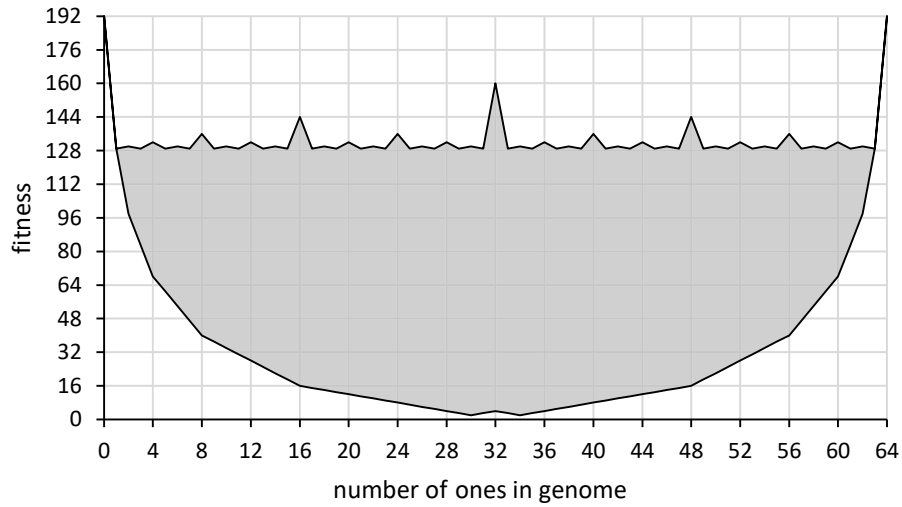


FIGURE 2.6: **Fitness landscape of the adjusted H-IFF function.** The global optima in this landscape are at the edges of the distribution and in there are local optima fractally dispersed between every other two local optima.

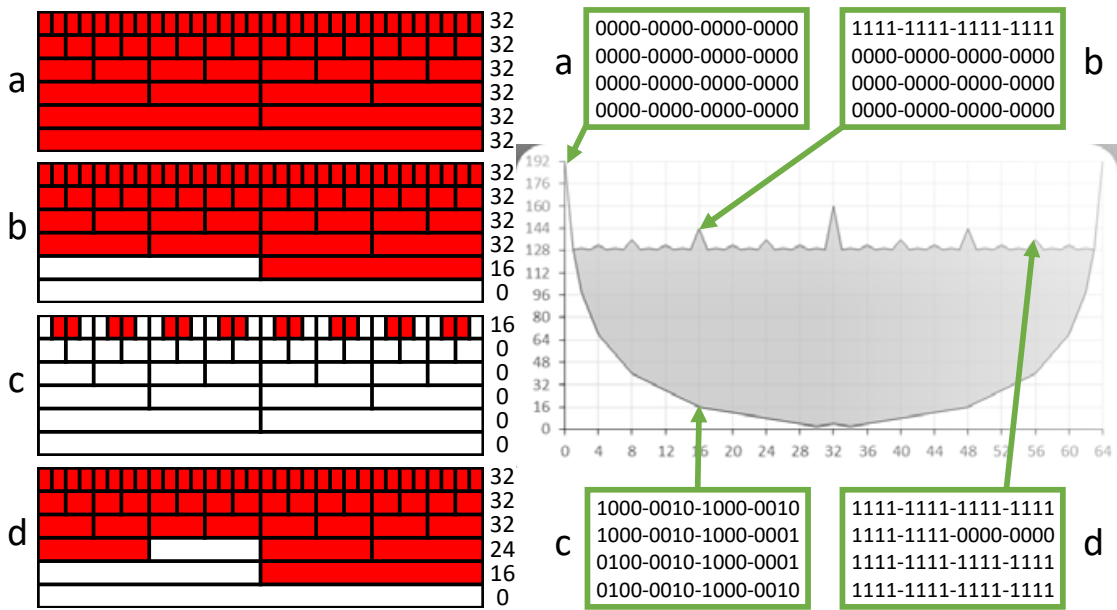


FIGURE 2.7: **Explanation of the H-IFF function.** Four genomes of length 64 are shown (a,b,c,d) with their corresponding fitness value. Left shows the scoring tables where red indicates a reward for self-similarity, as shown in Table 2.1. The red area directly translates into a fitness value of the individuals.

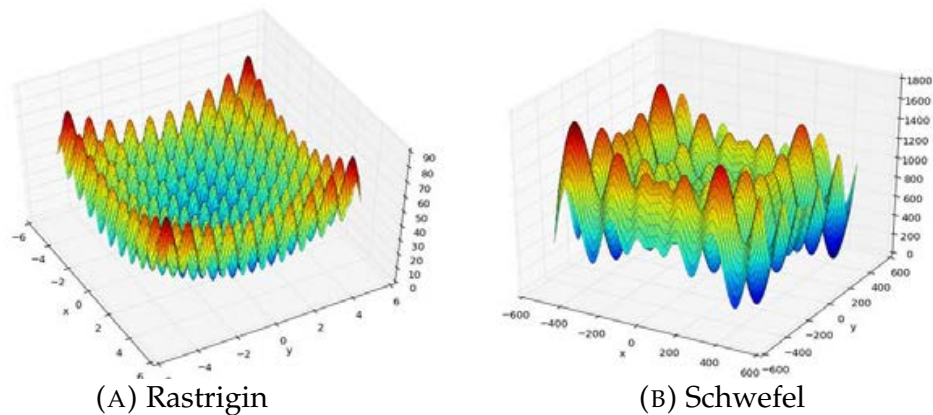


FIGURE 2.8: **The Rastrigin (A) and Schwefel function (B).** Shown are is the fitness landscape of genomes of size two that can be used for maximizing or minimizing n -dimensional genomes containing real values (Fortin et al., 2012).

of convoluted landscapes, such as the Rastrigin function (Figure 2.8a). However, the implementation of CMA-ES becomes problematic on more deceptive landscapes that are similar to H-IFF. One specific example of a difficult-to-solve deceptive continuous objective in this case, is the Schwefel function (Figure 2.8b). Covariance Matrix Adaptation Evolutionary Strategy has more difficulty finding the global optima since there is no gradient towards the optimum unless it contains a large enough population to already sample the local area of the global peak (or valley, since it is a minimization function). To emphasize, the control and creation of robots is probably also deceptive in certain regions of the search space. Hence, the ability of an algorithm to solve deceptive landscapes is important to evolutionary robotics where the fitness landscape is largely unknown.

2.7 Spatial Models

To further investigate the performance of parameters of evolutionary algorithms and to test biological hypotheses, spatial models can be used to see how a population of agents performs in a minimal environment. Predator-prey-like models can therefore be implemented to view how the relationship between predator and prey changes over time and shapes the evolutionary trajectories. Since H-IFF is a difficult function to solve for a population of agents, it has therefore been implemented in the spatial

model to evaluate how well a population of simulated agents is able to traverse the search space.

The spatial model implemented in this thesis was originally based on a Cellular Automata program architecture.²³ This is a C++ based application that has been modified to generate a predator-prey-like spatial model. In the spatial model, it is possible to create cells that are primary energy producers (plants), primary energy consumers, (rabbits) and secondary energy consumers (foxes). It has been modeled as an abstraction of energy flow in a natural ecosystem. Energy in nature is acquired by autotrophic creatures such as plants that, through the utilization of solar energy, create organic compounds. Considering a food chain, heterotrophs gather energy by consuming autotrophs; they cannot produce their own energy. Heterotrophs are in turn labeled primary energy consumers, secondary energy consumer, tertiary energy consumers, and so on, depending on their hierarchical level in the food web that is determined by the trophic structure. Rabbits are autotrophs that consume plants and convert the energy from plants into their own energy. Foxes, being secondary energy consumers, in turn gather this energy from rabbits.

The different cell types are defined by the properties of each cell. In the spatial model, each cell contains its own genome, its type, its movement efficiency, and subsequent fitness values. The type determines characteristics of movement and biomass. A type 0 cell is for example considered equivalent to a plant cell. However, most of the experiments done in this thesis focus on primary energy producers and primary energy consumers. The plant cells steadily acquire energy or biomass at a fixed rate (simulating light absorption) and each cell that is not occupied by either a rabbit or a fox will automatically increase the biomass of that specific cell. This biomass can in turn be consumed by the rabbit cells that transform the plant biomass into usable rabbit biomass. If the rabbit has accumulated enough biomass, it has a chance to reproduce, depending on the reproduction rate. A simple pseudocode for this spatial model can be seen in [algorithm 1](#).

Though the spatial model is similar to others, a few more factors are introduced in the model for studying complex evolutionary dynamics. First, to make the cells in the spatial model evolvable, each cell contains a genome and a corresponding fitness value. The fitness value represents an efficiency measure with which the rabbit cells are able to consume plant

²Original code used as a template can be found at: <https://github.com/Muzkaw/Cellular-War/>

³A minimal source code of the adjusted version used in this thesis can be found here: <https://github.com/FrankVeenstra/ALife2018>

Algorithm 1: Spatial Model Simple Pseudocode

```

initialize population  $P = \{X_1, \dots, X_N\}$ ;
while  $g := 1$  to  $G_{max}$  do New Cycle
  for  $n := 1$  to  $N_{max}$  do Update Cell
    if  $Cells[n] \neq X$  then
      | Mass := Biomass Production Rate;
    else
      | Mass := -Mass Loss Rate;
      | Move;
      | Eat;
      | Reproduce;
    end
  end
end

```

cells. The genome of the rabbit cells consists of a bit string either of size 8, 16, or 32. From this genome, the fitness value is computed by evaluating the genome with the H-IFF function at the start of a new individual rabbit cell. The model is now evolvable, and we can study how mutation rate, population size, and mass acquisition influence the evolvability of the rabbit population.

A model for development and mortality are implemented in the spatial model and experimented with in [Chapter 3](#). In the case of the mortality experiments, the cells can have a maximum age or terminal age, a probability of death, and the condition of being either mortal or immortal. The terminal age has been changed in [Chapter 3](#) to assess how this affects the evolvability. For investigating development, the algorithm includes multiple genomes that are expressed in different stages of their lifetime. This implementation is also further described in [Chapter 3](#). The full algorithm of the spatial model can be seen in [Appendix A](#).

The spatial model consists of a matrix that has the same width and height specified by the user. Smaller worlds require less computational power but also make the simulation more prone to extinction events. Considering computational requirements, the spatial world has been limited to a grid of cells that is 250 by 250 cells in width and height. In order to equate relationships in evolutionary algorithms to biology, spatial models are used as abstractions of ecosystems. Since spatial models can contain inherent elements, such as local competition, this would better represent changes in population dynamics than standard evolutionary

algorithms. With this spatial model system, implementing and evaluating existing biological concepts that are based on, for example, Lotka-Volterra equations are feasible. **Text Box 2.3** describes the implementation of the spatial model to recreate predator-prey equations from the Lotka-Volterra model.

Text Box 2.3: Simulating Predator-Prey Models

Population dynamics can contain counterintuitive elements that can be easily evaluated by the aforementioned spatial model without the knowledge of the mathematical formulation. Lotka-Volterra predator-prey state that in the absence of a predator, the prey reaches a population size close to the carrying capacity. Adding predators to the model influences the steady equilibrium, or stable or unstable limit cycles, of the population model. In a spatial model, one can discover equilibria to which the population size of predators and prey are converging. By simulating a 'rabbit' and 'fox' population, the same dynamics as explained by the equations arise. The population size of predators is greatly influenced by the reproduction resources available to the prey. Considering a spatial model initially starting with few predators and prey, one can see that there the population size of the predators is completely regulated by the resources available to the prey.

In this predator-prey model, increasing the resource available to the prey does not influence the population size much [Figure 2.9](#). A slight drop in population size can even be seen when more resources are available, which seems counterintuitive. The prey population was steady around a population size of 750, even after increasing the available resources ten-fold. The number of predators, however, do vary greatly, as there were about 500 predators when the biomass production was low, while there were around 1,250 predators when the biomass production was high. Varying the energy acquisition for the primary energy producer yields various equilibria and limit cycles, as represented in [Figure 2.10](#). Some of these parameters led to instability, as can be seen in the case of an energy accumulation rate of 0.064. The spatial model in this case uses three hierarchical steps in the food chain: plants, rabbits and foxes. Another surprising effect can be seen in the biomass of plants, with the absence and presence of predators [Figure 2.11](#). In the absence of predators, the average plant and rabbit biomass reaches a rather low equilibrium. The average plant biomass is increased again with the introduction of the predators, since the foxes protect the plants by eating the rabbits [Figure 2.11](#).

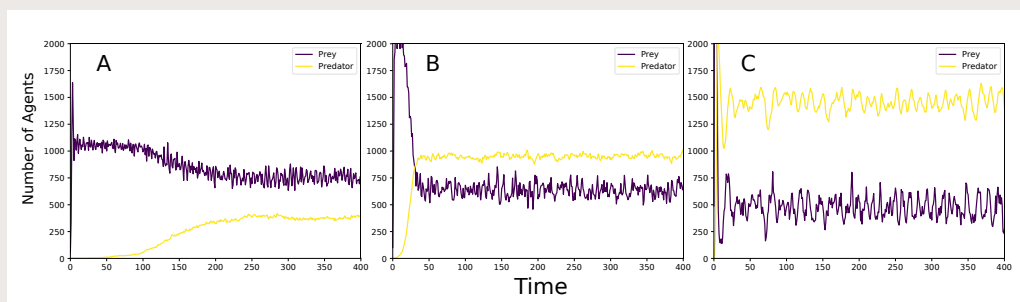


FIGURE 2.9: Number of prey and predators when varying the primary energy production. The predators and prey reach a somewhat stable population size while varying the energy production of 0.002 (A), 0.004 (B) and 0.032 (C)

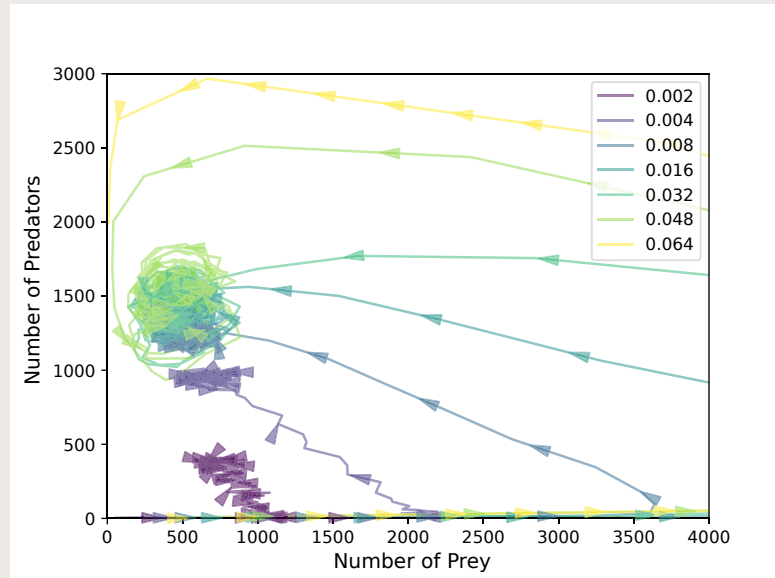


FIGURE 2.10: **Stable limits in predator-prey models.** The stable limit cycle in the spatial models using different biomass production rates ranging from 0.002 to 0.064. Every arrow represents a 20 iteration interval

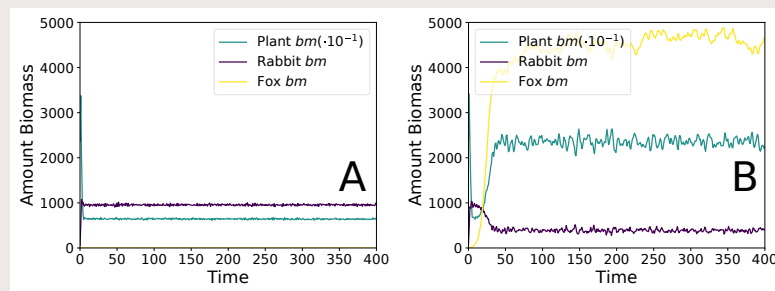


FIGURE 2.11: **Biomass of simulated plants, rabbits and foxes.** Excluding foxes (A), and including foxes (B).

The addition of another predator complexifies the spatial model even more, and one can now see that the prey population in turn grows larger with the addition of the extra predator. From an ecological standpoint this means that one cannot directly determine the health of an environment by looking at the plants or rabbits, rather, one should look at the state of the apex predators. The number of apex predators can thus determine how well the initial layer of preys, primary energy producers, are able to acquire energy and produce the biomass that can be distributed other layers of the strata. Hence, conservationists should not look at the population size of prey to determine the health of an ecosystem, but at the population size of the apex predators.

2.8 Concluding Remarks

This chapter has given an overview of the important factors influencing evolution in natural and artificial systems. These concepts are in turn taken into consideration and form the foundation on which the remainder of this thesis builds. These concepts should be taken into account in evolutionary robotics since they can significantly shape the evolvability of the simulated or real evolving entities. The importance of these concepts can be summarized as follows:

- The sequence space and corresponding fitness landscapes are important to consider for shaping evolutionary trajectories
- Mutation and selection operators determine both the robustness and evolvability of a population
- A measure of evolvability is valuable in describing the efficiency of an evolutionary algorithm
- Development can change the breadth of the search space by abstracting the genotype to phenotype mapping and allows for changes in behavior during the lifetime of an individual
- Simple evolutionary algorithms using binary or continuous objectives can serve as test beds to elucidate the effectiveness of them
- Spatial models are required for equating observable evolutionary dynamics to biology

In [Chapter 3](#), the evolvability of populations is evaluated on benchmark applications and the influence of mortality on the evolvability of a simulated population is of key interest. It concerns an *in silico* implementation not relying on any mathematical formulations. In [Chapter 3](#), both evolutionary algorithms and spatial models are used as testbeds. Development, as discussed, also forms a small part of the next chapter displaying some preliminary results of a developmental approach to evolvability. Principles of development are implemented in the form of generative encodings in [Chapters 5, 6 and 7](#). Ultimately, the evolvability of a population and the genotype to phenotype mapping of the individuals largely determine the emergence of specific phenotypes in an evolving population and are thus important for the effective design and implementation of evolutionary computation in robotics.

Chapter 3

Evolution and Longevity

The secrets of evolution are death and time—the deaths of enormous numbers of lifeforms that were imperfectly adapted to the environment; and time for a long succession of small mutations.

– Carl Sagan, *Cosmos*

Death is a seemingly frivolous property of life that is counterintuitive to the evolutionary advantage old age has when considering classical Darwinism. The older an individual can become, the more offspring it potentially is able to produce. Hence, the personal fitness of an individual is directly increased by living longer. Nature is, however, riddled with mechanisms that seem to be silly restrictions to longevity. For instance, senescence, the deterioration of function with age, is prevalent in a multitude of species. Though the individual benefit is usually related to fitness, Darwin (1872) already noted that longevity is likely a product of the complex interactions between a species and its environment. Could longevity itself somehow be a determinant for the evolutionary trajectories we see in nature?

Considering complex fitness landscapes that can be convoluted and deceptive, finding a good genotypic solution with evolutionary computation often requires the implementation of additional exploration operators. These types of operators are valuable for evolutionary algorithms since they promote the evolvability of a population by preventing the population from getting stuck in a local optimum. This chapter discusses the potential influence that mortality has on evolvability and why it could be a useful concept to implement in evolutionary computation. It furthermore demonstrates a potential evolutionary advantage of senescence in nature. In this chapter, I discuss some curiosities and theories of senescence (Section 3.1), the impact of mortality on genetic algorithms and spatial models (Section 3.2 & Section 3.3), and consider the potential evolutionary advantage of mortality in general

(Section 3.5). Moreover, some preliminary experiments were done to see how developmental mechanisms influence evolvability in a similar manner. These preliminary results on the influence of development on evolvability and age in spatial models are therefore also discussed in Section 3.4.

Approach

Although many examples of evolution can be seen as a gradual change over generational time, some might require evolutionary steps that would make individuals worse than their ancestors, thereby descending the fitness landscape. They would be required to cross valleys in the fitness landscape. Crossing these valleys would enable progeny to find a solution in the search space that is more distant, and perhaps more efficient than the ancestors' solutions. As a testbed for this potential leap, I used a deceptive fitness function: an adjusted version of the H-IFF function (Watson et al., 1998) that was described in Subsection 2.6.1. This deceptive fitness function is implemented on both an SSGA and a spatial grid model to simulate an evolvable population of individuals. The main hypothesis that is being addressed in this chapter therefore speculates on the nature of intrinsic mortality and its benefit to evolving populations (Hypothesis 1).

Hypothesis 1 *Intrinsic mortality benefits the evolvability of a population.*

3.1 Theory on the Evolution of Senescence

Most octopuses are semelparous, reproducing only once in their lifetime. Most types additionally only live around a year, an observation already mentioned in Aristotle's History of Animals (Aristotle, 1910). They "live young and die fast" (O'Dor et al., 1986). The process of senescence in the *Enteroctopus dofleini* for example, is regulated by secretions from an endocrine gland that normally causes death by starvation after reproduction (Anderson et al., 2002). After reproduction, the octopus suddenly stops foraging and instead takes care of eggs and hatchlings, followed by the eventual death of the octopus parent. However, by simply removing the endocrine gland, octopuses seemingly live significantly longer than usual, even being able to reproduce more than just once (Wodinsky, 1977). Senescence in octopuses is particularly elusive, and the true advantage of this type of senescence might be caused by various phenotypic traits and selection pressures. Does a short life have an evolutionary advantage? Or is the decreased life span a byproduct of the

mechanism inhibiting foraging behavior, which enables the protection of offspring with the dire side effect of mortality?

Other animals, such as particular salmon, undergo a similar process of senescence: dying after having laid eggs. Some spiders are cannibals and kill their male counterpart after sex (a feature also exhibited by octopuses). Elephants run out of teeth, a form of *mechanical senescence*, while some turtles express *negligible senescence* (not showing aging symptoms). Naked mole rats grow significantly older than other rodents. Artificial selection of *drosophila* can allow them to live 50% longer after a few generations (Nusbaum et al., 1994), and the proteins DAF-2 and DAF-16 directly regulate life span in *C. elegans* (Lin et al., 2001). As a final example, it has been shown that long-lived yeast mutants are outcompeted by short-lived wildtypes (Kyryakov et al., 2016). Longevity and aging thus seem to emerge differently across species. But what is the evolutionary value of senescence if there is any? We can ask ourselves whether aging is somehow beneficial to a species or if it is simply chance that individuals undergo senescence. To this day, this is a debated topic, as can be seen by the recent publications by Kowald et al. (2016) and Goldsmith (2016).

3.1.1 Summary of Theories on Senescence

Mortality is a fundamental component of natural systems that is caused either by intrinsic factors (senescence) or extrinsic factors such as predation, disease, and accidents. It initially seemed that aging is an evolutionary disadvantage for individuals since their personal fitness is lowered when an individual dies from internal mechanisms. There are, however, several theories explaining the cause and function of this biological phenomenon as an alternative to being a direct disadvantage. Darwin already mentioned in the Sixth Edition of *On the Origin of Species* (not in older editions) that longevity is related to the scale of organization, expenditure, and general activity of organisms, which has likely been determined by natural selection (Darwin, 1872). Although having an evolutionary disadvantage for the individual, it may have several advantages for the maintenance of a species. Weismann. (1889) has claimed that aging is determined by the “needs of the species”, which is subject to the same mechanical process of regulation as to other structures and functions of organisms.

An alternative theory by Medawar (1952) proposes that aging could just be a phenomenon that arises due to the simple neglect of selection pressure on older organisms, and older organisms are by chance more likely to have been prone to mortality-inducing factors that limit their lifespan. Considering a steady probability of death for each individual

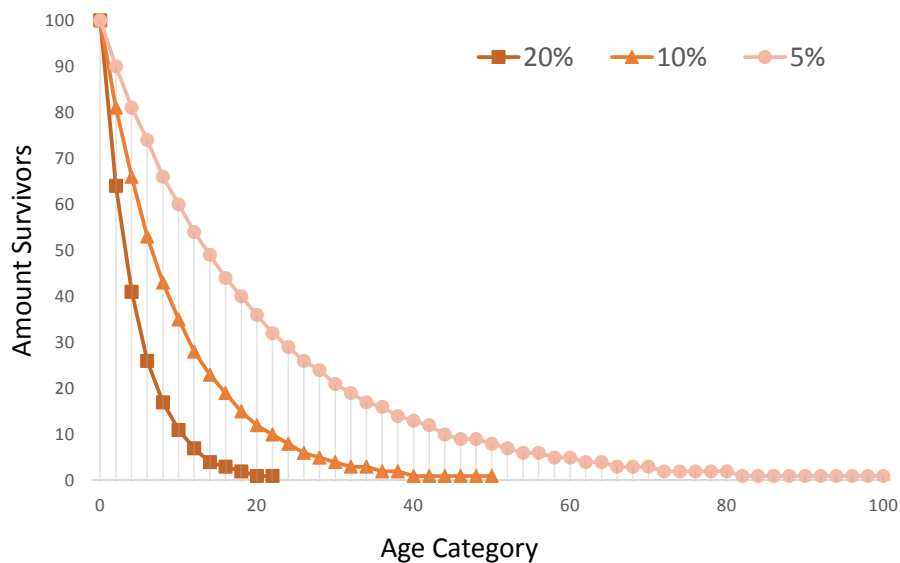


FIGURE 3.1: **Survivor curves.** Considering a fixed number of individuals entering the population every iteration, and accepting that there is a 5%, 10% or 20% probability of an individual dying by chance, it can be seen that after a certain number of iterations there will simply not be any survivors left in a specific age category. Based on Medawar (1952).

in a specific population, a survivor curve can be created displaying how many individuals in a specific age group are alive (Figure 3.1). Consider a population of 100 individuals wherein every month there is a specific chance for individuals to die—a fixed probability of death, or mortality rate. This probability shapes the age range that evolutionary selection can act on. In the case of a 20% mortality chance every month, the range of selection is quite low, whereas this range is quadrupled when the chance is only 5% (Figure 3.1). Depending on this rate, the number of older individuals in a population differs. Medawar proposed that a random accumulation could lead to this type of survivor curve. An *accumulation of mutations* leading to senescence could therefore be the result of a decline in selection pressure on older individuals in the case that individuals die due to mere chance or wear-and-tear processes. Genes beneficial in early life would therefore have a higher selective advantage and thereby a higher chance to propagate themselves into the next generation. Mere accumulation as an explanation for senescence is difficult to hold for most species since single genes that cause aging have been conserved throughout different species over evolutionary time (Guarente et al., 2000).

Genes could also have evolved to be beneficial early in life while being potentially deleterious later in life. This is a theory known as *antagonistic pleiotropy* (Williams, 1957). In this theory, a gene can have a pleiotropic effect by promoting reproductive success and survival early in life, while being detrimental later in life. Through this process, evolutionary biologists have argued that this inherent trade-off makes it difficult for natural selection to evolve old age in the first place. Although the scientific literature contains a ubiquity of examples of pleiotropic genes, it is more difficult to see how mutations in genes can improve the personal fitness in early life while having a deleterious effect later in life. These genes are sometimes referred to as “putative” disease alleles (Carter et al., 2011) since evidence that these types of alleles really have a benefit early in life has yet to be acquired.

Another alternative theory to the accumulation of mutations and antagonistic pleiotropy theories is the *disposable soma hypothesis* proposed by Kirkwood (1977), which is one of the dominating theories today (Shefferson et al., 2017). In the disposable soma theory, the body of an individual organism can allocate limited resources to various cellular processes and needs to make compromises between its metabolism, reproduction, repair and maintenance functions. For example, a population only focusing on repair can be outcompeted by a population that instead spends more energy on growth. Combined with Medawar’s survival curve, which can also be caused by extrinsic factors such as predation, this would suggest that maintenance and repair are also of lesser importance later in life, since the probability of an individual reaching old age by chance is already low. Not allocating any resources to the repair of an organism with increasing age would thus, in turn, lead to an organism’s deterioration with age as a side effect.

Other more recent theories consider the potential altruistic effect of senescence in which aging can be beneficial for coping with a changing environment (Yang, 2013; Mitteldorf et al., 2014; Herrera et al., 2016). In this case, it has been artificially shown that a terminal age is beneficial for a population in rapidly changing environments that necessitate adaptive changes in the genome. Similarly, a resulting benefit from senescence, or intrinsic mortality, is the reduction of over-consumption of environmental resources that gives a selective incentive for intrinsic mortality (Werfel et al., 2017). In addition, Lehman et al. (2015) showed that extinction events could lead to a better evolvable evolutionary algorithm, though in this case the extinction events were discriminative and kept certain *elites* in the population. From an optimization perspective, it seems that senescence, or simply mortality, can be beneficial for a population in terms of evolvability and altruistic aging.

The evolvability theory of senescence claims that senescence increases the evolvability of a population (Goldsmith 2014; see Section 2.5 for a discussion on evolvability). As I see it, mortality may aid evolvability in two ways:

1. When individuals die, a higher turnover rate of new individuals arises in the population
2. Mortality reduces selective pressure on the best individuals in the population and thereby decreases convergence, promoting diversification

For the first reason, a greater number of individuals that can live in a specific period leads to a larger proportion of acquired genetic adaptations by a population. Or, more individuals that have been 'evaluated' in the environment yield more individual fitness results from potentially differing phenotypes. Mortal populations thus contain a higher turnover rate of individuals as compared to immortal populations. For the second reason, if older fit individuals are prevented from outcompeting younger, slightly less fit individuals, the population can be stuck in a local optimum, or in a state that is less evolvable. This is of great interest since it not only has biological relevance, but can also change the efficiency of evolutionary computation.

The actual mechanisms of senescence would most likely be a combination of the theories of senescence that have been described. However, by considering the gene pool as a whole instead of thinking about the benefits of the individual, there is no reason for individually detrimental phenotypic traits to not pose an evolutionary advantage. Say mutations are the main factor driving senescence—mutations also drive evolution due to the introduction of new variations of genes in the population. A non-mutating population with an immortal life would reside in a zero-evolvability state (Goldsmith, 2008). The evolution of complex organisms can thus be a compromise between evolvability of the species and personal benefit to the individual (Goldsmith, 2008).

The antagonistic pleiotropy theory might surely be an explanation for senescence, though if senescence turns out to be beneficial for a population, this antagonistic effect would actually be an altruistic effect, or altruistic antagonistic pleiotropy. If the personal fitness of an individual could moreover be prolonged by adjusting the self-repair energy expenditure as mentioned by the disposable soma theory, there are evolutionary pressures towards better self-repair mechanisms. However, the lack in self-repair might also simply lead to more mutations, making a population more evolvable. Alternatively, the lack of self-repair mechanisms could lead to

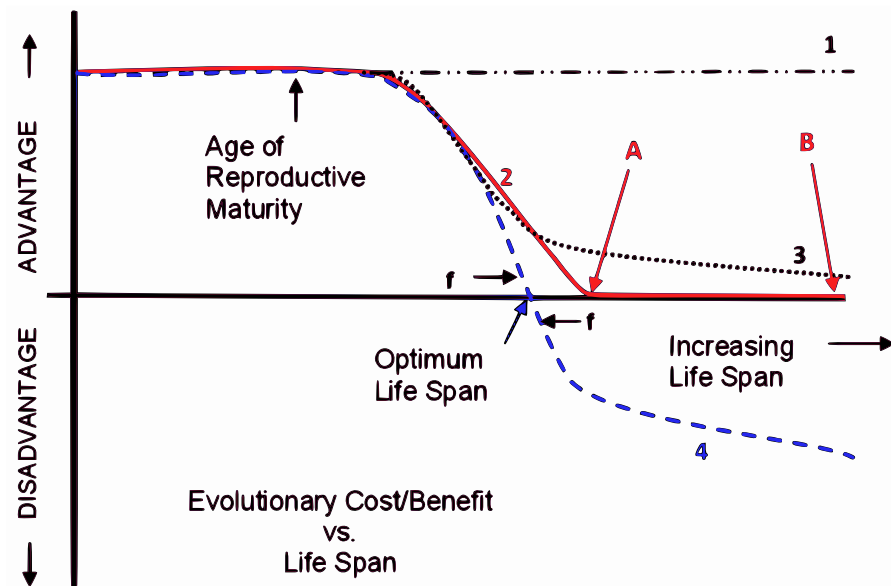


FIGURE 3.2: **Theories of aging plotted as an advantage and a disadvantage over time.** Line 1 denotes the advantage of longevity of an individual if longevity would not lead to a decrease in personal or inclusive fitness. Line 2 (solid red line) represents Medawar (1952), where the advantage of longevity would decrease with age but would not have a negative impact. Antagonistic pleiotropy and disposable-soma theories are represented by line 3 (dotted). In this case an increased lifespan does have an advantage to the inclusive fitness of a population but decreases with age. Weismann and Goldsmith support concept 4 where there exists an optimum life-span. Individuals that grow older than this life-span will have a negative impact on the population. The figure is taken from Goldsmith (2014).

senescence, which could be advantageous as well. If there is a selection pressure towards senescence, we can speak of programmed death.

3.1.2 Programmed vs. Non-programmed

The theories on senescence can be further categorized into non-programmed and programmed theories. Considering recent publications by Kowald et al. (2016) and Goldsmith (2016), this is a debated

topic. In Kowald et al. (2016), non-programmed senescence supports theories such as mutation accumulation, (Medawar, 1952), antagonistic pleiotropy, (Williams, 1957), and disposable-soma theory (Kirkwood, 1977). In contrast, Goldsmith (2016) supports programmed aging theories on evolvability (Weismann., 1889; Goldsmith, 2016; Mitteldorf et al., 2014) and altruistic aging (Yang, 2013; Werfel et al., 2017; Herrera et al., 2016) and claims this to be corroborated by biological examples. It seems that supporters of the non-programmed theories generally exclude programmed theories (Kowald et al., 2016; Shefferson et al., 2017) whereas the programmed senescence supporters do not specifically exclude non-programmed theories. The impact of the theories considering lifespan as an advantage or disadvantage are summarized in Figure 3.2, where the increased age is only considered to be a disadvantage in line 4 (supporting programmed senescence), but not in lines 1 (classical no negative effect), 2 (mutation accumulation) and 3 (antagonistic pleiotropy and disposable soma).

In Kowald et al. (2016), Figure 3.2 has been critiqued due to line 3 not representing the antagonistic pleiotropy and disposable soma theory well. It is critiqued since the parameters of these theories lead to a specific average lifespan that is optimal; hence, mortality is an emergent factor. However, if there were a gene that, at no cost, would improve the lifespan of the organism, it would have an advantage for both the antagonistic pleiotropy and disposable soma theories, but not for the programmed aging theories. Age itself is not the disadvantage, but rather the disadvantage is the product of the pleiotropic gene, or the trade-off between soma, respectively. Therefore, the distinction is still valid but should be taken lightly. The main distinction is the evolutionary advantage of the process of senescence—or the evolutionary disadvantage of long life, which line 4 displays.

3.1.3 Why Mortality Promotes Evolvability

To illustrate why mortality could potentially work in favor of evolvability, we should consider Nowak's model on the quasispecies in the sequence space again. As explained by Nowak (2006), when considering a sequence space of a specific gene, there can be several optimal regions (Schuster et al., 1988). Recall from Section 2.2 that the average mutation rate u shapes the eventual region the population occupies based on the threshold mutation rate values u_1 and u_2 (Figure 3.3). However, if genes in a population of individuals already reside in the broader less-fit state, how can they traverse the sequence space to end up in the narrow peak that is the better-fit solution? Traversing this fitness landscape would either require

an individual to drastically mutate into that region, or a population could gradually move to the region through genetic drift. Nowak's mutation rate threshold values are, moreover, only valid for a population of mortals. If immortality could occur, the immortal individual residing in the narrow peak will always stay there (since it cannot be outcompeted) and eventually, its offspring also have a chance to occupy the narrow region, no matter how high the mutation rate is. Mortality in Nowak's model is thus a requirement. Therefore, if we consider δ to be the mortality rate, I claim that there exists a mortality rate threshold δ_1 and δ_2 similar to the mutation rate thresholds (Figure 3.3). This is the initial hypothesis that has formed the premise of why mortality promotes evolvability.

A common issue with genetic algorithms is that the parameters for their optimal performance highly depend on the domain. Generational genetic algorithms inherently implement a mortality mechanism since the entire population is replaced by a new population of offspring every generation when no elitism is implemented. Moreover, deletion in steady state algorithms has also been investigated, for example, in dynamic environments and has shown to perform similarly to generational genetic algorithms (Vavak et al., 1996b). The application of a mortality rate in genetic algorithms can therefore also inform whether one should implement it in existing genetic algorithms to better traverse the fitness landscape. If mortality influences evolvability, it is thus not only of value to evolutionary biologists, but is also useful for optimization methods.

Accepting this premise that the mortality rate and mutation rate both affect evolvability, the next section attempts to experimentally verify it. Using H-IFF (Subsection 2.6.1) as the difficult-to-solve deceptive fitness landscape on both an SSGA and a spatial model can help us understand how this relationship influences the evolvability of a population. The SSGA is used as an abstract model to view the general effects of mortality on the evolutionary progression on this deceptive fitness landscape (Subsection 3.2.1). Additionally, as described by Werfel et al. (2017), spatial models can elucidate aspects of mortality that equate to natural systems which is discussed in Subsection 3.2.2. The spatial model—that contains an inherent extrinsic mortality rate emerging from local competition—is used to isolate the influence of intrinsic mortality to see whether it affects evolvability in natural systems.¹

¹The source code for both the SSGA and spatial model implementing mortality can be found at: <https://github.com/FrankVeenstra/ALife2018>

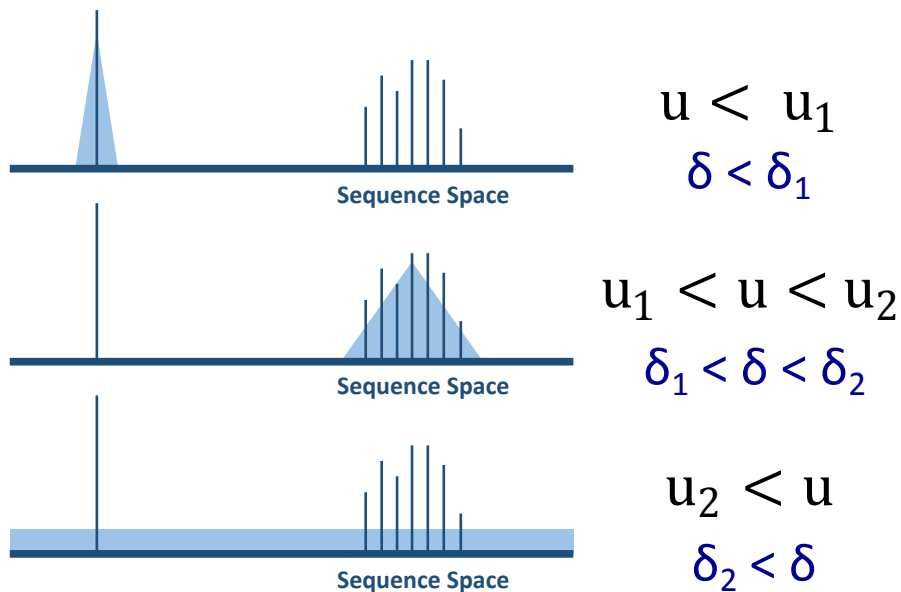


FIGURE 3.3: **Fitness landscape with one peak and a hill based on the sequence space.** u represents a mutation rate as in Figure 2.1. δ represents the mortality rate in a given population. The two threshold values for u and δ are believed to have a similar result on the stable region the population occupies on the fitness landscape.

3.2 Mortality and Mutation Rate Shape Evolvability

Though evolvability has been suggested to be a determining factor shaping longevity in nature, there have not been any artificial experiments directly investigating longevity and evolvability in this respect, apart from extinction events (Lehman, 2015; Lehman et al., 2015). Looking at the evolution of a species as an optimization problem, we can define fitness landscapes in simulation environments where simulated individuals composed of binary genomes can be evolved to conform to the maximum fitness. The experiments that back up the simulation results are divided into a benchmark optimization implementation and a spatial agent-based grid model. In both simulations, the fitness results from the H-IFF function (Subsection 2.6.1). The selection/deletion operators in the spatial model are inherent properties emerging from the interactions between the individuals and their environment as they are defined in the SSGA. Including an extrinsic mortality mechanic in the spatial model demonstrates whether the mutation rate can alter the stable region in the sequence space of

the genomes as explained by Nowak (2006) (Figure 3.3), as well as see whether an additional intrinsic mortality rate influences the evolvability of the population. The aim of the experiments in this section is to show how mortality alters the evolutionary progression of a population and to check whether this enables the population to traverse the fitness landscape more efficiently. The code is kept to a minimum; no crossover operators and only an asexual reproduction operator is used.

3.2.1 Mortality in the Steady State Genetic Algorithm

Methodology

The SSGA that includes a mortality rate uses a population of individuals containing binary genomes of length 64. The initial genomes were composed of a bit string of 1s and 0s that were randomly initialized. The bits in the genome were mutated with a probability given by the mutation rate. Note that mutating a gene randomly assigns a bit of 1 or 0, so the gene swaps a bit with half the contingency in mutation events. A mutation rate of 0.1 means that a gene is mutated with 10% probability, and thereby changes with only a 5% probability. This has been implemented to ensure a mutation rate of 1.0 would not produce offspring with the complementary bit string of their parent's genome, but rather an entirely random set of bits.

After initializing the population, each iteration of the SSGA is as follows:

1. Choose a random individual
2. Copy the genome and subsequently mutate and evaluate it
3. Compare the new genome to a random individual in the existing population and replace it when its fitness is higher

For a population size n , a generation consists of n iterations of this process. After each generation, individuals were independently checked for deletion with a probability given by the mortality rate. Deleted individuals were marked with a fitness value of -1, but were kept in the population to maintain the population size, though they were unable to reproduce. The population was logged after each generation. No crossover was implemented to isolate the effect of solely the mutation rate.

To see how mortality influences this SSGA, 20 evolutionary simulations ran for 100,000 generations on populations of 50 individuals with different values for the mortality rate and mutation rate. A mutation rate sweep from 0.0 to 1.0 was done, changing the mutation rate exponentially. A similar sweep was done for the mortality rate, although the 0.64 and 1.0 mortality

TABLE 3.1: **Number of times the optimal solution was found in the SSGA.** Different combinations of the mutation rate (u) and terminal age (δ) were used in each run. Results are taken from 20 runs of each set of parameters on 64-bit H-IFF. The subscript values represent the average number of generations (thousands) that had to be simulated before finding the global optimum. Mutation rates above 0.32 and below 0.01 have been omitted since the global optima is never found in these scenarios

$u \setminus \delta$.0	.005	.01	.02	.03	.04	.06	.08	.12	.16	.24	.32
.0	0	0	0	0	0	0	0	0	0	0	0	0
.005	0	0	0	0	0	0	0	0	0	0	0	0
.01	0	0	0	0	0	0	0	0	0	0	0	2
.02	0	0	0	0	0	0	0	0	0	0	2	2
.03	0	0	0	0	0	0	0	0	0	0	20 ₂₆	0
.04	0	0	0	0	0	1	0	0	0	14 ₄₃	3	0
.06	0	0	0	0	0	0	0	3	20 ₁₇	11 ₅₀	0	0
.08	0	0	0	0	0	1	18 ₂₂	19 ₁₉	3	0	0	0
.12	0	1	0	15 ₁₈	19 ₃₄	7	0	0	0	0	0	0
.16	2	11 ₃₅	17 ₃₅	3	0	0	0	0	0	0	0	0
.24	4	0	0	0	0	0	0	0	0	0	0	0
.32	4	0	0	0	0	0	0	0	0	0	0	0
.64	0	0	0	0	0	0	0	0	0	0	0	0
1.0	0	0	0	0	0	0	0	0	0	0	0	0

rates have been excluded since these values led to early extinction of the population and did not convey any important results.

Results

The number of times the global maximum was found in each of the 20 evolutionary runs is presented in [Table 3.1](#). In addition, the subscript values in the table represent the average number of generations ($\cdot 10^3$) it took the runs to find the global maximum, on average. As can be seen in the table, the relationship between the mortality rate and the mutation rate in the SSGA is very specific for finding the global maximum on 64-bit H-IFF within the given simulation time. Moreover, the mutation rate and mortality rate explain 89% of the variation seen in the ability to traverse to the global optimum in H-IFF ([Figure 3.4](#)). The results of the 20 runs using the SSGA are displayed in [Figure 3.5](#) (top). To see how single runs are able to traverse the fitness landscape, [Figure 3.5](#) (middle and bottom) depicts

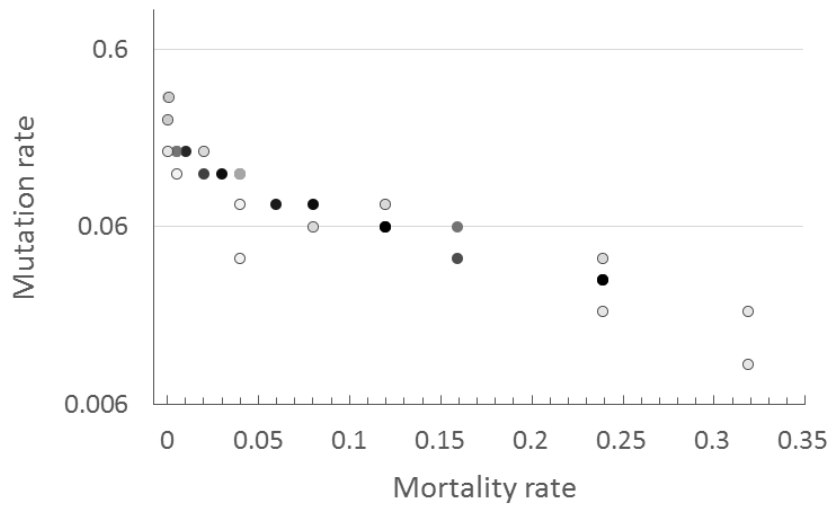


FIGURE 3.4: **Relationship between the mutation rate and mortality rate.** Mutation rate is shown in logarithmic scale. Symbols represent the number of optimal solutions found for 64-bit H-IFF. Darker colors represent more solutions for those parameters (up to 100% success). Exponential fit for the data: $y = 0.1538 \times e^{-7.28x}$, with $R^2 = 0.89$.

the fitness and diversity on the H-IFF landscape over generational time. Depicted by blue dots on the graphs are the individuals of a population at specific intervals.

Mortality rates or mutation rates that are too high lead to excessive variation, and consequently to less fit individuals. In contrast, if the mortality rate or mutation rate is too low, the population quickly stagnates at a local optimum. The proper ratio of mutation rate and mortality rate leads to a population residing in an unstable local optimum, but still fit enough to traverse the *top* of the fitness landscape and explore multiple peaks. Using the optimal mutation rate to mortality rate ratio, the ability of a population to produce adaptive diversity over generational time can be seen as creating diversity while still hugging the top of the landscape. I have informally called this phenomenon *hill hugging* since the genetic algorithm crosses valleys but does not step low in the search space compared to higher mutation/mortality rates.

Moreover, the experiments were done using 64-bit genomes on H-IFF. However, though the optimal ratio between the mutation rate and the mortality rate changes when varying the size of the genome, this optimal ratio still exists. On 128-bit H-IFF, which contains an immense number of

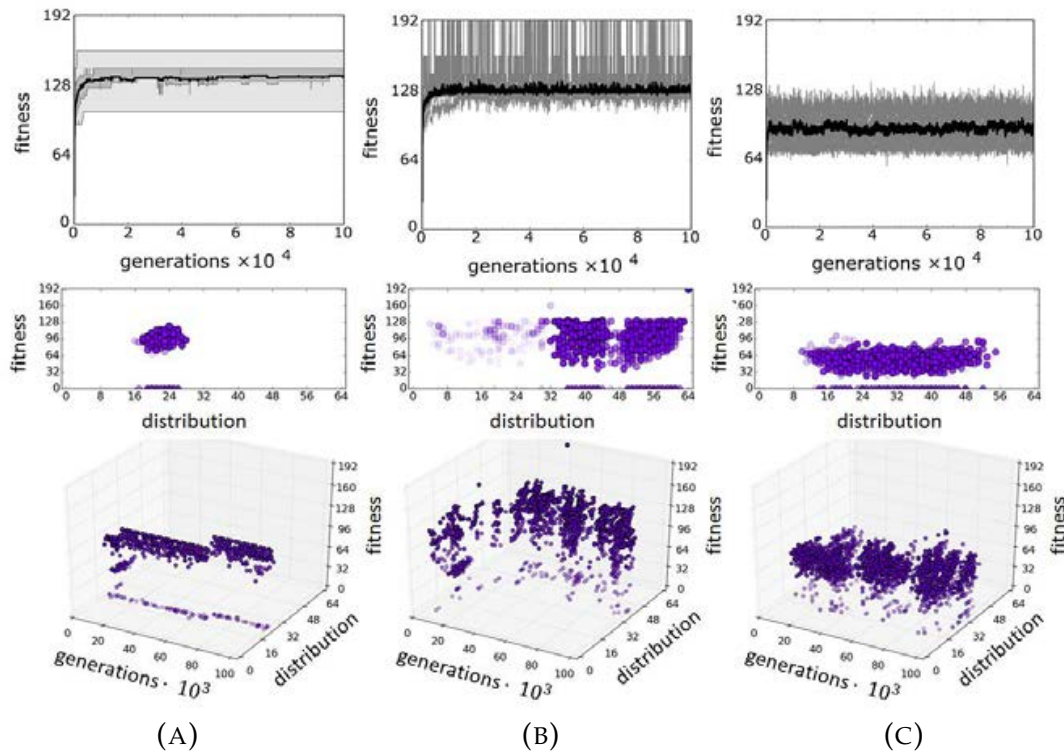


FIGURE 3.5: **Evolutionary progress for different mortality rates.** (top) The average fitness and percentiles (25-75 dark grey; 0-100 light grey) of 20 runs using a mutation rate of 0.08 and a mortality rate of 0.04 (A), 0.08 (B), and 0.16 (C). Also shown is a distribution of the population across the H-IFF landscape of a single run in comparing the distribution and fitness of individuals across the landscape (middle) and a plot of the distribution and fitness over generational time (bottom).

possible configurations almost 20 orders of magnitude larger than 64-bit H-IFF, it is still able to find the global maximum within 100,000 generations when a specific mutation rate to mortality rate ratio is used (Figure 3.6). On 128-bit H-IFF, the maximum achievable fitness value is 448, rather than 192. Out of 20 evolutionary runs, the global maximum on H-IFF, though highly unstable, was found 3 times in different runs when using a mutation rate of 0.03 and a mortality rate of 0.12. It was also found 3 times using a mutation rate of 0.06 and a mortality rate of 0.02, and it was found only once when using a mutation rate of 0.02 and a mortality rate 0.16. The other combinations of mutation and mortality rates that did not find the global maxima used similar values as done for the sweep in Table 3.1.

When performing 20 runs on only 32 H-IFF bit, the optimal mutation rate to mortality rate ratio is much broader, as can be seen when implementing a mutation rate of 0.1 and varying the mortality rate to 0.128 and 0.196 (Figure 3.7). Interestingly, implementing mortality in the form of an optimal extinction events ratio, similar to the implementation of Lehman et al. (2015), results in a similarly performing evolutionary run. In the case of the optimal extinction events ratio, 90% of the individuals were removed every 10 generations, as displayed in Figure 3.7.

In addition, the relationship between the evolvability and the mutation and mortality rate has been tested on the Chuang f1 (Chuang et al., 2010) function, as it has been implemented in Fortin et al. (2012). When changing the fitness function to the Chuang f1 function for 64-bit genomes, the optimal mutation ratio is comparable to 64-bit H-IFF as depicted in Figure 3.8. In this case, a similar mutation rate and mortality rate was required to find the global maximum in the evolutionary runs. Decreasing or increasing these rates had an effect similar to what they had on the H-IFF function.

When not implementing any type of mortality rate, the SSGA does not produce genetic variety and converges quickly. As can be seen in Figure 3.5a, the lower mortality rate clearly lessens the diversity of solutions found during the evolutionary progression. When only doing a mutation rate sweep on an SSGA, one might falsely conclude that the problem is similar to a needle in a haystack scenario. In the case of a needle in a haystack scenario, random search would be the best possible approach, and one might therefore falsely set the mutation rate of the evolutionary algorithm to 100%. However, as demonstrated by the mortality rate, this is not a fair conclusion; hence, the use of steady state approaches in general may be of limited value.

Though the relationship between a mortality rate and a mutation rate is interesting, as it significantly changes the performance of an SSGA, it is of limited interest to biologists since it does not represent a plausible spatial ecosystem. Therefore, the implementation of mortality and H-IFF on a spatial model is the main experiment conducted the next section.

3.2.2 Spatial Model

The spatial model is an agent-based grid model as discussed in Section 2.7. Like the SSGA, the spatial model implements the H-IFF fitness function producing the deceptive landscape. However, the genome size was limited to a 32 bit string genome. The genomes were only composed of 32 bits to reduce computational requirements, which were considerably

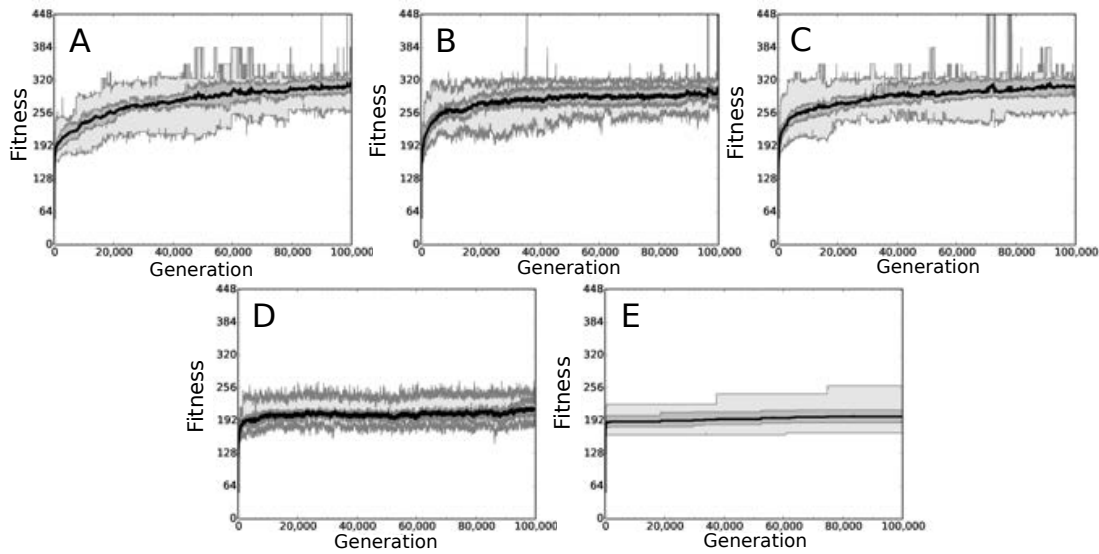


FIGURE 3.6: **Relationship between mutation rate and mortality rate on 128-bit H-IFF.** This figure illustrates the evolutionary progression of 20 runs with a mutation rate and mortality rate of 0.02 and 0.16 (A), 0.03 and 0.12 (B), and 0.06 and 0.02 (C), respectively. A relationship with a mutation rate or mortality rate that was too high led to the evolutionary progression depicted in D, while a too low mutation and mortality rate led to the progression depicted in E.

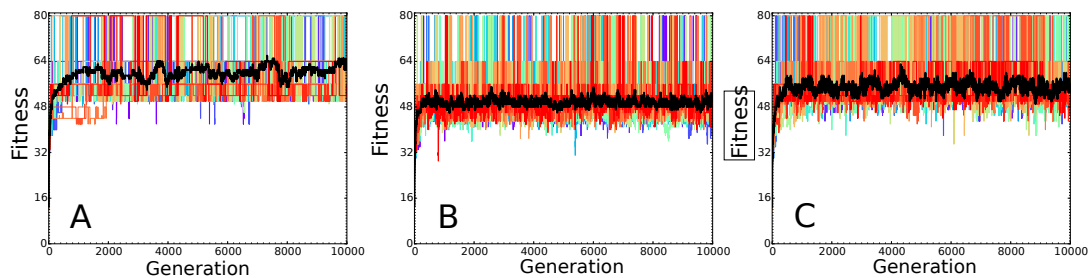


FIGURE 3.7: **Relationship between mutation rate and mortality rate on 32-bit H-IFF.** This figure shows the evolutionary progression of 20 runs with a mutation rate and mortality rate of 0.1 and 0.128 (A), and 0.1 and 0.192 (B). Using a mutation rate of 0.1 and removing 90% of the individuals each generations as extinction events results in a similar performance (C)

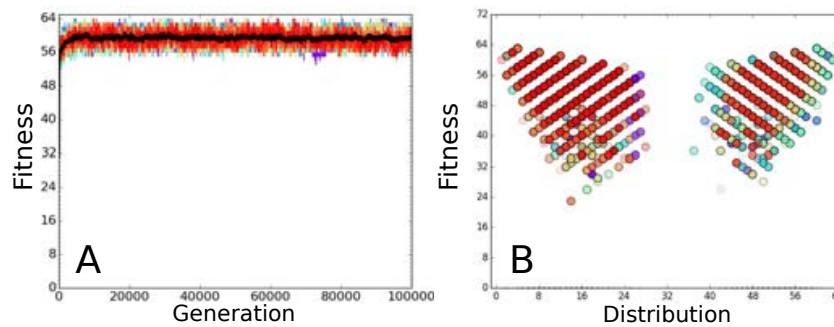


FIGURE 3.8: **Optimal mutation and mortality rate on 32-bit Chuang f1.** This figure displays the evolutionary progression of 20 runs (A) and the diversity off all individuals in the 20 runs (B). The mortality rate was set to 0.04 with a mutation rate of 0.1.

higher in comparison to the SSGA. Another difference to the steady state implementation was that the population was initialized with genomes in the middle local optimum of the H-IFF fitness landscape (i.e., 0000-0000-0000-0000-1111-1111-1111-1111, with corresponding fitness values of 64). From this starting genome, it is particularly challenging to find the global optima since no individuals of the population are close to any of the global optima, which could be the case when randomly initializing the genomes. Moreover, when randomizing genomes, the fitness of random individuals can by chance be so low that the population is never able to survive without a manual increase in their fitness. Hence, the middle local optimum was chosen as the initial genome of all individuals in the population.

The spatial model is similar to a predator-prey model and various features that were included serve as an analogy to natural systems. The experiments were performed on a 250×250 grid (same as in Kowald et al., 2016) where cells were either type 0 (prey) or type 1 (predator). One can imagine the prey and predators to be plants and rabbits, respectively, where predators were subject to evolution and each rabbit cell contained a binary genome. The fitness value derived from a rabbit's genome translates into food consumption efficiency, or metabolic efficiency. The ability to acquire food from the environment efficiently enables rabbits to grow faster, thereby producing more offspring. The spatial model is visualized in Figure 3.9 where green cells denote plants cells (the intensity of the color displaying how much biomass has been accumulated) and blue cells represent rabbit cells.

In the experiments, some parameters were set that shaped the

environmental dynamics of the system. These values were, however, flexible, and could be altered to produce similar dynamics as seen in the results. The specific values described here were set for computational efficiency. For example, the environment was not too large or too small, not too few or too many rabbit cells roamed around under certain conditions, and food was not too scarce or too plentiful. Changing these values with \pm half of the respective value did not change the presented results significantly.

At each iteration, all plant and rabbit cells were updated. A given amount of biomass was added to plant cells according to a biomass production rate, which was an absolute value of 0.0016. A plant cell from here on could grow to contain a maximum biomass value of 1.0. Rabbit cells attempted to move to a neighboring cell with an 8 in 9 chance at each iteration. If the target position was occupied by another rabbit, the rabbit would stay in position. For computational efficiency, the grid was sequentially updated from left to right and top to bottom, and it was ensured that rabbits moved only once per iteration. Furthermore, rabbit cells reproduced with a 1 in 10 chance if their biomass exceeded twice the reproduction cost (the reproduction cost being 0.4). The reproduction cost was subtracted from the biomass of the parent rabbit. Offspring started with a biomass equal to the reproduction cost multiplied by 0.8 (0.32). This multiplication stood for an additional reproduction cost, where a value of 0.08 biomass was lost during a reproduction event.

When a rabbit cell moved on a plant cell, it consumed the plant's biomass with an efficiency rate of ($fitness/maxfit$). The rabbits could not increase their biomass over the 1.0 limit; any unused plant biomass was left in the plant cell and thus stayed available for consumption. At every iteration, rabbit cells lost 0.02 biomass as a maintenance cost. Rabbits with a biomass below 0.01 were removed from the population, expressing extrinsic mortality through starvation (or local competition for food).

In contrast to the SSGA, the spatial model implements both intrinsic and extrinsic mortality. For intrinsic mortality, a terminal age was implemented and extrinsic mortality was thus the result from local competition. To explore the relationship between intrinsic mortality and mutation rate, different mutation rates and terminal ages were compared. The main results indicate how often, and how quickly, the global maximum was found on 32-bit H-IFF.

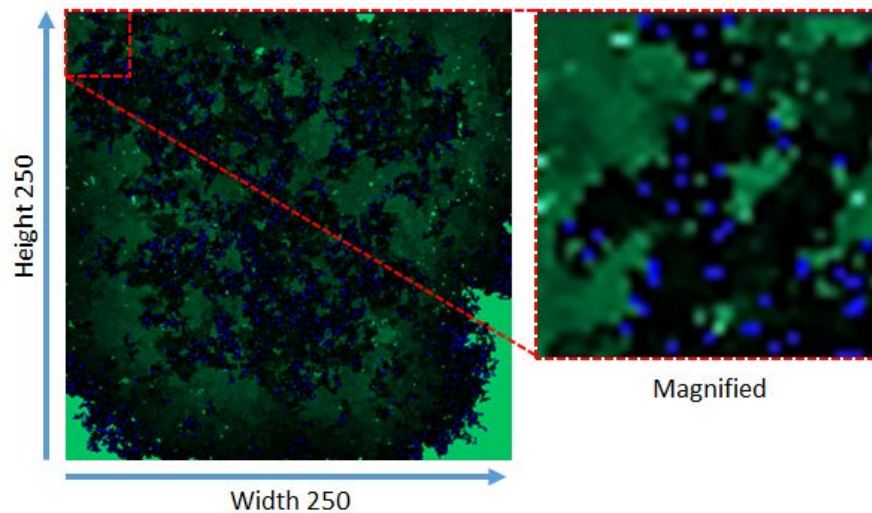


FIGURE 3.9: **Illustration of the spatial model.** Green represents plant biomass, blue rabbit biomass. Snapshot taken after the first few cycles of the spatial model.

Results

Similar to the steady state implementation, the spatial model showed a tight correlation between the mutation rate and mortality rate, as was revealed in [Table 3.2](#), for finding the maximum fitness on 32-bit H-IFF. Apart from these optimal ratios, the speed of finding the global maximum could also be determined ([Figure 3.10](#)). Though not tested for significance, a terminal age of 120 with a mutation rate of 0.06, and a terminal age of 160 with a mutation rate of 0.08, were optimal for finding the maxima the quickest.

Additionally, the optimal mortality rate to mutation rate ratio resulted in a population that was less prone to losing the global optimum once it was found ([Figure 3.11](#)). This might be because a better-fit population is able to sustain more individuals than a less fit population ([Figure 3.11](#)). There is, again, a specific ratio of mutation rate to terminal age that is optimal for the population to traverse the fitness landscape ([Figure 3.12](#)). The population of the spatial model, in contrast to the SSGA, was initialized with individuals residing in the middle of the fitness landscape. The results illustrate that despite being in a local optimum furthest away from the global optima, the solution can still be quickly found under the right parameters, the parameters being tuned to exhibit the optimal mutation rate and mortality rate.

Looking at the individual runs when using a terminal age of 60 ([Figure 3.11](#)), we observe a phenomenon similar as to the SSGA, in which

TABLE 3.2: **Number of optimal solutions for 32-bit H-IFF on a spatial model.** Results are taken from 20 runs for each combination of mutation rate (u) and terminal age (TA). ϵ marks combinations where the population went extinct in all runs.

$u \setminus TA$	40	50	60	80	120	160	500	1000	2000	-
0.01	5	0	0	0	0	0	0	0	0	0
0.015	19	8	0	0	0	0	0	0	0	0
0.02	ϵ	20	15	0	0	0	0	0	0	0
0.03	ϵ	10	20	20	0	0	0	0	0	0
0.04	ϵ	1	11	20	12	0	0	0	0	0
0.06	ϵ	ϵ	2	6	20	20	0	0	0	0
0.08	ϵ	ϵ	1	1	20	20	0	0	0	0
0.12	ϵ	ϵ	0	0	2	9	20	3	2	0
0.16	ϵ	ϵ	ϵ	0	0	1	20	20	20	17
0.24	ϵ	ϵ	ϵ	0	0	0	1	13	15	14
0.32	ϵ	ϵ	ϵ	0	0	0	0	1	0	2
0.48	ϵ	ϵ	ϵ	0	0	0	0	0	0	0

a lower mutation rate leads to premature convergence more quickly while a high mutation rate creates an unstable population. Moreover, the speed of finding the global maximum in the spatial model also differs when using different mutation rate and mortality rate variables (Figure 3.10). In this case, the speed was derived from the number of cycles the spatial model ran before finding the global maximum. A different measure of speed would be to count the number of individuals that had been simulated before the maximum had been found. For example, the only real difference to Figure 3.10 when comparing a mutation rate of 0.02 and a terminal age of 50 with mutation rate 0.16 and terminal age 500 is that the latter needs to simulate significantly less individuals before the maximum is found (two-sided Mann Whitney-u test p value 0.008). The difference in speed of the number of cycles was not significant in this comparison (two-sided Mann Whitney-u test; p value 0.2); i.e., the median number of individuals simulated before finding the global maximum was 971,792 in the low terminal age scenario and 188,148 in the high terminal age scenario. Thus, a higher terminal age in this comparison needed to simulate less individuals. Apart from this anomaly, the speed plot of using individuals as a measure looks almost identical to Figure 3.10. For evolutionary algorithms in general, the number of individuals simulated should be minimized, though

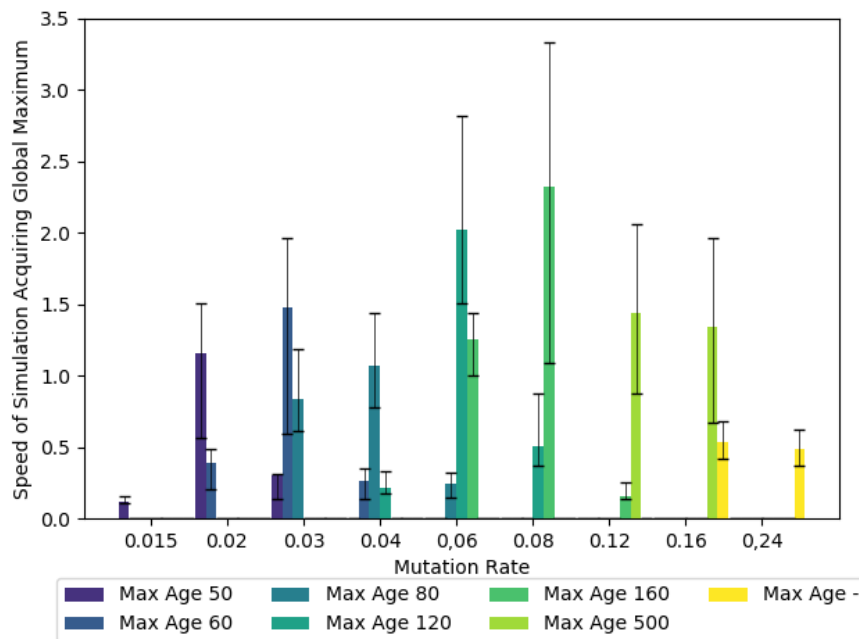


FIGURE 3.10: **Speed of solving H-IFF for the spatial model.** The figure displays the number of times the global maximum was found divided by the average number of iterations $\cdot 1000$ the spatial model ran varying the mutation rate (x-axis) and the maximum age.

simulation time in terms of cycles is what matters in real populations.

Another interesting result when using no terminal age is that although the average age of the individuals in the population remains relatively fixed across different mutation rates, the maximum age is significantly higher in high mutation rate scenarios (two sided Mann Whitney-u p value of $7 \cdot 10^{-8}$). These results were interpreted as the fittest individual being unable to produce many functional offspring due to the high mutation rates. This meant that the older, fitter individuals had, in turn, a higher chance to outcompete the other, likely less fit individuals in the populations. The elite thus became much older in scenarios where there was a high mutation rate—when looking at no terminal age, the maximum age of the population under high mutation rates could grow as high as 20,000 cycles. As displayed in [Table 3.2](#), as high mutation rates also lead to a greater number of less-fit individuals in the population, the mutation rate is necessarily low, otherwise the population would go extinct as denoted by ϵ in [Table 3.2](#). Both a mortality rate and a mutation rate could thus lead to an *error catastrophe*, where a species goes extinct due to excessive mutations. In

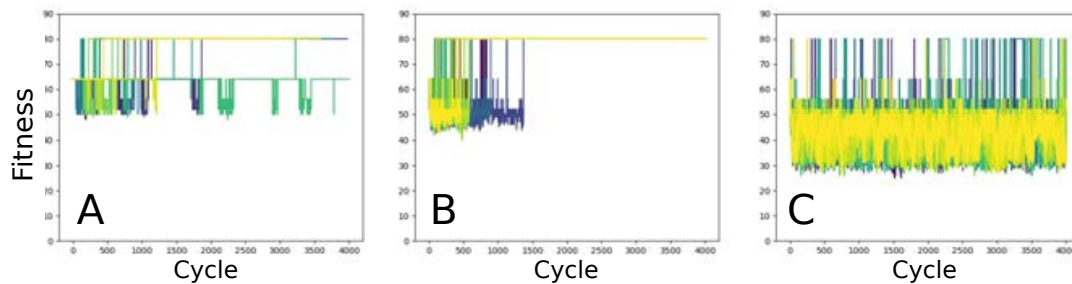


FIGURE 3.11: **Individual runs on the spatial model.** The graphs show individual runs (illustrated with different colors) display maximum fitness values across generational time when implementing a terminal age of 60 and a mutation rate of 0.02 (A), 0.03 (B), 0.04 (C). Each cycle represents 100 iterations of the spatial model.

conclusion, the mortality rate and mutation rate shape the evolvability of the population simulated in the spatial model.

3.2.3 Comparison to Other Algorithms

Although the results in this section were somewhat preliminary, the mortality rate implementation on the SSGA with H-IFF has been compared to some other conventional implementations of genetic algorithms. In particular, it has been compared with Age-Fitness Pareto Optimization (AFPO) (Schmidt et al., 2011), generational implementations, extinction events, and an SSGA containing a speciation operator as implemented in Stanley et al. (2002b). The results from these experiments do not display a miraculous efficiency of the mortality rate implementation, but they can provide evidence as to why mortality works for increasing evolvability, bolstering **Hypothesis 1**. This section therefore displays that types of evolutionary algorithms may be less effective compared to tuning existing algorithms.

Age Fitness Pareto Optimization

Since AFPO (Figure 3.13) has been used to create more diversity in evolutionary algorithms, the main initial aim of the mortality rate implementation was to beat this algorithm on a highly deceptive fitness landscape. This was initially done to see whether implementing mortality was indeed useful. Age-Fitness Pareto Optimization is a single population-based evolutionary algorithm that utilizes the deletion and selection of

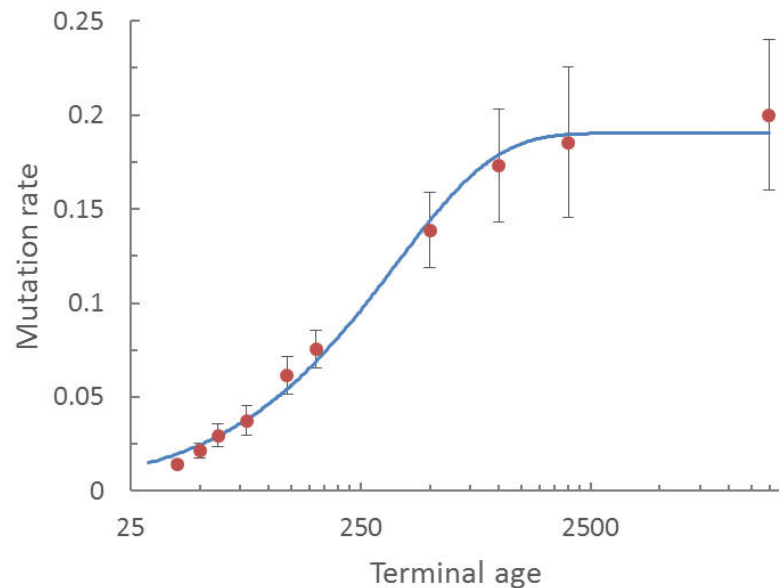


FIGURE 3.12: **Optimal mutation rate as a function of terminal age.** Note the logarithmic scale for terminal age. The continuous line shows an exponential fit: $0.1903 - 0.1907 \cdot e^{0.0028}$, with $r^2 = 0.9936$ (values closer to one indicate a better fit).

individuals based on Pareto dominance. Pareto dominance is based on two factors: a fitness value and the age of an individual; a lower age being advantageous. The different ages and fitness values in the population create a *Pareto front*—a region yielding optimal solutions for both objectives as displayed in [Figure 3.13](#).

Age-Fitness Pareto Optimization, as implemented for the experiments², starts by initializing a population of random individuals. Each individual has been assigned an age of 0, and these individuals and their asexually produced offspring increment their age value by one after each generation. After evaluating all of the individuals in this initial population, all fittest individuals in all possible age categories are kept while the rest of the individuals are discarded. Afterwards, all ages of all individuals are incremented, and a random individual is inserted into the population. This random individual has an age of 0, similar to the initial population. The new population, including the random individual, is used to generate an offspring population that is the same size as the maximum population size. Afterwards, tournament selection (with tournament size

²Researchers at the University of Vermont would call it ‘AFPO-101’ since no additional parameters were tuned

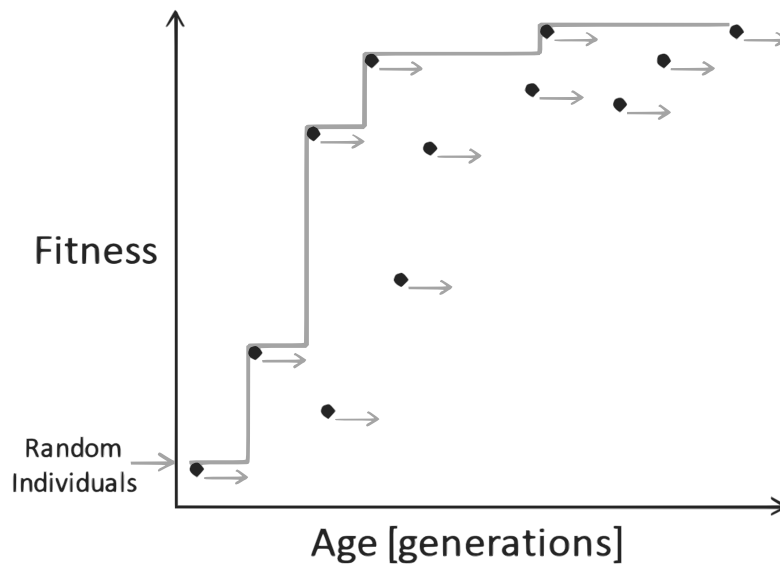


FIGURE 3.13: **Age Fitness Pareto Optimization.** The population in AFPO moves in a two-dimensional Age-Fitness Pareto space. Adjusted from Schmidt et al. (2011).

2) determines which individuals from the offspring population and the existing population will form the population in the next generation. This tournament is, however, also steady state since offspring are set up to compete with an individual in the original parent population sequentially. An individual can only outcompete and replace another individual if its fitness is higher and its age is lower, thus dominating on both fronts.

To compare how efficient AFPO is on H-IFE, an initial experiment was conducted, evaluating it on 32-bit H-IFE. Compared to the normal steady state implementation with and without mortality, it seemed to perform similarly to the mortality SSGA. Increasing the size of the genomes to 64 bits quickly changed the outcome of the performance of AFPO compared to the SSGA implementing mortality. A mutation rate sweep was done to determine the optimal mutation rate for AFPO. This optimal mutation rate was around 0.1 and was subsequently used in the experiments. The optimal mortality rate was tuned using the 0.1 optimal mutation rate of AFPO and was set to 0.05. The population size was, again, kept at 50 individuals, similarly to the previous SSGA implementations.

As can be observed in a single evolutionary run of 100,000 generations **Figure 3.14**, the mortality rate implementation seems to exhibit more diversity while remaining at the top of the landscape. In contrast, many individuals in AFPO reside in less-fit (lower) areas of the fitness landscape space, likely due to too frequent insertions of random individuals at each

generation. The evolutionary progression of the SSGA implementing mortality should convey the informal principle of hill hugging that was introduced in the previous sections since, especially compared to AFPO, the steady state implementation does not seem to occupy low fit regions of the fitness landscape. Two hundred of these evolutionary runs were done to compare the efficiency of both algorithms; the measure of efficiency was correlated to the number of times a global maximum has been found after running 100,000 generations (Figure 3.15). Out of the 200 runs, only one run of AFPO found the global maximum, whereas this number was 190 out of 200 runs in the SSGA with a mortality rate. Age-Fitness Pareto Optimization performs better than steady state algorithms when no mortality is implemented, but for this landscape, the mortality rate implementation was superior.

Other Conventional Implementations

There is a plethora of different algorithms that have been developed over the years that could be compared with the mortality rate. Simply looking at the difference between five strategies; an SSGA, a generational genetic algorithm implementing 5% elitism, AFPO, speciation, and extinction events, clearly shows how different strategies traverse the fitness landscape, as can be seen in the 2D diversity plots of Figure 3.16. However, elitism in generational genetic algorithms is known to increase the selective pressure on the best individuals of the population and thus leads to less diversity over time. Without elitism, the mutation rate needs to be quite low in order to ensure that the population does not lose the best solutions in the next generation. This is especially the case when the selection operator is not very strict.

To test whether generational genetic algorithms could exhibit the same performance as the SSGA implementing mortality, two generational algorithms were implemented, differing only in their selection operator. One used tournament selection with a tournament size of three, and the other used a roulette sampling method. With a sufficiently low mutation rate, it can be seen that the generational genetic algorithm can perform just as well as the SSGA that implemented mortality (Figure 3.17). The mutation rate, due to the higher selection pressure on the best individuals, needed to be 0.25% in the roulette wheel and 2.5% in the tournament approach and was 10% in the steady state approach with a 5% mortality rate.

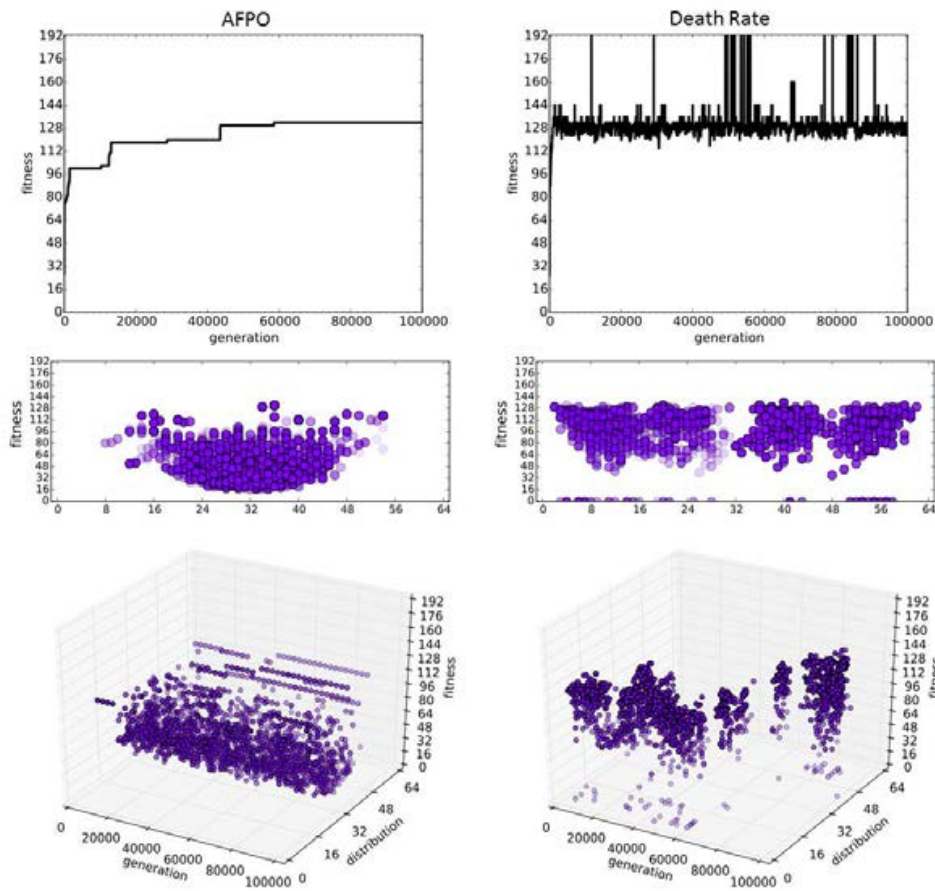


FIGURE 3.14: **Single evolutionary runs comparing AFPO and H-IFF.** The graphs display the maximum fitness (top), distribution of the population in 2d (middle), and the distribution of the population in 3d (bottom) of a single run. The blue dots represent individuals on the population plotted at intervals.

3.3 Changing the Domain

Single-objective continuous functions can be used as an alternative to the binary functions of H-IFF. The efficiency of evolutionary algorithms can then be evaluated as to how well an implementation is able to evolve the correct parameter sets. There are many different types of objective functions that are used as benchmarks for evolutionary algorithms, but of specific interest to this section are deceptive fitness landscapes where there is no apparent gradient towards the global optima. The convoluted landscapes with gradients (such as Rastrigin or Griewank functions) are comparatively

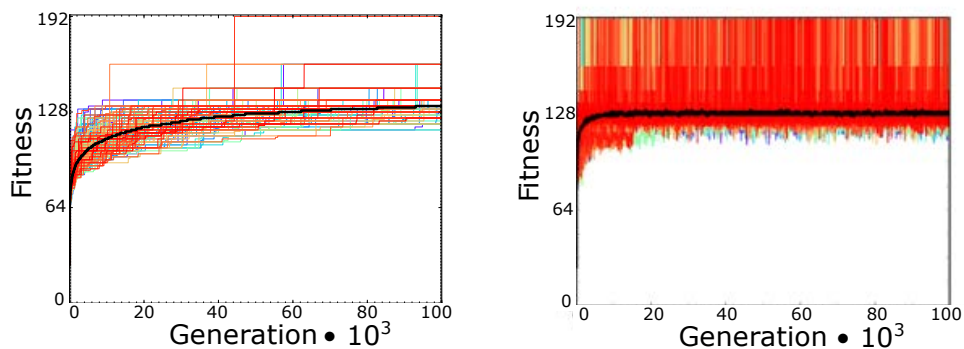


FIGURE 3.15: 200 evolutionary runs are shown for both AFPO and mortality rate. The solid black line represents the average value of the maximum fitness of each evolutionary run. Though it can be seen that the average maximum fitness in AFPO is ever-increasing, the average maximum fitness using the mortality rate stays around the same value. The graph's color is reddish due to the order of the curves of the evolutionary runs being plotted in the graph.

easy to solve, even with regular SSGAs, and thus not discussed. Since a continuous function is used, more advanced strategies like Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES; Hansen et al. 2001) can be implemented and compared with our implementations and can convey how the different algorithms work.

The alterable parameters of the continuous functions implemented in evolutionary algorithms are the mutation rate and the mutation spread. This spread is usually controlled by a Gaussian mutation operator, as in the case of the following experiments. The standard deviation of the Gaussian mutation operator is usually denoted by a σ value to indicate the spread. The change of the value of a gene becomes larger when σ is increased. In the SSGA with mortality, the σ value is thus additionally correlated with both the mutation rate and the mortality rate.

The Schwefel function (Figure 2.8b) was used to create the deceptive fitness landscape whose genomic values could range between -500 and 500. A σ value of 200 was therefore chosen as the mutation spread so that a gene having a value of -500 would not likely be mutated into a value of 500. Therefore, the landscape could not be directly crossed after one mutation step. The σ value of 200 means that a value that is being mutated has a 68.2% chance to be altered through the addition of a value between -200 and 200 (one standard deviation) and a 95.4% chance to be between -400

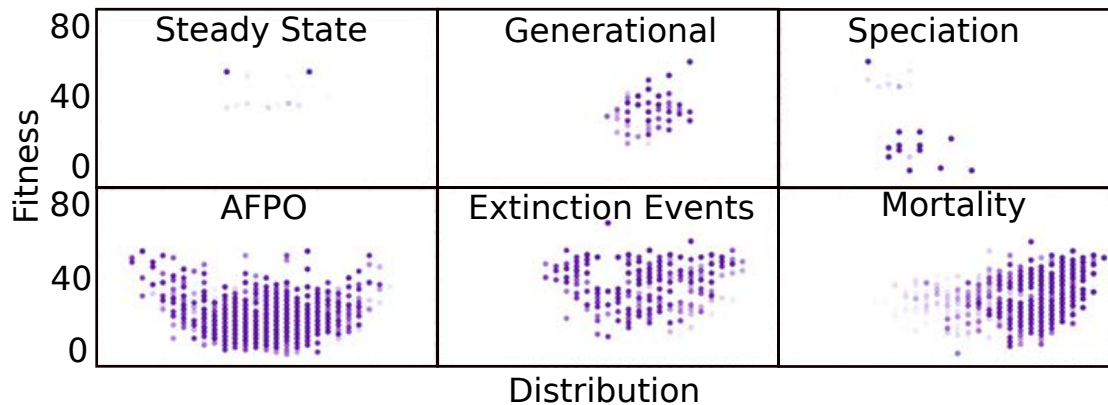


FIGURE 3.16: **Diversity on H-IFF using various genetic algorithms.** The blue dots represent individuals in the population over time for representative single runs of each strategy. Evolutionary runs contained a population size of 50, a mutation rate of 0.1, and were simulated for 1000 generations on 16-bit H-IFF. An SSGA (top left), a generational GA with 5% elitism (top middle), speciation implemented in an SSGA (top right), Age-Fitness Pareto Optimization (bottom left), indiscriminate extinction events (bottom middle), and a mortality rate (bottom right).

and 400 (two standard deviations). In CMA-ES, the mutations are directed and changed based on the spread of the population over generational time (Hansen et al., 2001). Hence, in comparison to CMA-ES, no specific mutation rate or *sigma* value was implemented.

For the experiment, a genome of size 8, containing 8 floating-point numbers, was used. A mutation rate sweep was done for AFPO to see what the best mutation rate to implement was. A mutation rate of approximately 20% yielded the best results. Using this mutation rate, the optimal mortality rate was determined for the SSGA. The optimal mortality rate was coincidentally also around 20%. These rates were however determined for quite a short run and could be subjected to further tuning, though they nicely illustrate how the genomes in the population change over time. The optimal mutation rates for AFPO and the optimal mortality rate for the SSGA were based on four evolutionary runs using different rates and simulating a population size of 50 for 2000 generations. Afterwards, 12 evolutionary runs were conducted with these optimal mutation rates, again for 2000 generations. Covariance Matrix Adaptation Evolution Strategy implemented a population size of 1000 for 100 generations. This was

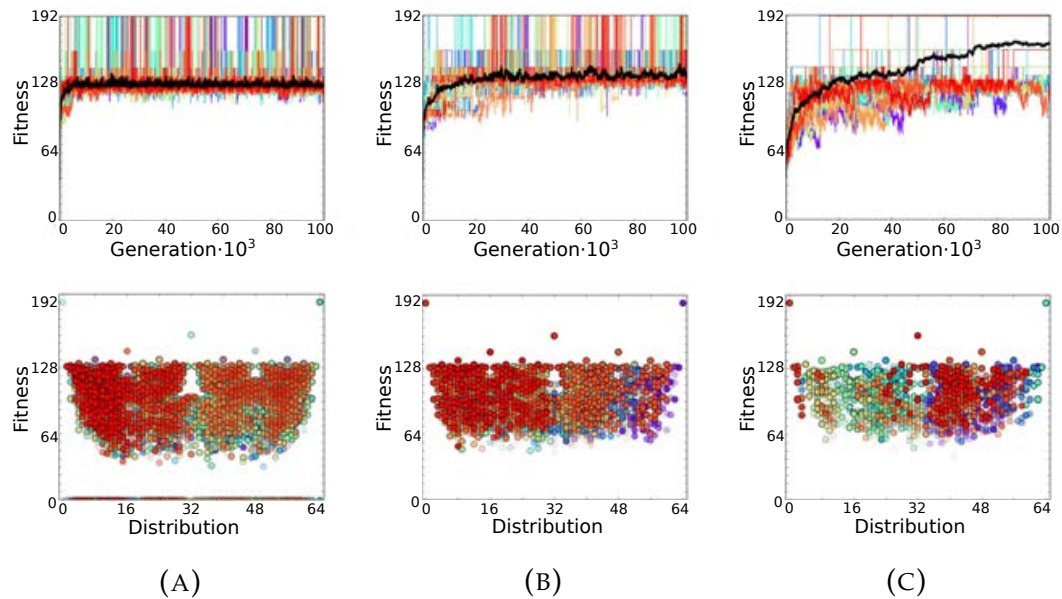


FIGURE 3.17: **Evolutionary progression on 64-bit H-IFF of generational Genetic algorithms vs. the SSGA implementing mortality.** The graphs show the evolutionary progression of 20 evolutionary runs for 100,000 generations on 64-bit H-IFF (top) and the genetic diversity of all individuals in all runs (bottom). Results are shown for the SSGA implementing mortality (A), the generational genetic algorithm implementing tournament selection (B), and the generational genetic algorithm using roulette sampling (C). The optimal mutation and mortality rates were used for each implementation.

done because larger populations are more efficient in CMA-ES, as smaller populations converge very quickly to a local optimum. However, the number of evaluations needed (100,000) stayed the same in all approaches.

Figure 3.18 depicts the difference in the evolutionary progressions of the different runs (top) and shows the change in the values of individual genes over time for a single representative run (bottom). As can be noted, within the 2000 generations, the CMA-ES strategy was the only one to find the global optimum in 2 out of the 12 runs. AFPO and the SSGA did not find the global optima. Longer simulations and larger population sizes would most likely lead to the global optima being found eventually. What is interesting to see in these plots is the genetic change of each implementation over time. The SSGA implementing mortality is shown to have varied the values of its genes over time, where a gene sometimes

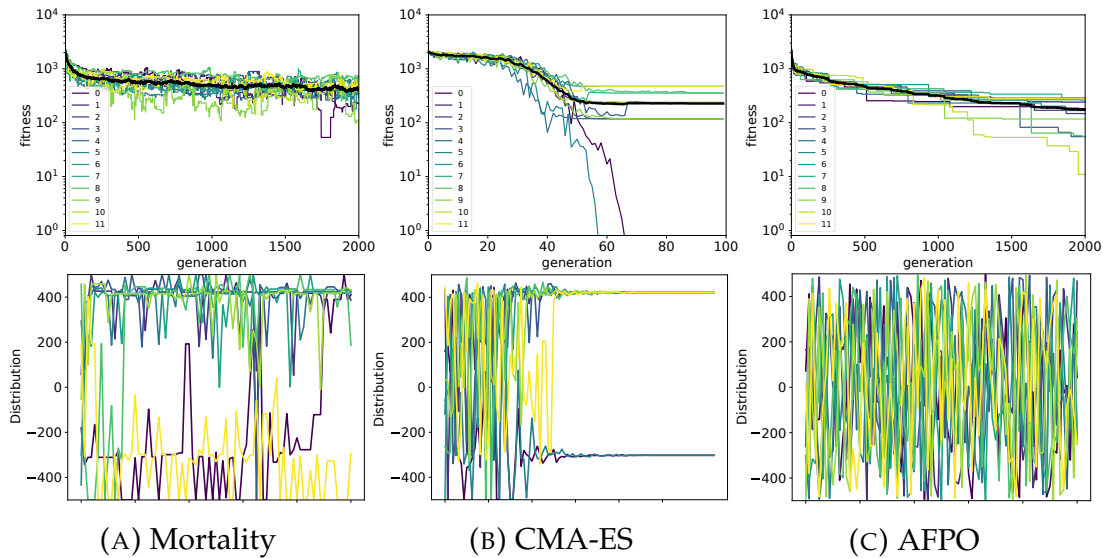


FIGURE 3.18: **Comparison of evolutionary algorithms on the Schwefel function.** The top graphs show the performance of 12 runs on the Schwefel function using different evolutionary algorithms. The bottom graphs show the change in the values of the 8 genes in the genome over time.

changed its value from one side of the landscape (close to -300) to the other side (close to 400). The SSGA implementing mortality thus retained genetic variation. AFPO kept much more genetic variation, on average, in the population due to the insertion of random individuals. The graph therefore illustrates that average values of each of the genes in the population drastically changes with time. Covariance Matrix Adaptation Evolution Strategy began with much genetic variation due to the large population size and converged to specific values quickly, after which time the genes no longer changed. Two runs of CMA-ES found the global maximum after approximately 60 generations. These effects on diversity are typical for each optimization type shaping the evolutionary progression of each type of genetic algorithm.

3.4 Development and Evolvability

Though mortality and mutation rate change the overall evolvability of an evolutionary algorithm, these are most likely not the only mechanism that increase evolvability. As mentioned in [Section 2.3](#), organisms contain

inherent developmental programs that enable the patterned expression and recursion of cells, tissues, and organs. This relates to how the genotype is mapped onto the phenotype. This section therefore focuses on the impact of development on the evolvability of the spatial model and discusses how this is similar to the addition of a mortality rate. In addition, an interesting emerging dynamic on the age of the individuals in the system was noted and included in the section. However, it should be taken into account that these results are also preliminary.

Hinton et al. (1987) have demonstrated how developmental plasticity could smooth the search space evolution operates in, but they considered only an abstract control system and a one-to-one genotype to phenotype mapping. In *needle in a haystack* problems, random search has been proven to be the best search strategy (Hinton et al., 1987). Random search could be seen as having a 100% mutation rate. It is therefore not viable in nature since it would lead to an error catastrophe or to spending valuable resources on sub-optimal individuals. A more gradual stepping stone mechanism which includes intermediate sub-optimal fitness values, would allow for individuals to more effectively traverse the search space by keeping the population close to the top of the search space, as has been shown by implementing mortality. As an alternative to increasing this traversability, or evolvability, of a population, an individual could exhibit a small genotypic sweep by evaluating multiple genes expressed at different time intervals. This is what I consider to be a minimal developmental representation in this section. The question posed here is: *can this minimal development model lead to an increase in evolvability?*

Methodology

A fixed patterned change of the genome, through the developmental processes of an individual, could increase the search of various phenotypic expressions in an individual's lifetime. To investigate how during the lifetime of an individual, one can explore multiple phenotypic traits, a spatial model was implemented again using H-IFF. The experimental setup consisted of a predator prey like spatial model where a binary 16 bit genotype was translated into the phenotype of a cell as a metabolic efficiency as has been implemented in [Subsection 3.2.2](#). Note that in previous experiments a 32-bit genotype was used. The only way for individuals to be removed in this experimental setup was through starvation.

Two experiments using the developmental stages were performed. The first experiment evaluated the efficiency of finding the global maximum

similar to the spatial experiment with the mortality rate (Subsection 3.2.2). Instead of varying the mortality rate, the number of developmental phases were altered. A timed developmental phase changed the expressed genome of one cell to another. The number of developmental phases, as well as the timing of these phases, were fixed, and the only mutable parameter in this instance was the mutation rate. The timing of moving from one developmental stage to another was set to 100 iterations. One hundred iterations were chosen since a cell, on average, lived around 500 iterations. An age of 500 iterations is thus enough for an individual to reside in all (up to five) developmental phases during its lifetime. To investigate how the population of cells in the spatial model is able to traverse the H-IFF landscape, 50 simulations ran for 200,000 iterations (2000 cycles) with differing parameters of mutation rate and number of developmental phases.

Results

Since genomes with different developmental phases can evaluate multiple genomes in one lifetime, it was expected that the developmental stages would be directly correlated to the evolvability of the population. More genomes mean more exploration; however, this might also mean a shorter lifespan since genes that are deleterious later in life will limit the lifespan. An elite that goes from an efficient developmental stage to a suboptimal one would be quickly outcompeted by fitter cells, and thus the relationship between the mutation rate and the number of developmental stages would likely be skewed. In this setup, evaluating more than one set of genomes could be considered equal to evaluating multiple individuals in conventional evolutionary algorithms, though the speed of acquisition of the global maximum in the spatial model might translate to evolutionary algorithms as well. The aim is to determine how these developmental phases can aid, or be a detriment to evolving spatial models.

Conventionally, in evolutionary robotics, the computational time for evaluating an individual mostly depends on the physics simulator since this requires the most computational power. The evolutionary algorithm itself usually does not require much computational power, and measuring the efficiency solely based on the number of individuals that have been simulated is therefore a good measure when considering the efficiency of the evolutionary algorithm. To evaluate whether development is computationally more efficient, it is thus considered how many individuals have been simulated until the global maximum was found. As depicted in Figure 3.19, a clear relationship between the mutation rate, the number of

developmental stages, and the times the global maximum has been found can be seen. Though this relationship exists, it is not known how useful this is for evolutionary computation.

In a second preliminary experiment, the timing of the developmental stages itself was subject to evolution, where each developmental stage could span between 10 and 1000 iterations. In this case, the global optima can be found in different stages of development of the populations, and this has an effect on the evolutionary progressions of the developing cells. Namely, if a population of cells finds the global optimum in the initial stage, it can be seen that the time the individuals spend in this stage is maximized (Figure 3.20; right). In contrast, there is no selection pressure to maximize this phase when the global maximum has not been found.

When comparing the different developmental times and mutation rates, a clear relationship between the mutation rate and the number of developmental phases can be seen, as shown in Figure 3.19. High mutation rates are efficient when the cells do not have developmental phases. However, with lower mutation rates, no developmental phase is very inefficient for finding the global optima compared to developing populations. When using mutation rates lower than 0.12, none of the non-developing populations found the global maximum, while the developing populations were still able to do so. Other than that, a 100% mutation rate seems to work well in the spatial model. Producing many random offspring is therefore still efficient on 16-bit H-IFF. This would, however, not have a high probability of working using larger genomes, as can be inferred from Table 3.2 where mutation rates larger than 0.32 do not yield optimal solutions.

The developing populations of these experiments can be altered by a timing mechanism as developmental change; single bits change in timed expression in the genomes of the individual cells. This development increases the breadth of the search, and in turn enhances the evolvability of a population. In Figure 3.20 (right), the period during which the maximum possible fitness is found alters the average age of the population. This limit in age can be regarded as being the result of different bits changing that in turn cause a deleterious effect. This change in bits causing a detrimental effect could be an antagonistic pleiotropic effect since it lowers the age of individuals, supporting the antagonistic pleiotropy theory. It could also pose a dual benefit, where development both increases the breadth of the search and limits the lifespan of individuals, further increasing the evolvability of a system. Though no quantifiable data is shown, it is an idea that might be worth investigating in the future. The negative effect of pleiotropic genes on the personal fitness might thus actually increase the

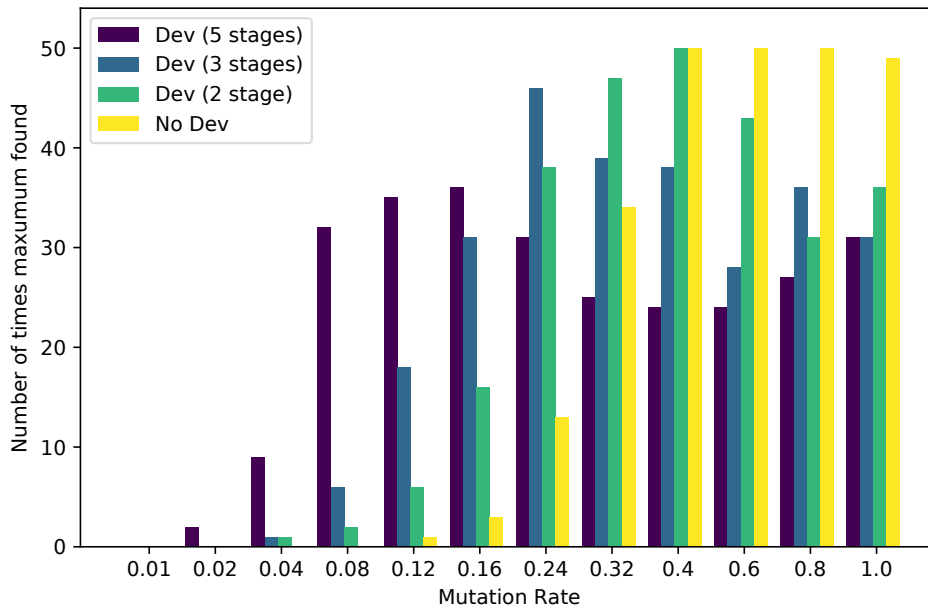


FIGURE 3.19: **Correlating developmental stages with finding the global maximum.** This figure displays the results, for the four different experiments, using 1 developmental stage (no development), 2 developmental stages, 3 developmental stages and 5 developmental stages. Fifty evolutionary runs were conducted for each parameter set and ran for 2000 cycles (200,000 iterations). Decreasing the mutation rate requires the additional developmental stages in order to find the global maxima on H-IFF.

inclusive fitness.

3.5 Discussion

As shown, evolvability is greatly influenced by the mutation and mortality rate ratio in both evolutionary algorithms and a spatial model. In particular, the H-IFF function, despite its deceptiveness, can be traversed by an SSGA through simply including an indiscriminate mortality rate. Since a fitness landscape in nature is likely highly convoluted and possibly deceptive, we speculate that programmed aging could be, as Goldsmith (2014) has mentioned, beneficial for the evolvability of a population. The better a

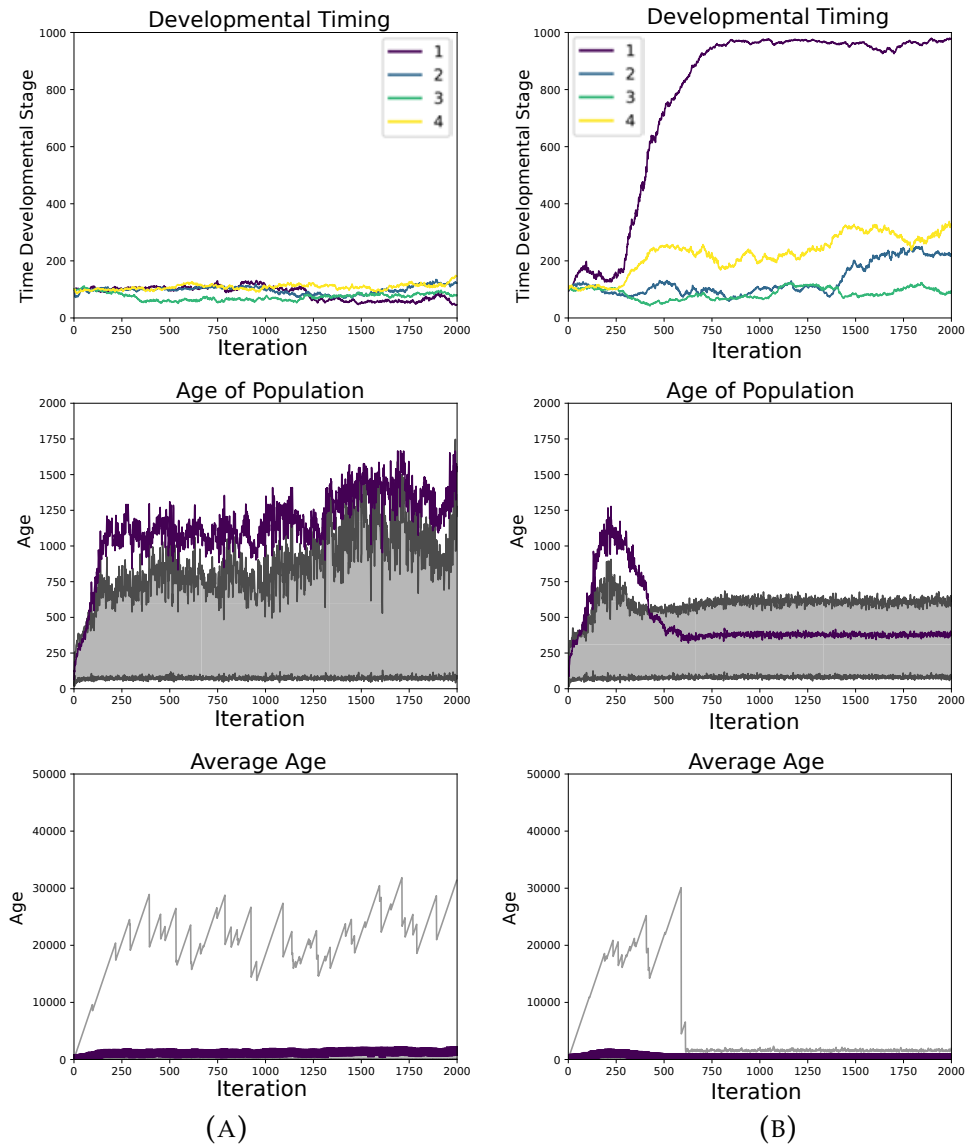


FIGURE 3.20: **Single evolutionary runs in the developing populations.** The left and right side of the figure represent developmental timings where no optimal strategy has been found (A) and where the optimal strategy has been found in the first developmental stage (B). The specific timing of the evolutionary developmental stages is displayed (top) as well as the average age (blue) and 25th - 75th percentiles (grey area) of the populations (middle). The age of the oldest individual in the population is depicted in the bottom two graphs.

species can traverse the fitness landscape without going through low fitness regions, the more plausible it is that the population finds more adaptive traits. This, in turn, makes the population better and potentially increases its ability to cope with changing environments.

In the presented experiments, the shortest path to the global maximum of H-IFF from the center of H-IFF was to mutate individuals in the appropriate lesser-fit local maxima. The results indicate that there is an optimal mortality rate for a given mutation rate, which could be considered to improve existing evolutionary algorithms, especially the steady state variants. Steady state genetic algorithms with no chance of removing elite individuals are prone to premature convergence, as demonstrated by the few times the global maximum has been found when no mortality rate was implemented. Nevertheless, the addition of indiscriminate mortality enhances such algorithms and allow them to efficiently traverse the state space landscape. It is further shown that this effect is likely transferable to other domains and landscapes with preliminary results.

AFPO relies on a new random individual being inserted in a specific region close the maximum fitness of the landscape by accident. From this starting point, the new pareto efficient individuals in this age category need to quickly find a good solution before they are outcompeted by a younger strain climbing the local hill. A potential improvement to AFPO is thus also to insert a new random individual at intervals, not every generation, so the individuals in the new age category have time to climb a local hill, preventing them from being outcompeted by chance. Despite the occasional loss of the best individual in a population, the entire population of individuals remains close to the top of the fitness landscape.

The correlation between the mutation rate and mortality rate indisputably works on SSGAs and spatial models for solving deceptive fitness landscapes. Even the most advanced optimization strategies have difficulties with solving these types of landscapes. Aside from that, there is something to be said for the ease of implementing a simple mortality rate into a genetic algorithm—simply remove individuals from the population at random to increase the performance.

3.5.1 Theoretical Implications

The mutation rate and mortality rate have an optimal ratio that depends on multiple factors. However, factors such as reproduction speed, development, population size, selection pressure, and crossover likely influence the optimal parameter set as well. However, a few essential factors that always shape evolvability include the mortality rate,

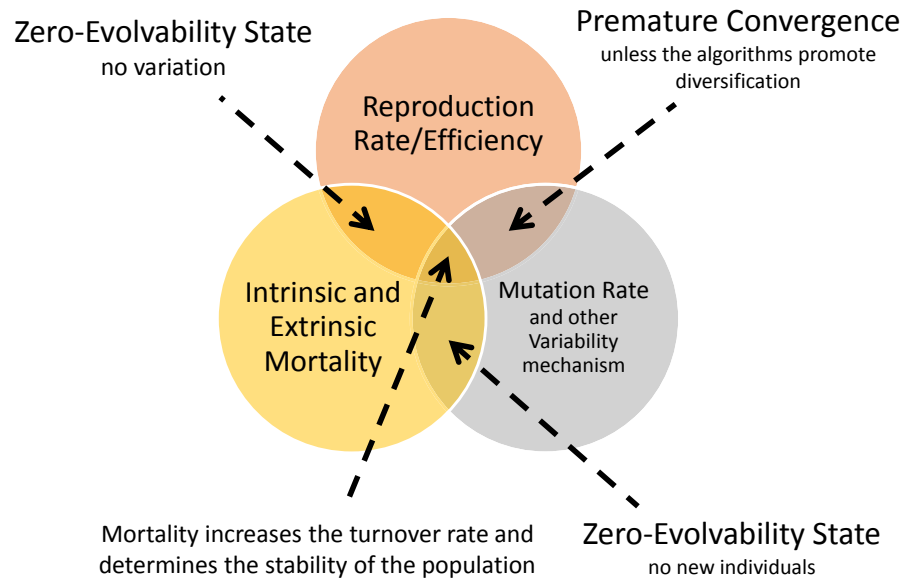


FIGURE 3.21: **Venn diagram depicting the relationship between reproductive rate, mortality, and mutation rate.** Reproduction, mortality and mutation rate shape evolvability. For a given sequence space, there exists an optimal relationship between these factors which is an overlap of the four elements. No mortality leads to premature convergence, and no reproduction and no mutation lead to a zero-evolvability state since no new individuals or variation is being introduced in the population.

reproductive rate, and mutation rate. This relationship is clarified in the Venn diagram shown in [Figure 3.21](#), where non-reproducing or non-mutating populations reside in zero-evolvability states. In this case, individuals are evaluated too long and genetic variation is low. I speculate that, through changing the parameters of, for example, mutation and mortality rate, one can map the deceptiveness of the landscape as shown in the plots that depict diversity and fitness over generational time ([Figure 3.5](#)). One could potentially change just one parameter without taking the other into consideration, as long as the other parameters are within certain bounds.

Species in natural environments suffer from both intrinsic (aging) and extrinsic (predation, accidents and parasitism) mortality. Extrinsic mortality is known to fluctuate, both in predictable ways (seasons) and depending on external factors (diseases, variable predator pressure). As there is a clear correlation between mortality and mutation rate for optimal

evolvability, this means that such fluctuations in mortality rates could have a negative or positive impact on the evolvability of populations. Evolving an intrinsic mortality factor may alleviate this problem: when external pressure is high, aging is not a dominant factor. If external mortality is decreased, then intrinsic mortality prevents the death rate to mutation rate equilibrium from being out of balance, preserving evolvability. Therefore, this would be an advantage of an intrinsic mortality rate.

Though mortality has been shown to be beneficial for the evolvability of a population, it is still difficult to see how mortality in itself would evolve. In a prisoner's dilemma scheme, one could define individuals that exhibit mortality to be cooperators, and immortals to be defectors. Since an immortal can produce more offspring than a mortal can during its lifetime, its personal fitness would be higher. However, if all individuals in the population are immortal (defectors), the overall benefit to the population is less than when all individuals are mortals (cooperators). Hence, the evolution of mortality could have a basin of attraction toward immortal individuals, evolving older individuals over generational time. However, if a population with mortals were more efficient, it would be more likely to eventually outcompete an immortal population.

Another interesting detail may explain the evolvability of mortality. Consider a population residing in a stable limit cycle. The number of individuals in this population fluctuates depending on how much prey is available to them, and how greatly predators decrease their population. A periodic fluctuation would mean that the extrinsic mortality rate also fluctuates. In a period when many individuals of the population are eaten, the population can exhibit more genetic drift. Once less individuals of the population are eaten again, this genetic drift will also decrease when we accept that the mutation rate does not change. This change in evolvability of a population could, however, be counteracted by continuously changing the mutation rate. Lower mutations rates when many prey are eaten would keep the evolvability the same. Similarly, lower predation rates would require higher mutation rates. This process of altering the mutation rate is likely cumbersome to evolutionary adaption. However, as mortality has been shown to change the evolvability as well, this mortality rate could instead have been evolved to provide a steady evolvability rate in the presence and absence of high predation rates. This would keep the evolvability of a population the same, and therefore the mutation rate would not need to be altered. A possible benefit to senescence could be to keep the evolvability of a population steady under conditions of fluctuating extrinsic mortality rates. As discussed by Herrera et al. (2016), intrinsic mortality seems to be beneficial in changing environments, and therefore

a steady optimal evolvability rate might be required. Intrinsic mortality could therefore be a mechanism that ensures a baseline evolvability rate in a dynamic, changing environment.

There is a major argument to be made as to why mortality is likely programmed by combining a changing environment, as in Mitteldorf et al. (2014) and Herrera et al. (2016), and the experiments of H-IFF shown here. Consider, for example, a H-IFF type function, and a changing environment where the global maximum switches from one side of the H-IFF landscape to the other side. How could one possibly hope to make a population traverse this landscape efficiently? Either the mutation can be increased to its maximum, or mortality could enable a drift across the top of the landscape. I claim that mortality is a more plausible explanation in natural systems since it gradually changes the genetic information in the population instead of producing drastic changes in the genome. A high mutation rate would produce many individuals that are non-functional, these individuals might need to be brought into the world, which can have a massive reproductive cost. Moreover, can such a potentially high mutation rate really be achieved locally in germ cells, or would high mutation rate mechanisms mean that the individual cells are also subject to high mutation rates? This could result in many detrimental effects in the individual and could lead to error catastrophes in species.

3.5.2 Hypothetical Explanation for the Benefit of Mortality

The issue of mortality also becomes more interesting when considering multiple niches. Consider a hypothetical single population of finches on an isolated island with different beaks. The population hasn't yet resulted in any speciation. Additionally, assume there are three types of food sources we simply call food source a, b, and c. Say that these food sources require appropriate beak shapes to optimize the consumption; these are called beak A, B, and C, respectively. Clearly, depending on the steady food supply, the beak frequency of beak types would roughly correspond to the availability of a food type. However, when food source a and b are low, beak C will begin to dominate. Say that this domination is so vast that all beaks A and B have been out-competed after 100 generations—a phenomenon in artificial life that can be termed *catastrophic forgetting*. The long-term effects of beak C dominating have led to a decrease in food source c and a subsequent increase in food source a and b. What is now the quickest pathway to rediscover beaks A and B? If the path to beak A and B is slightly deceptive, and the mutation rate is unalterable, a population of mortals will simply allow for better adaptive radiation of the population to rediscover

either beak A or beak B than a population of immortals. This hypothetical dynamic shift of niches therefore promotes a population of mortals and is arguably a conceivable logical explanation supporting the evolutionary advantage of intrinsic mortality.

3.5.3 Application to Evolutionary Robotics

Undefined domains with potentially deceptive landscapes such as robotics simulators may also have an optimal ratio between mutation rate and mortality rate. It is likely that this optimal ratio changes in different parts of the landscape. An additional feature of the mortality to mutation rate ratio is that it determines the broadness of the hills in the landscape that can be robustly occupied across generational time. This is derived from accepting the relationships of mortality and mutation rate to stable sequence spaces, as presented in [Figure 2.1](#). A higher mutation rate or mortality rate would then change the stable region the simulated population is occupying, and a broader stable region might result in more abstract, generic phenotypes that could better transfer to real robots; an unstable region might be a narrow peak containing a phenotype that isn't transferable. The field of evolutionary robotics is thus a promising field for future applications. It would be especially interesting to see whether an optimal ratio of mortality and mutation rate can surpass existing algorithms like Age Fitness Pareto Optimization (Schmidt et al., 2011), Age Layered Population Structures (Hornby, 2006), and novelty search (Lehman, 2012) that have been frequently applied to such simulators.

3.6 Conclusion

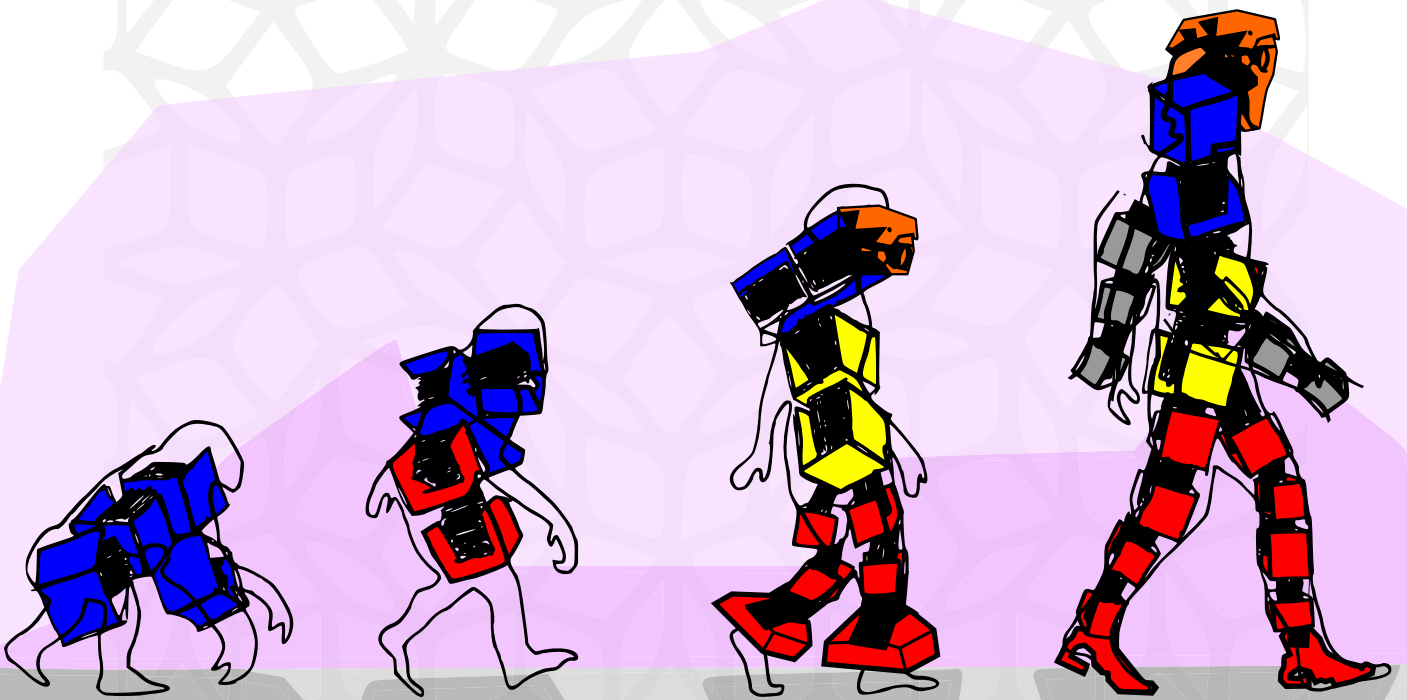
An explicit relationship between the mutation rate and mortality rate for optimal evolvability on a deceptive fitness landscape in both spatial and non-spatial evolutionary models has been presented. As an alternative to proposed theories on aging showing how intrinsic mortality is advantageous for altruistic aging, we claim that intrinsic mortality governs evolvability and that it is thereby a potentially evolvable trait, ultimately supporting the theories on programmed death. Moreover, in scenarios of fluctuating extrinsic mortality rates, an intrinsic mortality rate would keep the evolvability the same, which may further support why intrinsic mortality has an evolutionary benefit. Also, considering potentially unrealistically high mutation rates to otherwise grant the same level of evolvability, these high mutation rates in nature would likely lead

to an error catastrophe, leading to the extinction of the species altogether. Senescence might therefore be a better explanation for evolvability. The results not only increase our understanding of senescence, but hold potential benefit in applications to evolutionary algorithms and robotics. As it has been shown that the mortality to mutation rate ratio influences evolvability on deceptive landscapes, one question that can now be posed to bolster the significance of this chapter is:

Do natural systems contain deceptive dimensions that are traversable through mortality-induced evolvability?

Part II

Evolving Modular Robots



Chapter 4

Evolutionary Robotics Platform

4

Everyone knows that a machine tool is more complicated than the elements which can be made with it, and that, generally speaking, an automaton A, which can make an automaton B, must contain a complete description of B, and also rules on how to behave while effecting the synthesis. So, one gets a very strong impression that complication, or productive potentiality in an organization, is degenerative, that an organization which synthesizes something is necessarily more complicated, of a higher order, than the organization it synthesizes. This conclusion, arrived at by considering artificial automaton, is clearly opposite to our early conclusion, arrived at by considering living organisms

– John von Neumann, Theory of Self Reproducing Automata

In **Part I** the importance of evolvability and the genotype to phenotype map were discussed with respect to evolutionary dynamics. The part conferred how evolutionary computation can be implemented to evaluate theoretical hypothesis of biology and how its performance can be altered. In contrast to the theoretical undertone of the previous part, this part of the thesis illustrates scientific questions that arise from an engineering perspective; i.e., it expresses the application of the types of evolutionary computation on the creation and control of modular robots.

Three of the challenges of evolutionary robotics—exploration vs. exploitation, genotype to phenotype mapping, and the reality gap—are addressed in this part of the thesis. In particular, modular robots are being used since the reconfiguration of robotic modules allows a change in both the morphology and control systems of a robot. Non-modular approaches can also allow for morphological change, though the use of modular robot parts additionally enables the evolved robots to be easily transferred to the real world, which expands the potential real-world

solutions that can be evolved. Another advantage of using a modular approach is that incremental improvements or the exchanges of modules can alter the functionality of the modular robot through simply replacing or adding modules to the system.

To gain more insight into how to evolve the behavior and control of a modular robot, a large portion of this chapter is devoted to explaining the evolutionary robotics plugin developed for the robotics platform: Virtual Robot Experimentation Platform (V-REP; Rohmer et al. 2013). This plugin covers much of the evolutionary algorithm and its parameters, the inclusion of an encoder able to construct modular robots from a genome, and the implementation of various controllers. The modules, environment, and objective used can be changed in the plugin, allowing for the emergence of a variety of modular robots. The plugin itself is not necessarily unique since others have developed similar software platforms to evolve modular robots, as can be seen in Faíña et al. (2013) and Auerbach et al. (2014). The plugin was developed due to its ease of use, potential integration with other projects, and having previous experience with it (Veenstra et al., 2015). Though the plugin in its current form is unable to evolve real-world modular robots directly, due to the lack of a standardized performance feedback from a physical robot, the simulator can directly control a robot in the real world and in the simulation simultaneously. The robots that were transferred to reality in Chapter 7 were simply evolved in the simulation environment. Part III of this thesis considers online evolution directly applied to physical robots without using the plugin.

Modular robots are important since they can eventually give us insight into how to construct various (reconfigurable) robotic morphologies. This both eases the production of robots and allows for a feedback loop to evaluate various developmental models and their relevance to robotics. Chapter 5 and Chapter 6 implement a generative encoding and discuss it concerning developing a robot morphology. Chapter 6 specifically highlights the importance of genotype to phenotype mapping. Chapter 6 describes the evolution of modular robots evolved for locomotion, while Chapter 7 utilizes the same modules with the addition of solar panel modules, to evolve modular robots with energy autonomy.

The control of the modular robots described in Chapter 6 and Chapter 7 is based on neuroevolution. Each module could contain a neural network that was evolvable. However, experiments in the next three chapters did not implement neural networks to their full potential, due to the necessity of keeping the control as simple as possible. Nevertheless, since it has been implemented and shown to be useful, this chapter also gives a brief overview of contemporary neuroevolution strategies (Section 4.1) and the

manner in which neural networks were encoded for the evolutionary robotics plugin (Section 4.1). Hence, a brief overview of the control strategies that were implemented in the simulator (Subsection 4.2.3) is presented. Afterwards, a high-level overview of the various features of the plugin is given in Section 4.2. In Section 4.4, the real modules that were developed are presented.

4.1 Neuroevolution

Neuroevolution, or the implementation of evolving neural networks (ENNs), is a bio-inspired optimization strategy that is usually implemented to investigate the acquisition of behavior in artificial creatures (Sims, 1994b) and robots (Nolfi et al., 1994). A key question in neuroevolution is which type of neural network to use as the basis for evolution. This section therefore gives a brief overview of some prominent neuroevolution strategies.

Evolving neural networks can be optimized without the strict forms of learning algorithms such as backpropagation that are usually implemented in neural networks. Many different types of neural networks with various learning and evolutionary strategies have been developed over recent decades with auspicious results. In Stanley et al. (2002b), the technique Neuroevolution of Augmenting Topologies (NEAT) was introduced and subsequently implemented in many instances as an effective strategy to generate neural networks to control robots. Neuroevolution of Augmenting Topologies is an efficient implementation of an ENN since it begins with a population whose networks contain basic connections from the input neurons to the output neurons; it subsequently augments the network to become more complex through artificial evolution. The mutation operators facilitate the addition or deletion of neurons (nodes) and the connections (edges) between the neurons. The mutation operators additionally alter the activation function of the nodes and adjust the weights of the edges. One more feature that has been implemented to protect innovation is speciation, also known as niching (Stanley et al., 2002b). Speciation is made possible in the population of neural networks through implementing historical markings that track the historical origin of the evolved genes.

A prominent extension of NEAT is Compositional Pattern Producing Networks (CPPNs; Stanley 2007). In this case, a collection of various types of functions is employed to produce patterned outputs. This approach is not only used for control, but also for outputting a spatial pattern for the morphology of robots (Auerbach et al., 2011; Cheney et al., 2013). It has also

been used to create static 2D images (Secretan et al., 2011) and 3D objects (Clune et al., 2011). Another extension is the use of CPPNs to create the neural network itself, an approach called HyperNEAT (Stanley et al., 2009). This approach has been further extended by Risi et al. (2010) to evolve the substrates of the neural network; i.e., specific regions of the network could change the placement and densities of the neurons in that area.

The utilization of CPPNs as an abstraction of development is valuable as a generative encoding. Multiple neurons can be created from a range of input values. Other classical approaches of neural networks are usually produced by direct encodings, as in the original NEAT version and other approaches (Angeline et al., 1994; Yao et al., 1997). Moreover, within the evolutionary loop, other machine-learning strategies, like backpropagation, can be implemented in the neural network combining evolutionary algorithms with other learning strategies. For example, (Belew et al., 1992) have demonstrated that evolving the initial weights enables back propagation to find better solutions compared to without evolving the weights. An algorithm implementing both backpropagation and neuroevolution would thus be similar to the approach of Hinton et al. (1987), where development during an individual's lifetime could potentially increase the overall fitness. It should therefore be taken into consideration that a learning strategy could always be additionally implemented after a network has been evolved.

One more feature that is frequently implemented in neural networks is central pattern generators (CPGs). CPGs mimic neural circuitry by generating continuous activation loops within neurons. In most animals, rhythmic activation of motor neurons enabling locomotion is regulated by some sort of CPG (Bear et al., 2016). The activity patterns generated by CPGs are important for locomotive gaits since they provide rhythmic actuation of muscles (Still et al., 2006). As CPGs do not require any sensory information, they may be of particular importance in evolving robots. Therefore, patterned inputs to the simple neural networks used in Chapters 6 and 7 implement a patterned input to the neural networks of the modules by simple sinusoidal functions.

Neural networks are usually implemented as centralized control strategies in robots; one controller processes all the inputs and generates all of the outputs across the entire robot. A neural network creates outputs based on the network topology, internal activation, and the sensory input. In the case of CPPNs, the network has also been used to create the robots themselves, where control can be localized, producing a type of decentralized control. Auerbach et al. (2011) and Cheney et al. (2013) have shown how local body parts created by the CPPN also encoded for the

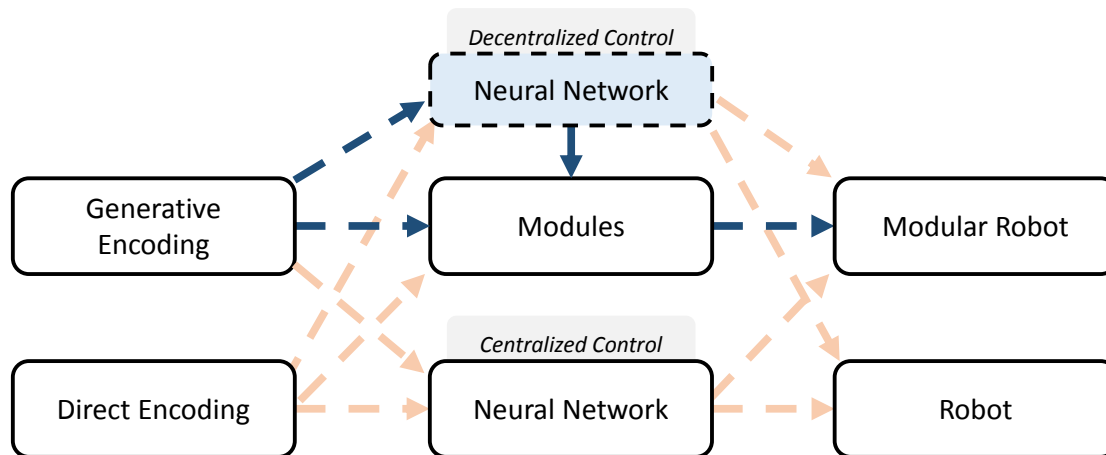


FIGURE 4.1: **Approaches to controlling a modular robot.** Illustration on how the generative and direct encodings are used to ultimately create robots. The dark blue lines represent the approach taken in this thesis.

control system in the robot. In modular robots, this type of decentralized approach is already prevalent where individual modules can be assigned “roles” and act accordingly (Støy et al., 2002). The ability of the change in these roles further increases the adaptiveness of the robotic system (Pfeifer et al., 2006).

As is described in Chapters 5, 6, and 7, a Lindenmayer system (L-system; Lindenmayer 1968b) has been implemented as the generative encoding to create robot morphologies, an approach similar to Hornby et al. (2003). The L-system was initially chosen as a plant-inspired approach to creating robots. Additionally, each of the different modules was controlled by a neural network where inputs and outputs could be distributed from one module to the next. Each module exhibited its own role through its neural network, modularity again being effective for isolating the functionality in specific parts of the robot (Støy et al., 2010). The basic approach to controlling the modular robot is thus guided by a generative encoding and results in a decentralized control, as illustrated by the dark blue arrow in Figure 4.1. The other arrows in Figure 4.1 depict a few other potential pathways one can take to create a decentralized or centralized approach to controlling robots when implementing neural networks.

Though natural systems might seem to exhibit control in a centralized manner, many organs and tissues also contain local control elements. One of the most prominent examples is the nervous system of octopuses whose neurons are largely distributed throughout their entire body. Each of their

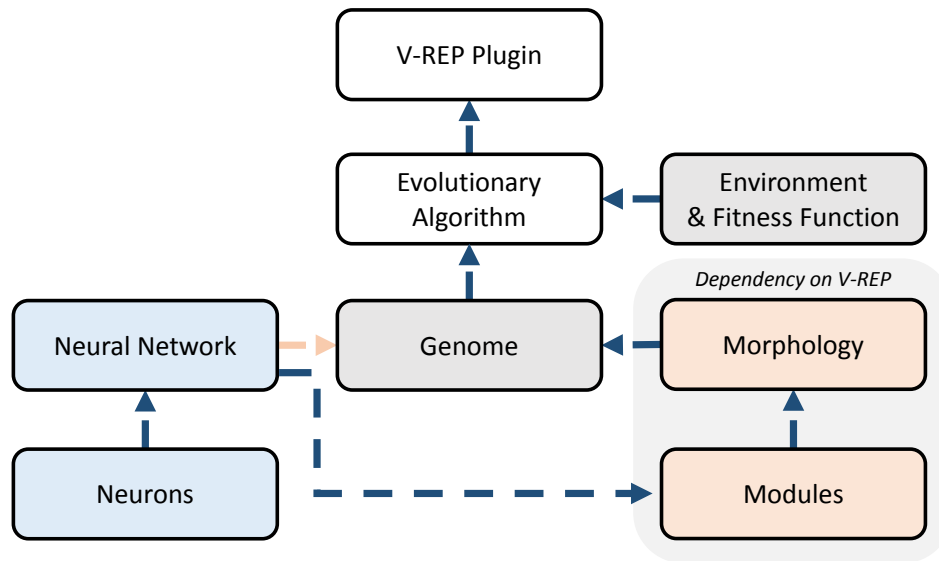


FIGURE 4.2: **Overview of the evolutionary robotics plugin.** The plugin starts an evolutionary algorithm which in turn incorporates an environment with a fitness function, and a population containing randomized genomes. A genome can, in turn, directly contain a neural network (dashed orange line), but in the modular approach, the neural network is actually included in the modules themselves. The morphology and modules are dependent on V-REP since they contain either specific functions to be called in V-REP or need to load V-REP-dependent model files. A full UML can be found in [Appendix B](#).

arms contains a staggering 3 million motor neurons (Levy et al., 2017). This can have several advantages for octopuses. For example, reflexes and reaction speed of arm movements are quicker when their control is localized. The octopus has therefore also been an interesting model in itself, as discussed by Laschi et al. (2012). The reflexes are similar to how they are controlled in, for example, the lower body of humans through a feedback loop from the spinal cord rather than the brain. In contrast to the usual centralized controllers, a modular approach can lead to decentralized control scheme where parts of the body are controlled locally—a decentralized embodiment of robotic control.

4.2 Overview of the Evolutionary Robotics Plugin

The robotics simulator used in this thesis was Virtual Robot Experimentation Platform (V-REP; Rohmer et al. 2013). This simulator was chosen since it is open-source and has many capabilities through which one can interface with it using various coding languages. To potentially maximize the performance, a C++ based plugin was developed and integrated with V-REP as a Dynamically Linked Library (DLL) plugin. The DLL plugin communicated with V-REP continuously: at start up, and, most importantly, at every simulation time-step. The DLL plugin was developed through using a C++ based template provided by V-REP that communicated with V-REP through functions that were called during the start, stop, and iteration of a simulation.

For the evolutionary algorithm, a loop was integrated automatically, starting and stopping the simulator based on a settings file that was modifiable. The main function of the DLL was to instantiate an evolutionary robotics class, which was linked to the evolutionary algorithm. This class encapsulated the environment, morphology, and control with three abstract factory patterns. The abstract morphology and control classes could moreover contain class instances of modules and neurons, respectively. The modules and neurons were also created using a factory pattern and were instantiated whenever the settings file contained information indicating that they should be used.

The total of five factory patterns created robots, with neural networks and modules, in specific environments, which depended on the initial settings. This approach enabled the quick replacement of a type of module with another type of module by simply specifying in the settings file which modules the factory pattern should instantiate. The aim of this approach was to keep the simulator as flexible and accessible as possible. In theory, another researcher could create and integrate another module to directly incorporate it and evolve this new module with the existing modules by either supplying a “.ttm” file (model file in V-REP) or a blueprint C++ class describing how the module should be constructed and controlled. However, though potentially a useful tool for other researchers, the main purpose of the simulator was to conduct scientific experiments, and therefore the functionality and the flexibility of the simulator in this state were obviously limited and contained messy code. Instead of a detailed description that consumes numerous pages, some important highlights are given to further explain its functionality. The basic functionality can

be located in [Figure 4.2](#), while a full UML description is provided in [Appendix B](#).

4.2.1 Evolutionary Algorithm

The evolutionary algorithm was implemented in two ways: (1) in the main loop of the DLL plugin, and (2) as a server-client implementation. In both cases it was initially written as an SSGA. It has been extended to encompass other types of evolutionary algorithms, though these were not implemented in the experiments in the remainder of this thesis. In the first approach, where the evolutionary algorithm ran in the main loop of V-REP, it simply initialized a random population of individuals by instantiating a fixed number of genomes for a given population size. The simulation subsequently began by sequentially evaluating the genomes in the population through the following steps:

1. Construct the Robot and its control from the genome
2. Initialize its position
3. Start the simulation and update its control at every simulation time step
4. Receive a measure of fitness based on, e.g., distance traveled or energy acquired after a specified period

From here on, it followed the same steady state evolutionary loop as explained in [Section 2.6](#). After a specified number of generations, the genetic algorithm elicited a stop function that shut down the simulator.

The later server-client implementation was used to evaluate multiple individuals in parallel, speeding up the evolutionary progressions and allowing parallelized single runs on computer clusters. This required one processor of a node on a cluster to be responsible for the evolutionary algorithm while a number of client applications (depending on the number of processors available on a node) of V-REP applications booted and waited for commands from the server. The server application ran the evolutionary algorithm, created genomes, and stored the genomes in a shared folder. Once a generation of genomes was created, the server sequentially assigned a client instance to evaluate a specific genome. After the client was done evaluating a genome, it sent a fitness value back to the server application that was in turn stored in the client application. Since genomes could result in different phenotypes, and thereby different computational requirements per client, as soon as a client was done evaluating an individual, it received a new genome from the server to evaluate. This process was repeated

in every client until the entire generation was evaluated. Once every individual in a generation was evaluated, the server incremented the evolutionary algorithm by going through the selection, reproduction, and replacement phases, producing a new generation of individuals. Based on the fitness values, all elites of the genomes were stored in a specific shared folder. The size of the robots that were simulated by the client applications could vary in size, which impacted the computational requirements in each different run. The runs that evolved smaller robots were therefore also significantly quicker to evaluate since the computational requirement was significantly reduced.

For testing the simulator, the plugin was used to directly evolve a population on a single terminal with a windows operating system. The server-client implementation was mainly used on a computer cluster, though could also be used on in a windows operating system directly. This server-client implementation is thereby still useful on a personal computer when using multi-core processors.

4.2.2 Morphology

Each genome stores a base class reference to a morphology. The morphology type, instantiated by the morphology factory, is acquired from the settings file. This morphology type indicated that the genome should load either a fixed or a modular morphology. This fixed morphology was used in Veenstra et al. (2015), and the modular morphology is used in this thesis. This modular morphology, construed by a generative encoding, was randomly initialized for the initial generation. In the simulation environment, each variable of the implemented L-system represents a specific module *state* which encompasses all of the parameters of the morphology, control, and attachment rules inherent to a module state. The genome of the generative encoding is thus composed of a fixed number of module states predefined before an individual is generated and evaluated in the simulation environment. The mutable parameters of the module state are used to create new types of attachment rules and connections of the modules. The manner through which these rules produce the modular robots is discussed in the next few chapters and is therefore not illustrated here. It is illustrated in detail in Figures 5.4 and 6.3.

All modules have a male connector site and zero or multiple female connector sites. When assembling a modular robot, the robot is constructed from an initial module (axiom). This initial axiom of the L-system is always the first module type that is defined in the settings file. Male connector sites of other modules can potentially connect to the female connector sites of

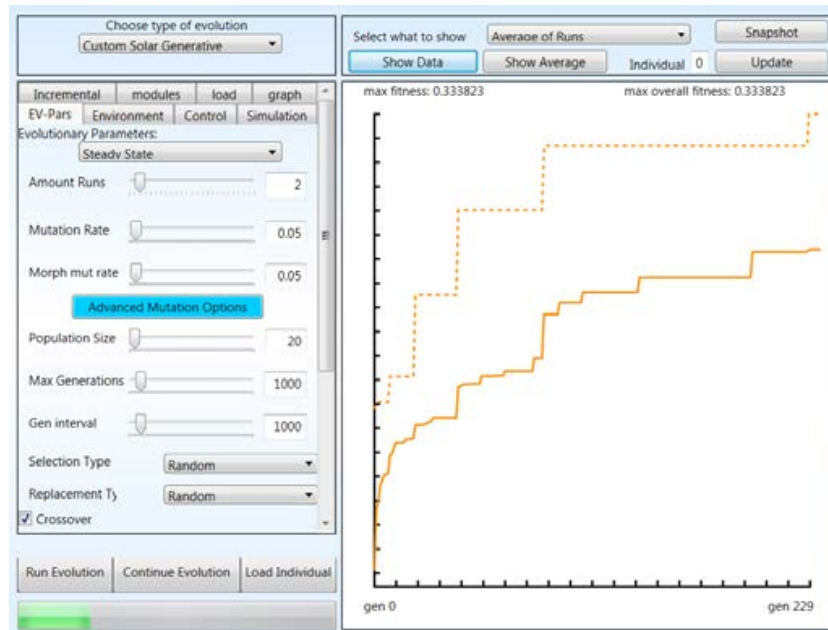


FIGURE 4.3: **Illustration of the interface designed for the evolutionary robotics plugin.** The interface shows a few adjustable parameters on the left with some menu options to select different parameters used, for example, for the modules, the type of controller or the environment. The right side can be used to quickly see the evolutionary progression. The interface is a standalone C# application that creates a settings file and begins V-REP by passing arguments to the application.

the initial module. Through this process, the modular morphology can be represented by a tree, and hence the use of an L-system is further justified. Each module type that has been elicited for use by the evolutionary robotics plugin stores information about the child modules and their orientations that specify how the modules are connected when incrementing the L-system. However, when this incremental step causes a generated module to collide with an existing structure, it will not be created. The outcome of the structure is thus also influenced by the environment as in a context-sensitive L-system (Prusinkiewicz et al., 1990). Each module is created directly by an instantiation of a virtual module class. This module class contains all of the instructions to construct and subsequently control the module if it contains any actuators.

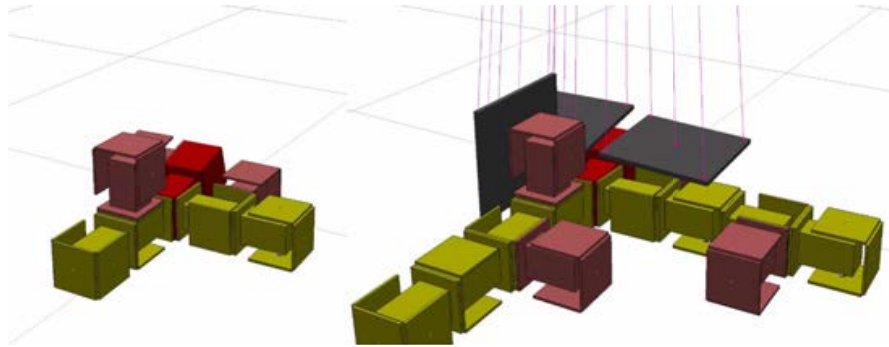


FIGURE 4.4: **Example of growth in the simulator.** These images display how the morphology can change after two additional developmental iterations (right) from an initial morphology (left), based on the L-system

4.2.3 Control

The control of the modules is regulated by neural networks that were generated for each module. In Chapters 6 and 7, the neural networks used for each module type were, however, simple and open-loop; the networks didn't utilize any sensory input other than predefined actuation functions. The neural networks consisted of three layers: a feedforward input layer, a potentially recurrent hidden layer, and an output layer. The input neurons could be connected to sensory input such as light absorption or servo position. The output neurons could be connected to available actuators in the modules. The input neurons could give a time-dependent patterned output that was defined by a sinusoidal function when specified. The amplitude, frequency, and phase offset in the sinusoidal input neuron were mutable parameters.

The neural network was initially developed to allow complex communication patterns to arise in the modular robot between modules. This was done through allowing additional input and output neurons across different modules to influence one another. For example, an output neuron could connect to the input neuron of an adjacent module directly. These implementations are reserved for future experiments using the evolutionary robotics plugin to evolve modular robots. This will hopefully shed light on how to improve designs in robots and whether it is efficient to control a robot in either a centralized or a decentralized manner.

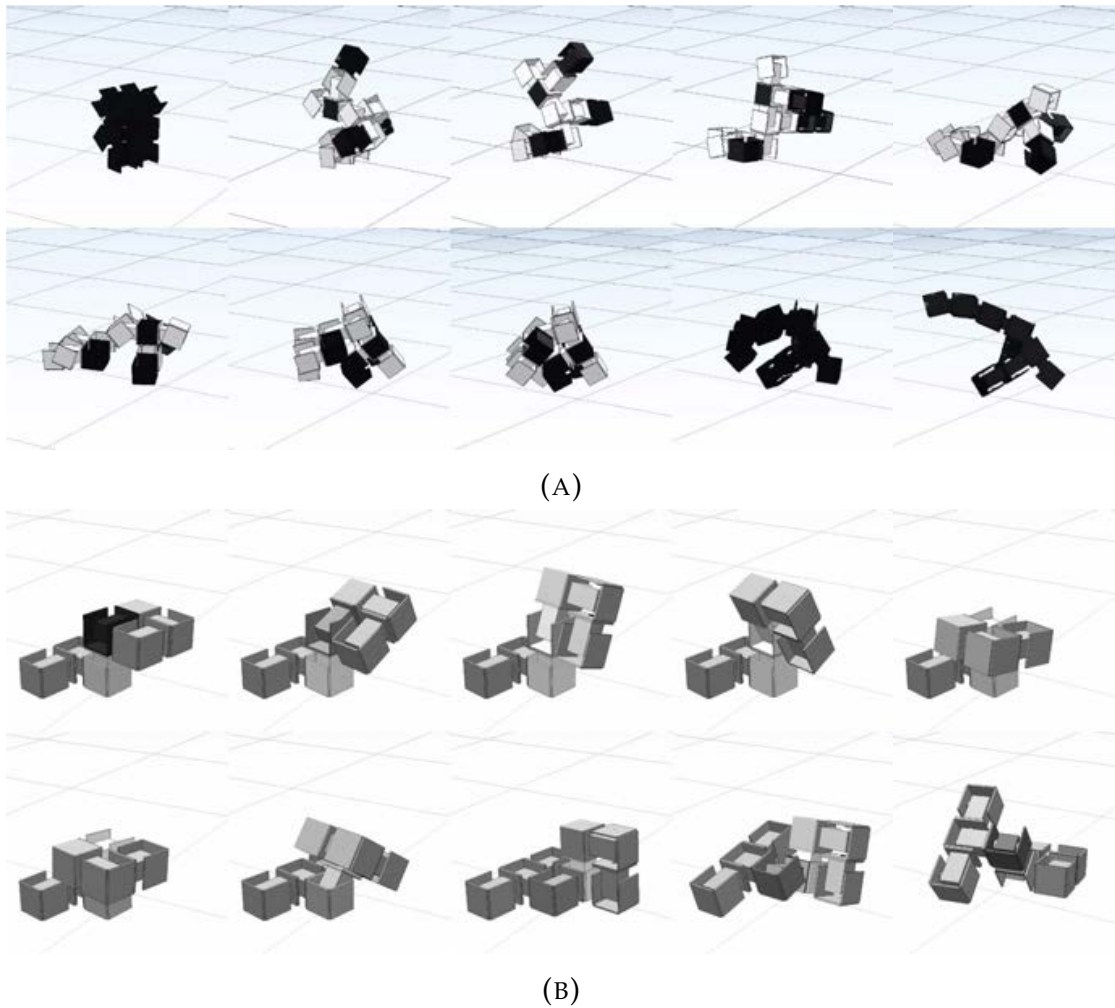


FIGURE 4.5: **Two modular robots controlled by decentralized neural networks.** The color of the individual modules represents the output of the neural networks which ranged from -1 (black) to 1 (white).

4.3 Current State and Functionalities

The current state of the simulator permits the adjustment and control of many parameters of the evolutionary robotics plugin. A small user interface was developed to quickly see the effects of a specific evolutionary approach (Figure 4.3). In this user interface, the parameters of the evolutionary algorithm, the encoding, the environment, the control type, and the fitness function can be specified.

Some noteworthy additional features were implemented to potentially improve the effectiveness of the evolved modular robots. One of these

features was online growth of the robot (Figure 4.4). This function was implemented since growth was one of the objectives of the plant-inspired approach to building modular robots, where the input of a light sensor input could potentially guide the addition of new modules towards a light source, similar as to Zahadat et al. (2017). However, growth is not used since the current setup would require a human-in-the-loop (requiring a human to physically adjust the robot during its lifetime) to adjust the phenotype of the modular robot. This would make the acquisition of modular robots more complex since a researcher would not only have to evaluate an evolved robot in the real world by assembling and controlling it, but also through adjusting it during its lifetime. A truly autonomous approach would be to facilitate the modular robot to self-reconfigure. One approach to accomplish this is through using a industrial robot arm to construct the modular robots; this approach is briefly discussed in Chapter 8.

Another feature not utilized but worth mentioning is the connectivity of separate neural networks in each module whose activation could be visualized as seen in Figure 4.5. The figure shows that neural networks with a single output ranging from -1 to 1 can control the desired angle of the servomotors. While an output of 0 keeps the servomotor in the center (grey), an output of 1 moves the servomotor +90 degrees (white), and a value of -1 moves the servo -90 degrees (black). The movements were controlled by PID controllers that were similar to the implemented Robotis AX-12a and AX-18a servomotors used in the real modules. Cheney et al. (2013) have similarly visualized the voxels of a simulated soft robot depending on their activation pattern useful to directly see how the phenotype adjusts itself to input signals changing over time.

The implementation of communication between modules sprouted from an investigation on the effect of the directionality of communication between neural modules. In a decentralized control, communication from the axiom module to the distant modules, *vice versa*, and communication in both directions were possible. Here, the sensory input and motor output can be transmitted across the modular robot. This approach may shed light on whether a decentralized embodied control of modular robots is advantageous. Which implementation is truly useful and whether any of these implementations could explain signal propagation in nature remain to be evaluated in future experiments.

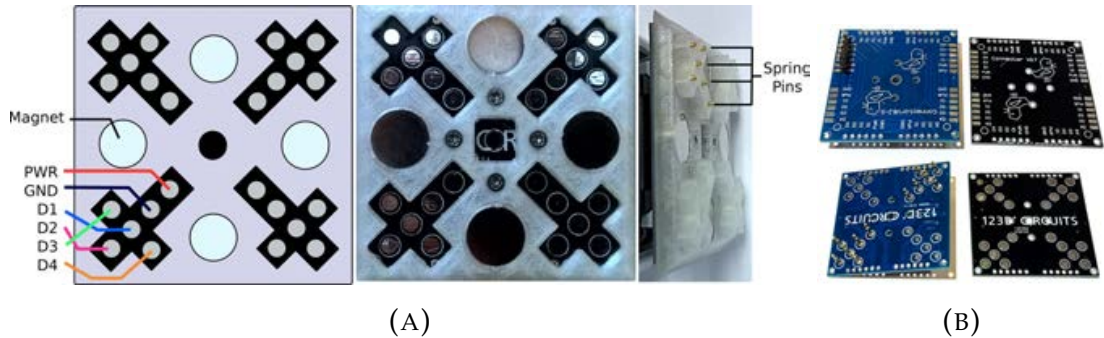


FIGURE 4.6: **Module connection site.** (A) The connection mechanism of the modules is composed of a female connector (left and middle) and male connector (right). Each module contains six channels through which modules can distribute power (PWR), ground (GND) and data (D1,D2,D3,D4) from one module to another. Both the male connector and the female connector have copper pads. Spring pins are attached to the pads on the male PCB. These spring pins (right) allow for a current to flow between modules (B). The connection sites contain magnets through which male sites can be connected to female sites. The blue (left) PCBs have places to connect the spring pins and is used for the male connector sites while the black (right) PCBs are used for the female connector sites.

4.4 Real Modules

In conjunction with the modules designed for the simulator, some of the modules were also created in reality. The modules that were used in [Chapter 6](#) were based on modules that were collaboratively developed. The EMERGE module, which is a variation of the module used in this thesis, is discussed in Moreno et al. (2017).¹ In the EMERGE module, PCBs have been developed with additional infrared sensors that have not been implemented here. Instead, I have designed PCBs with six, rather than 3 to 4 electrical channels. The six channels allow electricity to flow both to and from a power source allowing for the potential automated charging of a power source. However, more channels meant less space; hence, no infrared sensor was integrated.

¹An overview of the different module types and the parts needed to construct them can be found here: <https://sites.google.com/view/emergemodular/home>

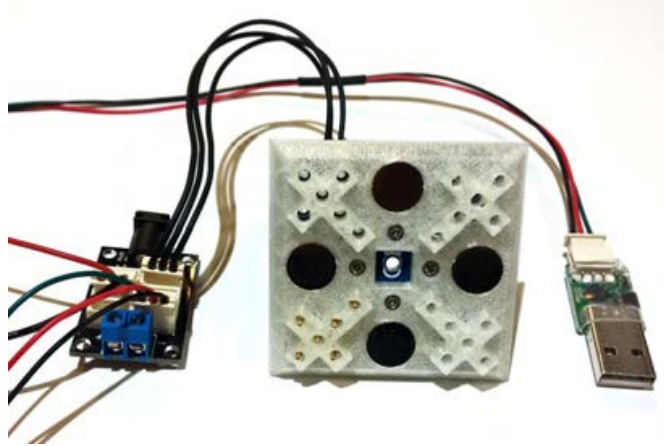


FIGURE 4.7: **Interface Connector.** The picture displays the connection site, USB cable and power hub that powers and controls the modular robot. This connection site could be connected anywhere on a female connection site of the modular robot.

The connection mechanisms and the servomotors used were the same in the different module types. The specific mechanism that connects the two modules, and the PCBs attached to these connection sites, is depicted in [Figure 4.6](#). The real modules are connected to one another via magnet-based connection sites ([Figure 4.6a](#)). The connection sites contained PCBs with pads and spring pins that enabled electricity to be routed between modules. As mentioned, the connection sites could be either male or female. The male connector sites contained spring pins that were soldered on the pads of the PCB ([Figure 4.6b](#)). The female connection site contained 3mm pads to ensure a connection with the spring pins on the male connection site. The PCBs included six separate channels through which electricity could be routed. 3D printed hulls housed four cylindrical NdFeB (neodymium, iron and boron) magnets (12mm diameter, 3mm depth). Two magnets could hold one another together with a force of roughly 13.5N. The male connector hull had protrusions that ensured the fit and connection of the male connector to the female connector.

Two solar panel modules have been designed for the modular robot and were implemented in the experiments discussed in [Chapter 7](#). One “flower module” has been collaboratively designed with a MSc. intern, Chloé Metayer. This flower contained an additional vibration sensor and a servomotor that could open and close the module base on the sensory input. In addition to the experiments in the next chapters, preliminary results of the automated assembly of modular robots is also presented

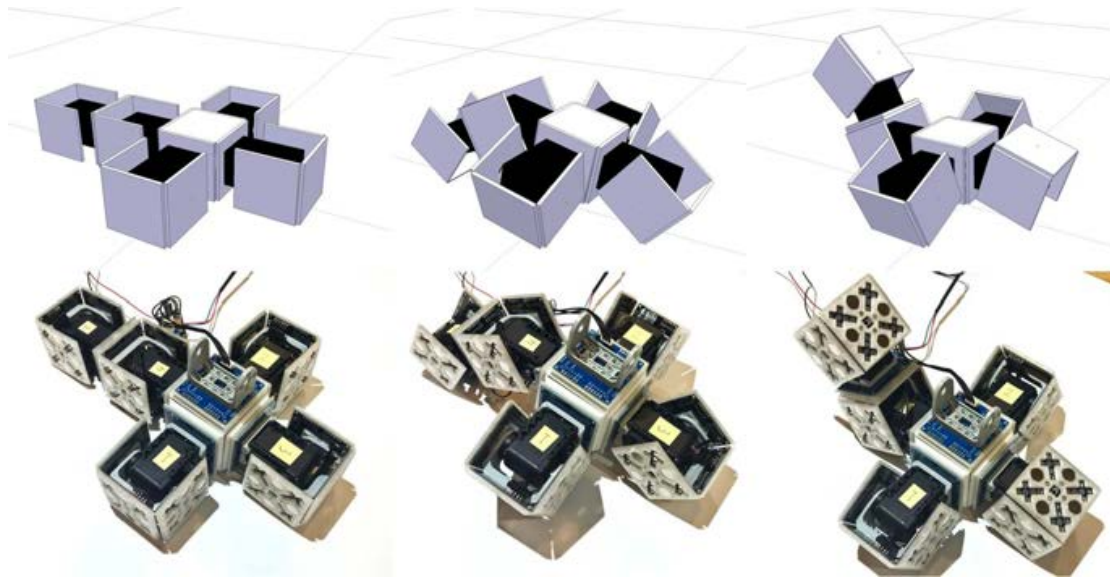


FIGURE 4.8: **Collection of Modules.** 5 solar panel modules, 7 servo modules, and 3 structural modules are depicted.

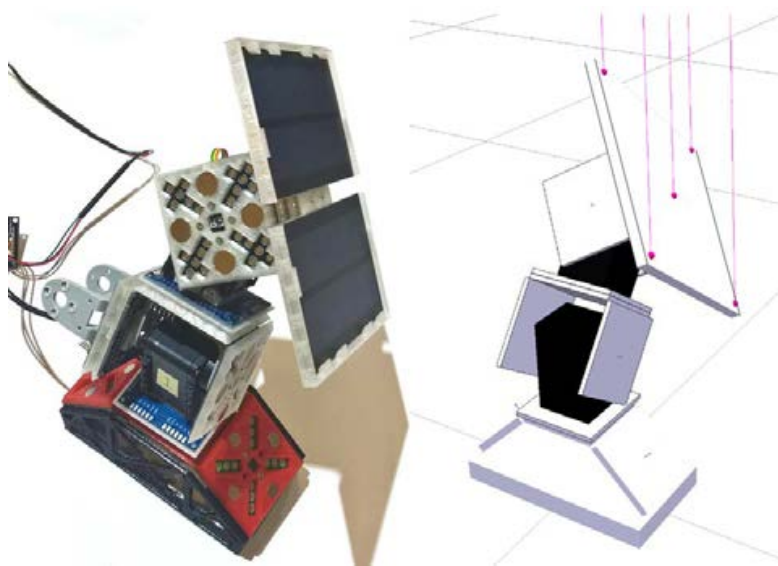
in [Section 8.2](#) and spurred from a collaboration with Rodrigo Moreno and Andrés Faiña. Since no evolutionary algorithm or three-dimensional simulation was implemented in this approach, it is briefly discussed in [Part III](#) of this thesis.

The modular robots created were connected to an external power source and a laptop via an additional connection site ([Figure 4.7](#)). This connection site could be connected to any open female connection site that would, in turn, automatically power and control all modules. A personal computer was connected to a central 3-pin power hub via a USB2AX interface. As can be seen in [Figure 4.8](#), the collection of modules was composed of servo modules, solar panel modules, and structural modules. The simulator could in turn limit the number of modules used based on the modules that were available in reality. Therefore, it would always evolve robots that were feasible to create in reality.

[Figure 4.9](#) depicts how a modular robot containing five servo modules, and a modular robot containing two servo modules and a solar panel, could be controlled in simulation and reality. Through moving the servo modules in reality, the real modular robot also adjusted its servomotors accordingly. The rays that come out of the solar panel module represent how the solar panels absorb light, the light absorption rate becoming lower when the solar panel is not perpendicular to the light source. Ultimately,



(A)



(B)

FIGURE 4.9: **Modular robots connected to V-REP via the plugin.** A collection of five servo modules and one cube module (B). The lines in the solar panel modular robot depict its connection to a light source.

the connection of the simulator to the modular robot can lead to a feedback loop of automated evolution, where multiple individuals can be evaluated in the simulator, while some elites are evaluated in reality.

4.5 Concluding Remarks

The evolutionary robotics plugin discussed in this chapter conveys a wide variety of potential applications in the study of the evolution of modular robots. The plugin implements an evolutionary algorithm, a morphology constructed through a generative encoding, and a decentralized method of controlling the modular robot with neural networks. It has been implemented for the evolution of plant-inspired virtual creatures ([Chapter 5](#)), comparing encoding strategies ([Chapter 6](#)) and evolving energy autonomy in plant-inspired modular robots ([Chapter 7](#)). There are many preliminary implementations that can shed more light on how to evolve and design robots in the future, such as growth and decentralized control. The use of modules is essential for the rapid prototyping of various morphologies with corresponding control strategies, which can aid in our understanding how to best create robots.

Chapter 5

Evolving Plant Morphologies with L-Systems

The chemical differences among various species and genera of animals and plants are certainly as significant for the history of their origins as the differences in form. If we could define clearly the differences in molecular constitution and functions of different kinds of organisms, there would be possible a more illuminating and deeper understanding of question of the evolutionary reactions of organisms than could ever be expected from morphological considerations.

– Edwin Ray Lankester

Plants are primary energy producers. Without them, we likely would not see the higher order terrestrial organisms we see today. All organism being tangled in a trophic structure, where organisms higher in the food chain are solely dependent on the lowest. Where these food chains are usually short due to inefficiencies of energy transfer across the chain (Reece et al., 2010). For natural evolving systems, energy acquisition is therefore most important. The design of an artificial entity can therefore start with energy acquisition. And what better suitable model for this than plants?

In this chapter, we therefore verge in realm of phytomorphogenesis, the acquisition of artificial plants optimized for the acquisition of light. Since the evolved creatures are plant-inspired and have light absorption as their fitness function, the morphologies are referred to as phytomorphologies (plant morphologies). Although the initial aim of the presented research was not directly related to modular robots, there are many features of plants that make it relevant for modular robots and robots in general. This chapter describes an approach, related to modular robots, to evolve the morphology and a minimalistic control of plant-inspired robots. In order to achieve this,

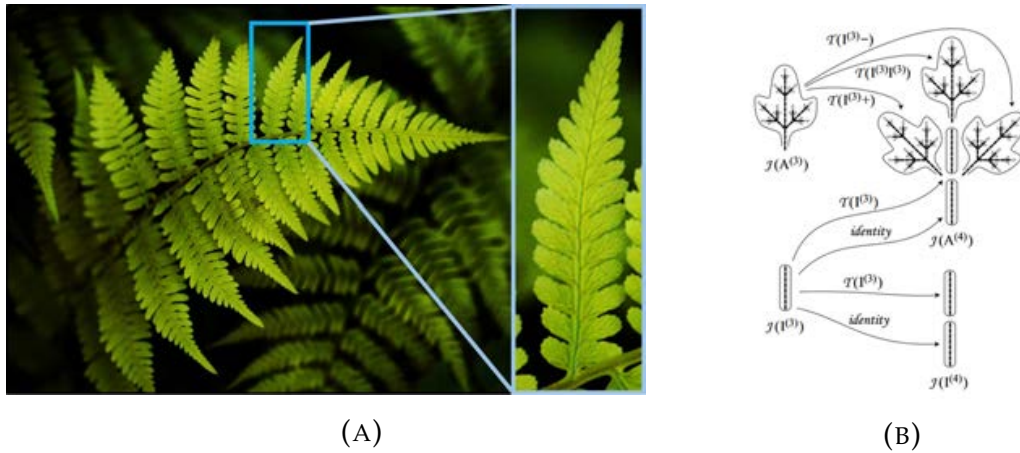


FIGURE 5.1: **Recursion and modularity in plants.** (A) Self-similarity as depicted in the fern. (B) How self similarity can be modeled through recursion and modularity (Prusinkiewicz, 2004).

a Lindenmayer-system (L-system; Lindenmayer 1968a; Lindenmayer et al. 1992) is implemented to see what type of morphologies emerge from the platform. The use of L-systems for the creation of 3D robots is thereby also used in other experiments using the robotics simulator as a generative encoding.

Considering macro structures in different organisms, plants exhibit many recursive patterns as self-similarity and modularity considering its leaves and branches (Figure 5.1). As described in Section 2.3, many of these characteristics are potentially beneficial for the evolution of any type of agent. As plants are relatively simple organisms, investigating how to evolve phytomorphologies might thus eventually lead to the creation of complex robots that could display characteristics of higher order organisms while also being ‘primary energy producers’. A curiosity that will be investigated in this chapter is whether phytomorphologies that have been evolved will exhibit movement to track a light source or not. Hereby formulating our second hypothesis:

Hypothesis 2 *Actuation in Evolving phytomorphologies is Beneficial for Optimizing Light Absorption.*

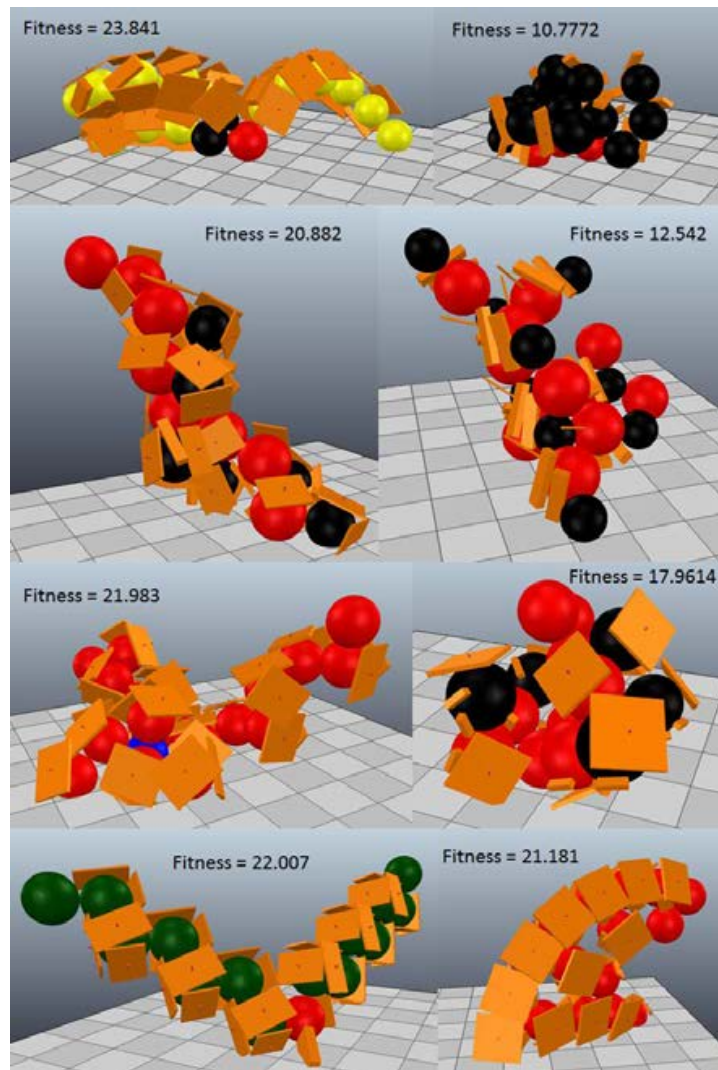


FIGURE 5.2: **Evolved Phytomorphologies.** Eight of the best evolved static individuals with their respective fitness values.

5.1 Introduction

The development of phytomorphological elements of plants ultimately arose from a dynamic interaction between genetic, ontogenetic, and environmental forces. Phytomorphological traits have emerged through the evolution and selection of plants, favoring those that were adequately adapted to their environment. Different environments stimulate the development and evolution of specific qualities in plants and contribute to the adaptation of plants to specific environmental niches, light absorption

being one of the most essential characteristics prevalent in almost all plants. The resulting role of plants as primary consumers conveys their fundamental impact on any terrestrial ecosystem.

Urban environments have replaced a large share of plant-rich environments, meaning that the potential energy uptake in these environments is exposed and primed for solar exploitation. For an efficient, but still aesthetically pleasing, deployment of solar cells, the developmental processes manifested by years of plant evolution is investigated. Hence, gaining insight into how plant development works and how this can be mimicked in intelligent robotic and autonomous systems is the main interest of this chapter. For investigating how to properly embody such systems, an evolutionary developmental model was used for investigating various factors that have contributed to the emergence of phytomorphologies.

In plants, various signaling mechanisms have evolved to communicate environmental factors to remote cells and tissues. Moreover, the cell walls of plant cells contribute to the relative immobility, as well as the rigidity of plants, limiting cell migration and thereby actuation. Lacking a nervous system, plants are forced to utilize relatively slow signaling molecules for communication. These molecules atone for the lack in efficient communication mechanisms through various diffusion and transduction pathways. The signaling molecules can be transported through an apoplastic (through the cell wall) or symplastic (via the cytoplasm; through plasmodesmata) pathway. Various molecules can also be transported over long distances through the vasculature of the plant.

Although plants acquired efficient dynamic behavior that directly influences morphogenesis, a complete modelling of plant signaling, as conducted by Zahadat et al. (2016), was not done since this might overcomplicate the design process and the potential increased computational requirement. Zahadat et al. (2016) have implemented a strategy that is similar to the concept of *morphogens* (Wolpert, 1969), in this case simulated hormones. Since actual robotic implementations of evolved phytomorphologies are likely not able to grow or move once created, grammars seem to be a suitable method to implement for generating potential phytomorphologies. Conventional approaches of these grammars can be simulated by simple grammars have been popularized with the use of Lindenmayer systems (L-systems; Lindenmayer et al. 1992). Furthermore, since L-systems work with variables, they can easily be extended to contain signal propagation algorithms, and even morphogens themselves, if one chooses to do so. One could potentially also model morphologies with direct encodings. A comparison of evolving robots with these two methods is discussed in the next chapter (Chapter 6). In

this chapter, an evolving context-sensitive L-system is implemented to engender phytomorphogenesis of artificial plants.

5.1.1 Phytomorphogenesis

Variation in plant features is influenced by many factors, including ecophysiological, phenological, morphological, and ontological traits. Other important factors driving plant evolution include resource allocation, biochemistry, metabolism, and leaf morphology and function (Ackerly et al., 2000). All of the genes are in turn subject to evolution, and specific genetic components are selected for across generations, where the absolute fitness value of a plant is, in biology, determined by the number of viable seeds it produces during its lifetime. Alternatively, as a more useful measurement, its fitness can be represented by the amount of genetic information it is able to propagate into the next generation. Both cases involve a difficult-to-measure value.

However, it has been shown that the photosynthetic rate of leaves in plants has a direct influence on the absolute fitness (the proportional change in the abundance of a genotype over one generation) of *arabidopsis thaliana*. One specific gene (At1g61800) causes leaves to produce more chloroplasts when plants were placed in a different environment where they were subjected to higher light intensity (Athanasidou et al., 2010). This is an example of the importance of dynamic feedback to plants. Solar cells do not have to worry about receiving too much light but a solar cells capacity to acquire energy can be modeled as a trade-off between average available light and amount of peak light intensity a solar panel is able to capture. Since dynamic behavior in plants is, evolutionarily speaking, a result of various compromises a plant takes to optimize its survival and reproduction, this behavior might not be necessary for plant evolution under controlled conditions. The experiments presented in this chapter therefore focus more on investigating intrinsic properties of plants that contribute to the generation of phytomorphologies.

Phyllotaxis is the main factor driving phytomorphogenesis (Cells, 1997). The most common patterns formed in plants through phyllotaxis include distichous, spiral, decussate and whorled patterns (Kuhlemeier, 2007). Notably, the divergence angles of *primordia*, tissues containing cells capable of triggering growth, of the plants differ usually by 180 90 137.5 Fibonacci; Newell et al. 2005) and some other more uncommon angles (Kuhlemeier, 2007). These mostly unimodal angles influence how well the leaves sprouting from the primordia can absorb light and overshadow other leaves (Falster et al., 2003). Leaves can also be positioned at a certain

level of steepness that is advantageous for either preventing self-shading or capturing light from the morning and evening sun (Falster et al., 2003). Steeper angles of lamina (blade of the leaf) are also more beneficial for plants that receive an amount of light higher than the maximum photosynthetic potential of a plant. When the leaves are steeply oriented, other leaves, which would otherwise be overshadowed, can receive more light and thus the overall photosynthetic activity of the plant is increased. Other evolutionary trade-offs that emerge in the leaves of plants include e.g. mass-to-area ratio, sap flow versus heat processing, CO₂ uptake to water loss ratio, and the leaf size-to-number ratio (Nicotra et al., 2011). Moreover, hormones, such as auxin, play an important role in embryonic development, cellular elongation and phyllotaxis (Prasad et al., 2013). The driving factors that induce phyllotaxis in plants can make the search space of an evolutionary model quite convoluted, which artificial plants do not necessarily have to take into account. As is discussed next, an L-system can result in a patterned formation that is similar to phytomorphologies.

5.1.2 Simulated models

Computer models of plants have generally been implemented in computer graphics (Habel et al., 2009) for accurate modelling of plant dynamics (Cournède et al., 2008; Prusinkiewicz et al., 2012; Merks et al., 2013; Runions et al., 2014) and for assessing the role of evolution on the emergence of plant traits (Valladares et al., 2000). Moreover, evolutionary computations and generative encodings have been implemented to efficiently simulate plant models (Zamuda et al., 2012; Zamuda et al., 2014) with some biological accuracy. In previous work on generating patterned morphologies, and for keeping the morphological encoding simple, generative encoding strategies, such as the parametric encoding used in the work of Sims (Sims, 1994a), are usually implemented since they can recursively generate body segments. Different types of generative encoding strategies have been developed over the past two decades to abstract developmental strategies towards generating both morphology and control of virtual creatures (Eggenberger-Hotz, 1997; Yeom et al., 2010). One strategy for generating artificial structures linked to neural networks is known as artificial ontogeny (Bongard et al., 2001; Bongard, 2003). In this method, an agent's simulated spherical elements can grow by increasing in size and splitting in two. As a result, repeated divisions can transit a single unit in a fully developed agent. Each separately created unit contains up to six joints and diffusion sites. These diffusion sites could in turn contain zero or more sensory, motor, and interneurons. Despite a promising application

of artificial ontogeny to produce plant-like structures, the implementation of neural networks can result in a substantial increase of the search-space making it a less attractive system to implement. Nonetheless, it would be an interesting methodology to implement in the future.

A Lindenmayer system (L-system; Lindenmayer 1968a) is another grammatical generative encoding approach originally used to mimic plant development by iteratively rewriting variables and constants through a set of rules (Lindenmayer et al., 1992). L-systems can be seen as a developmental representation of a virtual plant. Like some other generative encoding strategies, the similarity of L-systems to biology includes its reuse of rules and variables comparable to how organisms reuse genes. Further relevance L-systems have to biology can be derived from the fact that cells, or parts of plants, can change their state, or cell fate. This determines the behavior and ultimately the phytomorphogenesis of plant form and structure. L-systems are thus an attractive method to implement for creating artificial plants, both because they somewhat mimic biological development, and because they are extremely simple and efficiently encoded. L-systems have furthermore been used to create the morphological structure of virtual creatures with reactive controllers (Hornby et al., 2001). This approach can similarly be effective for the generation of virtual plants.

5.2 Approach

Virtual Robot Experimentation Platform (V-REP; Rohmer et al. 2013) is used as the simulator to create and evaluate plant-like robotic morphologies. The simulated components are controlled via a C++ based DLL plugin created with visual studio 2013 as described in Chapter 4. The plugin is divided into three parts: a genetic algorithm, a morphology generator and a control part. The genome of the morphology is encoded as the rules and parameters of the L-systems. Two experiments were done to simulate 16 evolutionary runs for evolving *static* phytomorphologies as well as 16 runs for evolving phytomorphologies, which contained joints that could rotate.

5.2.1 Genetic Algorithm

The implemented genetic algorithm is a steady state genetic algorithm (SSGA; Wu et al. 1995). In our case, a random offspring is generated asexually, without crossover, and evaluated against a random individual in the population. Random selection and a population size of 100 individuals

was used to keep the population somewhat diverse and to slow the convergence of the evolving L-system to a local optimum. The genomes of the initial population were further randomly initialized. The individuals were evaluated based on their ability to absorb light in an environment that only contained a flat surface, a light source, and the individual itself. When comparing an evaluated offspring with a random individual of the population, the offspring would only replace the selected individual if its fitness value were higher. Based on preliminary experiments, the mutation rate was set to 5%, meaning that each variable of the genome had a 5% chance of being changed in an offspring. When mutating the variables, either a completely random new variable could be assigned, or a local mutation could cause the variable to change slightly. Where local mutations cause a spread that used to explore the local search space, whereas the random variable was intended to potentially promote diversity. To recall a morphology without needing an L-system to construct it, the evaluated morphologies were also stored.

Ten evaluation steps contributed to the eventual fitness value of a virtual plant. At each timestep, the amount of light absorbed by the simulated leaves of an individual was calculated. The orientation and surface area of the leaves have a direct influence on the amount of light absorbed by the leaves. The amount of light absorbed is calculated by the multiplication of one light-sensitive surface area of the artificial leaf by the z-directional vector of the leaf relative to the directional vector that is oriented from the leaf's origin to the origin of the light source. Furthermore, if there is anything between the artificial leaf and the light source, the leaf does not contribute to the overall fitness value of the individual. The light source that directly influences the fitness of the virtual plants is moved at each timestep. Starting at the Cartesian coordinate (2.0, -4.0, 10.0) and ending at the coordinate (2.0, 5.0, 10.0), this light source moves in the direction of y with a directional vector of (0.0, 1.0, 0.0) as illustrated in [Figure 5.3](#).

The fitness function for each individual is given in [Equation 5.1](#).

$$F = \sum_{i=1}^n \left(\sum_{j=1}^o \alpha \delta - \sum_{k=1}^p \beta \right) \quad (5.1)$$

The fitness F is the sum of the acquired fitness values at 10 timesteps. n represents the upper bound of the number of timesteps. The total number of leaves is given by o , and p represents the total number of objects formed by the individual. α represents the surface area of the artificial leaves, which is multiplied by the z directional vector δ . β represents the volume of the objects.

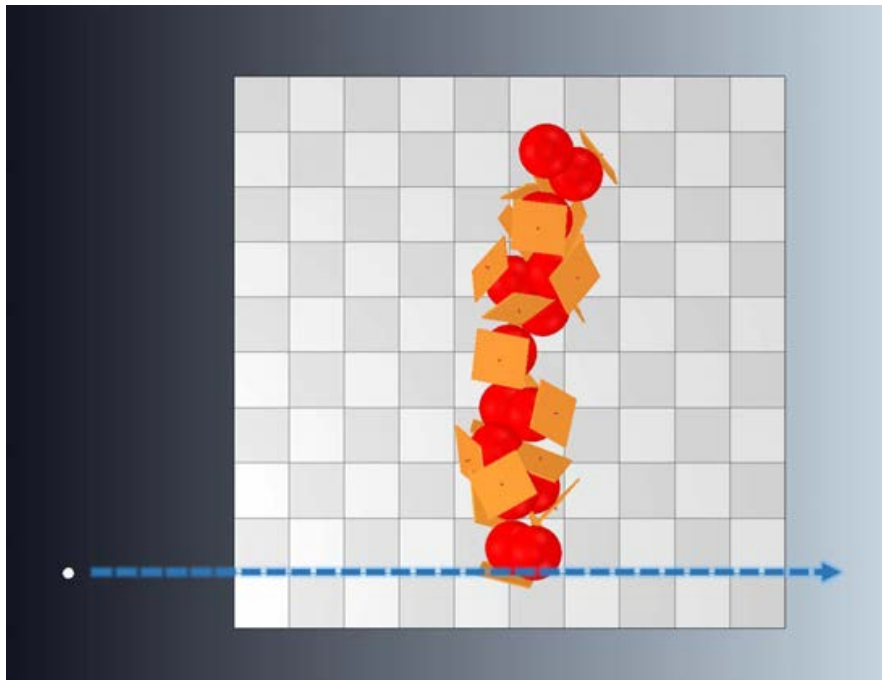


FIGURE 5.3: **Movement of a light source with respect to robot.** This figure shows the top-down view of the simulation environment with an omnidirectional light source shown as a white dot in the bottom left corner. The dashed blue line represents the movement of the light source.

5.2.2 L-system

The implemented L-system was context-sensitive (Prusinkiewicz et al., 1990). The context refers mainly to the simulated environment. For example, in order to prevent objects from overlapping, a feedback loop to the L-system ensures that the created morphology does not contain any overlapping/colliding objects. The L-system contains a ten variables that are referred to as the specific states of the objects that are created. Each state of the object contains corresponding rule sets that define which child objects are created. An example of how the states, rules and constants of the L-system influence morphogenesis is displayed in [Figure 5.4](#)

The L-system generates morphologies by iterating seven times through the state parameters of the morphology. Seven iterations were subjectively chosen, as they seemed to display a good diversity of morphologies without requiring too much computational power. The axiom of the L-system is a state 0 object. Before the first iteration of the L-system, an object in state 0 is therefore created at the center of the environment on top of the floor.

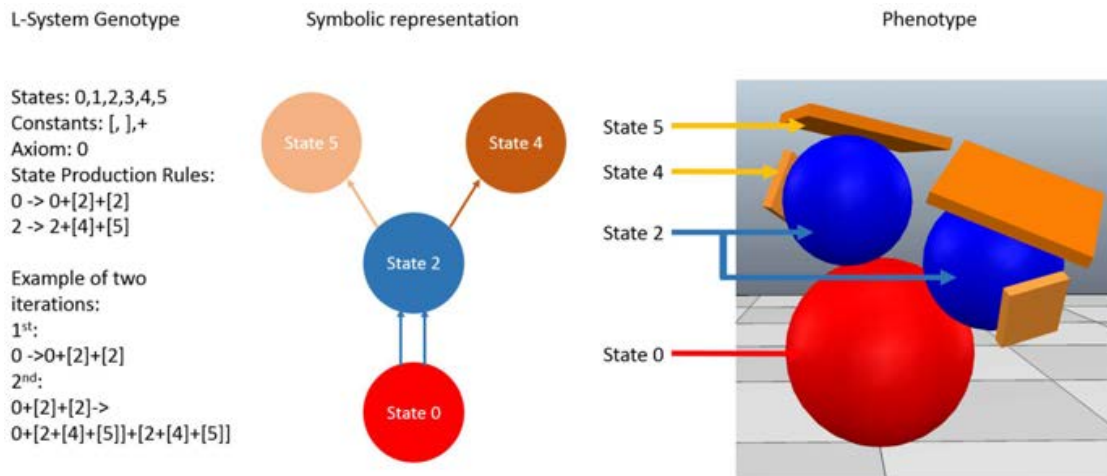


FIGURE 5.4: **Three illustrations of the implemented L-system.** The genotypic representation shows how the production rules result in the generation of the morphology. The symbolic representation shows the developmental instructions and the relationships between states as similarly represented by the work of Sims (Sims, 1994a). The phenotype generated by the example is shown on the right. Note that the + constant represents a three-dimensional orientation to which a new object is rotated relative to its parent.

Afterwards, the first iteration of the L-system will generate objects that the rules in state 0 produce. Having only seven iterations, an object chain from the initial object to the outermost child consists of a maximum of eight objects. Some loopholes in the L-system can quickly result in a very high computational demand, and thus specific constraints are implemented. Every object in a given state can potentially create up to six new child objects. The maximum number of objects that can be created is therefore limited to 50. Likewise, the number of loops the L-system can make for generating these objects is limited to 200 to further limit the calculation time that would otherwise arise. To enable individuals to absorb light from the environment, two object states of the L-system genome represent artificial leaves that are expressed as rectangular cuboids. These leaves are colored orange. All other states represent spherical objects that shape the overall morphology. Spherical objects were chosen to effortlessly calculate the position of new objects without needing to worry about collisions and overlapping objects. The objects in four other object states are colored red, blue, green, and yellow, while the remaining objects are colored black by

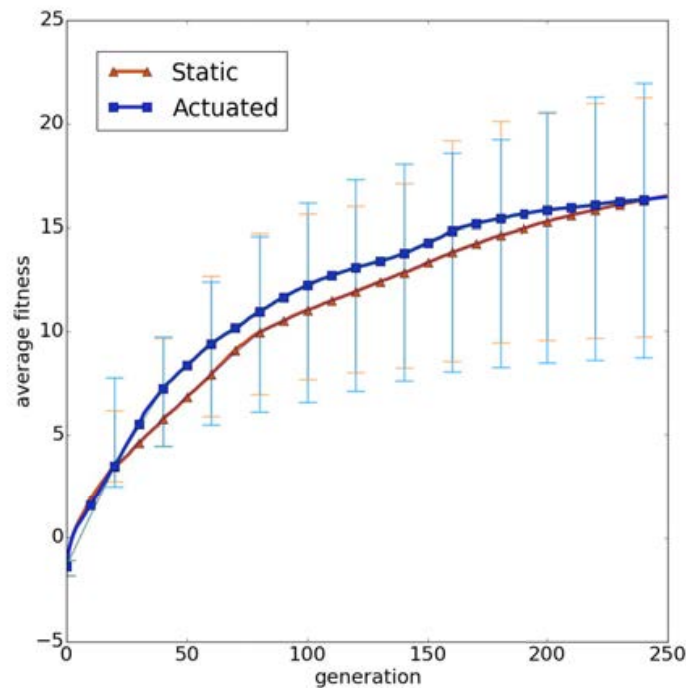


FIGURE 5.5: **Evolutionary progression of static and actuated phytomorphologies** The average fitness values of the populations across generations. The runs are not significantly different from one another (Mann-Whitney U test p-value was 0.782).

default. Note that the first object created is always in state 0, which is always colored red. An illustration of how the L-system generates the phenotype from a specific genome is depicted in [Figure 5.4](#).

Additional components are included in the L-system to enable movement of the joints in simulation time. Whether a joint moves is represented by one Boolean value. The angular rotation a joint can make per timestep is limited to 36 degrees, meaning that a joint can rotate a maximum of 360 degrees in a positive or negative direction during one evaluation.

5.3 Results

As can be seen in [Figure 5.5](#), the average acquired fitness values of the population with static plants is similar to the fitness of the population of

plants that could potentially actuate their joints. Since the evolutionary runs were not normally distributed (confirmed by a Shapiro Wilk test) a Mann-Whitney U Test was performed to see whether the results were significantly different. As can be inferred by looking at the graph (Figure 5.5), the Mann-Whitney U test confirms that the data is insufficient to reject its null hypothesis. For the number of generations that we have simulated the populations, no statistical difference between the efficiency of static versus actuated phenotypes could be observed. Considering that the runs shown in Figure 5.5 did not plateau, a difference might emerge when simulating far more generations. Although a few phenotypes did utilize moving parts (such as Figure 5.8), the majority of the phenotypes that evolved did not move. Out of the 16 evolutionary runs of rotating individuals, the best individuals of the final generations seldom utilized any actuation in joints that would change the shape of the artificial plants significantly.

The fitness values depicted in the graph of Figure 5.5 are quite arbitrary at first sight. However, with some additional knowledge, the fitness value can make sense. For example, the fitness value of the best-evolved individual (Figure 5.6) was 23.841. Without the negative contribution of the volume of the individual, its fitness would have been 31.9399. The division of this value by the amount of timesteps results in the average surface area of the artificial leaves that was exposed to the light source. This area is corrected by the relative angle the leaves had in respect to the light source. 3.19399 m^2 is thus the two-dimensional projection of the average light-absorbing surface area of the artificial leaves. The total volume of an individual could also be extracted by checking the negative contribution of the volume. For the example, the total negative fitness contribution of the volume of the individual discussed in this paragraph was 8.0989. The total volume of the simulated individual was thus 0.80989 m^3 . Thus, the phenotype seen in Figure 5.6 represents a structure with an average light absorption area of 3.19399 m^2 and a volume of 0.80989 m^3 .

The phenotypes of the evolved phytomorphologies are quite diverse, and different spiral patterned morphologies can be seen (Figure 5.2). In Figure 5.7, the best evolutionary run is mapped across different generations. Looking at the top view of this figure, one can see that the total amount of surface area exposed by the artificial leaves (orange rectangles) gradually becomes larger.

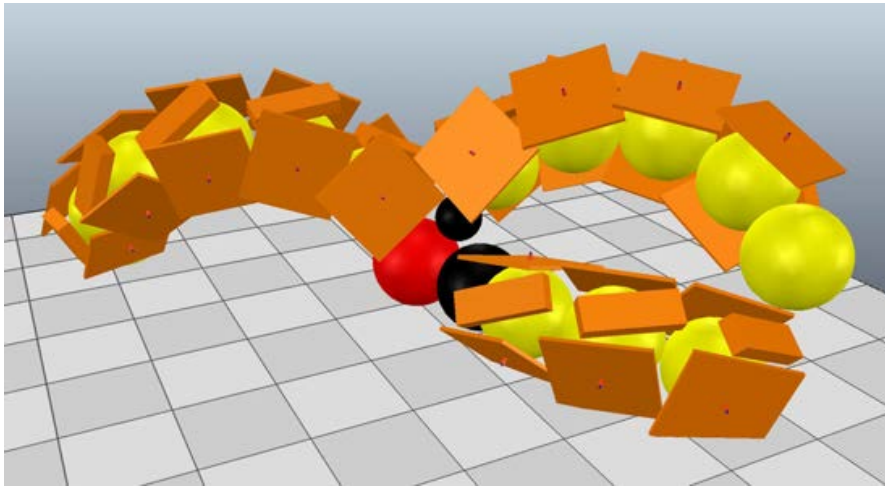


FIGURE 5.6: **The phenotype of the best evolved individual.** Note that object chains are surrounded by artificial leaves

5.4 Discussion

An evolutionary developmental algorithm was employed to engender various phytomorphologies optimized to absorb light. As can be seen in [Figure 5.2](#), a wide variety of phytomorphologies evolved. Functionally, these evolved morphologies do not look particularly optimal for light absorption as one would expect all of the orange surfaces to point somewhat upwards, instead of in the various directions shown in the resulting morphologies. Making longer evolutionary runs could shed more light on whether the evolutionary L-system can generate models that are more efficient. Moreover, actuating the morphologies did not change the population fitness values significantly when compared to the statically simulated populations. Blind tracking of a moving light source may have caused the search space to become more convoluted, making the algorithm inept for finding solutions where actuation was more beneficial than not actuating anything. Another explanation is that growing many leaves is simply more useful than considering movement and actuation. Considering that plants do not exhibit complex light tracking behavior in most cases, the conditions set in the experiment were not enough to confirm [Hypothesis 2](#).

The evolved virtual plants were quite voluminous considering that the volume has a negative effect on the fitness value. However, making large objects and dispersing the morphology over a large area,

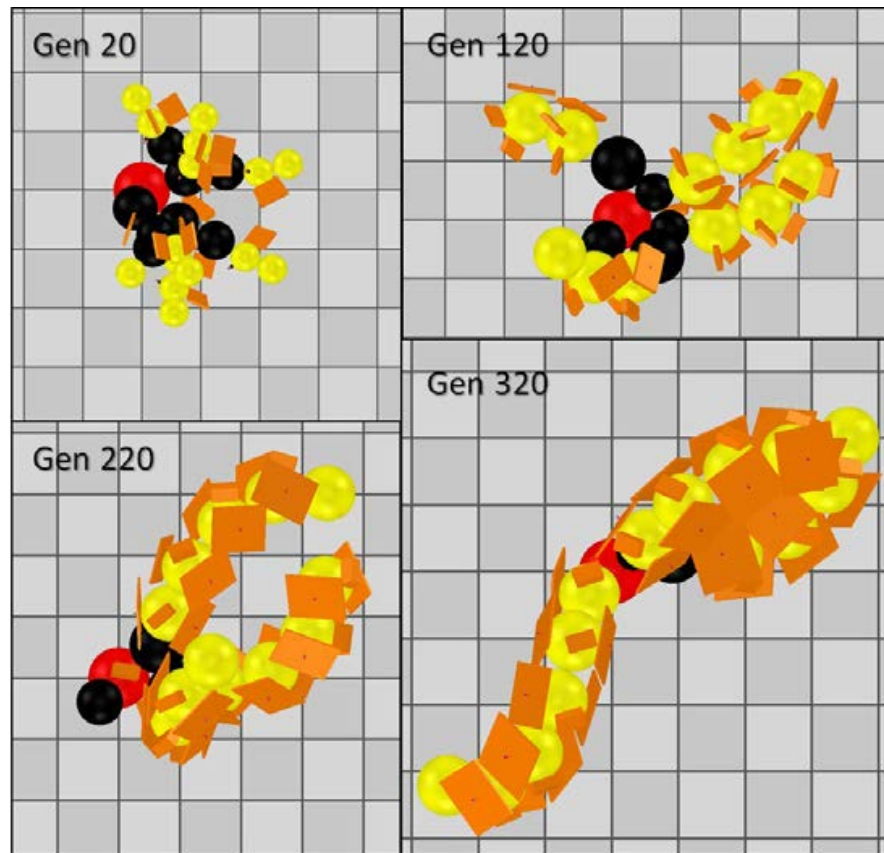


FIGURE 5.7: Illustration of an evolutionary progression
 A top view of individuals of one evolutionary run is depicted to illustrate how evolution shapes new more efficient individuals. The figures depict individuals from generation (gen) 20, 120, 220, and 320.

while making leaves with a thin volume but large surface area, is an intuitive result given the simulation environment. It is expected that different phytomorphologies arise when artificial plants have an additional restriction to grow horizontally. In biological environments, factors such as the overshadowing of neighboring plants form an additional pressure that stimulate specific types of plants to grow tall quickly. Co-evolving the same L-system can therefore yield results that are more diverse than the ones described in this chapter.

Considering the results, various future improvements of the evolutionary algorithm may increase the efficiency of a population to traverse the search space. Since no crossover function was implemented, this might definitely increase the efficiency of the evolving L-system, considering that

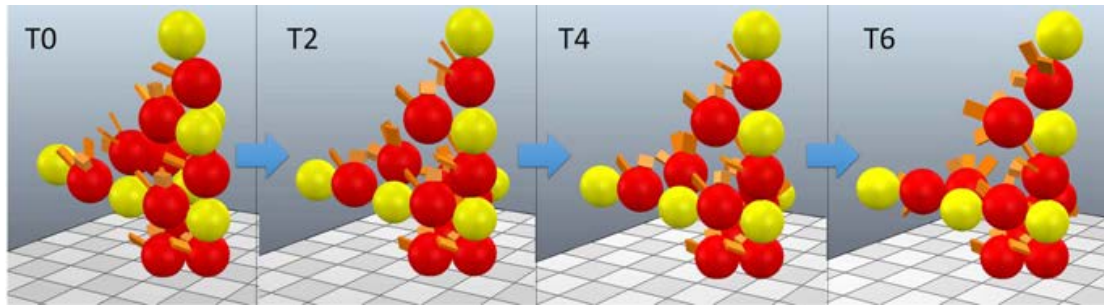


FIGURE 5.8: **Evolution of movement?** An individual that rotated some of its joints during the simulation. T0, T2, T4 and T6 represent the respective timesteps 0, 2, 4 and 6

specific states and rules of the L-system can be recombined between individuals within the population. As mentioned earlier, the implementation of neural networks in addition to artificial development can be interesting for developing more dynamic morphologies (as in Bongard et al. 2003). Moreover, morphogens (Wolpert, 1969) are also an attractive strategy to implement to mimic long-range communication in plants. An algorithm that checks for diversity, besides quality, as has been implemented in novelty search (Lehman et al., 2008), might also be useful to speed up the search process. Moreover, novelty search can lead to the evolution of very distinct morphologies making it more useful for people that possibly want to generate phytomorphological structures for aesthetic purposes. Additional implementations that have not been published, but which might be interesting for the reader, are displayed in [Text Box 5.4](#).

Text Box 5.4: Additional Implementations

Prior to the publication of the work shown, an approach that was more related to Sims (1994a) was taken to create the actual morphologies. The difference between the approach described in this chapter was that objects were able to collide and simple rectangular shapes were used. Rectangular shapes were eventually less computationally demanding. This approach was, however, altered with the thought of crossing the reality gap in the future. The spherical approach was therefore easier since collisions were easy to avoid. Aesthetically, the rectangular phytomorphologies look more interesting, though this also depends on personal taste (Figure 5.9).

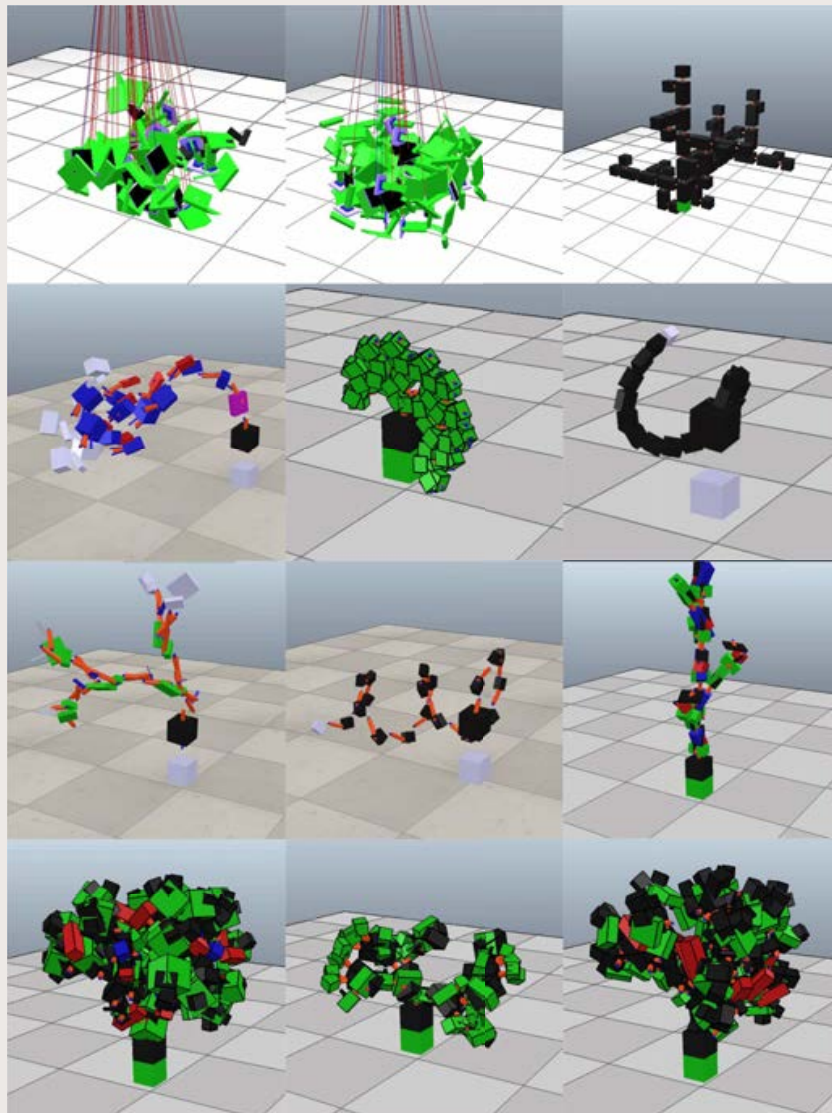


FIGURE 5.9: **Generated phytomorphologies.** Collection of various randomly created and evolved phytomorphologies using the robotics simulator.

5.5 Conclusion

L-systems can be utilized to create a wide variety of three-dimensional phytomorphologies that can be optimized for light absorption. These phytomorphologies were generated with the aim of implementing them in urban environments for both functional and aesthetic purposes. Interestingly, evolution did not exploit possibly beneficial joint actuation but instead converged on various types of static phytomorphologies. The plots did seem to somewhat plateau and noticing that movement has not evolved leads us to reject **Hypothesis 2**. In our system, it is more likely to simply grow many leaves than to additionally actuate them. This evolving L-system can be extended by implementing additional algorithms to increase the effectiveness of traversing the fitness landscape for acquiring both more efficient and unique phytomorphologies. It can moreover include a potential dynamic feedback controller than might lead to movement in leaves that aids the individual.

Chapter 6

Comparing Encodings for Evolving Locomotion in Modular Robots

Today's organisms are phylogenetically descended from others which were vastly simpler than they are, so much simpler, in fact, that it's inconceivable, how any kind of description of the latter, complex organism could have existed in the earlier one. It's not easy to imagine in what sense a gene, which is probably a low order affair, can contain a description of the human being which will come from it. But in this case you can say that since the gene has its effect only within another human organism, it probably need not contain a complete description of what is to happen, but only a few cues for a few alternatives. However, this is not so in phylogenetic evolution. That starts from simple entities, surrounded by an unliving amorphous milieu, and produce, something more complicated. Evidently, these organisms have the ability to produce something more complicated than themselves.

– John von Neumann, Theory of Self Reproducing Automata

The use of physical modules in robots enables both the morphologies and the controllers of robots to be subjected to evolution. The modular robotics approach additionally allows us to potentially quickly create the modular robot in the real world. It can then serve as a fast prototyping mechanism to evaluate different phenotypes in the real world. However, one major constraint is the modeling of the modular robot. To subject a modular robot to evolutionary algorithms requires an encoding from the genotype to the phenotype. This encoding greatly influences the emerging phenotypes of the robotics platform and should be considered in order to either explore or exploit specific parts of the search space.

Andres Faina had already developed a system capable of evolving modular robots originally developed for ROS (Robot Operating System). Since we thought V-REP was a promising new robotics simulator, Andres implemented the direct encoding he wrote for evolving modular robots (Faiña et al., 2011; Faiña et al., 2013) to V-REP. This enabled us to compare both the algorithmic implementation of the direct encoding and the generative encoding of the L-system as developed in [Chapter 5](#). Although his API was written in Java, and mine in C++, the results of the evolutionary experiments were observed in two separate implementations. The modules, stored as .ttt files, were simply loaded by the algorithms. For this implementation certain optimal parameters were set based on evaluating the performance of various parameters (tuning mutation rate etc.). These settings were afterwards fixed for our experiments. Since a generative encoding creates a mapping from genotype to phenotype that includes more recursiveness compared to than a direct encoding, the expectation was that the generative encoding would be more efficient than the direct encoding for evolving modular robots. Hence the third hypothesis of this thesis:

Hypothesis 3 *A generative encoding increases the efficiency of evolving modular robots compared to a direct encoding.*

6.1 Introduction

Evolutionary robotics has covered a vast amount of research on the automated design of robotic entities via artificial evolution (Lipson et al., 2000; Hornby et al., 2003; Eiben et al., 2013). To rapidly explore different robotic morphologies and control systems that can be physically assembled in the real world, robotic modules are useful as evolutionary building blocks, a robotic module being an independent unit that encapsulates part of its functionality (Stoy et al., 2010). This encapsulation is important for the (re)configuration of modular robot compositions. In contrast to static robotic entities, modular robots can be reconfigured to enable researchers to quickly explore different morphologies.

However, it is difficult to design a representation of the genotype to phenotype mapping of a modular robot, and we can either evolve all parameters of every simulated module or reuse parts of the genome to construct and control a modular robot. The latter approach—a generative encoding—would require a smaller genome and could in turn evolve decent morphologies and controllers more quickly. In contrast, evolving all parameters of every robotic module enables us to fine-tune behavioral

parameters, but also increases the search space. So, is it more efficient to fine-tune or to have recursive morphologies and control? Both encodings greatly influence the functionality of the evolutionary approach since being able to fine-tune is more likely an exploitative evolutionary approach while the abstractions taken in a generative encoding are likely more explorative.

In nature, most multicellular organisms develop from a *zygote* (Reece et al., 2010). The zygote and its genome comprise the developmental representation of the organism (Floreano et al., 2008). The resulting developmental process allows for the reuse of genes, which can give rise to recursive structures in the phenotype. Computational models representing an artificial organism's phenotype either use a direct or generative encoding (also known as indirect encoding). A direct encoding constitutes a one-to-one mapping of genotypic components into the phenotype, meaning that the genes encode for every simulated module. In contrast, generative encodings—similar to the development of an organism from a zygote—reuse elements of the genome for constructing the phenotype. Generative encodings have a smaller genotypic space due to this reuse of genes.

Since the morphological search space in modular robots is limited to the number of connection sites available on each module, encodings that directly map the assembling process of modular building blocks have been implemented for the generation of robot morphologies (Marbach et al., 2005; Faña et al., 2013; Guettas et al., 2014). Usually, these direct encodings implement an additional symmetry operator that increases the effectiveness of artificial evolution. A multitude of simple generative encodings have been implemented to evolve robot morphologies and control (Sims, 1994b; Hornby et al., 2001; Auerbach et al., 2011; Cheney et al., 2013) and discussed how to quickly evolve useful robot morphologies. It is, however, unclear whether designing platforms that evolve modular robot morphologies use a direct or generative representation, and if the generative encoding is still useful when just a few modules are being used. A generative encoding should no longer have an advantage if the genetic sequence space in both encodings is of similar size since the number of mutable parameters is equal.

Whether a generative encoding or a direct encoding is more useful for evolving modular robots for locomotion is the main concern of this chapter. Moreover, the experiments that described use modular robots of different sizes, only allowing a certain number of modules to be simulated in an individual. Both encodings make use of evolutionary algorithms to optimize the simulated robots. The direct encoding utilizes

the “evolutionary designer of heterogeneous modular robots” (Edhmoor; see [Subsection 6.2.2](#)) system (Faíña et al., 2013). Edhmoor contains a specific evolutionary algorithm that evaluates robots through multiple mutation phases, as explained in [Subsection 6.2.2](#). The generative approach utilizes a steady state evolutionary algorithm (Syswerda, 1991). The generative encoding is based on a parallel rewriting system called a Lindenmayer System (L-system; Lindenmayer 1968a) as discussed in [Chapter 5](#). The main difference between both implementations is the size of the search space. The search space of the direct encoding grows exponentially with the number of modules that determine the number of mutable parameters. The search space of the generative encoding stays roughly the same size. Although more variants in phenotype are available when allowing more modules to be simulated, the genome of the individual stays the same size. The ability of the direct encoding to mutate parameters of individual modules enables more local, fine-tuning improvements. In contrast, since small mutations in the generative encoding can lead to drastic phenotypic changes, the generative encoding might be more prone to stagnate in local optima. Though the scope of this chapter does not encompass transferability, the implemented encodings serve as a stepping stone towards evolving feasible modular robotic entities in reality.

6.2 Methodology

Many modular robotic systems make use of central pattern generators for controlling the modules (Kamimura et al., 2005; Sproewitz et al., 2008; Bonardi et al., 2014). These central pattern generators are derived from their natural equivalents seen in biology (Still et al., 2006; Ijspeert, 2008). The implementation of modifiable central pattern generators seems a logical step toward evolving modular robots. However, this convolutes the search space of the evolutionary system unnecessarily for the aim of this chapter. To still achieve a patterned output in the modular system, sinusoidal functions control each module individually in a decentralized manner. By fixing the morphological parameters of the simulated modules and limiting the control parameters of the modules to sinusoidal functions, we were able to analyze how the different encodings can be implemented for evolving robotic structures. For evolving simulated robot morphologies, two evolutionary platforms were used to evaluate the direct and the generative encoding. Both platforms employ the robotics simulator “Virtual Robot Experimentation Platform” (V-REP; version 3.32; Rohmer et al. 2013). The

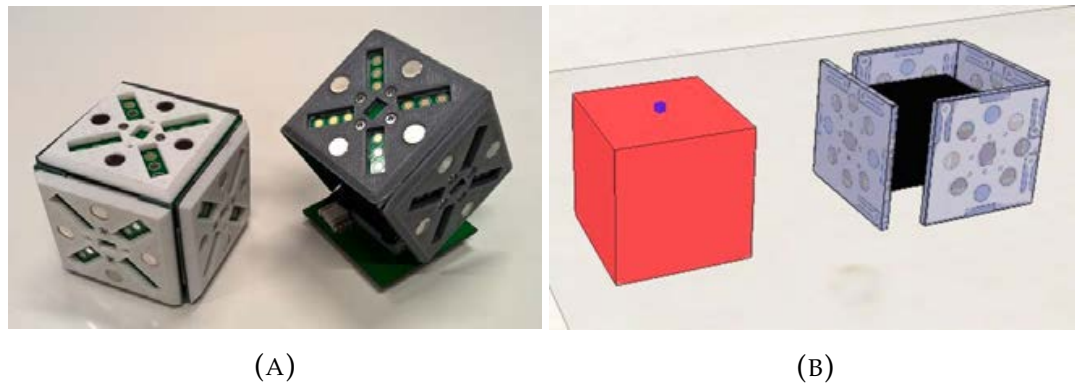


FIGURE 6.1: **Illustration of simulated and real modules.**
Modules in the real world (A) and in the simulator (B).

next sections discuss the common elements, as well as the differences for each platform.

6.2.1 Common Elements for Both Platforms

Both encodings simulate the exact same modules modeled in V-REP. A cube module and a servo module were designed for the platforms. The modules are based on earlier designs of physical modules (Figure 6.1a). In turn, the simulated modules (Figure 6.1b) are modeled according to the physical properties of these modular units. The real modules can be attached to one another via magnets, and the breaking force and torque parameters resulting from these connections is modeled in the simulated modules.

The modules contain male and female connection sites that enable the modules to connect. The connections are modeled with a force sensor in V-REP. If the force on a connection site exceeds 1.7 N m of torque or 80.0 N of force, the force sensor between the modules breaks leading to fragmentation of the morphology. Ten consecutive threshold violations for the force sensor had to be registered before a connection could break.

The cube module (dimensions x , y , z are 55mm, 55mm, 55mm; weight is 100g) is used as an initial building block for the modular robot to which other modules are attached. This cube has five female connection sites (top, right, left, front, back). The servo module (dimensions x , y , z are 55mm, 55mm, 80mm; weight is 160g) has three female connection sites (top, right, left) and one male attachment site (bottom). The bottom male connection site of the servo module is thus able to connect to any of the female connection sites of other cube or servo modules.

The joint of the servo module implements a PI controller (P is 0.1 and I is 0.01) and could exert a maximum torque of 1.5 N m. A sinusoidal wave function controls the position of the joint in the servo module. The maximum amplitude of the sinusoidal wave function ranged from -90nd +90degrees from its original position. The offset, phase, and amplitude of the sinusoidal function are mutable parameters. When a new module is added to the simulation, only the male connection site of the new module can be connected to any female connection site of the robot. The new module has four different orientations in which it can attach to a new connection site (note that the number of orientations is a bit different in the direct encoding: [Subsection 6.2.2](#)). The servo modules implemented the default simulation material, while the cube module used the “rest_stack_grasp_material” as material types simulated by V-REP.

The goal of the simulated robots was to move as far away from its initial position in a horizontal direction as possible within 20 seconds of simulation time. This distance is measured by the horizontal distance that the initial cube module has traveled. Before starting a simulation in V-REP, modules are joined together to form a robot morphology. The entire robot is then shifted upwards so that its lowest point is 0.1 millimeter above the simulated ground. To take into account the movement due to the robot simply falling over, the distance traveled in the first 2.5 seconds of the simulation is discarded. An additional cost function was added to compensate for modules that were disconnected due to the breaking of a connection site. The fitness value of each individual is directly correlated to the horizontal distance traveled, multiplied by the number of connections broken between the modules to the power 0.8, and can be derived from [Equation 6.1](#).

$$F = \sqrt{(p_e x - p_1 x)^2 + (p_e y - p_1 y)^2} * \eta^{0.8} \quad (6.1)$$

F represents the fitness value obtained by calculating the eventual position (p_e) minus the position after 2.5 seconds (p_1) traveled in both x and y directions. η represents the number of broken module connections of the morphology after 20 seconds of simulation time.

A simulation environment consisted of a default floor and was simulated using the bullet dynamics engine (version 2.78). The dynamics settings were set to accurate (default) with a timestep of 50ms. Six experiments were done comparing the different encodings. Three of the experiments ran 12 evolutionary runs whereby a maximum of 5, 10, or 20 servo modules and one cube module were allowed. These three experiments were done to see how the direct encoding performed. The

other three experiments analyzed the efficiency of the generative encoding and were composed of 12 evolutionary runs simulating a maximum of 5, 10 or 20 modules. The runs were limited to a fixed number of evaluations. In the simulations that could simulate a maximum of five modules, 12,500 evaluations were done. The other runs were limited to 25,000 evaluations; more evaluations were performed in these runs since the search space was larger when increasing the number of simulated modules. Twenty five thousand evaluations were chosen as a trade-off between performance and computational requirements. Since a high-end physics simulator is used, the computational requirements are considerable. The next sections cover the direct and indirect encodings in more detail.

6.2.2 Direct Encoding

The “evolutionary designer of heterogeneous modular robots” (Edhmoor; Faiña et al. 2011; Faiña et al. 2013) system was used as the direct encoding strategy to assemble and evaluate robot morphologies. The Edhmoor system is organized as a tree representation, where nodes represent control parameters of a module and its type and edges represent how a module is attached to a parent module. The direct encoding is used together with a constructive algorithm. This algorithm starts building a random population of robots with just a few modules. Afterwards, different mutation phases are applied cyclically. The mutation phases of the algorithm are:

- *Add Module*: Add a module into a morphology.
- *Mutate morphology*: Change the orientations or the place where some modules are connected
- *Mutate control*: Change the control parameters of some modules
- *Prune robot*: Test all morphologies generated by removing a module and its children

In every phase, a mutation operator was applied several times to produce different random mutations of the same individual, which were tested in the simulator. For example, when adding a new module to a robot, five different robots were generated and each of them had a new module placed in different positions and orientations. These phases revert to the previous robot if the mutation did not increase the fitness of the robot, except in the “add module” phase. This phase forces morphological evolution to take

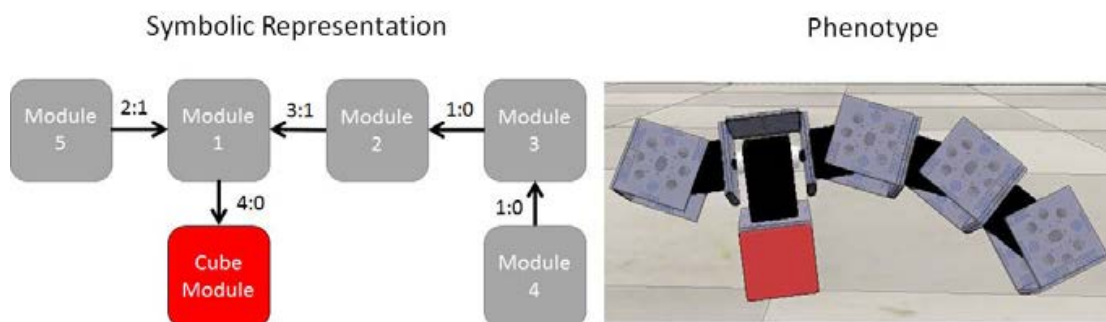


FIGURE 6.2: **Representation of the direct encoding.** (left) Symbolic representation of the direct encoding: each rectangle represents a module and each arrow represents a connection between modules. There are two numbers for each connection, which indicate the face of the parent where the child node is attached and the orientation of the child node. (right) The symbolic representation of the direct encoding that encodes for a phenotype.

place, which has been shown to be advantageous when evolving virtual creatures (Cheney et al., 2016).

The evolutionary algorithm of Edhmoor was furthermore generational; the 10% worst performing robots were removed from the population every cycle. Half of them were replaced by random robots with a low number of modules; the other half were generated by applying symmetry operators to the best robots. The symbolic representation and its phenotype are depicted in Figure 6.2. A more detailed overview of the system can be found in Faña et al. (2013).

6.2.3 Generative Encoding

The implemented generative encoding is based on a context-sensitive Lindenmayer-system (L-system; Lindenmayer 1968a; Lindenmayer et al. 1992)—a parallel rewriting system. The variables used in the L-system represent the modules employed to construct a robot (Figure 6.3), similar to the L-system used Chapter 5 and as described in Chapter 4. The generative encoding was limited to using five different module states. The first state (the axiom) represents the cube module, and the four other states represent the servo module. The four states that represent a servo module encode for the same module but can differ in their mutable parameters responsible for the sinusoidal function that controls the servo module. The attachment rules of the cube module included the information for which

module is connected to what connection site and in which orientation. The same attachment rules are possible in the servo modules but the servo modules only contain three attachment sites. The implemented attachment rules are in essence similar to the rewriting rules of a normal context-sensitive L-system (Lindenmayer et al., 1992). It is context-sensitive since a module cannot be placed at an attachment site if another module is already occupying it. Furthermore, modules cannot be created if this causes a collision with other created modules.

The internal sinusoidal function that controlled the PI controller of the modules could be mutated in the genome of the module states. This means that the robot can actually not have more than four distinct sinusoidal controllers. To illustrate the different object states, they are colored by phenotype. The modules could be either red, yellow, blue or pink depending on their state. Four iterations of the L-system were done to create the robot phenotypes starting with the cube module as the axiom.

All parameters of the module states were subject to evolution. There was a 15% chance of a morphological parameter being mutated and a 5% chance of a control parameter being mutated. A symmetry mutation operator enabled an object state to arise at the opposite site of a module where it originally was expressed. Though symmetry is an inherent trait to an L-system, the symmetry operator enhanced the probability of creating symmetrical phenotypes. Since the genome of an individual is represented by different module states, a crossover operator enabled different states to be exchanged between individuals. The crossover function had a 20% chance that a module state of an individual came from a different individual than its original parent.

6.3 Results

The results of the different evolutionary runs were divided in a performance analysis and a phenotype analysis. The performance analysis was done to obtain clear insight into the efficiency of the encodings. Knowing what type of phenotypes resulted from the evolutionary runs gives us more insight into what the prominent evolved characteristics were and how we can ultimately improve the simulator for the design of actual modular robots.

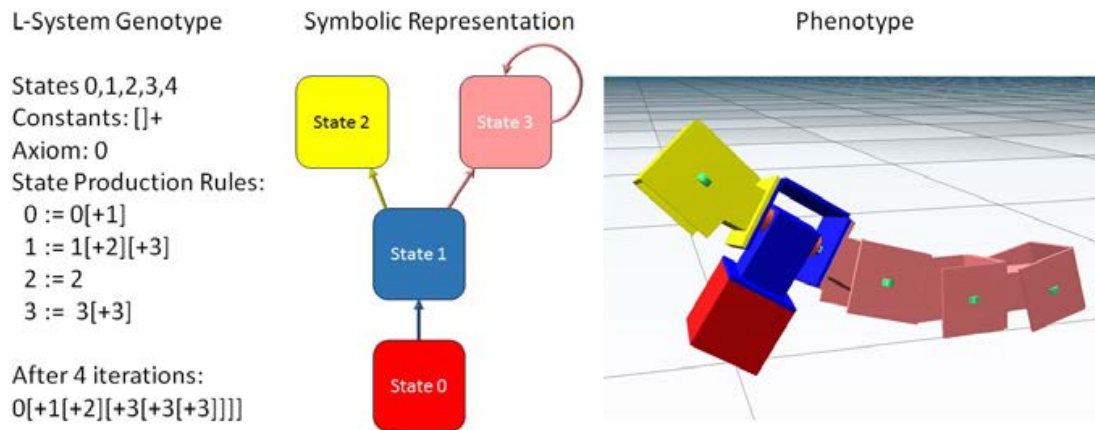


FIGURE 6.3: **Representation of the generative encoding.** (left) The L-system parameters form the genotype of the morphology whereby the variables of L-system are replaced by module states. The ‘+’ constant represents the placing of the next module at the specific attachment sites of a module. The symbolic representation of the genotype (middle) serves as a visualization on how the genotype constructs the phenotype (right).

6.3.1 Performance Analysis

As can be seen in [Figure 6.5](#), the average fitness values—as well as the averages of the maximum fitness values—of the evolutionary runs is quite different per encoding. The generative encoding seems to be able to quickly find decent behaviors that are rewarded with a high fitness value. A Mann-Whitney U test was performed at specified intervals to check whether the encodings performed significantly different. The performance difference was measured using the average fitness values of the maximum fitness of each individual evolutionary run at a specified time. The test resulted in a significant difference between evolved populations after 6,250 evaluations (p-value: 0.000612) and 12,500 evaluations (p-value: 0.003674) when simulating a maximum of 5 modules. There was also a significant difference between the two encodings at 6,250 evaluations (p-value: 0.00328), not at 12,500 evaluations (p-value: 0.0124106) but again at 25,000 evaluations (p-value: 0.001617) when evolving a maximum of 10 modules. The runs of the simulation evolving a maximum of 20 modules were also statistically different at evaluation 6,250 (0.00332) but not at evaluation 12,500 (p-value: 0.177805) or at evaluation 25,000 (p-value: 0.209462). The maximum and average fitness values of the individual runs can be seen in [Figure 6.4](#).

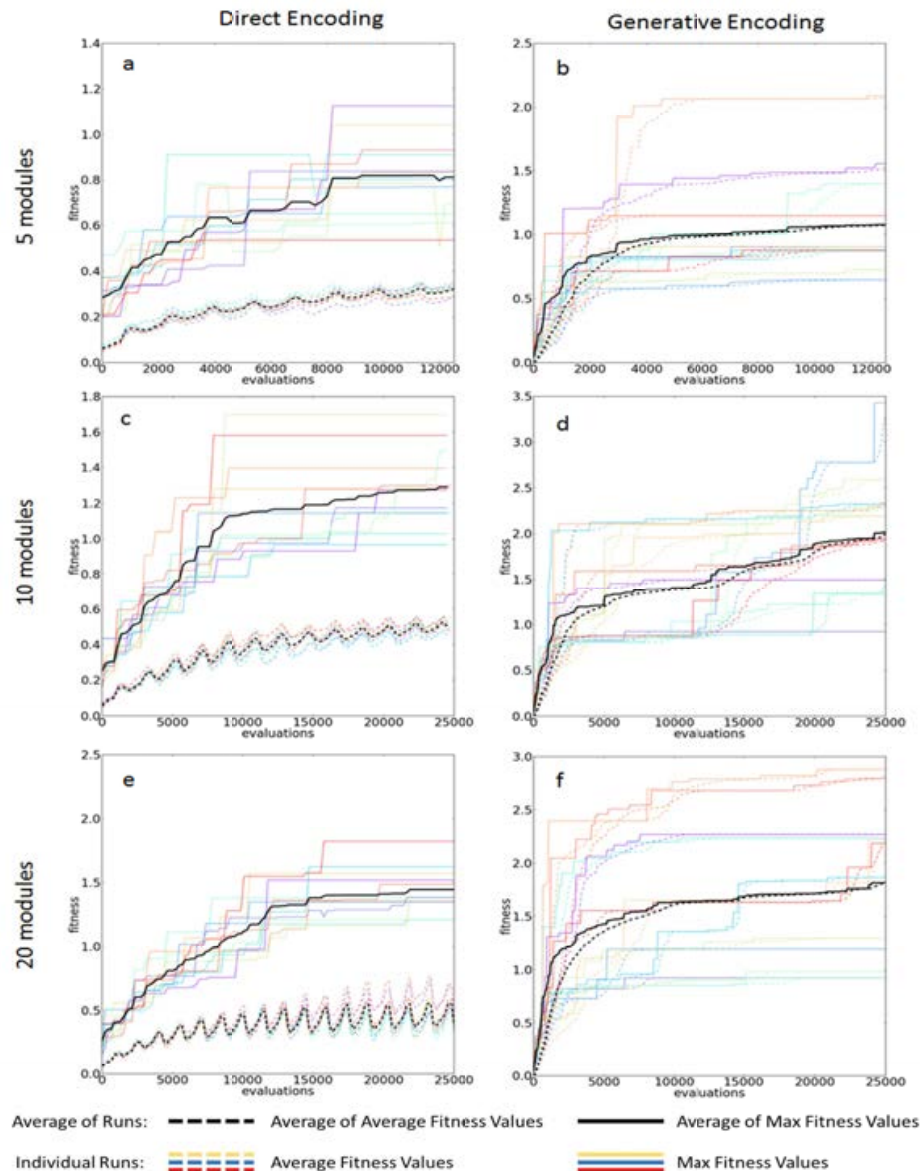


FIGURE 6.4: **Evolutionary progression of individual runs.** The six graphs represent the direct encoding simulating a maximum of 5 servo modules (a); generative encoding simulating a maximum of 5 modules (b); direct encoding simulating a maximum of 10 servo modules (c); generative encoding simulating a maximum of 10 modules (d); direct encoding simulating a maximum of 20 servo modules (e); generative encoding simulating a maximum of 20 modules (f). The bold black line represents the average maximum fitness values for all runs, while the black dotted line represents the average of the average fitness values of all runs. The colored lines represent individual runs; solid lines representing the maximum fitness of the population and dotted lines representing the average fitness.

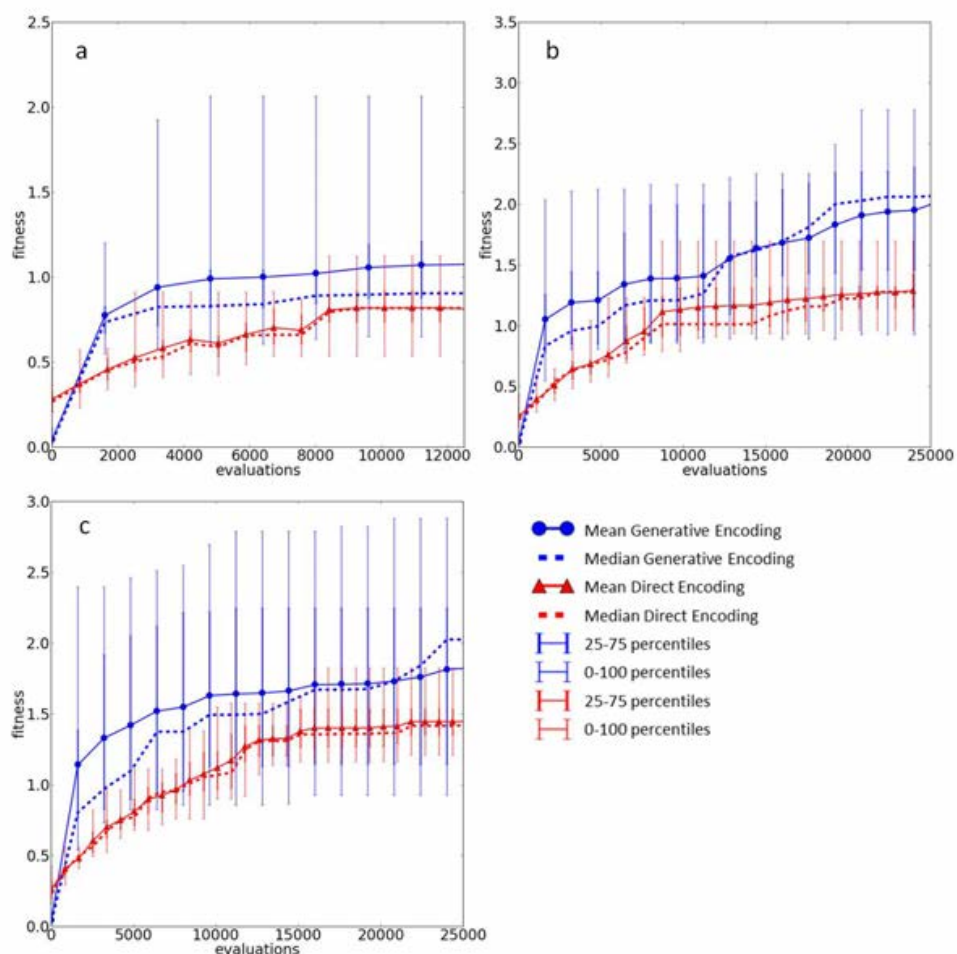


FIGURE 6.5: **Evolutionary progressions of the direct and generative encoding.** The above graphs display the average maximum fitness values of the different evolutionary runs when simulating a maximum of 5 modules (a), 10 modules (b), and 20 modules (c). The solid blue line marked with circles represents the average maximum fitness value of all of the runs of the generative encoding. The red solid line marked with triangles represents the average maximum fitness values of the direct encoding. The dotted lines represent the median of the two types of encodings. The thick error bars depict the 25th-75th percentiles and the thin error bars depict the 0-100th percentiles.

6.3.2 Phenotypes

Different distinct phenotypic behaviors emerged after a certain amount of evolutionary time. The direct encoding evolved various kinds of strategies,

though the generative encoding had evolved simpler, distinct types of locomotion due to the similarity in behavior seen in several modules. Caterpillar like behavior could be seen in evolved conglomerates that were composed of a single chain of modules (Figure 6.7b). A single chain of modules could also result in a different type of rolling locomotion, as seen in Figure 6.6a and Figure 6.6b. For some evolved robots, there was no apparent logic as to how they moved. Two robots tossed their weight around which resulted in complex rolling (Figure 6.6c, 6.7c and 6.7d), and one robot performed a crawling (Figure 6.7e) behavior. The types of behavior should become more evident when consulting the supplementary video ¹.

The constructive strategy of the direct encoding tends to add modules to the robot. This results in the best individual of all of the different evolutionary runs to be composed of 5 modules when simulating a maximum of 5 modules. In the case that the maximum number of modules is 10, 8 out of 12 runs have reached the maximum number of modules, and the average is 9 modules. When 20 modules are allowed, the average is 12.33, with a standard deviation of 2.87. In this experiment, the number of modules was limited by the fact that the excess of torque breaks the connections between the modules, which are heavily penalized by the fitness function.

All of the robots with a maximum of five modules developed similar morphologies, linear structures, with a rolling behavior. One of them being shown in Figure 6.6a. With a limit of 10 modules, branches in the structure of the robots appear. Despite the rolling behavior still being predominant, a crawling behavior can be found in some individuals (Figure 6.6b). When increasing the maximum number of modules to 20, some unspecified conglomerates of modules are found but most of the behaviors roll or crawl, as in Figure 6.6c.

In the generative encoding, there was a recurrence of simple friction-based phenotype (Figure 6.7a) when simulating a maximum of 5 modules. This friction-based phenotype seemed to exploit friction parameters of the simulator. Exactly half of the evolutionary runs that allowed for a maximum of 5 servo modules stagnated in a local optimum with this kind of phenotype. This phenotype was, moreover, found when simulating a maximum of 10 or 20 servo modules (Figure 6.5). The fitness values of these individuals were quite low, while other simple, more effective morphologies, such as the phenotype shown in Figure 6.7b, were possible to evolve. Considering the number of modules of the resulting phenotypes,

¹The video on Evolving Modular Robots can be found here: <https://www.youtube.com/watch?v=HCDftic1AdA>

the number of modules present in all robots was considerably smaller in the generative encoding compared to the direct encoding. The average number of modules in the best-evolved individuals of all runs was 3.5, 6.83 and 8.615 for the runs allowing 5, 10, and 20 modules max, respectively. Five out of 12 runs, when simulating a maximum of 20 modules, led to the evolution of morphologies composed of more than 10 modules; 10 out of 12 led to the use of more than 5 modules. Seven out of 12 runs simulating a maximum of 10 modules led to the evolution of using more than 5 modules.

In the generative encoding, not all genes of module states are represented in the evolved phenotypes. On the contrary, it seems that the evolutionary algorithm actively selects against the use of more modules. Out of all of the evolutionary runs, the runs with a maximum of 5 modules only evolved phenotypes with, on average, 1.33 expressed servo module states. The runs simulating a maximum of 10 modules had an average of 2.66 expressed servo module states, and the runs of simulating a maximum of 20 modules had on average 2.23 expressed servo module states. The phenotypes seen in [Figure 6.7d](#) and [Figure 6.7e](#) are examples of large phenotypes using only two types of servo module states.

6.4 Discussion

As can be derived from the graphs ([Figure 6.4](#) and [Figure 6.5](#)), there is a difference in performance between the generative and direct encoding. The most striking difference in performance can be seen in the initial phase of the generative encoding, where it outperforms the direct encoding. Over time, the direct encoding was able to catch up with the generative encoding and the performance differences were no longer statistically significant. The generative encoding still had an advantage in the long run when only a maximum of 5 modules could be simulated. This result was counterintuitive since it was expected that the generative encoding would perform better in the long run when more modules could be simulated.

A smaller portion of the genome of the generative encoding can lead to modular robots containing more modules than the direct encoding. The number of servo module states used in the generative encoding was 2.23 on average in the final evaluations of the evolutionary runs of the generative encoding. Since four servo module states could be stored in the genome it is noteworthy to see that not all genetic information is expressed in the phenotype of the generative encoding. This result illustrates the usefulness of reusing the genome for creating modular robot morphologies. Being able to evolve robots with just a few genotypic parameters is an advantage, and

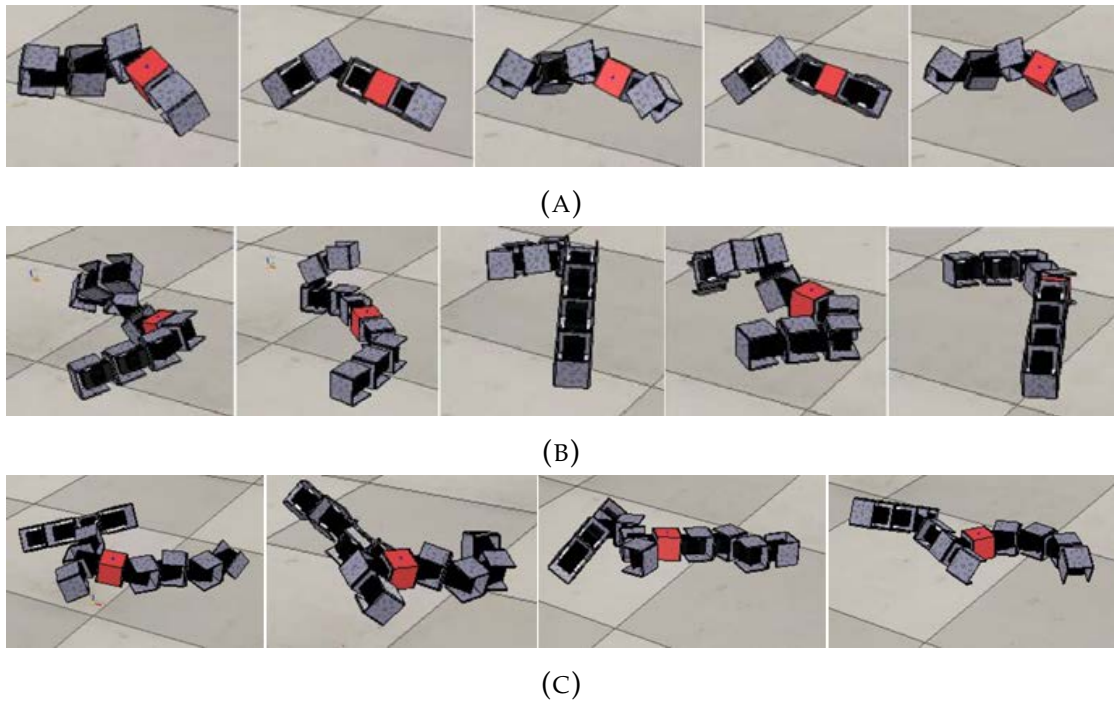


FIGURE 6.6: **Phenotypes obtained through the direct encoding.** Resulting phenotypes simulated with a maximum of 5 modules (A), 10 modules (B), and 20 modules (C).

it might lead to discovering abstract recursive mechanisms that are useful for the specified objective.

All evolutionary runs of the direct encodings led to phenotypes that utilized more modules compared to the generative encoding. This is due to a strong pressure in the direct encoding for adding new modules to the existing morphology. The mutations in the generative encoding can lead to destructive genotypes more quickly, potentially posing a limiting factor to the number of modules simulated for the individuals. Although the generative encoding outperformed the direct encoding, the generative encoding was still prone to premature convergence. This premature convergence was not seen in the direct encoding due to other evolutionary parameters that were used in the encoding. An improved version of the evolutionary algorithm could implement methods to increase diversity and evolvability, as done in speciation (Cook, 1906), as implemented in Neuroevolution of Augmenting Topologies (NEAT; Stanley et al. 2002a), novelty search (Lehman et al., 2008) or Age Layered Population Structure (ALPS; Hornby 2006; Hornby 2009). Regarding the L-system, an alternative

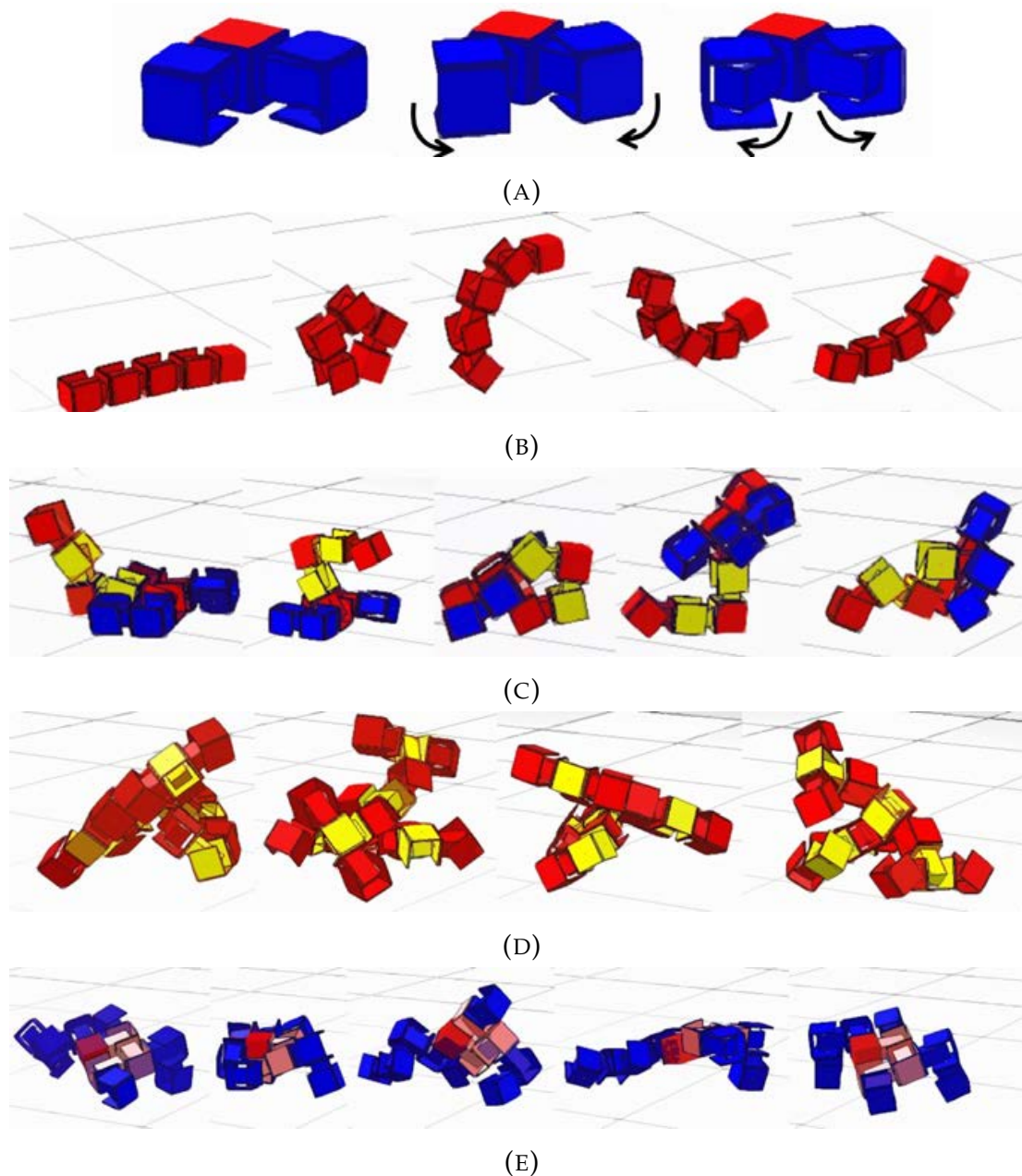


FIGURE 6.7: **Phenotypes acquired through the generative encoding.** Resulting phenotypes simulated with a maximum of five modules (A;B), ten modules (C), and 20 modules (D;E).

generative encoding, such as a Compositional Pattern Producing Network (Stanley, 2007), can be a relevant alternative generative encoding for evolving modular robots (as applied in Auerbach et al. 2015). Though as

a subjective interpretation, the L-system did evolve robots that seem to exhibit more generic behaviors that may well aid in their transfer to the real world (Figure 6.5).

However, in the scope of this chapter, the presented data is of limited use for robotic applications since it is not known how well the evolved behaviors transfer to reality. However, I expect that a hybrid approach of the two encodings would be a useful strategy to cope with the reality gap. The generative encoding can be used to evolve the global morphology and control of the robot, while the direct encoding would tweak morphological and control parameters online or in a feedback loop with the simulator. This would be beneficial since the generative encoding cannot locally change parameters specific to individual modules. Nonetheless, it might also be better to evolve phenotypes using the generative encoding and have an online learning system—such as a form of local decentralized learning (Christensen et al., 2013)—adjust the control of the modules accordingly.

The presented semi-homogeneous modular robot system presents a promising step in the direction of evolving feasible modular robots. Increasing the heterogeneity in the system would give us additional insight into how the modules can be better modeled in the future to produce even better robots. One could think of applying additional structural modules that have a variable stiffness. Since many organisms exploit various biomechanical attributes—be it elasticity, friction, or strength—adding this type of module can enable evolution to come up with morphological solutions (Pfeifer et al., 2005) and reduce the need for every part of the robot morphology to be actuated. Additionally, sensory modules can be implemented to extend the functionality of the system, giving the robot inputs to its control system. The products of evolution of these potentially evolved heterogeneous modular robots can become experimental platforms that can be consulted before designing and building a non-modular equivalent.

Text Box 6.5: Generic Phenotypes

When considering [Figure 6.6](#) and [Figure 6.7](#), the generative encoding, through looking at its more generic behaviors, are easier to see how they work. [Figure 6.8](#) depicts two phenotypes from the direct and generative encoding. Since the top robot does not use many modules to move, its workings are very easy to deduce. The locomotive pattern is likely harder to imagine for the direct encoding.

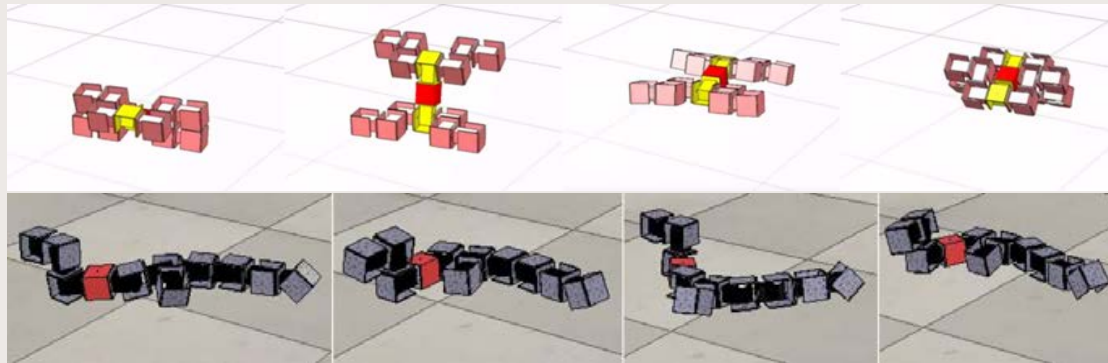


FIGURE 6.8: **Illustration of evolved phenotypes.** Evolved morphology using the generative encoding (top) and the direct encoding (bottom)

Curiously, the phenotype frequently evolved as depicted in [Figure 6.7a](#), considerably altered the performance of the generative encoding when being able to simulate up to 20 modules, as seen in [Figure 6.9](#). Being unable to evolve a new morphology was detrimental to the overall performance difference between the encodings that might have well changed the statistical significance if it did not evolve this phenotype.

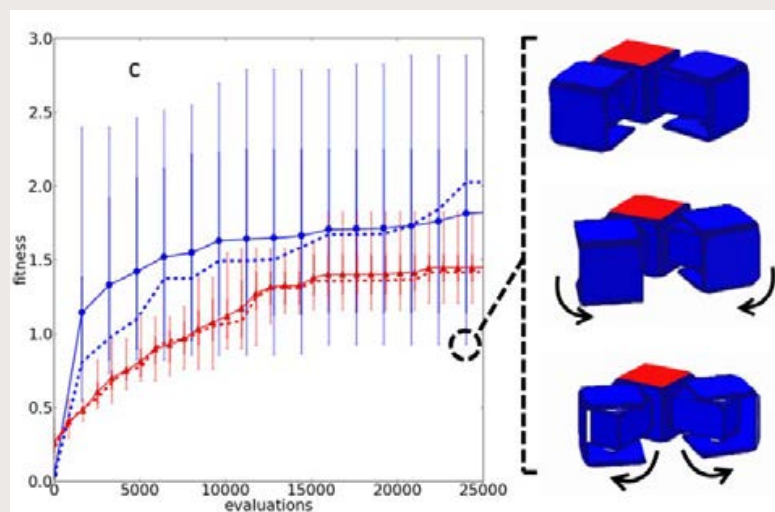


FIGURE 6.9: **Prematurely converged phenotype** The figure illustrates the low fitness solution at the end of the evolutionary run and its corresponding phenotype.

6.5 Conclusion

Much work in evolutionary robotics has been devoted to brain-body optimization strategies, though few studies consider the transferability of the evolved morphologies and control systems. This approach attempts to decrease this gap and enable researchers to quickly evolve and evaluate robots in simulation and in reality. The presented robotic platform that evolves conglomerates of modules has shown that a generative encoding, despite having less optimization freedom, is more effective for evolving locomotion in simulated robots. The reuse of genes in the generative encoding seems to work well for the evolution of robot morphologies and control. This is a great advantage when constructing a robot out of many modules since many of them can be assigned with the same control parameters. To facilitate both exploration and exploitation, I conceive that the generative encoding is able to evolve more abstract and simple robots, and suspect that a hybrid system would be ideal for experimenting with the reality gap of the evolved robots. This hybrid system can initially use a generative encoding in simulation, followed by a direct encoding that locally optimizes parameters in a real robot.

Chapter 7

Toward Energy Autonomy Through Evolving Plant-Inspired Modular Robots

I, on the other hand, am a finished product. I absorb electrical energy directly and utilize it with an almost one hundred percent efficiency. I am composed of strong metal, am continuously conscious, and can stand extremes of environment easily. These are facts which, with the self-evident proposition that no being can create another being superior to itself, smashes your silly hypothesis to nothing.

– Isaac Asimov, I, Robot

As stated in [Chapter 5](#), plants as primary energy consumers are the initial source of energy that accumulates in higher-order terrestrial organisms. This chapter explores how to design a robotic system with a similar objective as plants in a potential artificial ecosystem to act as primary energy producers. In this case, the robotics simulator, as implemented in [Chapter 6](#), was used, and the solar panel modules as described in [Section 4.4](#) were implemented. The experiments convey how modular robots can be optimized for energy autonomy and describes the potential applications of solar panel modules in robotics. The current modular robots make use of energy-intensive servomotors, but these can potentially be replaced with other, less energy-demanding structural modules. This chapter will have a more abstract hypothesis since there is not much comparable preliminary work that has used solar panels to evolve robots for optimal energy consumption. Hence, the title of this chapter begets the use of the word ‘Toward’. The hypothesis that was formulated to guide this chapter was therefore:

Hypothesis 4 *Energy autonomy in modular robots can emerge from implementing solar panel modules.*

7.1 Introduction

dec

Energy autonomy in artificial systems is beneficial for long-term autonomous behavior in single or multi-robot applications required for, for example, monitoring and exploration. However, it can be complicated to design energy autonomous systems since this depends on the energy demand and energy acquisition of the robot. In evolutionary robotics, locomotion and object manipulation are among the most prominent objectives for robots (Vargas et al., 2014). However, the same principles in evolutionary robotics can be implemented for energy autonomy in robotic systems. Energy autonomy is usually implemented on different robotic systems where the robot can utilize energy from light (Noth et al., 2006; Afarulrazi et al., 2011) or microbial fuel cells (Ieropoulos et al., 2003; Philamore et al., 2015). Being able to automatically design robotic systems that are geared towards energy autonomy could give us unintuitive solutions that might be more effective than traditional solutions. Since plants have mastered extracting energy from light in terrestrial environments, they are taken as a source of inspiration. Although plants have many unique features, we do not consider the implementation of many of them since this is either impractical or infeasible to implement in robotic systems. However, an abstraction of plant development is implemented in the form of a generative encoding (Subsection 7.2.3). Having a modular robotic system that conforms to energy optimization could give rise to self-reconfigurable robots that maximize energy acquisition.

The ultimate mechanisms that drive evolution in organisms are enacted through the external energy influx into an open system that innately works against its thermodynamic equilibrium. Considering the earth as an open system, this energy influx is mainly acquired from the sun in the form of light, and to a lesser degree in the form of heat and chemicals from Earth. Being able to acquire this energy from the sun to generate complex organisms is thus vital for evolution. The initial multicellular organisms that roamed the planet acquired energy from their environment in the form of chemicals and light (Reece et al., 2010). These carbon-based life forms were the precursors to plants that became the expert terrestrial organisms for acquiring energy from light. Although plants

appear to be slow, they are highly optimized for gathering energy from their environment. Many plants contain track and actuation mechanisms that optimizes their productivity by absorbing light more efficiently (Ehleringer et al., 1980). This tracking behavior emerges in the form of circumnutation through *heliotropism* (Graham et al., 2013). In *phototropism*, a form of heliotropism, the plant actively grows towards a light source, as in the case of the sunflower (Atamian et al., 2016; Kutschera et al., 2016). Some other species of plants can adjust their leaves or flowers to bend towards the light with a structure called the *pulvinus*. The pulvinus is a structure below the leaves and flowers that twists and pivots them through adjusting turgor pressure (Song et al., 2014). In general, leaves adjust their angle and move their surface perpendicular to the sun when conditions are optimal in a process called *diaheliotropism*. These are some general adaptive mechanisms through which a plant can adapt its morphology based on specific stimuli. Other morphological traits are hard-coded in the genome, such as *phyllotaxis* (Cells, 1997), the arrangement of leaves on a plant. Since the number of solar modules used in the robotic system is limited, as will become apparent in the next sections, no fair comparison can be made with the complex forms of phyllotaxis seen in plants. However, phyllotaxis is mainly driven by innate factors requiring limited feedback from the environment, which is similar to the open-loop control implemented in the generative encoding and evolutionary algorithm.

A mechanism exactly like phototropism is a feat that we are unable to implement in the robot since this would require some form of growth. This could, however, be accomplished with soft robots (Sinibaldi et al., 2014; Sadeghi et al., 2017; Heinrich et al., 2016; Vergara et al., 2017). Although allowing for continual growth shows promising bio-inspired applications in robotic systems (Sadeghi et al., 2014), this also brings forth difficulties regarding the reconfigurability and reuse of robotic parts. The modular robotics approach therefore does not allow continual growth, but enables reuse and reconfigurability. The bio-inspiration of the implemented robotic platform is motivated by the rotational movements of the pulvinus since this structure allows for movement of the leaves without major morphological change. With an open-loop control system, simple control mechanisms allow the optimal energy absorption. While artificial plant systems have been implemented in cellular automata (Hogeweg, 1988; Balzter et al., 1998) as well as virtual creatures (Zamuda et al., 2014; Zahadat et al., 2016; Veenstra et al., 2016; Corucci et al., 2016), they have rarely been investigated in a three-dimensional embodied approach other than light-tracking solar panels (Prinsloo et al., 2015). Many evolutionary robotics experiments have focused on acquiring behavior typical of consumers

(Sims, 1994b; Pfeifer et al., 2006; Vargas et al., 2014), whereas this chapter looks at how primary energy producers can evolve in artificial systems. Some robotic platforms have been designed to cope with energy autonomy (Greenman et al., 2003; Ieropoulos et al., 2003; Philamore et al., 2015) although research in this area is still limited. The evolutionary system, instead of modeling nature, aims at implementing feasibly evolved designs into real modular robots. The modules are therefore based on a heterogeneous modular design, a robotic module being an independent unit that encapsulates part of its functionality (Stoy et al., 2010). The modular robotics approach eases the construction process of different morphologies, as well as the capability of changing the morphologies on the spot. By simply implementing the same connection mechanism on each module of the modular robot, different modules can be joined together to form unique and feasible robot phenotypes. The platform thus allows for light-absorption in a robotic system that implements actuator with similar degrees of freedom as the pulvinus with the aid of a plant-inspired developmental algorithm.

7.2 Material and Methods

A simulation environment is used to evolve modular robots, and the evolved phenotypes are transferred to the real world. The simulated robots were optimized for harvesting energy from light using simulated solar panels. The physical robot simply follows the parameters that have been evolved by the simulator. Different environments were simulated that in turn shaped the search space of the robot. Moreover, five different types of modules were designed that could be simulated. These modules could all be connected using the same connection mechanisms. Enabling morphological change and open-loop evolution of the control system allows the platform to evolve unique robot phenotypes that can be transferred to reality. The simulated robots could never use more modules of a type than was maximally allowed. Every evolved robot could thus be created in reality. Varying the number of solar panels that we can implement in the robotic system gave us an idea on the number of modules we require to evolve sufficient robotic phenotypes. A modular robot was constructed by attaching one module to another module manually. Since the connection mechanism is based on magnets, the construction process is a simple snap-on procedure.

Virtual Robot Experimentation Platform (V-REP; version 3.4.0; Rohmer et al. 2013) was used as the robotics simulation platform and the

evolutionary algorithm was implemented as a C++ based DLL plugin. The simulation environment consisted of a default floor and was simulated using the bullet dynamics engine (version 2.78). The dynamics settings were set to accurate (default) with a timestep of 50ms. The modules used for the simulator are based on the physical properties of the real modules (Figure 7.1), approximating similarity in size and weight. The modules consisted of a base module, a servo module, a cube module, and two types of solar modules, as is discussed in the next section. All modules contained connection sites that could be either male or female. The pairing of these two connection sites established a connection between two modules, allowing for the composition of a modular robot (Subsection 7.2.2). A list of the materials used to construct these methods can be found on our website.¹

7.2.1 Modules

Five different types of modules were used: a cube module (Figure 7.1a), a base module (Figure 7.1b), a servo module (Figure 7.1c) and two types of solar modules. One of the solar modules simply contained two solar panels that were joined (Figure 7.1d). The other, more elaborate, solar panel module (flower module) was designed with the aim of allowing for more plant-like adaptive behavior in the system (Figure 7.1e). However, the increased complexity of this flower module also brings about increased complexity in the simulator. Therefore, this flower module was not used in the experiments described in this chapter, but rather serves the role of informing the reader about potential future implementations of solar modules. Throughout different modules, all custom parts were 3D printed using polylactic acid (PLA).

The base module (Figure 7.1b) was composed of a simple custom structure with three female connection sites. Three female connection sites were used to reduce the number of possible connection sites limiting the search space of the robot. However, more connection sites could make more sense in a future implementation of the base module, though this makes the state space landscape more convoluted due to more possible configurations of the modules. This base module was simulated statically, meaning that its physical properties were not simulated. Instead, the module was fixed in place, ensuring that the structure stayed in the same position conforming to the sessile nature of plants. The three female connection sites were 55 by 55mm and were placed next to each other at a 45 degree angle. The

¹List of Materials can be found on our website at: <https://sites.google.com/view/emergemodular/projects/energy-autonomy>

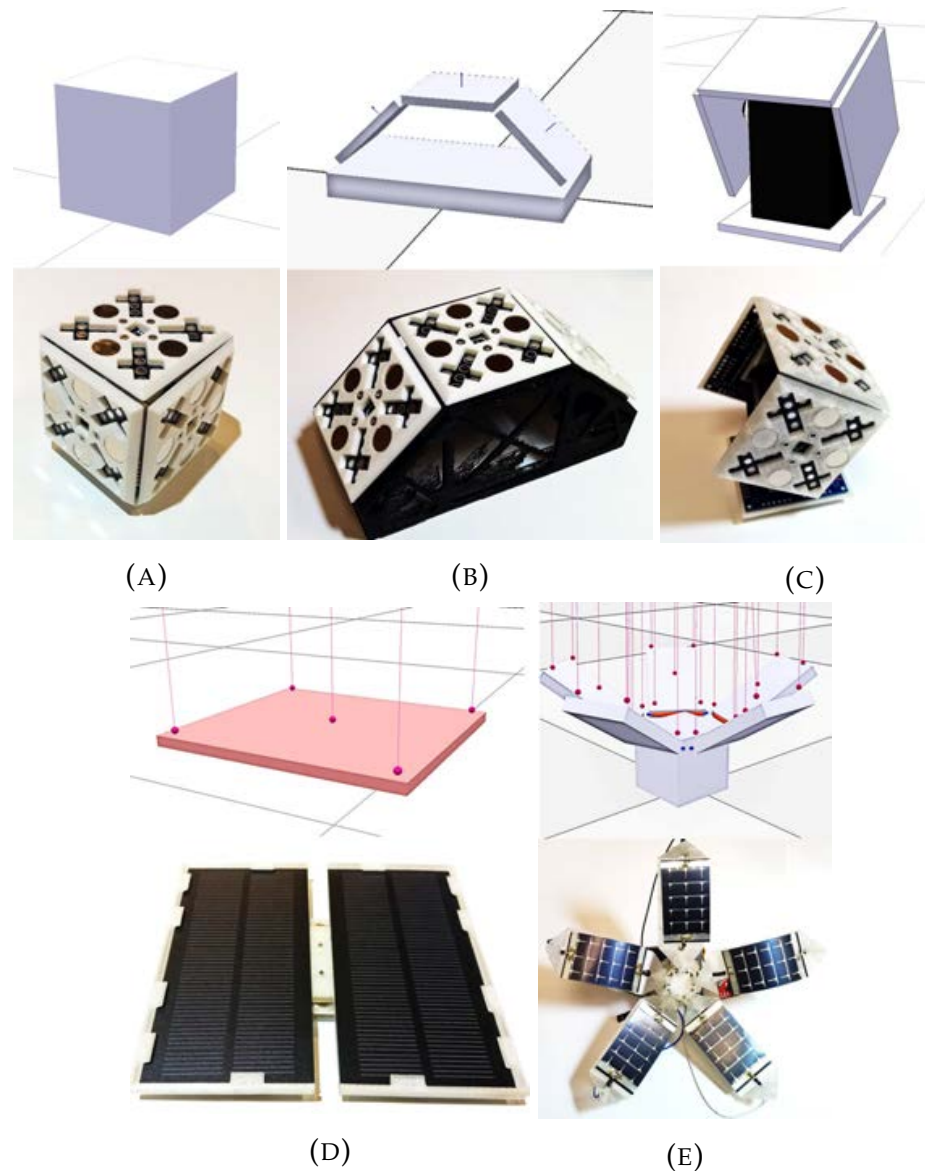


FIGURE 7.1: **The modules used in the simulated and physical robot.** The cube module (A), base module (B), servo module (C), solar panel module (D) and flower module (E).

cube module (55mm by 55mm by 55mm; weight is 300g) was used as a structural building block for the modular robot to which other modules were attached. This cube had five female connection sites (top, right, left, front, back) and one male connection site (bottom).

The servo module (80mm by 55mm by 55mm, weight is 160g) is based on a module from the EMERGE modular robotic platform (Moreno et al.,

2017). This module contains three female connection sites, attached to the shaft, and one male connection site on the bottom, attached to the chassis of the servomotor. It houses a dynamixel AX-18a servomotor. In the experiments, the range of the servomotors was limited to $+90^\circ$ and -90° . The difference of the servo module used in this chapter is the attribute of having four data channels, compared to the two data channels implemented in the EMERGE module. In addition, infrared sensors that were present in the EMERGE module are not implemented in the servo modules used. Infrared sensors were not implemented since they take up additional space on the PCBs and consume additional energy. At the location of the infrared sensor in the emerge module there is simply a 3mm-wide hole that can be used to adjust a screw to tighten the connection between the servo bracket and the servomotor. This tightening can in turn lead to a stiffer joint that is harder to actuate, but is also harder to move passively. This is beneficial to adjust on the spot since we do not want an inactive servo joint to buckle under the weight of the robot. Servomotors closer to the base of the robot can therefore be tightened more while those at the end can be looser. The dynamixel AX-18a servomotor is controlled via a PID controller that is directly connected to the robotics simulator.

The first type of solar panel module (referred to as solar module) implemented two simple solar panels (141mm by 124mm by 6mm, weight is 66g) on a 3D printed bracket (Figure 7.1d). Two 0.8W, 5.5V monocrystalline silicon solar panels (118mm by 63mm) were mounted on the case of the solar module. These two solar panels were connected in series and could output a voltage of 11V (Figure 7.2a). An additional diode was attached to the solar module to prevent current flowing in the opposite direction. The current acquired from the solar panels was converted to 12V using a step-up voltage regulator (not shown in the figures) to ensure a 12V output. This output voltage was connected to two data channels, connected to the base module. The solar module was simulated with a basic morphology with similar dimensions and contained five proximity sensors to determine light absorption. These proximity sensors were positioned at each corner and in the middle of the solar module. Each proximity sensor served as a ray tracer simply determining whether there was an object in-between the starting position of the ray and a light source. Additionally, the difference between the z-directional vector of the ray compared to the orientation of the solar panel gives us the angle at which light impacts the solar panel. This angle of attack linearly influenced the contribution of light absorption to the fitness of an individual.

The flower module (Figure 7.1e) is composed of five SP3-37 flexible solar panels. Each panel is mounted on a 3D printed petal. These artificial

petals could not bend, making the use of the flexibility in the solar panels redundant. The flexible solar panels thus effectively function the same as a regular solar panel. The five petals are connected together through a system of cranks and a small circular platform. A rack and pinion system combined with a small servomotor pushes the platform and actuates the petals. Two hinge joints connect the circular platform to the petals, one at the inner edge of a petal (center of the flower) and closer to the middle of the petal. A hinge joint on the edge of the petals is attached to the platform containing the rack, while the hinge joint closer to the middle of the petal is connected to the piston. When the piston moves the rack up and down, the petals actuate and open or close the flower, respectively. The petals can open to a certain degree and this position can be optimized depending on the solar intake. The energy harvested is stored in a LiPo battery. This battery was used to power the servomotor actuating the petals, making the flower module (not the modular robot) energy-autonomous. Though the flower module is not used in the evolutionary runs, it can also be connected to the same power grid of the modular robot to charge the 12V battery. Additionally, the flower module contained a MMA842Q accelerometer (3-axis), which enables the flower module to detect movement. This sense of movement can inform the flower module whether it should close or open its petals under harsh or favorable conditions. For example, at night, during heavy rain or in windy scenarios, it would be better for the flower module to close and protect its solar panels. Especially if the solar panels can unfold in an origami-like manner (the original intend of implementing the flexible solar panels), the surface area would be relatively weak, highlighting the need for the flower and solar panels to close. The flower module contained five solar panels and five shapes were used to simulate them. Similar to the solar module, each solar panel simulated five proximity sensors that were used to measure the light absorption. This led to the increased computational requirements since 25 proximity sensors and six shapes were used in the flower module. Therefore, the flower module was excluded in the robot simulator.

One base module, one cube module, eight servo modules, five solar modules and two flower modules were created in reality, and the simulator was thus restricted to use this number of modules. This limits the potential resulting phenotypes of the evolutionary algorithm and constrains the search space, but enables evolution of feasible modular robots. A direct feedback from simulated modular robots and actual modular robots is thus present. All of the morphological parameters of the individual modules were fixed to represent their physical counterparts. The eventual physical modular robot was constructed based on the phenotypes that evolved in

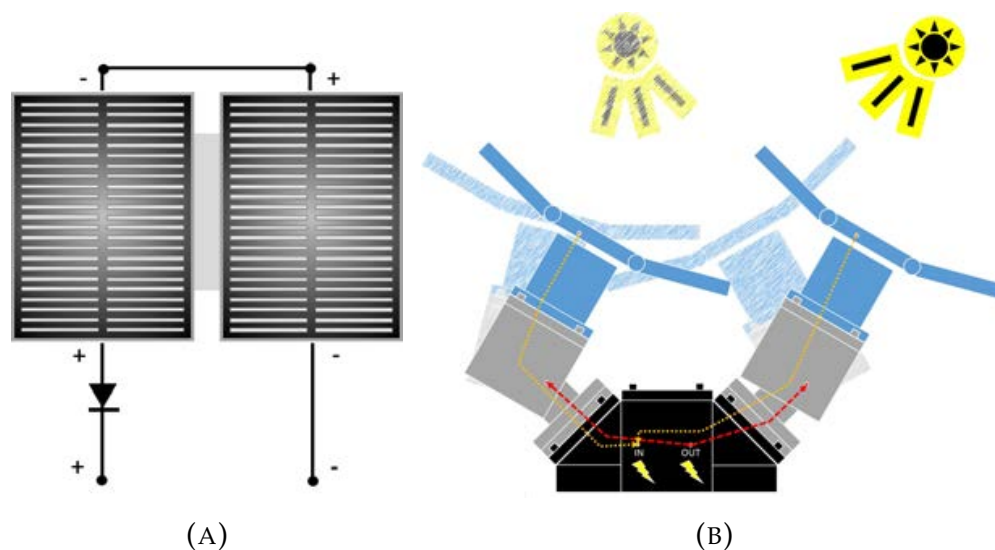


FIGURE 7.2: **Schematic diagram of the modular robot containing solar panels.** The Solar panel module consists of two solar panels that are connected to one another in series 7.2a. A diode is attached to prevent current from flowing in the reverse direction. As can be seen in 7.2b the solar panel modules pass power down to the base module where the battery is located.

the simulation environment. An implementation of the dynamixel protocol (protocol used to communicate with the servomotors) in the plugin allowed for the direct control of the actual robot in the simulation environment. The dynamixel protocol utilizes serial communication to communicate with the dynamixel AX-18a servomotors. Through sending instruction packets, individual motors can be controlled using a single communication bus. The values of the dynamixel servos needed to be transformed into hexadecimal values that indicate parameters such as the desired speed and position of the servomotors. The solar modules implemented in the evolutionary algorithm did not have any type of actuation. Connecting a few modules together to form a robotic entity enabled current to flow from a power source towards the servo modules while also allowing for the flow of current from the solar panels to the power source (Figure 7.2b). The solar modules could directly charge a 12V lithium ion battery that was used, though the efficiency of charging the batteries greatly depended on the light saturation of the solar panels.

7.2.2 Connectivity

The real modules are connected to one another via magnet-based connection sites, as was presented in [Figure 4.6](#). Two of the data channels implemented in the PCBs were used to enable a current to flow from solar panels back towards the initial power source. Three-dimensionally printed hulls housed four cylindrical NdFeB (neodymium, iron and boron) magnets (12mm diameter, 3mm depth). Two magnets could hold one another together with a force of roughly 13.5N. The male connector hull has protrusions that ensure the fit and connection of the male connector to the female connector. The connectivity of the modules in the robotic system is depicted in a block diagram ([Figure 7.3](#)). Two of the channels were dedicated to power and ground, while the four other channels routed data channels. Two of these data channels were used to enable a current to flow from solar panels back towards the initial power source.

In the simulated modules, a force sensor was put in between two connection sites to detect the torque and force between two modules. Since the modules have not yet disconnected in the physical robot, the modules could only break from one another if 100 consecutive threshold violations occurred where the threshold was set to 10 000 N m of torque and 80 000.0 N of force. The chosen values are incredibly high to ensure that the modules stay connected to one another except in the case of a faulty collision. When the simulated robot simulated modules that inaccurately collided, disconnection between modules could occur. These inaccuracies did not occur in the eventual simulator. As a failsafe, the fitness value of individuals that contained broken force sensors was set to 0. The construction of the modular robot occurred before the simulation started. A generative encoding ([Subsection 7.2.3](#)) translated the genome and created the robot phenotype. This genome consisted of simple morphological parameters and an additional neural network ([Section 7.2.4](#)) for the servo modules.

7.2.3 Generative Encoding

The generative encoding was based on a context sensitive Lindenmayer-system (L-system; Lindenmayer 1968a; Lindenmayer et al. 1992)—a parallel rewriting system—as implemented in [Chapter 6](#) (Veenstra et al., 2017a). The variables of the L-system represented the different modules of the robot. A variable describes a *state* that is decoded as the morphology, control and attachment rules of the modules. In the simulation environment, these modules represent the cube, base, servo and solar modules. One state was dedicated to the base module and was the axiom

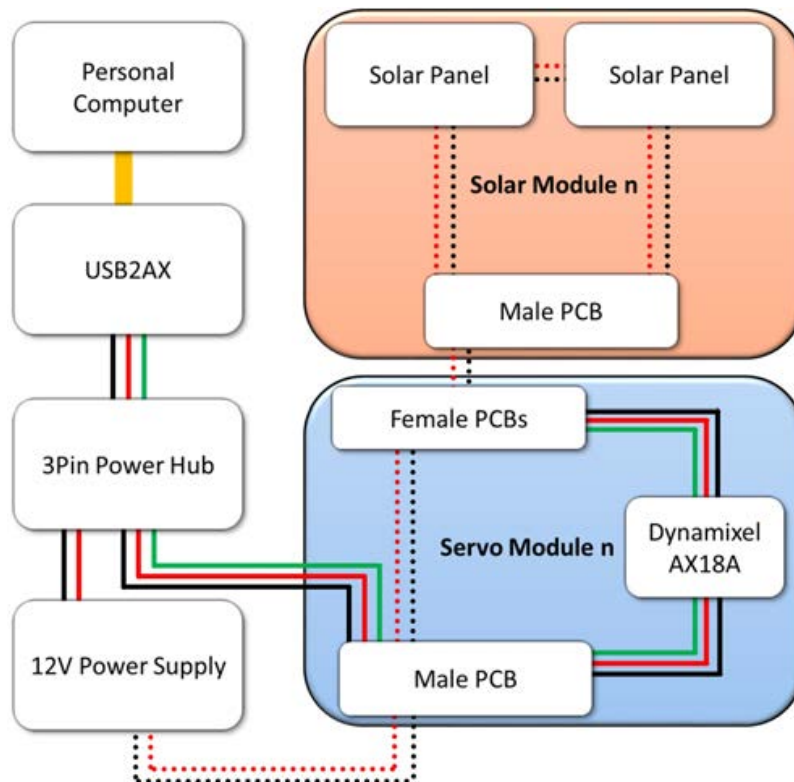


FIGURE 7.3: **Block diagram of the modular robotics platform.** A terminal is connected to a central 3-pin power hub via a USB2AX interface. A 12V power supply is also connected to the same 3-pin power hub. The power hub is in turn connected to a male or female connector face of the modular robot. This connection distributes power to all connected modules. The block diagram only shows the Servo and the Solar module. The base and cube module simply transfer all electrical current from their male connection sites to their male connection face and *vice versa*. Two of the channels are connected separately to the solar panel modules which transfer power from the solar panels to the power source. The red and black connections indicate power and ground while the green connection represents the communication wire. The dotted lines represent the power and ground connections to the solar panels.

of the L-system. Another state represented the cube module. Four states represented the servo module and two states contained the parameters of the solar module. Every state can assign unique attachment rules and control parameters. The generative encoding does not necessarily lead

to all genotypic parameters to be expressed in the phenotype. Instead, some genotypic parameters could be dormant, which allowed genetic drift to occur across generations. For example, eight servo modules could be generated in the phenotype if one servo state is expressed eight times in the phenotype, but this phenotype could also be the result of one state being expressed six times and one state being expressed two times in the phenotype. The phenotypic representation of the modules is determined by the attachment rules of the L-system that were stored in the genome. The attachment rules of the modules included the information of which module is connected to what connection face in which orientation. The L-system was moreover context-sensitive since a module cannot be placed at an attachment site if another module already occupies it. Modules cannot be created if this causes a collision with other created modules. The different modules were colored in the phenotype based on their states so that recursive expression could easily be detected. Five iterations of the L-system were done to create the robot phenotypes, starting with the base module as the axiom.

7.2.4 Evolutionary Algorithm

The implemented evolutionary algorithm was based on a steady-state genetic algorithm (Syswerda, 1991). The simulations were limited to 46,000 evaluations per evolutionary run. 46,000 evaluations were chosen as a trade-off between computational time and performance. A population size of 92 individuals was used and was simulated for 500 generations. The population size is a multiple of the 23 simulation instances that ran in parallel on a cluster node containing 24 computing cores. One core was dedicated to running the evolutionary algorithm itself, while the remaining cores evaluated individuals in V-REP. For each experiment, 12 evolutionary runs were performed with different initial seeds. The evolutionary algorithm was generational and randomly selected one parent from the population to produce an offspring. The initial population consisted of individuals created from randomized genomes. Though the offspring were haploid, a crossover function allowed certain states of another parent to be transferred to the offspring with a 20% chance. Up to eight states were simulated. After crossover occurred, the offspring were mutated with a morphological mutation rate of 0.15 and a control mutation rate of 0.1. The morphological mutation accounted for any aspect of the generative encoding replaced by a random value with a 15% chance. Only the four states of the servo modules contained a neural network, and the neural network had several mutation operators that could be activated with

a 10% chance. There were four mutation operators working on the neural networks: *change connectivity* that altered the edges between the neurons: *add neuron*, *remove neuron*, and *change neuron* swapping an interneuron for a new interneuron of another type (Section 7.2.4). The maximum amplitude of the servo arm ranged from -90nd +90

Neural Network

In a previous experiment, simple sinusoidal wave functions were implemented to control simulated servo modules (Veenstra et al., 2017a). The same sinusoidal patterns are implemented here, though they are implemented in a network of neurons. This artificial neural network, implemented in the servo modules, consists of one input neuron, up to six interneurons, and one output neuron. The input neuron is always activated. A recurrent interneuron layer consisted of neurons that simulated a fixed sinusoidal output pattern or a neuron with a binary step function. The phase, amplitude and frequency of the sinusoidal neuron could be altered, and the mutable parameters of the binary step neuron were the threshold value (between -1.0 to 1.0) and the output weight (-1.0 to 1.0). The output neuron simply outputs a value between -1.0 and 1.0 based on its inputs. This value is in turn transformed into a value that represents the absolute position of the servomotor of the module. In this neural network, the connections were not weighted; instead, once a neuron is activated, all of the neurons connected to the activated neuron receive the same output. This was implemented due to the small size of the neural network (one input and one output) and to limit the search space. A neural network could be implemented in each state of the servo module. This means that multiple servo modules could express the same neural network if the generative encoding created these neural networks from the same gene.

Fitness function

The goal of the simulated robots was to absorb light within five seconds of simulation time. The simulation time was limited to five seconds since the modules were able to appropriately adjust the positions of their joints to the light source within the given simulation time. Approximating an actual day cycle in the physical world depends on the transformation of the simulation time that can be stretched according to a given environment. The fitness of each individual was determined by the amount of light that was absorbed by each ray of all the solar panels present on the robot, and can be derived from Equation 7.1. The gathered light was calculated by subtracting the

z directional vector of the z-axis of each ray on the solar panel from the z directional vector of x-axis of the solar panel. An additional cost function was added to represent an arbitrary energy expenditure of the robot.

$$F := \sum_{i=1}^m \rho_i - \sum_{j=1}^n \epsilon_j \quad (7.1)$$

F represents the fitness value obtained by calculating the total amount of light absorption that occurred within the 5 seconds of simulation time. The ‘:=’ operator represents an update of the fitness value with right hand side at each timestep. ρ represents the amount of energy gathered by each ray i at every timestep. For each servo module j , the amount of energy ϵ used by all servo modules is subtracted. An additional cost function was added to compensate for modules that were disconnected due to the breaking of a connection site (not shown in equation). Breaking of modules never occurred in experiments, though the initial fitness cost was implemented to ensure we did not reward malfunctioning robots.

7.2.5 Experimental Setup

The evolutionary runs were performed in four different environments (Figure 7.4). The first environment contained a light source located directly above the modular robot. The second environment consisted of four walls surrounding the modular robot constricting direct outward growth of the phenotype of the modular robot. The third environment contained an object that blocked the phenotype from receiving direct light absorption. This was done in to motivate outward growth of the modular phenotype. The last environment consisted of a moving light source that mimicked the trajectory of the sun in winter in the northern hemisphere of Earth. It was expected that the last environment would promote the evolution of blind control systems that enabled solar panels to tilt towards the sun. The position of the light source was calculated by two sinusoidal functions that were transformed into Cartesian coordinates (Equation 7.2). The ‘:=’ operator represents an update of the left hand side variable with the term on the right hand side as it is performed at each timestep.

$$P_x := P_{x_start}\alpha_x + \tau\alpha_x P_y := \sin(\tau)\alpha_y P_z := \sin(\tau)\alpha_z \quad (7.2)$$

α represents a specific scaling factor for transforming the position in Cartesian coordinates. The x y and z positions of the light source are updated at each timestep denoted with τ .

In the environment with the moving light source, the energy cost was a custom value. This cost value was either 0.0, 0.1, 0.5, 2.0 or 8.0. For every energy cost, 12 evolutionary runs were done. These values were chosen as they changed the evolutionary trajectory of the different evolutionary runs whereas above a cost of 8.0, no change in the evolutionary runs could be seen. Additionally, for each simulated environment, twelve evolutionary runs were performed when simulating a maximum of one, two, or five solar modules. This was done to see whether movement evolves in environments where a different number of solar panels can be simulated. It was expected that when simulating a maximum of one solar panel, on average, more angular movement would be detected in the solar panels compared to simulating more solar modules. This hypothesis was tested by measuring the difference in angular movement of each solar panel at each timestep. This arbitrary measure of angular movement is stored together with the fitness values of the individuals.

7.3 Results

The results shown in this section are separated by the resulting phenotypes of the eventual population of robots in different environments and the results on the impact of movement and energy cost on evolving the modular robots. The phenotypes give a clear overview of the types of robots that could be evolved, while the energy costs explain emergent behaviors seen in the evolved robots.

7.3.1 Phenotypes Evolved in Different Environments

The four types of environments led to major differences in evolved phenotypes (Figure 7.5). The individuals in the environment with the stationary light source evolved simple morphologies where all solar panels were pointing upwards avoiding, collision with one another. The individuals in the constrained environment were more difficult to evolve, as can be seen in the graph depicting the evolutionary progressions (Figure 7.8). In particular, only three individuals in the last generation of the evolutionary run displayed phenotypes that were different from the top right and bottom right individuals seen in Figure 7.5. It can therefore be said that the search space of this environment is much more rugged than the search space in the other environments, making it harder for the evolutionary algorithm to escape a local optimum. As expected, in the environment where the light source was blocked directly from above, the

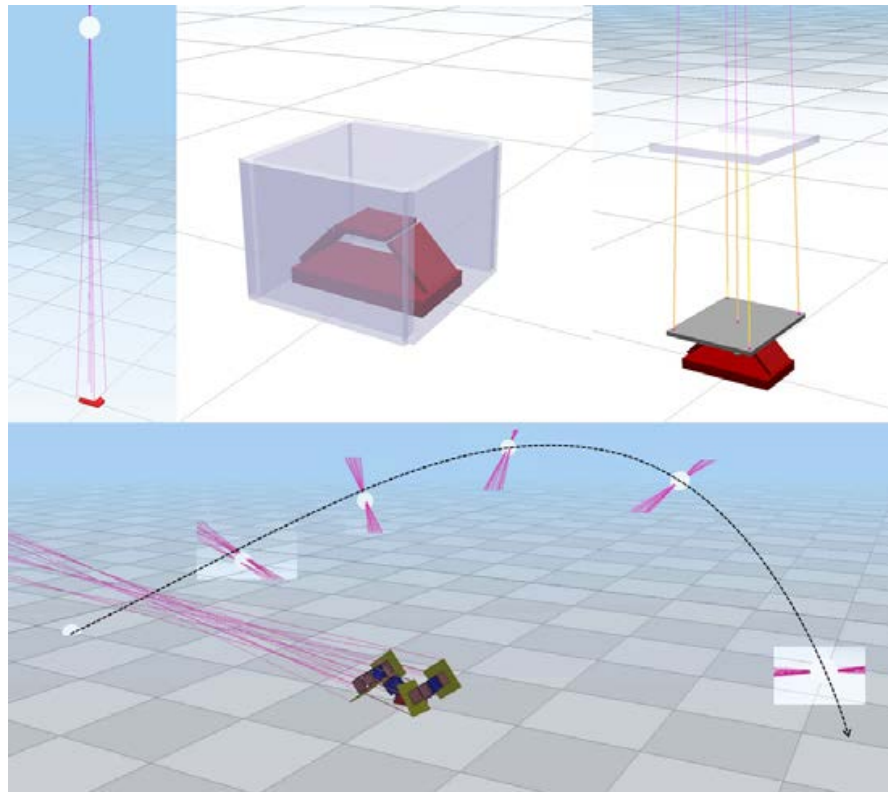


FIGURE 7.4: **Different simulated environments.** Environment with stationary light source (top left); an environment where the initial base module is constrained (top middle); an environment where direct light is blocked by an object (top right); and an environment where the light source is moving in an arc (bottom). The line represents is an approximation of the path the light takes.

solar panel modules were located on the outside of the modular robots. Some of these evolved phenotypes were easily implemented in the real world since they did not actuate any servo modules (Figure 7.11). As can be seen in Figure 7.7, the eventual population of the modular robots that evolved in the environment with the moving light source were also easy to transfer to the real robot. Although it is a conceptual model, the modular robot that incorporates the flower modules is depicted in Figure 7.10.

7.3.2 Movement in Energy-Harvesting Modular Robots

As could be seen in the simulation environment with the moving light source, movement can increase the fitness value of evolved robots as compared to the evolutionary runs if the cost of moving is low enough

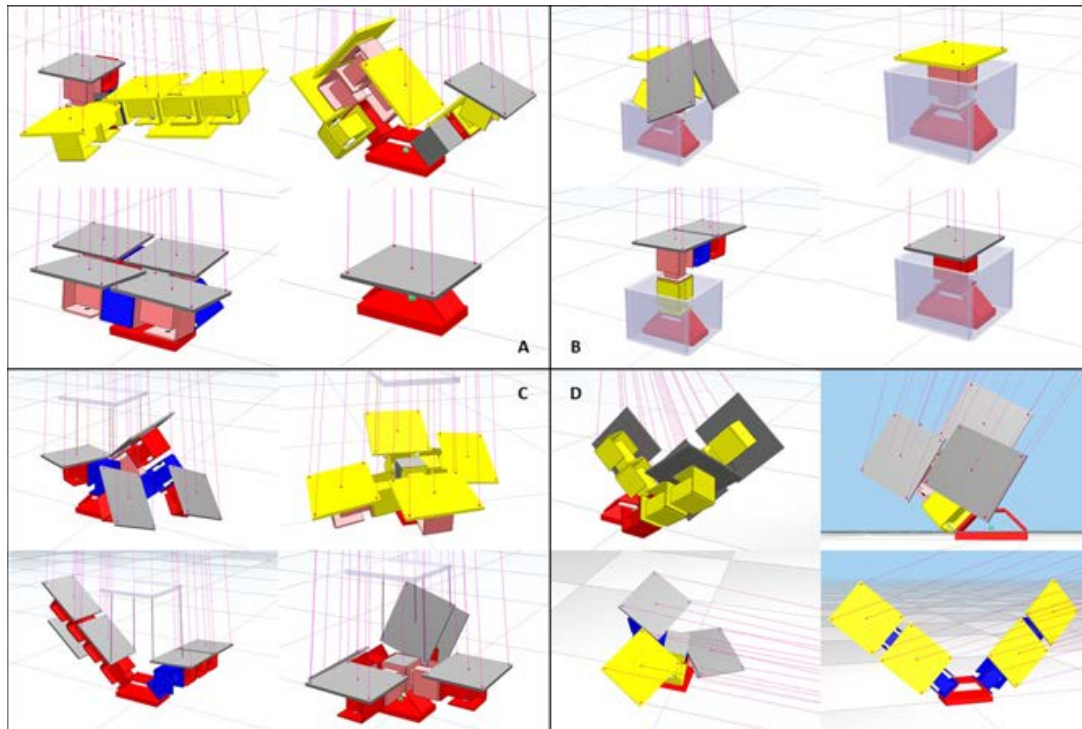


FIGURE 7.5: **Different evolved phenotypes.** Four individuals of each simulation environment are shown. (A) The top left individuals were evolved in the environment with a stationary light source and no objects in the environment. (B) The top right individuals were evolved in an environment where the surroundings of the initial module were blocked by four adjacent wall structures. (C) The bottom left depicts individuals evolved in the environment where the light source was blocked by an additional structure. (D) The bottom right individuals were evolved in an environment where the light source moved.

(Figure 7.9). A statistically significant difference can be seen when comparing the average movement angle of all solar panels when comparing maximum and minimum energy cost. The difference in fitness for a high energy cost versus a low energy cost for movement is highest when simulating one module and lowest when simulating five modules. Based on a Mann-Whitney U test, we can see that there is a significant difference in the angular movement of the leaves when a large energy cost is applied when simulating a maximum of 1 (p-value: 0.0000779), 2 (p-value: 0.0000300), and 5 (p-value: 0.0050966) solar panel modules. The difference in angular movement between simulating a maximum of 1 and 5 solar

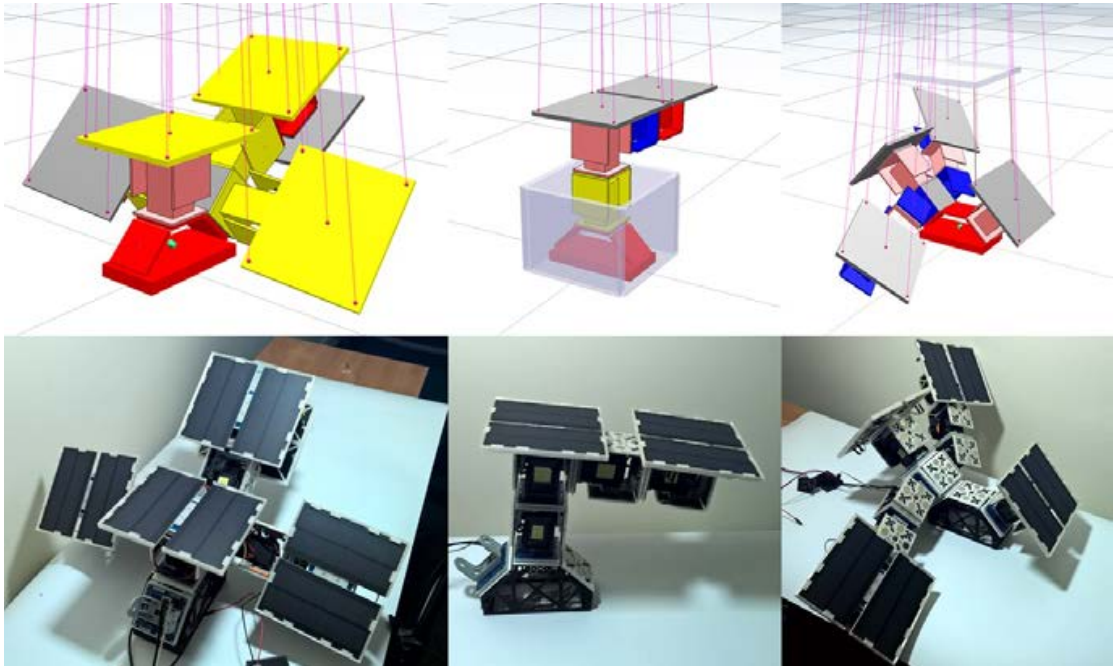


FIGURE 7.6: **Evolved phenotypes in different environments.** Simulated (top) and real (bottom) modular robots evolved in the open environment (left), the constrained environment (middle) and the environment where the light source was blocked.

modules without an energy cost was also significant (p-value: 0.0030495). Similarly, the difference in angular movement when simulating a maximum of 1 and 5 solar modules was significant, with the highest energy cost (p-value: 0.0070689). When comparing the maximum of 2 solar modules with the other maximum number of solar modules no significant difference could be found. This is due to the angular movement of the solar panels when simulating a maximum of 2 panels in-between simulating 1 and 5 solar panels. These differences indicate that less modules lead to increased movement of solar modules. Based on these results, the number of solar modules directly influences the amount of movement in the solar panels.

7.4 Discussion

The aim of this chapter was to provide a deeper understanding of how evolution of modular robots could shape robotic entities towards energy autonomy in different environments. The resulting phenotypes of each environment indicate that the simulated individuals adjusted

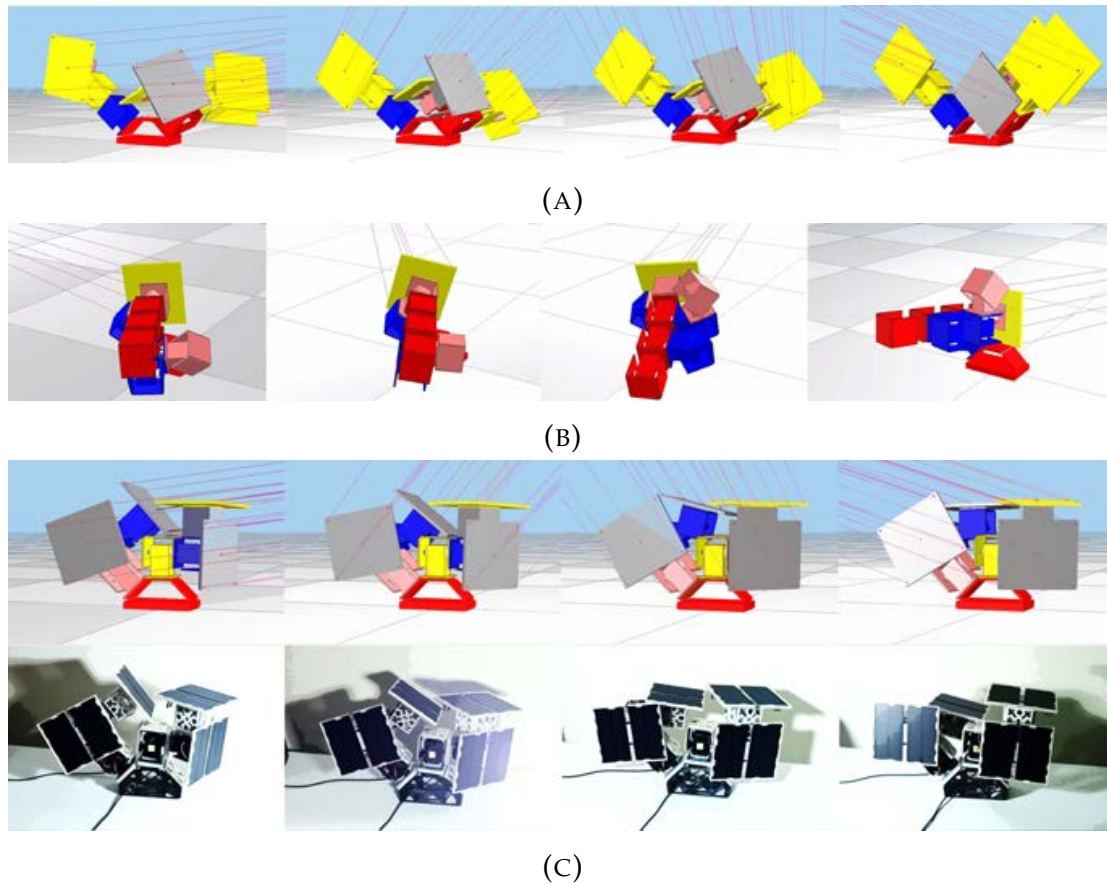


FIGURE 7.7: Various evolved phenotypes in the environment with the moving the light source. Simulating five solar modules (A), a maximum of one solar module (B), and a four solar module phenotype that has been transferred to the real world (C). The camera stayed in the same location after each subsequent picture.

differently to optimize for light absorption. Some of the simulations led to solutions in local optima, indicating a rugged search space for the evolutionary algorithm, as was the case of the constrained environment. The other environments typically evolved more varied and more complex phenotypes. This demonstrates that our simulator is versatile in creating novel robots using a fixed number of modules in different simulated environments, though improvements to the evolutionary algorithm as well as to the encoding, could be made.

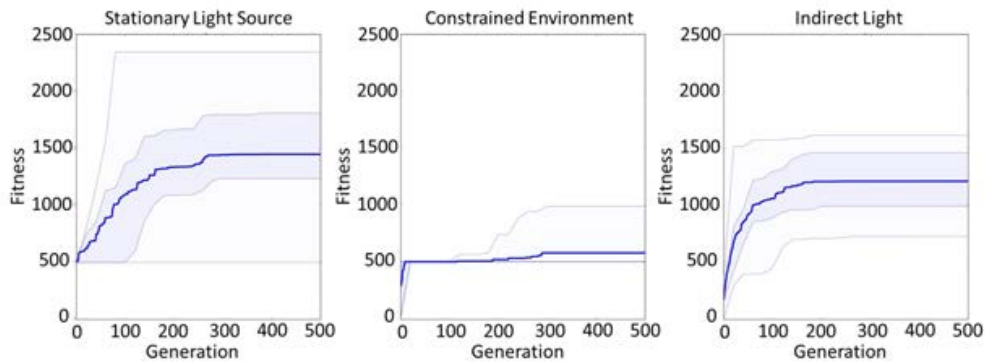


FIGURE 7.8: **Evolutionary runs conducted in the different environments.** The three graphs depict the individual runs where bold solid line represent the average maximum fitness value all evolutionary runs. The lightly colored surrounding area represents the 25th-75th percentiles while the less intense colored surrounding area represents the 0-100th percentiles. The evolutionary runs are shown of the environment with a stationary light source (left), the constrained environment (middle) and the indirect light environment (right).

7.4.1 Light Tracking

When movement was costlier and when more solar panels were available to be implemented in the modular robot, I expected to see a relative decrease in angular movement of the leaves when compared to runs without implementing cost of movement and simulating less solar panels. Our results indicate that when less solar modules were allowed in the evolutionary runs, more movement could be observed in the phenotypes that evolved. The reason that this difference occurred was due to the increased search space of simulating more than one solar module, as well as the constrained available positions for solar modules when simulating a limited number of modules. Our results indicate that having more than one solar panel does lead to robots tilting their solar panels towards the light source. However, when only one solar panel was being used, the robot evolved more movement, and movement has a clear evolutionary advantage when it tilts the solar module towards the light. Similar to Veenstra et al. (2016), evolution does not necessarily pick up movement of solar panels, even when there is no cost for movement attributed to it. A significant difference could, however, be seen when a large cost was implemented for movement in the modular robots. Having more solar

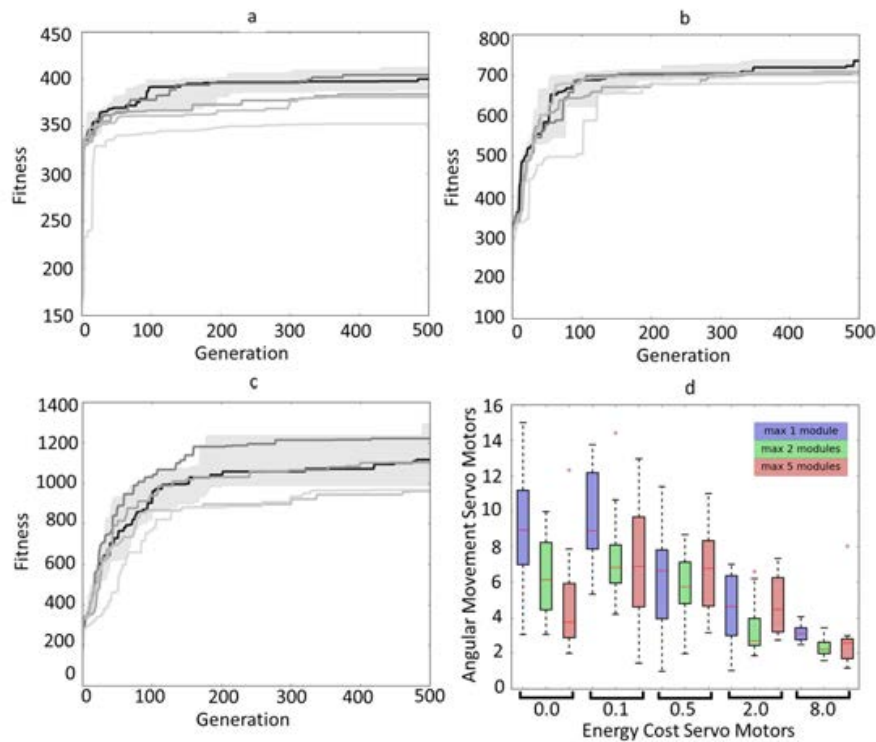


FIGURE 7.9: **The effects of an additional energy cost.** A maximum of 1 solar panel (a), 2 solar panels (b) and 5 solar panels (c) were simulated. The black line depicts the average of the maximum fitness values of 12 individual runs. The lighter gray lines depict the average of the maximum fitness of 12 individual runs where energy cost was applied to the simulation. The lighter gray lines represent higher energy cost. The box plot (d) depicts the average movement of the solar panels in each simulation with different energy costs. Though much disparity could be seen between runs, a trend is present that when energy cost is higher, the resulting phenotypes, on average, move their solar panels less. This distinction can most clearly be seen when a maximum of 1 solar panel was simulated (blue). The different colors of the box plot represent the different numbers of maximum solar panel modules that could be generated.

modules may be more beneficial than requiring tracking a light source. When robots are constructed in the real world, the actual movement costs and solar uptake of the entire robot can be modeled and used as a feedback mechanism for the simulation environment. In this scenario, we could

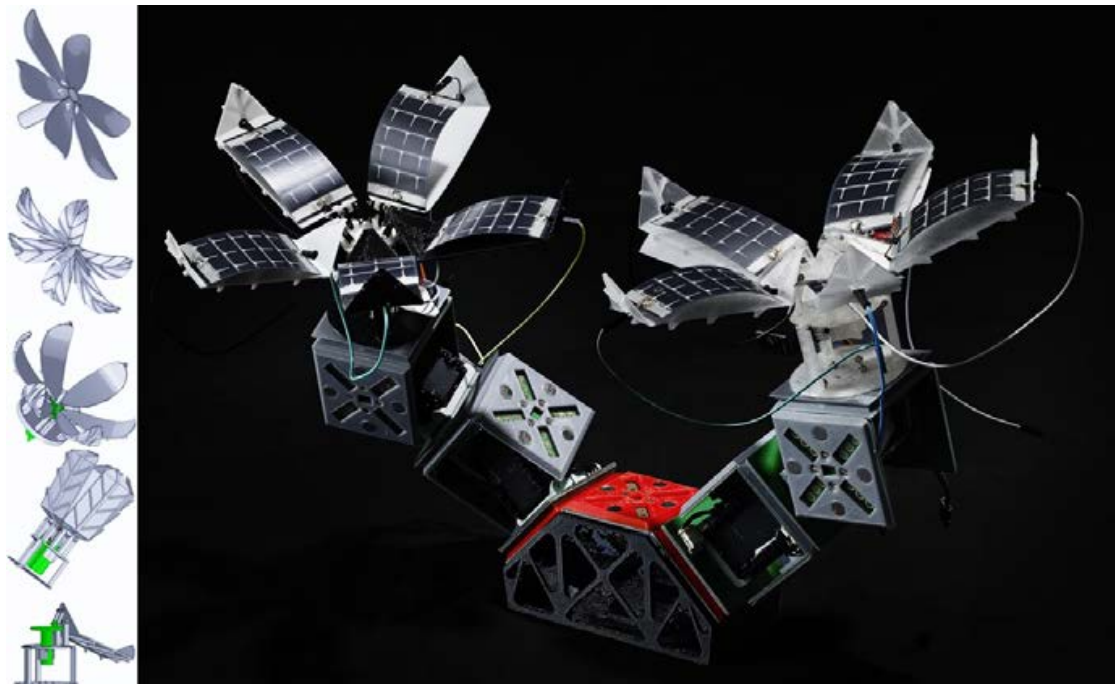


FIGURE 7.10: **Incorporation of the flower module.** (Left) different design stages of the flower module. (Right) an example of an assembled modular robot with flower modules. Note that the servo modules are an older version with one data channel instead of four. Photograph by Phil Ayres.

determine whether it is better to make small modular robots at different locations only containing a few solar modules, or to create a single modular robot containing many solar panels.

7.4.2 Challenges

One major challenge of the presented robotic modular platform is the size and weight of the modules. Since the modules are only connected via magnetized connector sites, the structure will become quite heavy, which is especially detrimental at the extremities of the modular robot. Modules connected to the base module would need to cope with more force than the modules at the extremities. Similar to plants, the main stem or tree trunk is usually the strongest and heaviest, while its branches are lighter, and its leaves even lighter than the branches. Being unable to distribute weight accordingly is, however, a common issue in modular robotics, and one solution is to remove redundant connection sites. Another

solution is to vary the shape, size, and weight of the modules. Such a system might require modules that isolate their mechanical connections and communications. Since every connection face of our modules contains the same number of magnets, each connection between modules has the same strength. A future implementation could allow for modules near the base to be connected more forcefully and modules at the extremities to be connected with less force.

Only a limited number of generations were simulated in the evolutionary runs due to conversion of the evolutionary algorithm. Improvements to the evolutionary algorithm itself would be valuable to cope with this. These improvements could consist of implementing speciation (Stanley et al., 2002b), novelty search (Lehman et al., 2011), and/or Pareto optimization techniques (Schmidt et al., 2011; Brodbeck et al., 2015). However, doing more evolutionary runs might lead to the discovery of better-performing individuals by having an initial population that can be exposed to incremental improvements. The demonstrated robotic platform has potential for integrating interactive evolution (Sims, 1992; Graf et al., 1996). A human-in-the-loop could, for example, design the modular robot by simply connecting the modules to form a robot morphology. The simulator could in turn evolve behaviors for that specific robot's configuration in the simulation environment, similar to Wagyu et al. (2015) with potential continuous self-modeling (Bongard et al., 2006). Although evolutionary algorithms were used to generate the control and morphology of the modular robots, online adaptation could be a more beneficial strategy, especially for real-world robots. A hybrid approach of initially evolving arbitrary modular robots followed by an online learning strategy could therefore lead to robots that are more feasible. This would be especially valuable if the adaptive control can be localized and its genetic information could be reused similarly to the generative encoding.

The generative encoding abstracts the complexity of the genome and has been shown to work well to quickly acquire decent performance. However, as is common in evolutionary algorithms, some runs stagnate in a local optimum, making it harder for the evolutionary algorithm to discover novel strategies. However, the variety of modular robots that evolved in each environment varied a great deal. It is unknown whether doing more evolutionary runs will increase this variety of decent modular robots. The generative encoding implemented might also be improved through using more implementations that have been used as abstractions of development such as compositional-pattern producing networks (Stanley, 2007).

The evolutionary system limited the number of usable modules to the number of robotic counterparts in the real world. Simulating more

modules could lead to phenotypes that are more effective. However, simulating less modules could potentially lead to good solutions as well, when considering that many modules that have been implemented in the evolved robot did not seem to be of any particular advantage (See [Figure 7.7b](#)). The implemented modules are also not ideal when optimizing for light harvesting in modular robots. For example, the servo modules contained a hinge-like joint, although it might be better to implement a twisting joint in the environment with the moving light source. Moreover, other modules with less energy-demanding actuation, or elastic and rigid structural properties, could increase the performance of the evolutionary algorithm to find more suitable solutions for absorbing light from the environment. One possible new module type could be a soft-module similar to Vergara et al. (2017). If a soft module could be used to inflate multiple chambers, directed movement of solar panel modules towards a light source could be implemented with ease. Such a soft module would be similar to a pulvinus structure observed in many plants. Even the energy-gathering system could potentially be combined with modules that gather energy from different sources, such as microbial fuel cells. The robotics platform does allow for the effortless integration of new modules in the system—for both the hardware and the software—although explorations of these systems is left for future work.

Four types of environments were used in the evolutionary runs to optimize for light absorption, while the fitness function stayed the same across environments. An additional approach could be to evolve a population of individuals initially in one environment, and afterwards in more advanced environments, which can either be done incrementally (Bongard, 2008) or through encapsulation of behavior in specific evolved environments (Lessin et al., 2013). The light source could also be used as a stimulus in mobile modular robots. In this case, the evolutionary algorithm could optimize the robot to locomote towards light as a main stimulus that could potentially give rise to modular Braitenberg vehicles (Braitenberg, 1986). Evolving locomotion would add an additional reality gap that was not present in our sessile modular robots. Additional experiments could also include lattices with many solar panels, which greatly increase the uptake of light. As seen in the evolutionary runs, modular robotic structures with many solar panels would not necessarily give any freedom for complex behavior to evolve. Moreover, the robotics platform can be used to evolve any arbitrary task that evaluates compositions of robotic modules. If the same connection sites are used for the modules, any type of module could be implemented. For locomotion, it would be useful to have additional modules that can act as feet or tendons. The robots evolved for

different tasks could then be further evolved towards energy autonomy by allowing solar panels to be implemented in the evolved individuals.

7.4.3 Multi-Robot Systems

Considering food chains in natural ecosystems, organisms higher up the food chain extract the energy that has been harvested by primary energy producers (Reece et al., 2010) simply through consuming and reusing the molecules they have produced. Hence, most plants and other photosynthetic organisms are known as primary energy producers. Primary, secondary, tertiary, etc., energy consumers subsequently depend on the acquisition of chemical energy that has been created by their prey. The modular approach allows for a potential implementation of a multi-robot system, where a robot specialized in absorbing light could share its energy with other robots, potentially making robot energy autonomy viable in environments that are off the electrical grid. Moreover, humanity's ecological footprint is already unsustainable and in overshoot (Toth et al., 2016). This calls for an approach towards energy autonomy in robots that will not increase our already existing ecological footprint. Rather than relying on renewable energy sources to be implemented in our energy grid, we can imagine a robotic system where robot entities can specialize in gathering energy from the environment while higher-order robots can tap into the power grid of these robot entities to recharge themselves—possibly tapping into humanity's existing electrical power grid to release their energy. Since light in urban or desert environments is largely wasted, as a limited number of plants grow in these locations, robots in these environments can be greatly beneficial if they are able to gather solar energy in say, windows, rooftops, or the like.

Since the task of our robotic platform is largely undetermined, it is unclear what the eventual robotic entity will look like, and how it will move, manipulate objects, and gather information from the environment. Instead of having robotic entities created for specific functions, a range of robotic modules can be implemented that can be optimized for a variety of tasks. In an approach where multiple small modular robots are used to gather energy, an additional modular robot could be used to extract energy from the energy-harvesting modular robots simply by connecting a docking module to the solar-harvesting modular robot. This docking module could simply be a male connector face that attaches to a female connector face of the energy harvesting robot. This can, in turn, give rise to an artificial ecosystem whereby some robots are specialized in energy uptake, while others can be specialized for different tasks. Such a system could enable

Text Box 7.6: Transferring locomotion from simulation to reality

The modular robot has also been used for other objectives such as locomotion. There is, however, a severe reality gap when testing the evolved phenotypes in the simulator with reality. Although as [Figure 7.11](#) depicts, the behavior of the real and physical modules is almost identical. As illustrated in this chapter, the modules are both electrically and mechanically connected by the connection sites. However, when the robot performs a strong movement that causes it to forcefully slam on the ground, the connections of the spring pins can temporarily or permanently be lost. This loss in connection due to mechanical stress could therefore be isolated by isolating the mechanical and electrical connections between the modules.

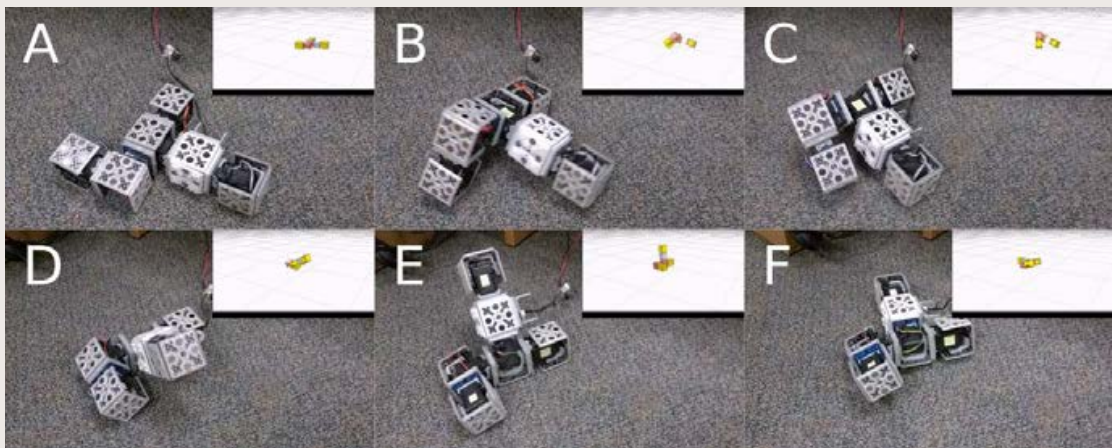


FIGURE 7.11: **Simulated and real modular robot behavior.** The figures A to F show the behavior of an individual instance of a modular robot optimized for locomotion with the simulator environment in the top right corner of each figure.

symbiotic relationships between modular robots that are optimized to harvest energy and other robots that are able to extract energy from the energy-harvesting modular robot. The modular connection mechanisms would allow the energy to flow directly from one modular robot to another

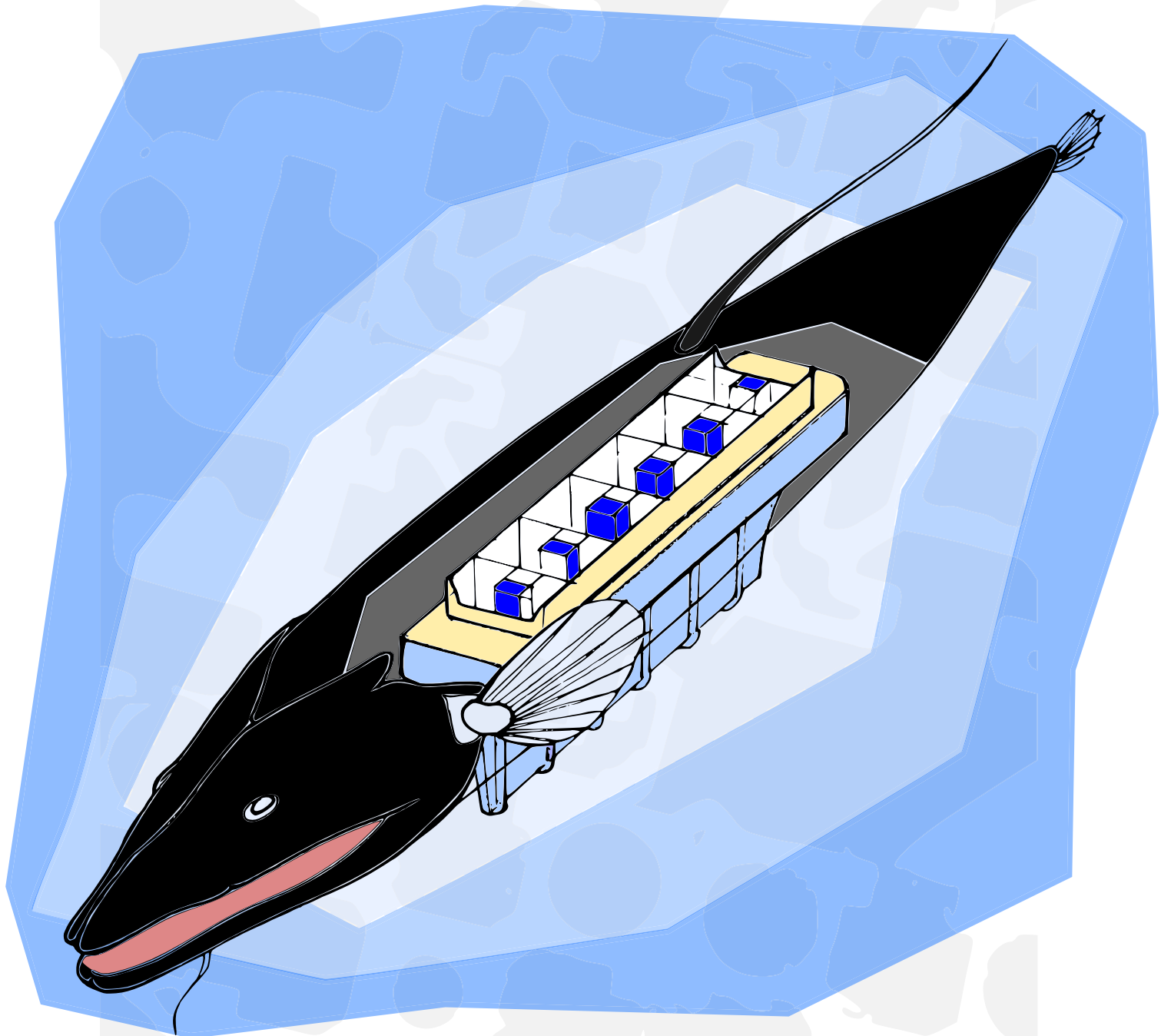
without too much of a hassle. The advantage of energy autonomy in an ecosystem of robots, instead of a single robot, is that some robots can specialize for very specific tasks, not having to worry about expending large amounts of energy. This division of labor in multi-robot systems conforms to symbiotic energy exchange that can contribute to the survival of the entire robotic system.

7.5 Conclusion

It has been shown how an evolutionary algorithm can evolve light-harvesting robots that can easily be transferred to the real world using a modular robotics approach. Furthermore, there is an advantage for evolutionary algorithms to exploit movement when only one solar panel module is simulated, but not when more are simulated, given the constraints of the simulator. It has been demonstrated that the robotic platform is able to evolve a variety of modular robots based on the various objectives in the environment. This might prove especially advantageous when evolving a multi-robot system that can form a stepping stone towards an artificial ecosystem. This artificial ecosystem can consist of multiple modular robotic entities that interact with one another based on energy. The described platform was limited to the available types of modules, and the flower module was not used in the evolutionary algorithms. However, a robotic module containing more complex electronics could simplify evolutionary search as a smart morphology and online learning is implemented. All of the described experiments and proposed methods can thus lead to an automated design of homeostasis in robots that can potentially scale up to artificial ecosystems.

Part III

Evolving Physical Robots



Chapter 8

Evolutionary Approaches in Physical Robots

It means that each factory will be making Robots of a different color, a different nationality, a different tongue; they'll all be different—as different from one another as fingerprints; they'll no longer be able to conspire with one another; and we—we people will help to foster their prejudices and cultivate their mutual lack of understanding, you see? So that any given Robot, to the day of its death, right to the grave, will forever hate a Robot bearing the trademark of another factory.

– Karel Čapek, R.U.R. ¹

Embodiment of robots conveys robots experiencing the world directly, as opposed to simulated robots, where “the actions of robots have an immediate feedback on the robot’s sensations” (Brooks, 1991). An embodied robot is a real, physical robot whose behavior can be observed in the environment (Pfeifer et al., 2006). Though already here, some sort of embodiment can also take place in the simulator, since the “environment” itself is also simulated. Pfeifer et al. (2006) especially advocate that embodiment is a prerequisite for the emergence of intelligence. However, a major problem with the application of evolutionary algorithms in physical systems directly is that the evolutionary process can take a long time to complete (Nolfi et al., 2000) as compared to a robotics simulator. Hardware failures are common in the real world, which adds an additional requirement of a robust robotic system when implementing evolution on that system. There are therefore many ways to acquire evolved behaviors in robotic systems. One pathway has been described in **Chapter 7** by directly building a modular robot based on the evolved simulated phenotype. In this case, due to the sessile nature of the plant-inspired robot, the *reality gap* was negligible. If the evolved robots would have looked more like the work

¹This epigraph has been added after the acceptance of the official thesis manuscript.

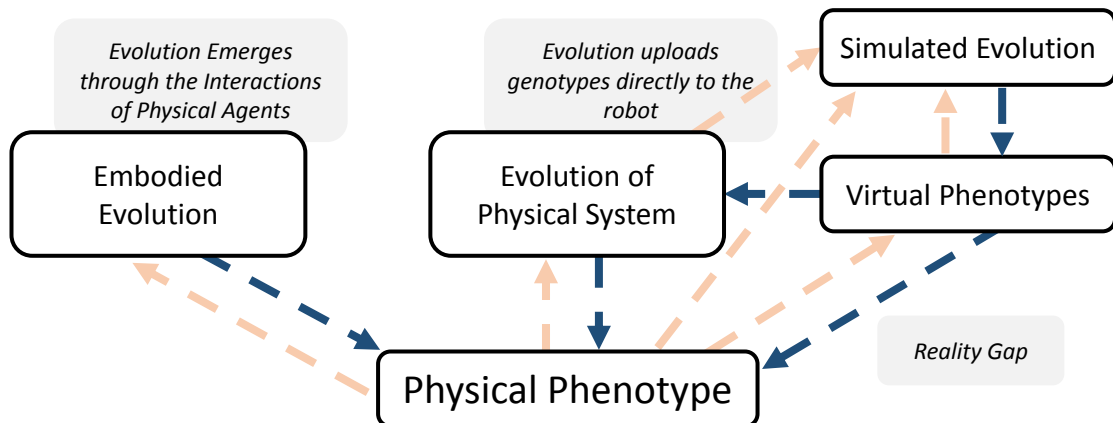


FIGURE 8.1: **Evolutionary robotics approaches to acquire a behavior.** This can be brought about by first simulating the robot, evolving the physical system directly or through embodied evolution.

of Sims (1994b), or the phenotypes displayed in Figure 6.8, the transference of the evolved phenotypes to the real world would be problematic.

An alternative strategy to circumvent the problem of the reality gap is to directly evolve the robot in the real physical world. This bypass of the simulation environment is especially beneficial for robots that are hard to simulate, e.g., robots that contain many parts, springs, or soft structures. This chapter therefore provides a brief overview of evolutionary approaches in physical robots (Section 8.1) and discusses an implementation of the automated assembly of a modular robot (Section 8.2). The latter implementation would allow an autonomous robot platform to automatically evolve and evaluate simulated and real world robots that differ in their morphology. In Chapter 9, an experiment was done to directly evolve a physical knifefish-inspired soft robot using the strategy.

8.1 Evolving Physical Robots

Different evolutionary robotics approaches can be taken to evolve the morphology or behavior of robots, as summarized in Figure 8.1. One of the most direct approaches to evolving robot behavior is through having a terminal run the evolutionary algorithm, and either uploading the genomes, or directly controlling the robot itself. These types of approaches are not common but are discussed by Floreano et al. (1994) and Zykov et al. (2004). Alternatively, an evolutionary algorithm can first evolve

simulated robots and afterwards transfer and further evolve the robots in the physical world (Nolfi et al., 1994; Pollack et al., 2000). In this case, the reality gap is present but is reduced by evolving the evolved behaviors for a few generations in reality afterwards. However, this additional evolutionary step might not be needed with the increasing accuracy of simulation environments. Jakobi et al. (1995), for example, saw that including noise in a simulation environment makes the evolved simulated robots more transferable to reality. Furthermore, this approach relied on the additional simplification of most behaviors to minimize the reality gap (Jakobi, 1998). The reality gap can also be decreased through implementing adaptive traits in the simulated models that can be adjusted in the real robot through, for example, plasticity in neural networks (Floreano et al., 2001). A feedback loop from the robot and the simulator has even been used to reconstruct the parameters and the dynamics of a physical robot themselves through continuous self-modeling (Bongard et al., 2006). In this case, an evolutionary algorithm can self-adjust its simulation environment based on the model it creates of itself. Even the simulator itself can be evolved based on the acquisition of real data from the environment. Another strategy that has been implemented is called Map-Elites (Mouret et al., 2015). In this approach, elites with different characteristics are transferred to the real-world and a heat map of behavioral traits that are efficiently transferred can be constructed, which, in turn, illuminates the search space of the robot. This map can inform the robot which behaviors are likely to be transferable compared to others, as has been demonstrated in Cully et al. (2015).

Another type of implementation bypassing both the reality gap and the evolutionary algorithm running on a terminal is known as embodied (artificial) evolution. The term was originally proposed by Watson et al. (1999) and embodies the evolutionary process itself in a population of physical robots. As proposed by Eiben et al. (2012), embodied artificial evolution is a specific form of embodiment where (1) physical units are involved, as opposed to virtual ones, (2) real birth and death is implemented, (3) the environment shapes the fitness of an individual, and (4) individuals decide when and how to mate. This embodied evolutionary approach is perhaps the most cumbersome to achieve, but could in a future with autonomous robots be an emergent property of artificial life, since robots that make robots would also be bound by evolutionary principles in a struggle for artificial life.

As depicted in [Figure 8.1](#), many approaches can be taken in evolutionary robotics to acquire a physical phenotype. The simplest form, without requiring a robotics simulator, is to directly evolve the behavior of a robot in a physical system, as implemented in Floreano et al. (1994). This is the

type of evolutionary strategy that is implemented on a knifefish-inspired soft robot, as described in [Chapter 9](#). Before discussing this approach to evolving the soft robot, the next section presents a method for the automated reconfiguration of modular robots directly in the physical world, which can allow for the automated evaluation of simulated and real robots simultaneously. This is an approach toward autonomous evaluation and self-reconfiguration of modular robots.

8.2 Automated Assembly of Modular Robots

In Brodbeck et al. (2015), an approach has been described for the automatic generation of modular robots. The robots, composed of active and passive modules, were picked up by grippers attached to a UR5 robot arm and connected to other modules with hot glue adhesives. This process allowed Brodbeck to automatically assemble and evaluate a variety of modular robotic structures that were evolved. The main disadvantage of this approach was the difficulty of removing the glue once a robot had been constructed, and reconfiguring the morphology afterwards. The implementation of a similar assembly strategy using the magnet-based modular robotics system described in [Chapter 4](#) and [Chapter 7](#) would allow for a versatile automated assembly, and disassembly, of robotic modules, without the necessity of hot glue or other adhesives. This makes the platform potentially resilient and enables robots to be sequentially reconfigured and evaluated. This section describes a methodology for automatically recognizing modules and how to subsequently pick and place the modules to create a robot morphology.

8.2.1 Methods and Environment

For the assembling process of a modular robot, a robot arm and a webcam were used in an environment where modules could be automatically detected, connected, and disassembled. The environment consisted of a rectangular plate with four fiducial markers (markers used as a point of reference) on the corners. These four fiducial markers were used to extrapolate the position of the modules in the environment. An affine transform was used to translate the pixel coordinates in the positions in the reference system of the robot. The modules themselves also contained fiducial markers. These markers were oriented so that the male connector site was always facing in the same direction. The positions and orientations

of the fiducial markers on the modules were used to plan how the robot arm would connect the modules to one another, similar to Faíña et al. (2017).

The robotic platform used a magnetic gripper attached to a Universal Robotics UR5 robot manipulator. The gripper utilized permanent magnets to detach and attach magnetic modules for the creation of the robot, as can be seen in [Figure 8.2](#). A module is attached to another by attaching the male connector site to another specified female connector site of another module. The gripper did not contain any moving parts, and thus relied on the specific movement of the arm for connecting and disconnecting itself to the modules. The end effector moved in an arc perpendicular to the male connector site to disconnect itself from the modules. When detaching a module from the modular robot containing several modules, the direction of the movement was changed to produce an arc around the two connected module sites.

The basic assembly steps are:

1. Move and align end effector above the marker
2. Move the end effector down until a force of 30N is applied; avoiding jamming and ensuring that the module is well connected to the gripper
3. Lift the module up to a safe distance above the floor
4. Move the end effector to align with a desired site to connect to
5. Move the end effector down, hovering above the floor
6. Move the end effector towards the site to attach
7. Perform the movements required to release the module that is attached to the gripper.

With this system, various types of morphologies can easily be constructed. However, the morphologies in this case could only be assembled in 2D since the fiducial markers were only connected to one connector site of the modules, which was identified by a camera from a top view. Hence, only planar configurations (one layer of modules) were considered. The process of assembling modular morphologies is displayed in [Figure 8.3](#). The assembly of the morphology was achieved without difficulties due to the self-alignment properties of the connectors. It was more challenging to disassemble the morphology where the robot arm would sometimes jam. The main problem here was that the complex movements of the end effector were sometimes near singularity points leading to a safety stop, causing the jam. Additionally, the end effector sometimes caused the attached module to be pressed against the arena floor



FIGURE 8.2: Assembly (left) and disassembly process (right) of modules

while rotating, also causing a jam. These challenges should be taken into consideration in future implementations.

8.2.2 Prospects on the Automated Assembly of Modular Robots

The process of creating modular robots discussed in this section allows for the automated generation of modular robots that would be especially useful for chain-type modular robots (Stoy et al., 2010), where some self-reconfigurations are not possible due to kinematic restrictions. It is

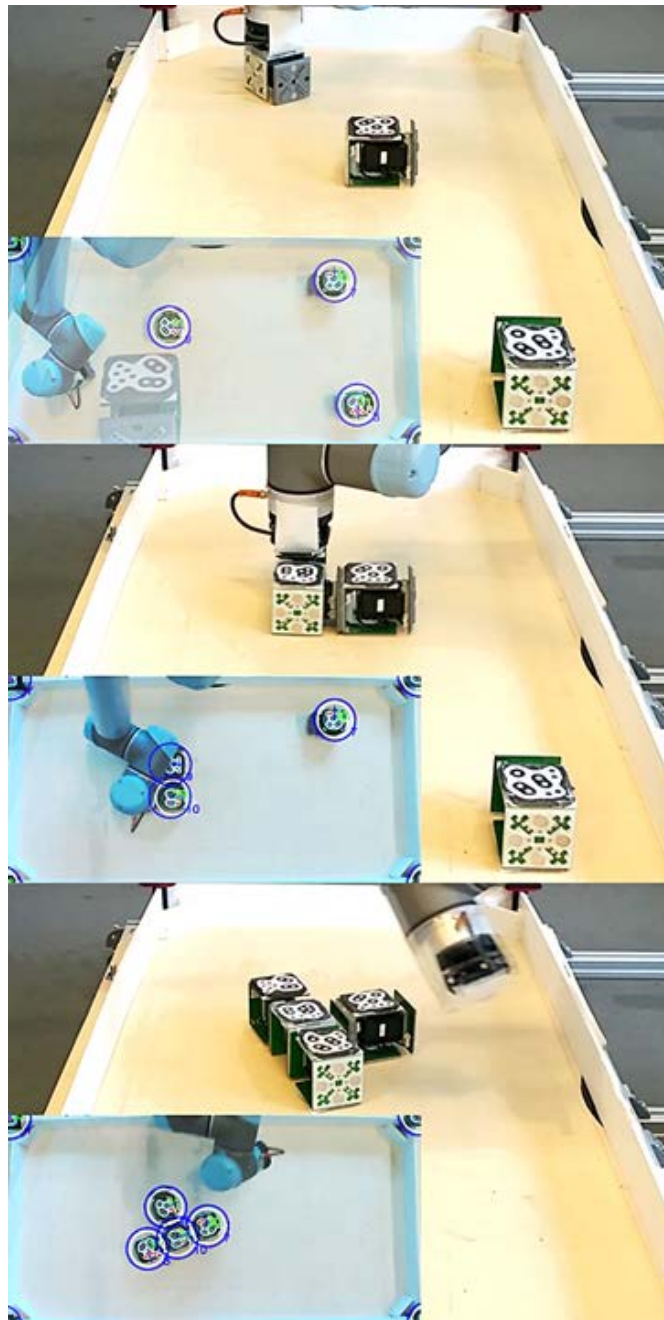


FIGURE 8.3: **Sequence of automatically assembling modular robots** The bottom left images depict the computer vision. The fiducial markers that are recognized are marked with a blue circle.

moreover useful as a testbed for rapid prototyping. When combined with a robotics simulator that can automatically optimize potential modular robot

configurations, this platform would allow us to automatically generate robots with a unique morphology and control. A fruitful prospect is to combine the approach with the evolving modular robots, as discussed in Chapters 456 and 7.

Some challenges remain to be addressed to improve the system to a state in which continuous experiments can be done with reconfiguring robot morphologies. Visual fiducial markers can currently only be attached to the modules at specific parts occupying and obstructing a potential connection site. A redesigned fiducial marker or another way of recognizing the robot modules should be used in future implementations. Another limitation of the current setup is that the robot is unable to be controlled and acquire power without the help of an operator attaching a control cable to one of the modules. This problem could simply be bypassed by adding a docking station that the modular robot can be connected to. To have a fully automatic assembly process, a base module that couples to a docking station can be implemented. The base module/docking station assembly can then be used as the initial module from which robot morphologies are assembled. Battery modules with wireless capabilities can also be added to the system when the assembled robot must move away from the docking station. Three-dimensional configurations still represent a challenge, as they would require more complex reconfiguration movements and adjustments to the visual tracking system.

Automatic reconfiguration can be especially beneficial in fields that optimize the morphology and control of robots. It can be specifically well suited for evolutionary robotics experiments that are usually time consuming due to the number of morphologies that need to be tested. Although some approaches already show promising results by implementing evolved robots in simulation environments and afterwards transferring them to the real-world (Auerbach et al., 2015), they are still time-consuming considering that all parts need to be glued or screwed together. This is also addressed in Brodbeck et al. (2015) by using an evolutionary algorithm to generate the robot morphologies; however, the approach presented here allows for different robot morphologies to be more quickly tested in an environment, and a combined approach of using simulations to optimize the robots with the subsequent transference of the evolved simulated robots to reality is therefore promising. Furthermore, this can give insight into reality gap-related issues that would allow us to improve simulators and evolutionary algorithms to minimize the gap.

8.3 Concluding Remarks

In the first part of this chapter, an overview has been given on approaches to evolve physical systems. This can be done in various manners, either including a simulation environment or not. However, to rapidly evaluate various robotic morphologies that can be evolved, as discussed in [Part II](#), an automated assembly process of these modular robots is beneficial. [Section 8.2](#) has shown how modular robots can be automatically assembled and disassembled using a robot manipulator. Without the need for power-consuming autonomous modules, the merit of the approach would potentially alter the way we evolve robots. Otherwise, human-approached prototyping can take many hours, whereas the assembly and positioning of the robots with the robot arm spans only a few minutes. The automatic reconfigurability thus enables fast prototyping of different robotic morphologies and control systems, insightful for constructing an efficient robotic end-product for a given task. The visual feedback system could be improved by being able to identify modules in 3D, and the robot arm can be improved by having an active gripper and a better positioning algorithm. In addition, the potential integration of a docking system could allow for the automated repositioning and recharging of modular robot conglomerates. Many improvements can be made to engender the automated design and creation of robot behavior and morphologies that would truly allow us to evolve the simulated and physical robot in an automated manner.

Chapter 9

Evolution in a Knifefish-inspired Soft Robot

Many were increasingly of the opinion that they'd all made a big mistake in coming down from the trees in the first place. And some said that even the trees had been a bad move, and that no one should ever have left the oceans.

– Douglas Adams, *The Hitchhiker's Guide to the Galaxy* ¹

The physicality and embodiment of robots accompanies the morphological complexity that can be difficult to simulate. One way to deal with the limitation of not being able to simulate a robot is to directly implement the optimization strategy on the robot. Bypassing the simulator ensures that the strategies found conform to reality, not violating the laws of physics. In this chapter, evolution is implemented directly on a bio-inspired soft robot that was based on the black ghost knifefish. This knifefish species is interesting since it has an undulating fin underneath its body responsible for its movement [Figure 9.1](#). The robot fish was made from soft materials used to cast the entire body and included solid fin rays connected to servomotors for actuating the fin. This chapter discusses the design, the algorithmic implementation, and the resulting behaviors compared to natural knifefish. The comparison of the robotic knifefish to the actual knifefish through evolving its controllers combines the paradigms of biorobotics and evolutionary robotics, also known as evolutionary biorobotics (Long, 2012). Through evolving the robotic knifefish, we can learn how to better optimize soft robots and start to understand the natural functionality of undulation in knifefish. It is a step toward automatically generating the control for robotic

¹This epigraph has been added after the acceptance of the official thesis manuscript.

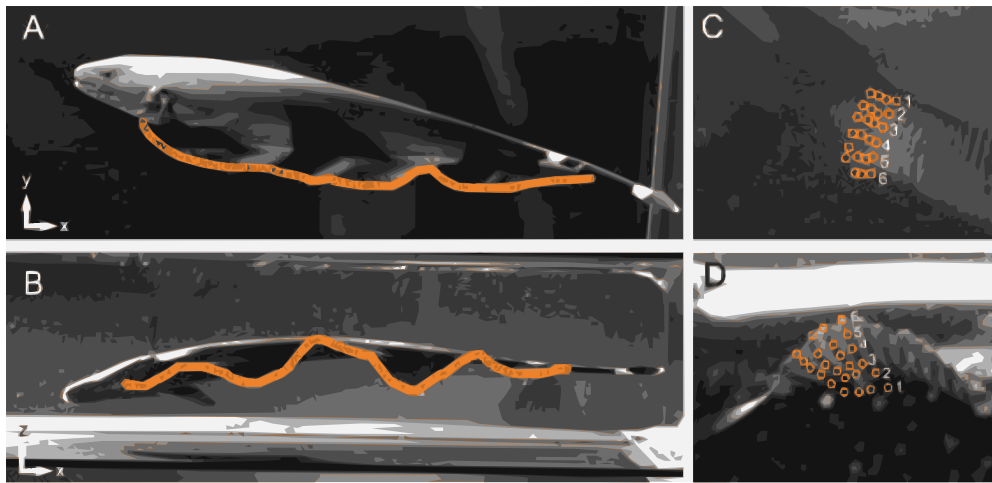


FIGURE 9.1: **The black ghost knifefish.** Vectorized image taken from Youngerman et al. (2014). Images show the *apteronotus albifrons* from a lateral (A) and ventral (B) view; orange line depicting digitized points across the tip of anal fin of the knifefish. Six digitized points are shown by orange circles along the individual fin rays laterally (C) and ventrally (D)

soft robots. Hence, this chapter brings us to the fifth and final hypothesis of this thesis discussing evolution of a physical robot:

Hypothesis 5 *Evolutionary computation finds better solutions for controlling the designed soft swimming robot compared to manually encoded behaviors.*

9.1 Introduction

Despite recent advances in evolutionary robotics, the reality gap (Jakobi et al., 1995) is still a prevalent issue. Especially in the emerging field of soft robotics, it becomes more difficult to simulate the physical properties of soft materials accurately (Rus et al., 2015). In cases where this was accomplished successfully, it has required high computational power and complex algorithms (Coevoet et al., 2017). For aquatic robots, the integration of flexible materials can lead to increased performance by the principle of morphological computation; i.e., by exploiting that, dynamic interactions with the environment can be useful for achieving a desired behavior efficiently. The complex mechanics of silicone and its hydrodynamic interactions are, however, computationally heavy to simulate, especially

when the morphology is driven by multiple actuators. For these reasons, an evolutionary approach was implemented in directly evolving physical systems (Rieffel et al., 2017) as a feasible alternative method to evolve efficient behavior of a bio-inspired soft robot.

Soft robots have been proposed for several applications that include exploration and search and rescue operations. For such tasks, high maneuverability is usually necessary. Since the family of ghost knifefish (Apterontidae) contain examples of dexterous aquatic animals capable of high multidirectional maneuverability at low speeds (MacIver et al., 2004), this fish was chosen as the model whose control was subjected to evolution. Knifefish are able to produce thrust in many directions by undulating a single anal fin located underneath the body. By generating propagating waves across their fin, they can easily move backwards and forwards depending on the directionality of the wave (Curet et al., 2011a). Vertical thrust is accomplished through sending counter-propagating waves towards and away from the center of the fin, canceling out longitudinal forces. In undulatory swimming, the thrust is produced through a reaction force on the fluid adjacent to the body or fin surface. Bending of the body part, the fin, enables wave propagation. The combination of the lateral forces produced on both side of the fin should cancel each other out to produce a net forward thrust (Biewener, 2003).

9.1.1 Evolution of Soft Robots

The evolutionary robotics approach to soft robotics has thus far only been implemented in simulation environments such as VoxCad (Cheney et al., 2013; Cheney et al., 2016; Kriegman et al., 2017) or off-the-shelf physics engines where morphologies are represented by tetrahedral meshes and the controls and morphology have been evolved (Rieffel et al., 2013). Computational power is, however, a major constraint when using simulations. Computational requirements usually scale proportionally to the number of tetrahedra or voxels simulated, usually exponentially. Morphologies found through the VoxCad approach have only been replicated physically by means of soft volumetrically expanding materials that require changes in the pressure of the surroundings for actuation (Hiller et al., 2012), though these results are again taken from evolving the soft robot in a simulator and transferring its phenotype to the real world afterwards.

Controllers for simulations of existing partially soft morphologies have also been evolved and in some cases transferred to hardware. A genetic algorithm with a “lumped” dynamic model simulation has been used

to evolve the gait of a soft caterpillar-inspired robot and has resulted in an increase in performance of a physical prototype (Saunders et al., 2011). In another instance, both an objective-based and a novelty-driven (*novelty search*; Lehman et al. 2011) approach have been utilized to optimize the design of a crawling octopus by discovering self-stabilizing dynamic gaits (Corucci et al., 2015). A differential evolution algorithm was used to optimize a model-free adaptive controller (MFAC) in a simulation of a robotic fish with a flexible caudal fin (Clark et al., 2015a). For the same morphology, an evolutionary multiobjective optimization technique (NSGA-II algorithm) found morphological and control parameters in simulation that maximize the swimming speed and minimize the power usage with subsequent validation in hardware (Clark et al., 2015b). However, in this approach it was found that the “best speed” parameters of the evolved simulated model were considerably faster than those seen in the experiments due to hardware limitations. This illustrates that although reasonable performance can be transferred from the simulation to reality, discrepancies are still persistent. In the above examples, the evolution of soft robot morphologies and controllers was made possible by confining the search space to highly abstracted morphologies (i.e., only a simple tail is present as a flexible, caterpillar-like shape) or by decomposing the morphologies into a finite number of voxels. While such approaches have yielded interesting results, they are still lacking in relation to realizing the full potential of soft robotics technology as they limit the design space to very simple or highly abstracted shapes. By evolving the controller in the physical hardware instead, an effective strategy can be attained through (1) a bio-inspired design that mimics a natural model closely and (2) the automated discovery of its most optimal behavior.

9.1.2 Knifefish-inspired Swimming Robots

Due to their unique morphology, knifefish have served as inspiration for a number of research robots. Building on the work of Low et al. (Low et al., 2006; Low, 2009), Siahmansouri et al. constructed an untethered robot with six fin rays capable of regulating the direction and depth of swimming by moving the fin relative to a buoyancy tank (Siahmansouri et al., 2011). Curet et al. built a knifefish-inspired robot with 32 individually actuated fin rays and were able to show that its optimal actuation parameters were similar to the ones of the black ghost knifefish (Curet et al., 2011b). They were also able to generate upward forces on the robot with counter-propagating undulation waves (Curet et al., 2011a). Sfakiotakis et al. (2015) devised a

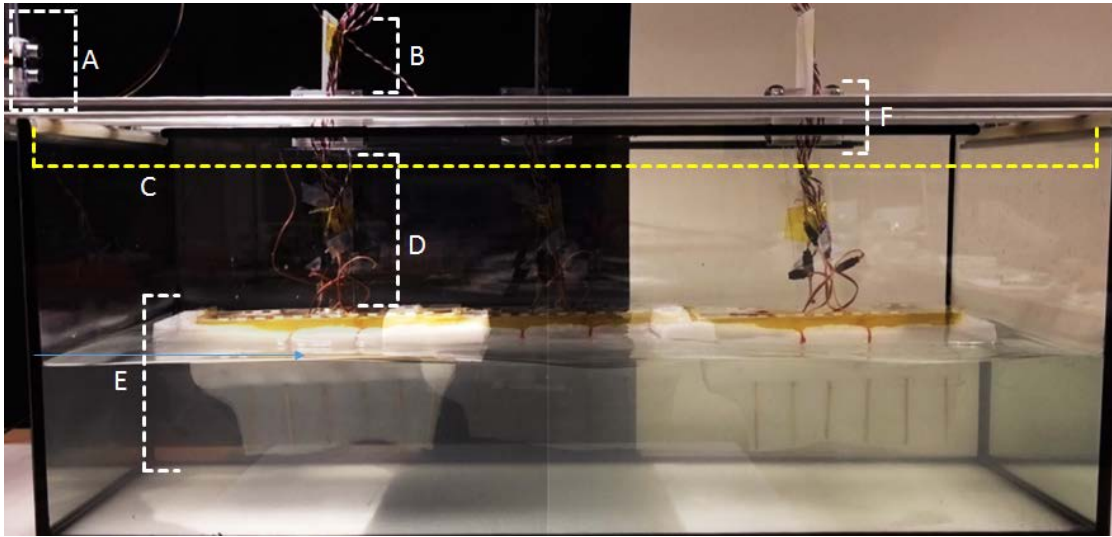


FIGURE 9.2: **Experimental setup for evolving the knifefish.** (A) Ultrasonic distance sensor, (B) plastic plate for bouncing back the sound of the ultrasonic sensor (C) T-slot linear slide, (D) plastic plate connecting the robot (E) to the cart (F). The evolutionary goal was to move the robot as fast as possible along the slide from the left to the right side of the aquarium.

linear slide equipped with a fin composed of eight individual fin rays and implemented open-loop velocity control and closed-loop position control.

A common denominator of the previous work on knifefish-inspired robots is the use of sinusoidal functions as an undulation pattern for the fin. This occurs even though a sine function is only an approximation of the actual undulation pattern of the species, which could be reproduced more accurately (Youngerman et al., 2014). The design of our robot also departs from the earlier work as it is an integrated silicone morphology constructed through contemporary soft robotics fabrication techniques. This approach simplifies the fabrication of the fin and fin rays significantly. Moreover, elasticity is added to the fin, which has been hypothesized to be a means of increasing energy efficiency (Low, 2009).

9.2 Methodology

A soft swimming robot with a single undulating fin inspired by the anatomy of the black ghost knifefish was designed.² To evaluate its swimming speed with different motion patterns, the experimental setup illustrated in [Figure 9.2](#) was constructed. As only the forward swimming speed was evolved, the robot was fixed on a linear slide. It was not submersible and kept at a level of neutral buoyancy. The robot (E) is placed in the water surface of a $100\text{cm} \times 40\text{cm} \times 40\text{cm}$ aquarium. It was tethered with power and signal cables for its 6 servomotors. It was attached to a cart (F) with four ball-bearing wheels that were mounted on a T-slot beam linear slide (C) atop the aquarium. A plastic attachment piece (D) connected the cart to the linear slide and prevented the robot from turning. The slide was equipped with two IR sensors to measure when the beginning and end of the slide had been reached. For the evaluation of an undulation pattern, the robot started on the left side of the track at the first IR sensor. During evaluation a swimming pattern was played on the robot and an ultrasonic distance sensor (A) measured the distance to a plastic plate (B) on the cart. The cumulative sum of the distance readings was used directly as the fitness value for the undulation pattern that was evaluated.

9.2.1 Mechanical Design of the Robot

The main parts of the robot are its hull, frame, and fin rays ([Figure 9.3](#)).³ The hull and fin of the robot were constructed from Ecoflex 00-30 silicone (Young's modulus approx. 0.1 MPa, Shore A hardness 00-30; Mosadegh et al. 2014). The uncured material was degassed after mixing and poured into a three-part 3D printed mold (two sides and one inner part). The inner mold part holds the fin rays in place during casting and blocks out a compartment for the rigid inner frame, which was mounted after casting. The inner frame was constructed from laser-cut acrylic parts that were glued together. The servomotors are held in place with bolts and nuts.

Six bamboo sticks (approx. diameter 3mm) serve as fin rays. With 6 fin rays it is theoretically possible for the robot to hover and to move forward, backward, up, and down by generating traveling and counter-propagating waves (Curet et al., 2011a). Each fin ray is attached to a servomotor via a servo bracket. The servomotors used were initially six H-KING HK 15148

²A video of the robot can be found here: <https://www.youtube.com/watch?v=3XjgZbs0t2g>

³The CAD files can be accessed at: <https://cad.onshape.com/documents/51d2c0394f6e3aa7b3fc06b3>

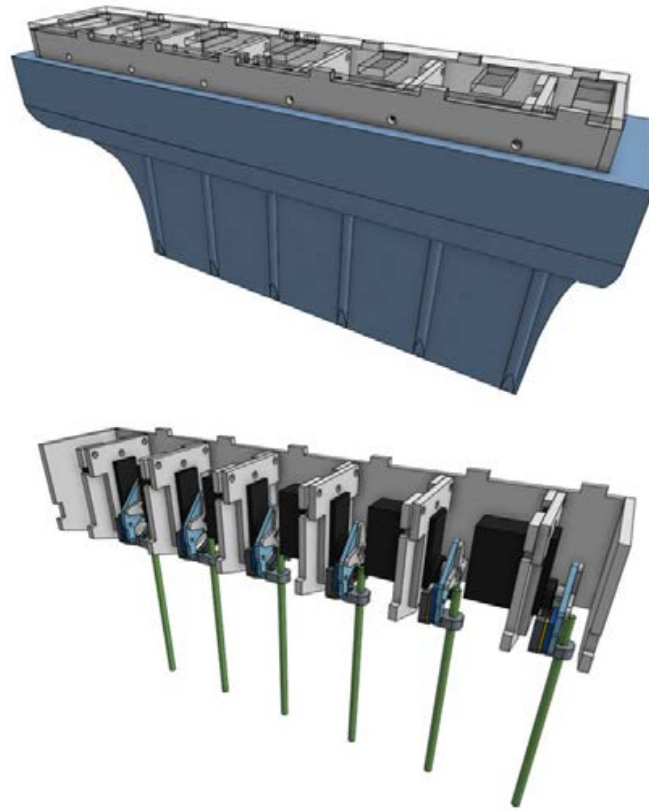


FIGURE 9.3: **CAD design of the robotic knife-fish.** The white parts represent the laser cut acrylic parts, the blue part is the silicone part (top), and the black parts depict the 6 servomotors that were used for each fin ray. The bamboo sticks that serve as the fin rays are displayed in green. The robot's full dimensions are 272x60x136mm, and the fin is 70mm high and 210mm long. The fin rays are each spaced 40mm apart.

mini servomotors. Due to malfunctions three of them were replaced with two tower Pro SG90 micro servos and one EMAX ES08AII servomotor. The servomotors are connected to the fin rays with a crank-like mechanism (Figure 9.4). The angle of a fin ray as a function of the servo angle is given by:

$$\alpha = \tan^{-1} \left(\frac{\sin(\theta) \cdot 21}{30 - \cos(\theta) \cdot 21} \right) \quad (9.1)$$

where the constant 21 is the distance (in mm) from the center of rotation of the servo to the piston that connects to the fin ray, and the constant 30

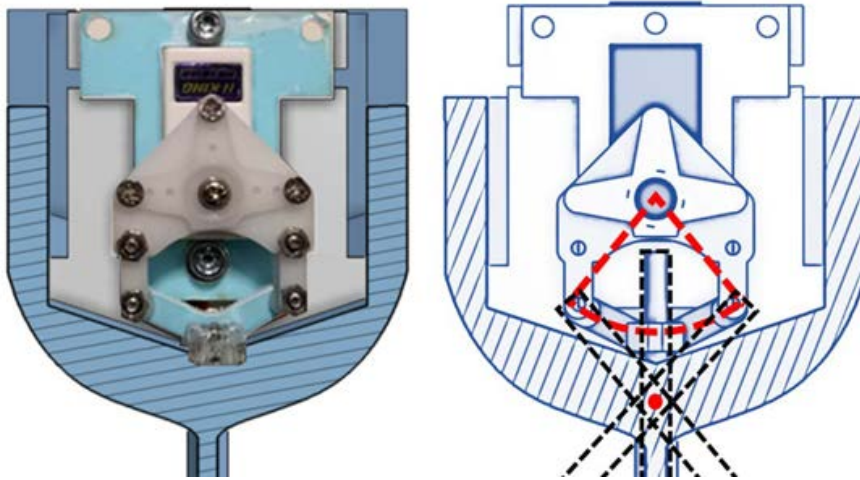


FIGURE 9.4: **Cross section of the fish design.** (left) A photo of the servo mechanism. (right) The red arc depicts the range of motion from the center of rotation of the servomotor to the plastic part that is connected to the crank mechanism. The red dot at the bottom in the hull depicts the approximate center of rotation of the fin ray.

is the distance from the center of rotation of the servo to the approximate center of rotation of the fin ray (see [Figure 9.4](#)). The change in the bamboo angle based on the angle of the servo is displayed in [Figure 9.6](#).

This equation, however, does not take into account the additional angular deflection caused by slack between the pistons and the fin ray, the elasticity of the soft body resisting rotation (see [Figure 9.5](#)), and the tilt of the soft body when actuating the fin rays. The average maximum angular excursion was therefore close to 28 degrees, instead of the approximately 45 degrees that was calculated when not taking into consideration these issues.

9.2.2 Evolutionary Experiments

In the pre-experiments, a generational evolutionary algorithm without crossover was implemented to create the genome for our robot controller. Due to the long evaluation time of the generational evolutionary algorithm, and servos being prone to overheating, we decided to implement Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES; Hansen et al. 1997; Hansen et al. 2003) instead, to quickly find the basin of attraction

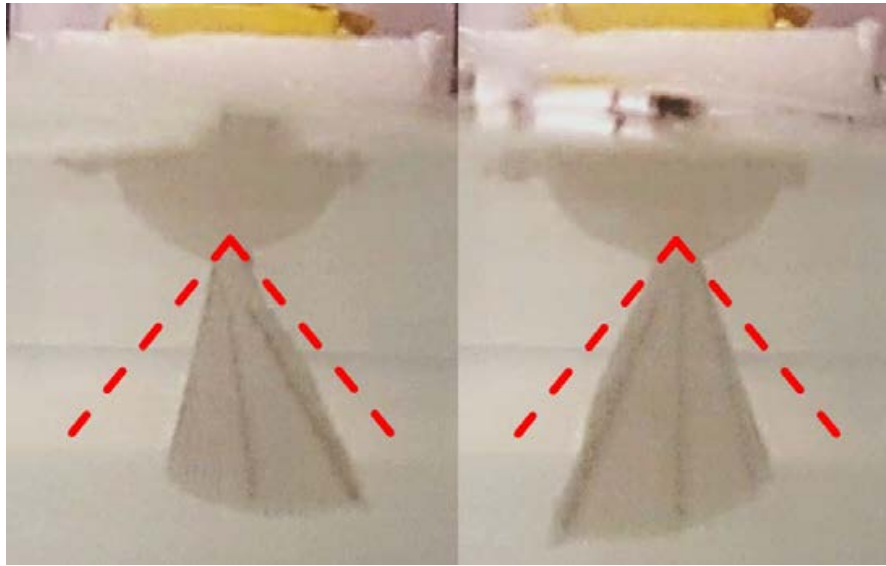


FIGURE 9.5: **Angular deflection of the fin.** Front view of the robot showing the angular deflection of the fin. The actual maximum angle of the fin can be seen to be less than the calculated angle (red dashed lines).

and thereby speed up the evolutionary process.⁴

Encoding

The genome created for an individual is composed of a string of 15 bytes. Each group of three bytes translates into a sinusoidal function with a specific frequency, phase, and amplitude. The five sine functions are summed to yield the first five terms of a standard Fourier series. With this function, an arbitrary continuous periodic function can be approximated and used as a fin undulation pattern on the robot to be evaluated. The mutable parameters were the amplitude, phase, and frequency of each sinusoidal function. These parameters are converted into servo angles α_n for the 6 servomotors with the following function:

$$\alpha_n(t) = \left(\frac{g_1}{255} \cdot \theta_{max}\right) \cdot \sin((g_3 \cdot t) + (g_2 \cdot s_n)) \quad (9.2)$$

where g_1 , g_2 and g_3 represent the mutable parameters of a genome triple as bytes. θ_{max} is the maximum angle that the servomotors can move.

⁴Our full implementation and the source code of the evolutionary algorithm and Arduino code can be found at https://github.com/FrankVeenstra/Knifefish_GECCO2018

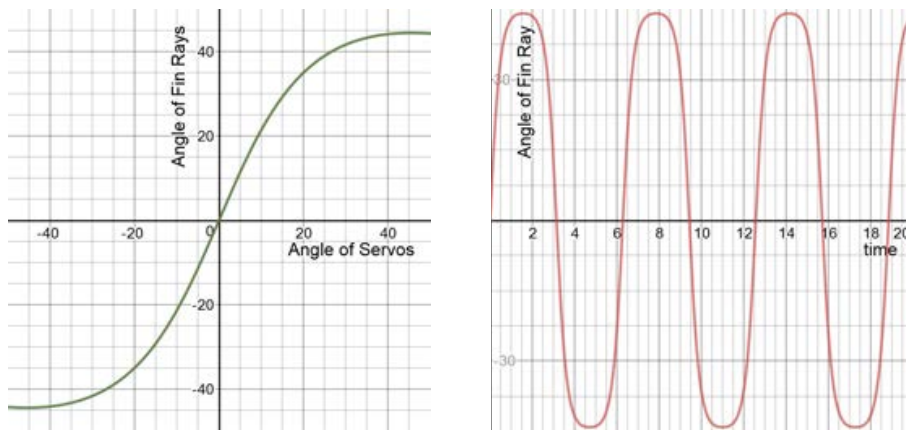


FIGURE 9.6: **Change in angle of fin rays.** The calculated angle of the fin ray based on the angle of the servomotors is shown on the left. The right side depicts the angle of the fin ray based on a simple sinusoidal input. The wave is not a perfect sine wave, but approximates one.

s_n stands for adjacent servomotor numbers (values from 0 to 5), and t represents the timesteps.

In the second experiment, the wave patterns that have evolved using the Fourier series were evaluated. In this case, the mutable parameters are transformed into outputs through summing up 5 sinusoidal functions of the type in [Equation 9.2](#).

Evolutionary algorithm

The evolutionary approach was divided into a control system and an evolutionary algorithm. The evolutionary algorithm made use of functions from the Distributed Evolutionary Algorithms in Python (DEAP) library, which included an implementation of CMA-ES (Fortin et al., 2012). The CMA-ES implementation implemented a population size of 10 and ran for 20 generations. Covariance Matrix Adaptation Evolution Strategy was able to find solutions in 20 generations similar to running a normal generational evolutionary algorithm for 100 generations, which was advantageous for limiting the duration of the experiments. Our CMA-ES implementation included an initial standard deviation value of 50 and a centroid value of 125 for every gene (half the maximum value of the bytes in the genome).

Controller system

An Arduino Mega 2560 controlled the robot by actuating the servomotors and received the sensor readings of the ultrasonic distance and infrared sensors. Through serial communication, a genome was uploaded from a PC running the evolutionary algorithm to the Arduino Mega. The Arduino Mega evaluates an individual using the genome it received. This evaluation consists of:

1. Move robot to the starting position (by using a manually coded swimming behavior)
2. Move the servos to a central position and wait for six seconds (this delay was implemented to prevent the overheating of servos and reduce waves in the tank)
3. Evaluate genome for 10 seconds
4. Send back a fitness value based on the distance the robot has traveled within the 10 seconds

All steps take roughly between 20-30 seconds for one individual, depending on how far the robot was able to swim. When the same genome was evaluated multiple times, the error difference in fitness was negligible (standard deviation of samples of size 4 was less than 1% for each run). Each individual was therefore only evaluated once.

There was a 20 ms delay inserted between each timestep for updating the servo angles. Five hundred timesteps were done for each individual. The fitness value of each individual is calculated as a summation of the ultrasound distance measurements at every consecutive update of the servo positions. At each timestep, the ultrasonic distance sensor initiates a sound pulse and measures the time difference between the pulse and echo. This time interval becomes higher the further the robot moves away from its initial position. The fitness value for a controller that is not moving robot lies at around $100 \cdot 10^4$. At the start of the evaluation of a genome, the entire wave pattern for each servo was calculated for each timestep. This required six arrays to store 500-byte values derived from the genome. Although this occupies significant memory on the Arduino Mega board, it circumvents doing calculations on the spot that might have caused an additional delay between every timestep. Such a delay was, however, caused by the ultrasound sensor, which required an 8 microsecond delay for measuring the distance.

Experiments

Since earlier examples of robotic knifefish have been able to swim with only a single sinusoidal wave function as a control signal for the fin, experiments were conducted where the genome was reduced to three bytes that translate into the frequency, phase and amplitude of a single sine function. We tested whether evolution is able to efficiently optimize these three parameters for increased swimming speed. Our second set of evolutionary experiments evaluate functions that are generated from all 15 mutable parameters and yield the first five terms of a Fourier series. This is done to see whether an arbitrary periodic function can increase the performance compared to a single sine wave. For both sets of experiments, whether evolution will find swimming behaviors similar to the ones of actual knifefish, and if the performance of the evolved controllers can rival a manually programmed controller, were also tested.

For both the sinusoidal and the Fourier series approach, 5 evolutionary runs were done with the exact same hardware setup. Since the slightest change in hardware and the environment can influence evolutionary runs drastically, all 10 runs were done consecutively. A manually coded swimming behavior is used as a baseline to compare with the evolved controllers. This behavior was the fastest swimming behavior we were able to find by manually adjusting the genome parameters during a two-hour trial session with the platform. Its control function is:

$$\alpha_n = 40 \cdot \sin((64 \cdot t) + (100 \cdot s_n)) \quad (9.3)$$

These control parameters correspond to a genome with the following three bytes: 255 for the amplitude, 64 for the phase, and 100 for the frequency.

9.2.3 Comparing Behaviors of the Robot with Actual Knifefish

Bale et al. (2015) found that diverse groups of aquatic animals that use median/paired fin swimming, including knifefish, have evolved a similar optimal swimming strategy. More specifically, the result of dividing the length of an undulation on the fin by the mean amplitude of undulations along the fin, during steady swimming, consistently yields around 20. This wavelength, which maximizes the force generated by the body and the swimming speed, is referred to as the optimal specific wavelength (OSW). The specific wavelength (SW) of the evolved undulation patterns was therefore calculated to compare them with the swimming behaviors of the knifefish. The SW was calculated by dividing the wavelength of

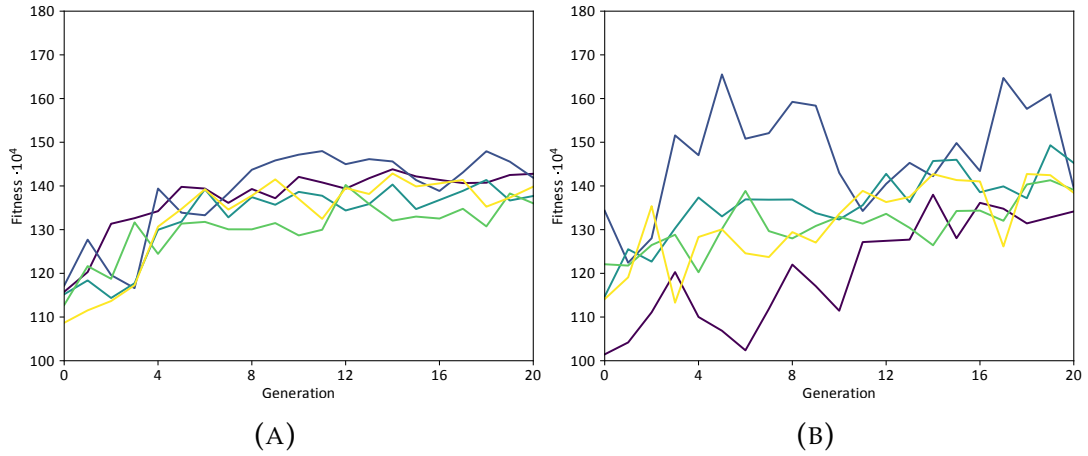


FIGURE 9.7: **Evolutionary progressions of 5 runs.** The sinusoidal approach (a) and the Fourier series approach (b) showing the maximum fitness (hall of fame) of the evolutionary runs.

undulation λ by the average amplitude of oscillation \tilde{a} . In general, this average amplitude \tilde{a} is given by

$$\tilde{a} = h_{mean} \sin(\theta_{max}^{avg})/2 \quad (9.4)$$

where θ_{max}^{avg} is the mean maximum angle of excursion of the fin rays and h_{mean} is the mean height of the fin.

9.3 Results

9.3.1 Performance Analysis

After running CMA-ES for 20 generations using the sinusoidal and the Fourier series approaches, different wave patterns were acquired. Both evolutionary progressions of the 5 runs of each approach (Figure 9.7) evolved decent swimming behaviors, though the Fourier series evolutionary progressions seem to have more variation in performance and did not plateau as clearly as the sinusoidal evolutionary progression. This corresponds to a larger, perhaps more convoluted, search space when evolving Fourier series.

The periodic control signals that have evolved in the sinusoidal approach are similar to each other, while the best individuals of the Fourier series exhibit more erratic wave patterns (Figure 9.8). Investigating the

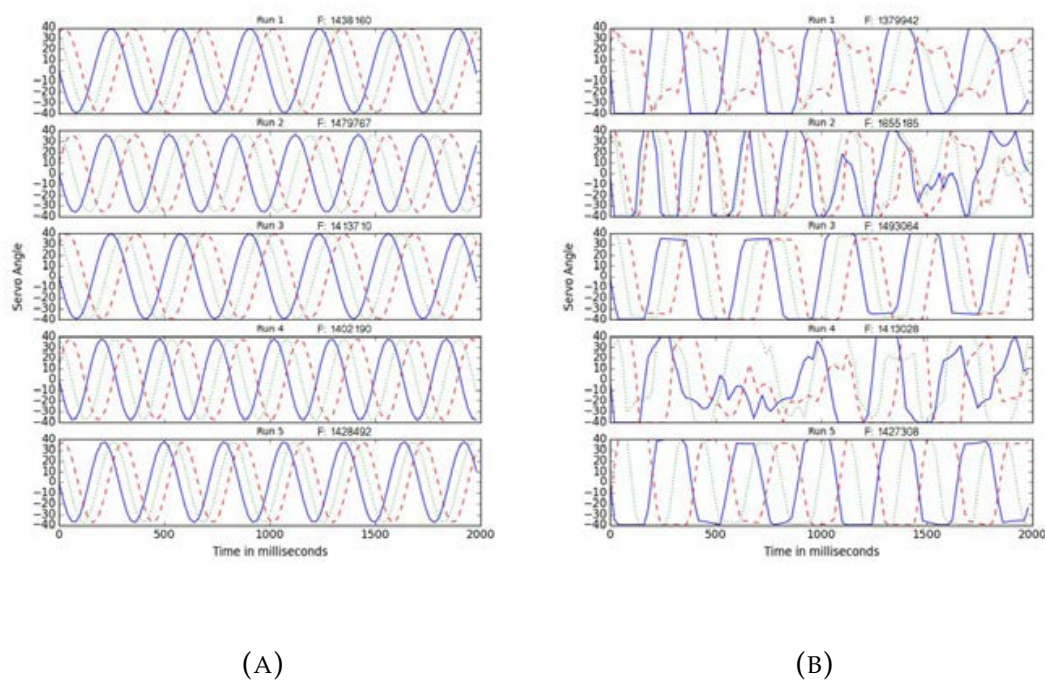


FIGURE 9.8: The best-evolved wave patterns in 5 distinct evolutionary runs using the sinusoidal approach (a) and the Fourier series approach (b). The graphs display two seconds of a resulting wave from each genome. The blue line represents the trajectory of the first servomotor, while the green dotted and red dashed lines depict the positions of servos 2 and 3, respectively. The trajectories of servo 4, 5 and 6 are not depicted. The difference in the wave of different servos visible in some of the Fourier series is due to including potentially high frequencies and querying the function every 20ms.

individual wave patterns and their corresponding fitness values, the best individual evolved in the Fourier series has a significantly higher fitness value than the others.

In [Table 9.1](#), the evolved swimming behaviors of the best candidates were compared to see whether the OSW ratio also applies here. The approximate wavelengths of the traveling waves were obtained from ventral-view video recordings of the robot with the best candidates and the manually coded behavior controlling its swimming. The average amplitude of oscillation was calculated from [Equation 9.4](#) using a maximum angular excursion of 28 degrees (derived from video recordings),

as was that the fin height is 7 cm. The average travel speeds were also measured from video recordings of the manual behavior and the best-evolved individuals being replayed on the robot (Bale et al., 2015). The black ghost knifefish that was the inspiration for our robot has an SW of around 18 (Bale et al., 2015). From Table 9.1 it can be seen that the best-evolved sinusoidal controller has a specific wavelength of 16, i.e., it approximates, but is lower than, the optimal specific wavelength found by Bale et al. (2015). Although our manually programmed controller has a SW of 17 and comes closest to the actual knifefish, in reality it performed considerably worse than most of the evolved controllers (see Table 9.1).

TABLE 9.1: **Specific wavelengths and travel speeds of behaviors.** The evolved behaviors resulted in wave patterns with varied wavelengths and speeds. (Wavelength of Four. (Run 4) has been omitted as the wave function was too erratic for it to be measured)

Genome	Wavelength	SW	Speed (cm/s)
Manual	28 cm	17	3
Sine (Run 1)	26 cm	16	8
Sine (Run 2)	23 cm	14	6
Sine (Run 3)	26 cm	16	6
Sine (Run 4)	23 cm	14	6
Sine (Run 5)	24 cm	15	8
Four. (Run 1)	26 cm	16	4
Four. (Run 2)	26 cm	16	2
Four. (Run 3)	24 cm	15	5
Four. (Run 4)	-	-	5
Four. (Run 5)	22 cm	13	1

Being able to evolve wave patterns to control the swimming behavior of the robot is of limited use if their phenotype cannot be reproduced. Since the robot was slightly worn down after many different experiments, and after having replaced several malfunctioning servomotors, we evaluated the performance of the evolved wave patterns again. When comparing the evolved Fourier series wave patterns with the evolved sinusoidal wave patterns, it can be seen that the sinusoidal wave patterns also outperform the manually encoded wave pattern significantly in terms of fitness value (Figure 9.9). Though this could have been caused by many factors, it seems that a sinusoidal function is a more robust general approach that might be suboptimal but resilient to morphological/environmental change

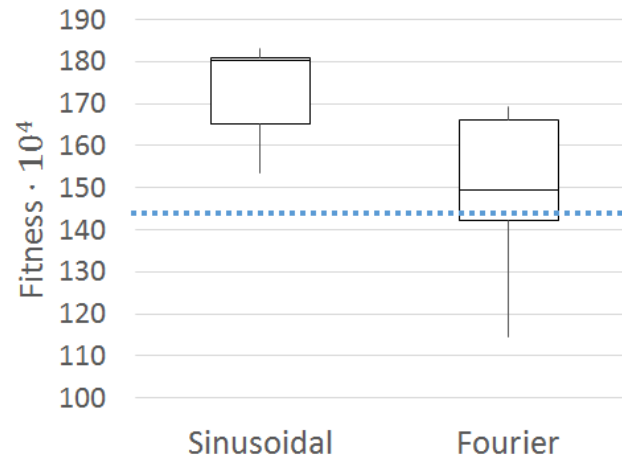


FIGURE 9.9: **Performance difference between the best-evolved sinusoidal and Fourier series individuals.** The box plot displays the quartiles of the best individuals of the 5 runs of the sinusoidal approach and the Fourier series approach. These results were obtained from replaying the best genomes of the different approaches using a patched-up version of the robotic fish (i.e., where the servomotors had been replaced). The blue dotted line represents the baseline performance of our manually encoded genome.

9.3.2 Phenotypic Analysis

To analyze the type of behaviors that evolved, the position of the tip of the fin rays was tracked in the best-evolved individuals using footage taken from a ventral view of the robot (Figure 9.10). This tracking was done to analyze the actual undulation patterns across the fin as opposed to the calculated control patterns. Looking at the best-evolved individuals from both the Fourier series and the sinusoidal approach, the wave propagates strikingly similar along the fin of both individuals. The phase and frequency between these two individuals are different, though the sinusoidal wave pattern generates the same wavelength with a higher frequency. The sinusoidal wave pattern makes roughly six undulations, while the Fourier series makes five.

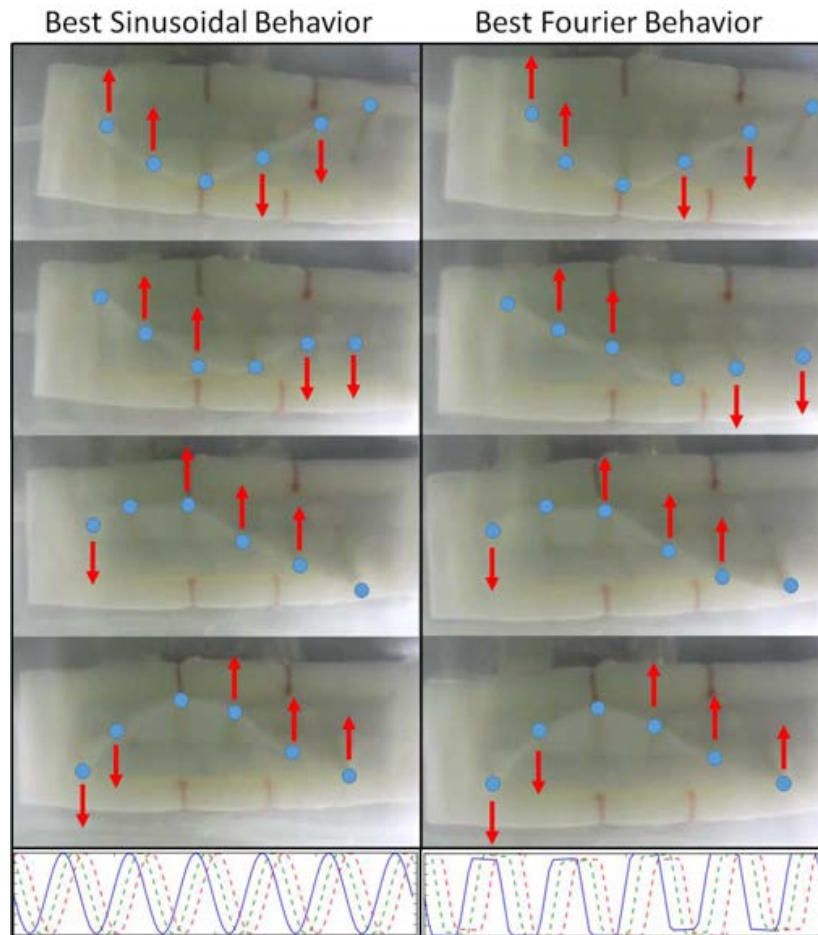


FIGURE 9.10: **Evolved wave patterns.** The wave patterns of the best (highest fitness) evolved sine wave and Fourier series as seen from below. Both waves are almost identical to one another, having a wavelength that is slightly larger than the length of the fin. The blue dots illuminate the tips of the fin rays, while the red arrows depict the motion of the individual fin rays. The function plots below correspond to the fin undulations depicted above and are the best *reproducible* evolved wave patterns shown in [Figure 9.8](#) (Sine [Run 1] and Fourier [Run 3])

9.4 Discussion

Covariance Matrix Adaptation Evolution Strategy proved to be an efficient method for automatically evolving the swimming behavior of our soft swimming robot inspired by the ghost knifefish. Although the search

space was quite small, failing hardware is usually a problem that makes evolving physical robots arduous. Predefining the controller by only utilizing periodic wave functions and only running CMA-ES for a brief period was enough to generate efficient swimming behavior. One of the main challenges when evolving physical robots is how to deal with malfunctioning hardware. Considering a death toll of 17 servomotors during these experiments, using CMA-ES seemed much more viable compared to initial experiments with a generational evolutionary algorithm that took almost five times longer to get similar results to the CMA-ES approach. Covariance Matrix Adaptation Evolution Strategy was, however, not completely resistant to the hardware failures of the servomotors, as discussed in [Text Box 9.7](#).

The robotic platform presented in this chapter was constrained by predefined functions and the limited movement sets acquired in the evolutionary runs. However, the presented robot fish could evolve many different behaviors that the knifefish is also capable of. This could make it a viable option for autonomous underwater vehicles. A next submersible iteration of the fish could evolve vertical thrust through sending counter-propagating waves towards and away from the center of the fin, canceling out longitudinal forces, as discussed by (Curet et al., 2011a). A selection of these behaviors can be evolved and encapsulated in a fixed environment, removing manual programming of the behavioral repertoire.

Zoological studies of knifefish kinematics have shown that the wavelength of the propagating wave varies across the fin during steady swimming (Youngerman et al., 2014). Given that the swimming behavior of the knifefish has been optimized through natural evolution, implementing this feature in the encoding of the controller could lead to better performance. Additionally, this could be accomplished by using a *compositional pattern-producing network* (CPPN; Stanley 2007) with servo number and time as inputs. A similar approach has previously been used successfully to generate the oscillatory controller for a quadruped robot (Morse et al., 2013). To discover a greater variety of controllers that perform well, novelty search (Lehman et al., 2011) or other diversity-enhancing methods can also be applied instead of a goal-directed approach that is often prone to premature convergence or over-fitting. Another aspect worthy of further inquiry is the materials used for the fin. It is possible that a material with another elastic modulus will better exploit the interactions with the water to facilitate the emergence of dynamics that aid in swimming.

With this robotic platform, we were able to automatically evolve the behavior of an intuitively functional soft robot using CMA-ES. Considering

the increasing advances of automated manufacturing methods and readily available materials to create detailed robots with various features, we think this evolutionary approach on physical soft robots can become viable as a tool for directly optimizing the behavior of the physical systems.

Text Box 9.7: Servo malfunctions

One of the main issues of the experiments demonstrated here was the malfunctioning of the servomotors as described by the ‘death toll’ of 17 servomotors. It should be considered for any future implementations that the servomotors are robust enough to survive the evolutionary progression. Meaning that we should have bought more resilient servomotors for the experiments. However, considering this limitation, CMA-ES was an essential implementation since other implementations took longer to be achieved similar fit individuals. The implementation of CMA-ES relieved us from having to tweak the parameters of the evolutionary algorithm to get the correct results. In order to have comparable results of multiple evolutionary runs, it was a requirement that the same set of servomotors survived for all the ten evolutionary runs of the sinusoidal approach and the fourier approach. However, in many cases a servo failure actually occurred. When plotting the percentiles and the average fitness values of the population in this run the resulting evolutionary progression contains a sudden change in performance as depicted in [Figure 9.11](#).

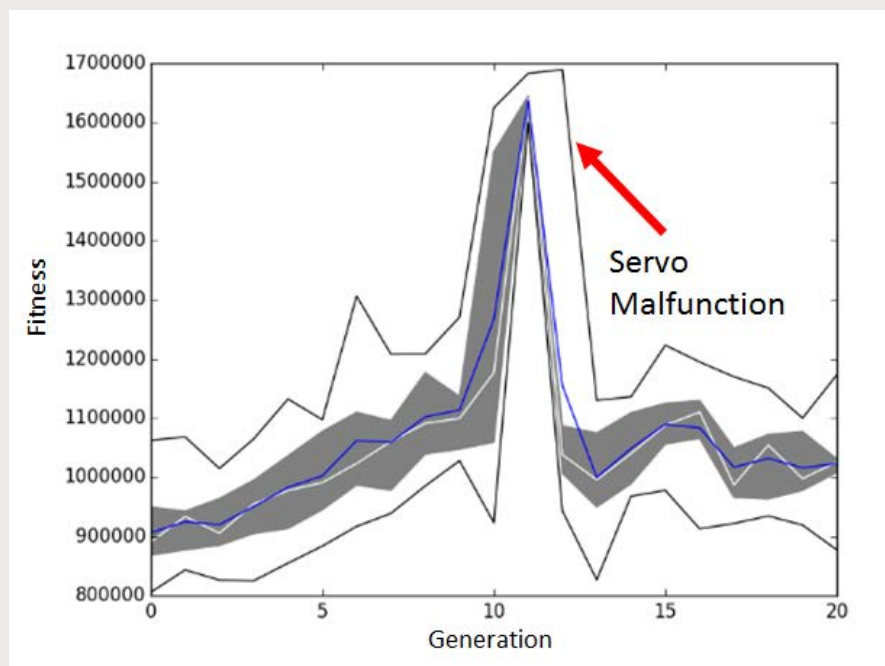


FIGURE 9.11: The evolutionary progression of a Fourier Series with a servo malfunction occurring around generation 10. The grey area represents the 25-75 percentiles of the population, the upper and lower solid lines the maximum and minimum fitness values in the population, the blue line the average fitness value of the population and the white line is the median fitness value of the population.

9.5 Conclusion

This chapter has demonstrated that evolving the controller for a knifefish-inspired soft robot is feasible directly on the physical robot. The majority of the evolved behaviors outperformed a hand-designed controller in terms of speed. Additionally, evolution was able to exploit the dynamic properties of the flexible material to produce feasible swimming strategies for the robot that have similar phenotypes but different genomes. Evolutionary experiments on physical robots, which have so far only been applied to traditional non-soft robots, are especially relevant for soft robots that are difficult to simulate computationally. In the future, the presented approach could be combined with more explorative search methods, such as novelty search and different fish models, to solve tasks for which even a simple hand-designed controller is an infeasible option.

Chapter 10

Discussion

Complex now [the mechanical reproductive system], but how much simpler and more intelligibly organized may it not become in another hundred thousand years? or in twenty thousand? For man at present believes that his interest lies in that direction; he spends an incalculable amount of labour and time and thought in making machines breed always better and better; he has already succeeded in effecting much that at one time appeared impossible, and there seem no limits to the results of accumulated improvements if they are allowed to descend with modification from generation to generation. It must always be remembered that man's body is what it is through having been moulded into its present shape by the chances and changes of many millions of years, but that his organisation never advanced with anything like the rapidity with which that of the machines is advancing. This is the most alarming feature of the case, and I must be pardoned for insisting on it so frequently.

– Samuel Butler, Erewhon

Artificial life—the paradigm concerned with the recreation of biological phenomena observed in life—is slowly progressing as a field, enabling its emergence. From the perspective of artificial life, this thesis has provided an overview on theory of evolutionary dynamics and displayed both simulated (Chapters 3, 5, 6 and 7) and physical (Chapters 7, 8 and 9) experiments that attempted to engender the recreation of phenomena of life using an evolutionary approach. The results of these experiments not only teach us about life, but also promote the development of future technologies. Researchers of evolutionary robotics simply try to nudge the evolutionary search process to give rise to a potentially wide spectrum of robots—making us the watchmakers. We allocate building blocks and divide laborious tasks across isolated functionalities of robots. Some functionalities have straightforward solutions, like rotating a wheel through a DC motor, while others require crossing convoluted fitness

landscapes. Machines have already been designed to, for example, exhibit optimal arithmetic, and these attributes could simply be implemented as a module, forming one more building block in our artificial system. The composition of the various theories, methods, and materials together can build a cascade of incremental steps engendering the emergence of a robotic conglomerate that can be optimized toward any objective. Toward this objective, this thesis has discussed the role of death on the evolvability of a population, the evolution of modular robots, and using evolutionary computation to directly evolve behavior on a physical robot.

10.1 Evolutionary Dynamics of Intrinsic Mortality

To improve our understandings of life and its emergent phenomena, it is important that we understand as much as we can about its evolutionary dynamics. For evolutionary robotics, this particularly means the manners in which we can traverse fitness landscapes. For robotics applications there may be many landscapes that are interconnected and could potentially be isolated. Additionally, some landscapes might be dependent on others, where a seemingly unrelated stepping stone needs to be found before the population can be further evolved (Stanley et al., 2015). For example, a robot might need to see before it is able to grasp objects. Although sight might be a direct evolutionary advantage when discovered, other potential strategies can emerge from precursors that had different functionalities when they initially appeared, such as the feathers that gave rise to flight in birds. Evolution therefore has a seemingly random explorative nature, and thus has the capacity to exploit any type of property that deems sufficient. The competence of a population to traverse the fitness landscapes empowers evolvability but is likely produced due to happenstance.

Chapter 3 has specifically shown that one factor, mortality, drastically changes the evolvability of a population when implemented on deceptive fitness landscapes. I contemplate that deceptive landscapes in particular are the most important analogy to natural systems. In nature, these deceptive landscapes are dynamically shaped by the environment. This dynamic aspect can come from us changing, removing, or adding an objective to the artificially simulated environment or physical robot. The mortality implementation shows how a population can traverse the top of the fitness landscape. This *hill-hugging* phenomenon—emergent from a simple decrease in selective pressure on the fittest individuals—might well

be a powerful way through which artificial systems can find solutions by means of apparent drift. Most algorithmic implementations instead rely on diversification methods through the insertion of random individuals (Schmidt et al., 2011), or the active search for novelty (Lehman, 2012). Though diversification is promoted when using mortality, we could postulate that this implementation is *functional novelty through death*, or *mortality-induced adaptive radiation*.

As discussed in Kowald et al. (2016), senescence might well be emergent through a trade-off between advantageous cellular machinery (disposable soma) or a coincidental effect of genes that are early-life fitness promoters and detrimental later in life (antagonistic pleiotropy). Proving that senescence increases algorithmic performance does solicit the advocates of non-programmed aging to take into account that a mortal population might simply be more effective on a time-scale of hundreds of generations. A difficult question still remains is still: when is senescence an evolvable feature, rather than merely a feature that increases evolvability? To answer this question, future experiments should make the parameters of mortality themselves evolvable. Depending on the fitness landscapes that are being used, intrinsic mortality is likely not advantageous on landscapes with an obvious gradient towards a maximum. Even when there is a landscape that makes mortality advantageous, individuals exhibiting no mortality could certainly take over the population in the long run. When considering aging individuals to be cooperators, and immortals to be defectors, the defectors likely have a direct advantage over cooperators creating a prisoner's dilemma scenario. A prematurely converged population stuck in a local optima has the chance to outcompete mortal populations that have not yet found better-fit solutions in the landscape, though is likely in trouble when the environmental niche it occupies changes. Nevertheless, intrinsic mortality affects evolutionary search and is thereby likely an evolvable trait.

10.2 Evolving Modular Robots

The importance of using modularity in robots is the isolation of the functionality in specific parts of a robot. Additionally, implementing generative encodings can engender a recursive succession of modules. This recursion of simple parts could in turn give rise to complex phenotypic traits. This process of recursion, or genotype to phenotype mapping, has been implemented in [Chapter 5](#) for creating artificial plant morphologies. Its specific importance has subsequently been discussed in [Chapter 6](#) for the acquisition of locomotion in modular robots. Although the L-Systems

that were implemented may not be the most efficient type of generative encoding, they are a simple method to quickly generate a patterned phenotype that can be represented as a hierarchical tree.

To ensure that the robots evolved in a simulator are feasible, the type and number of modular parts available can be specified to determine the morphological bounds of the evolutionary search process. This evolutionary search process can furthermore be altered by simply adding modules with their own unique attributes to the same modular system with a different objective. The addition of solar panel modules is an example of changing the objective and building blocks of the evolutionary search process (Chapter 7). The application of solar panel modules could potentially be extended in artificial settings through implementing various solar panels that are optimal for different light intensities, potentially maximizing light absorption while minimizing resource requirements.

In a future simulation environment implementing different types of modules, specific modules of the robots could be subjected to evolution while others remain fixed. For example, if a bipedal robot has been constructed, the leg modules could be subjected to evolution potentially changing the size and control mechanisms of various parts. In turn, when a bigger brain or additional arms are incorporated into a robot, the evolutionary algorithm could specifically re-evolve the structure of the legs to compensate for the added mass and shifted center of mass. Hence, a feedback loop of quickly tweaking local parts of a robot can be achieved at various scales.

The methods that were presented were, however, limited by the mechanical properties and requirements of the modules. In the case of the presented modules, the mechanical and electrical connections were distributed through the same connector site. From an engineering perspective this approach is flawed since a disconnection—even if temporary—of two modules would result in the inactivation of the modules, which in turn briefly lose power and communication. The future design of modules for a modular robot could consider this by separating the electrical wiring and the mechanical connections. This can be done through additionally having magnet-based wires distribute power and communication to other modules. However, this implementation would also make the robot more difficult to construct, especially automatically. In addition, mechanical connections close to the center of the robot would require more power and mechanical strength to actuate parts of the body as compared to parts that are more distant. Recursive modules that change in scale and strength are therefore likely contributors to more efficient modular robots. Despite the aforementioned limitations of the presented

work, the modular approach to evolving robots is an efficient way in which morphological designs can quickly be altered and explored in robotics.

The use of generative encodings can lead to phenotypes that are more transferable to the real world. As generative encodings can allow the same behavior to occur in multiple modules in a system, this recurrence can mean that the solutions found are transferable because the result of the patterned behavior is likely more generic. It is not fine-tuned to its environment, as would be more likely the case in a direct encoding. Fine-tuning in the simulator, in contrast, may lead to the evolution of a behavior that is not transferable. In addition, a generative encoding is able to make a wider sweep across the search space, while a direct encoding increments more locally. This wider sweep might also contribute to the generative encoding finding strategies that are more generic. Combining the approaches might however be an even better method for designing robots: initially making a somewhat large sweep with a generative encoding to find generic solutions and afterwards fine-tuning the phenotypes using a direct encoding. By this approach, a generative encoding can evolve the robots in a simulator, while the direct encoding can be implemented to tune the real robot.

The presented approach to evolving modular robots has demonstrated that modular robots can automatically emerge by giving the simulator a specific objective, be it energy acquisition or locomotion. With the addition of being able to reconfigure the modular robot physically, the approach considers a genotype to phenotype mapping emphasizing the morphology and decentralized control while also working toward a minimizing the reality gap. Being able to correctly abstract the morphology and control of robots, as well as being able to efficiently traverse the search space, has potential to yield a plethora of robot types optimized for any objective. This could potentially be transferred to real world scenarios in which a user specifies the environment and the objective and the evolutionary plugin automatically evolves the morphology and control based on the available modules.

10.3 Evolution of Physical Systems

Chapter 8 has given a brief example of a methodology to automatically assemble modular robots using fiducial markers and a UR5 robot arm. The results depicted in this approach are preliminary, but are aimed towards the automated production and evaluation of modular robots. Since the modules presented in this chapter were not active, some additional hurdles need to be overcome before a complete autonomous modular robot

assembler can be created. Moreover, though easy to use by researchers, the reconfiguration of modular robots can be difficult since disconnecting modules requires some force due to the strong magnetic connections between modules. Many improvements can still be made. These include, for example: using active grippers, having mechanical connectors, adding a power source module perhaps with wireless communication capabilities, and adding a docking station to automatically charge the robot and reset its position.

The evolution of the soft robot inspired by the black ghost knifefish has been presented in [Chapter 8](#). In this case, due to hardware limitations, it became apparent why the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) was so efficient. Implementing sinusoidal wave functions that were mutable quickly led to the discovery of various undulation controls. Evolving a Fourier series and a regular single sine wave function showed a similar evolutionary progression, the Fourier series being a bit slower. After replacing all of the servos with new ones, a drastic change in performance was noted. On average, solutions found by the Fourier series performed worse than the solutions found by the sinusoidal controller. The sinusoidal approach being simpler probably indicates that a more generic solution is more robust and is not likely to exploit specific behavioral characteristics that the Fourier series may have done. However, the resulting behavior in the Fourier series might also have emerged due to the exploitative nature of CMA-ES. Implementing another strategy, like the mortality rate, might be less exploitative and instead promote diversity.

There is an argument to be made for bypassing the simulator and implementing evolutionary approaches on robots directly, taking the embodied robotics approach. For bioroboticists, an organism can be physically reconstructed and its behavior can subsequently be evolved. This would elucidate not only principles of biological (perhaps extinct) organisms, but also aid in the construction of effective robots (Long, 2012). Moreover, the embodied approach—either implementing evolutionary computation directly on robots or the embodiment of the evolutionary algorithm itself in embodied evolution—does not have a reality gap. Simulators frequently exploit inaccurate dynamics/physics that constitute a major problem when dealing with the reality gap. Having efficient evolutionary methods that can adequately explore the fitness landscapes is thus a prominent technique for acquiring behavior in robots directly.

10.4 Concluding Remarks

This thesis has discussed death, modularity, and physicality for the ultimate acquisition of robot morphologies and control through evolutionary computation. Programmed death induces evolvability on deceptive fitness landscapes and might thereby be an explanation for senescence. Beyond being a mere explanation, it can prevent populations in evolutionary computation from prematurely converging, enabling more solutions to be found. Generative encodings in modular robots increase the performance of evolving modular robots and illustrate the importance of the genotype to phenotype mapping and how development is beneficial. While the behavior in robots can be transferred from the simulator when the phenotypes are simple, directly evolving behavior is also feasible in the absence of simulation models, bypassing the simulator. The exploration vs. exploitation trade-off, genotype to phenotype mapping, and physicality are interlinked concepts that, in conjunction, can be implemented in the generation of even better-adapted robots.

Moreover, the issues presented in this thesis are relevant to the general paradigm of AI. Conventional methods used in artificial intelligence implement learning strategies that update control parameters usually in a directed manner, following gradients and promoting novelty. Since evolutionary computation mainly works on the phylogenetic time scale, the incorporation of concepts in artificial intelligence on the ontogenetic time-scale can provide us with a toolset for improving existing machines and algorithms in general. The implementation of evolutionary computation can be beneficial for a wide range of applications, but especially for the generation of complex robots from mere building blocks.

All biological machinery has evolved through the blind process of evolution. The elegant evolutionary dynamics have in turn formed the backbone of this thesis, where they have been utilized with the aim to create robots by defining the environment, the task, the building blocks, and the encodings. The presented methodology is valuable for both science and engineering, where self-adaptation through evolution is key. Prospective improvements on the presented methodologies may alleviate the design burdens of watchmakers, such that instead of being designed, robots are evolved.

Appendix A

Full Spatial Model Pseudocode

The spatial model updates the biomass of all cells at every iteration¹, where g represents the cycles and n all of the cells on the grid. If the cell that is being checked is not part of the evolvable population ($\neq X$) then the biomass of the cell is updated depending on the biomass production rate set by the initial parameters. The cells that are part of the evolvable population can have a developmental trigger and a mortality trigger. If the cell should develop, its genome is swapped with another one that is stored in the subsequent developmental phase. If the cell is mortal, and the terminal age is reached or a random death factor function returns true, the cell is deleted and part of its biomass is left as plant biomass. After the developmental and mortality triggers, the cell can move, eat, and reproduce. The data of the entire grid is saved every 100 cycles.

¹The full implementation can be found here: <https://github.com/FrankVeenstra/ALife2018>

Algorithm 2: Spatial Model Extended Pseudocode

```

Initialize and Evaluate population  $P = \{X_1, \dots, X_N\}$ ;
while  $g := 1$  to  $G_{max}$  do New Cycle
  for  $n := 1$  to  $N_{max}$  do Update Cell
    if  $Cells[n] \neq X$  then
      |  $Mass :=$  Biomass Production Rate;
    else
      |  $Mass :=$  -Mass Loss Rate;
      if Developmental Trigger then
        | Develop Cell;
      end
      if  $Cell[n] \neq immortal$  then
        | if  $Cell[n].age > Cell[n].maxAge \ \|\ Cell[n] = Selected\ by$ 
          | Probability then
            | Remove  $Cell[n]$ ;
          end
        end
        Move;
        Eat;
        if Reproduction then
          | Reproduce;
          | Mutate Offspring;
          | Evaluate Offspring;
        end
      end
    end
  end
  if  $Cycle \% 100 == 0$  then
    | Log State
  end
end

```

Appendix B

UML Evolutionary Robotics Plugin

The UML of the robotics plugin is presented in [Figure B.1](#). The UML illustrates the relationships between the genetic algorithm, genome, morphology, modules, control, neurons, environment, and V-REP. Five factory patterns, from which two were dependent on V-REP, were implemented.

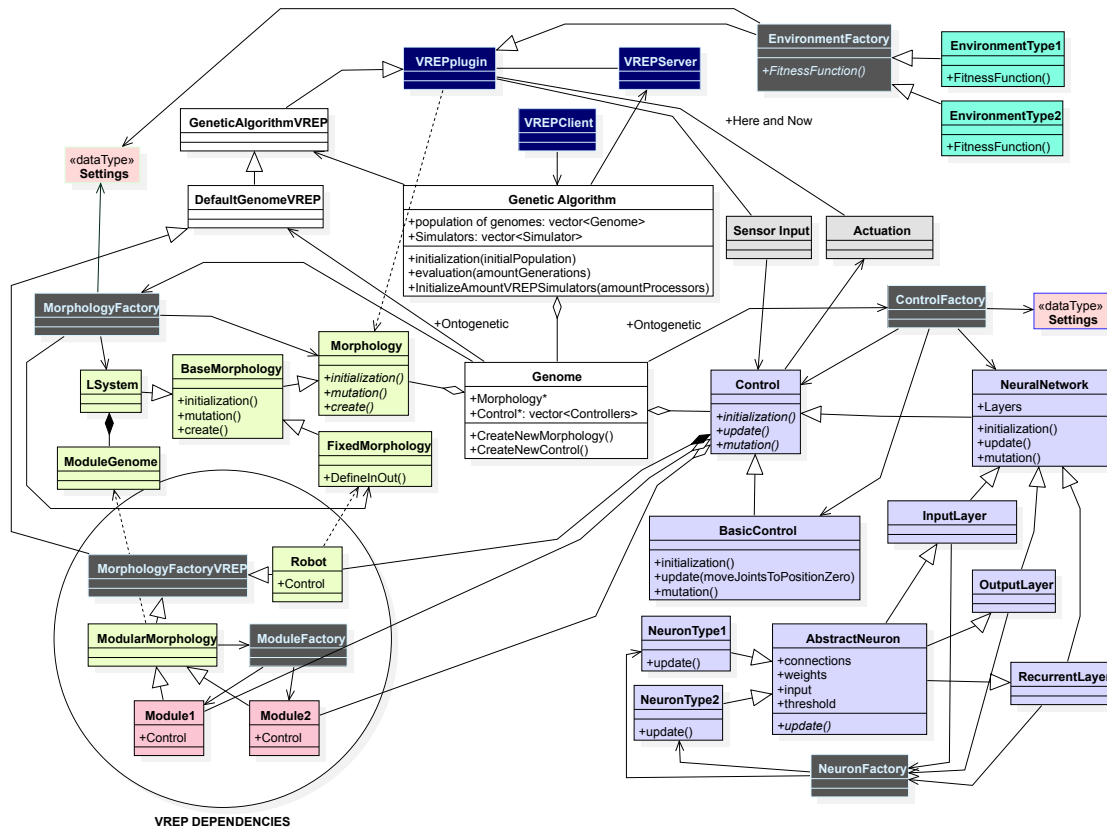


FIGURE B.1: The UML diagram of the Evolutionary Robotics Plugin. The dark blue boxes with white letters depict the main functions. The white boxes are the features of the genetic algorithms which were the evolutionary algorithm and the genome. To the left of the genome, in yellow, the classes relevant to the morphology are depicted. On the other side, blue depicts the control architecture of the neural network that was implemented in the modules/robots. Finally, in green the environment and corresponding fitness functions are defined. Dark grey boxes additionally present the factories enabling factory patterned instantiation of classes.

Bibliography

- Ackerly, D. D., S. A. Dudley, S. E. Sultan, J. Schmitt, J. S. Coleman, R. Linder, D. R. Sandquist, M. A. Geber, A. S. Evans, T. E. Dawson, and M. J. Lechowicz (2000). "The Evolution of Plant Ecophysiological Traits: Recent Advances and Future Directions". In: *BioScience* 50.11, pp. 979–995. DOI: 10.1641/0006-3568(2000)050[0979:TEOPET]2.0.CO.
- Afarulrazi, A. B., W. M. Utomo, K. L. Liew, and M. Zarafi (2011). "Solar tracker robot using microcontroller". In: *ICBEIA 2011 - 2011 International Conference on Business, Engineering and Industrial Applications*. Kuala Lumpur, pp. 47–50. ISBN: 9781457712807. DOI: 10.1109/ICBEIA.2011.5994256.
- Altenberg, L. (1994). "The Evolution of Evolvability in Genetic Programming". In: *Advance in Genetic Programming*, pp. 47–74. ISSN: 03032647. DOI: 10.1016/S0303-2647(02)00134-X.
- Anderson, R. C., J. B. Wood, and R. A. Byrne (2002). "Octopus senescence: The beginning of the end". In: *Journal of Applied Animal Welfare Science* 5.4, pp. 275–283. ISSN: 10888705. DOI: 10.1207/S15327604JAWS0504_02.
- Angeline, P. J., G. M. Saunders, and J. B. Pollack (1994). "An evolutionary algorithm that constructs recurrent neural networks." In: *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council* 5.1, pp. 54–65. ISSN: 1045-9227. DOI: 10.1109/72.265960.
- Aristotle (1910). "Book IV". In: *The History of Animals*. Trans. by D. W. Thompson.
- Atamian, H. S., N. M. Creux, E. A. Brown, A. G. Garner, B. K. Blackman, and S. L. Harmer (2016). "Circadian regulation of sunflower heliotropism, floral orientation, and pollinator visits." In: *Science (New York, N.Y.)* 353.6299, pp. 587–90. ISSN: 1095-9203. DOI: 10.1126/science.aaf9793.
- Athanasίου, K., B. C. Dyson, R. E. Webster, and G. N. Johnson (2010). "Dynamic Acclimation of Photosynthesis Increases Plant Fitness in Changing Environments". In: *Plant Physiology* 152.1, pp. 366–373. ISSN: 0032-0889. DOI: 10.1104/pp.109.149351.
- Auerbach, J. E. and J. C. Bongard (2011). "Evolving Complete Robots with CPPN-NEAT: The Utility of Recurrent Connections". In: *Proceedings of the 13th Annual Genetic and Evolutionary Computation Conference - GECCO'11*, pp. 1475–1482. ISBN: 9781450305570. DOI: 10.1145/2001576.2001775.

- Auerbach, J., D. Aydin, A. Maesani, P. Kornatowski, T. Cieslewski, G. Heitz, P. Fernando, I. Loshchilov, L. Daler, and D. Floreano (2014). "RoboGen: Robot Generation through Artificial Evolution". In: *Proceedings of the Fourteenth International Conference on the Synthesis and Simulation of Living Systems - Artificial Life'14*: pp. 136–137. DOI: [10.7551/978-0-262-32621-6-ch022](https://doi.org/10.7551/978-0-262-32621-6-ch022).
- Auerbach, J. E., G. Heitz, P. M. Kornatowski, and D. Floreano (2015). "Rapid Evolution of Robot Gaits". In: *GECCO'15*, pp. 743–744.
- Baldwin, J. M. (1896). "A New Factor in Evolution". In: *The American Naturalist* 30.354, pp. 441–451. ISSN: 0003-0147. DOI: [10.1086/276408](https://doi.org/10.1086/276408). arXiv: [arXiv: 1011.1669v3](https://arxiv.org/abs/1011.1669v3).
- Bale, R., I. D. Neveln, A. P. S. Bhalla, M. A. MacIver, and N. A. Patankar (2015). "Convergent Evolution of Mechanically Optimal Locomotion in Aquatic Invertebrates and Vertebrates". In: *PLOS Biology* 13.4, e1002123. ISSN: 1545-7885. DOI: [10.1371/journal.pbio.1002123](https://doi.org/10.1371/journal.pbio.1002123).
- Balster, H., P. W. Braun, and W. Köhler (1998). "Cellular automata models for vegetation dynamics". In: *Ecological Modelling* 107.2-3, pp. 113–125. ISSN: 03043800. DOI: [10.1016/S0304-3800\(97\)00202-0](https://doi.org/10.1016/S0304-3800(97)00202-0).
- Bear, M. F., B. W. Connors, and M. A. Paradiso (2016). *Neuroscience Exploring the Brain 4th edition*, pp. 304–307. ISBN: 0026397560. DOI: [10.1111/j.1365-313X.2005.02390.x](https://doi.org/10.1111/j.1365-313X.2005.02390.x).
- Belew, R. K., J. McInerney, and N. N. Schraudolph (1992). "Evolving Networks: Using the Genetic Algorithm with Connectionist Learning". In: *Artificial Life II* 10, pp. 511–547.
- Biewener, A. A. (2003). *Animal Locomotion*. Oxford University Press, p. 281. ISBN: 978-0-19-850022-3.
- Bolhuis, J. J. and L.-A. Giraldeau (2005). *The Behavior of Animals*, p. 515. ISBN: 9780631231257.
- Bonardi, S., M. Vespignani, R. Moeckel, J. van den Kieboom, S. Pouya, A. Sproewitz, and A. J. Ijspeert (2014). "Automatic generation of reduced CPG control networks for locomotion of arbitrary modular robot structures". In: *Proceedings of Robotics: Science and Systems*.
- Bongard, J. (2003). "Incremental Approaches to the Combined Evolution of a Robot's Body and Brain". PhD thesis, p. 193.
- Bongard, J. (2008). "Behavior Chaining : Incremental Behavior Integration for Evolutionary Robotics". In: *Artificial Life* 11, pp. 64–71.
- Bongard, J. and R. Pfeifer (2001). "Repeated Structure and Dissociation of Genotypic and Phenotypic Complexity in Artificial Ontogeny". In: *Proceedings of the Genetic and Evolutionary Computation Conference - GECCO'01* 1998, pp. 829–836.
- Bongard, J., V. Zykov, and H. Lipson (2006). "Resilient Machines Through Continuous Self-Modeling". In: *Science* 314.5802, pp. 1118–1121. ISSN: 1095-9203. DOI: [10.1126/science.1133687](https://doi.org/10.1126/science.1133687).

- Bongard, J. C. and R. Pfeifer (2003). "Evolving Complete Agents using Artificial Ontogeny". In: *Morpho-functional Machines: The New Species (Designing Embodied Intelligence)*, pp. 237–258. DOI: [10.1.1.26.4442](https://doi.org/10.1.1.26.4442).
- Bongard, J. C. and G. S. Hornby (2010). "Guarding Against Premature Convergence While Accelerating Evolutionary Search". In: *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation - GECCO'10*. ACM, pp. 111–118. DOI: [10.1145/1830483.1830504](https://doi.org/10.1145/1830483.1830504).
- Braitenberg, V (1986). *Vehicles: Experiments in Synthetic Psychology*. Cambridge: MIT Press. ISBN: 9780262521123.
- Bremermann, H. J. (1962). "Optimization through evolution and recombination". In: *Self-organizing systems* 93, p. 106.
- Brodbeck, L., S. Hauser, and F. Iida (2015). "Morphological evolution of physical robots through model-free phenotype development". In: *PLoS ONE* 10.6. ISSN: 19326203. DOI: [10.1371/journal.pone.0128444](https://doi.org/10.1371/journal.pone.0128444).
- Brooks, R. A. (1991). "New Approaches to Robotics". In: *Science* 253.5025, pp. 1227–1232. ISSN: 0036-8075. DOI: [10.1126/science.253.5025.1227](https://doi.org/10.1126/science.253.5025.1227).
- Budelmann, B. U. (1995). "Cephalopod Sense Organs, Nerves And The Brain: Adaptations For High Performance And Life Style". In: *Marine and Freshwater Behaviour and Physiology* 25.1-3, pp. 13–33. ISSN: 10290362. DOI: [10.1080/10236249409378905](https://doi.org/10.1080/10236249409378905).
- Cagnoni, S., M. Mirolli, and M. Villani (2014). *Evolution, complexity and artificial life*, pp. 1–280. ISBN: 9783642375774. DOI: [10.1007/978-3-642-37577-4](https://doi.org/10.1007/978-3-642-37577-4).
- Carter, A. J. and A. Q. Nguyen (2011). "Antagonistic pleiotropy as a widespread mechanism for the maintenance of polymorphic disease alleles". In: *BMC Medical Genetics* 12. ISSN: 14712350. DOI: [10.1186/1471-2350-12-160](https://doi.org/10.1186/1471-2350-12-160).
- Cells, K. (1997). "Phyllotaxis". In: *The Algorithmic beauty of plants*. Chap. 4, pp. 63–123.
- Cheney, N., R. MacCurdy, J. Clune, and H. Lipson (2013). "Unshackling Evolution: Evolving Soft Robots with Multiple Materials and a Powerful Generative Encoding". In: *Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation - GECCO '13*, p. 167. ISSN: 19318499. DOI: [10.1145/2463372.2463404](https://doi.org/10.1145/2463372.2463404).
- Cheney, N., J. Bongard, V. Sunspirial, and H. Lipson (2016). "On the Difficulty of Co-Optimizing Morphology and Control in Evolved Virtual Creatures". In: *Proceedings of the Artificial Life Conference 2016 - ALIFE XV*, pp. 226–234.
- Christensen, D. J., U. P. Schultz, and K. Stoy (2013). "A distributed and morphology-independent strategy for adaptive locomotion in self-reconfigurable modular robots". In: *Robotics and Autonomous Systems* 61.9, pp. 1021–1035. ISSN: 09218890. DOI: [10.1016/j.robot.2013.05.009](https://doi.org/10.1016/j.robot.2013.05.009).

- Christian, J. L. (2012). "Morphogen gradients in development: From form to function". In: *Wiley Interdisciplinary Reviews: Developmental Biology* 1.1, pp. 3–15. ISSN: 17597684. DOI: [10.1002/wdev.2](https://doi.org/10.1002/wdev.2).
- Chuang, C.-Y. and W.-L. Hsu (2010). "Multivariate Multi-model Approach for Globally Multimodal Problems". In: *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*. Portland: ACM, pp. 311–318. DOI: [10.1145/1830483.1830544](https://doi.org/10.1145/1830483.1830544).
- Clark, A. J., P. K. McKinley, and X. Tan (2015a). "Enhancing a Model-Free Adaptive Controller Through Evolutionary Computation". In: *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. GECCO '15. ACM, pp. 137–144. ISBN: 978-1-4503-3472-3. DOI: [10.1145/2739480.2754762](https://doi.org/10.1145/2739480.2754762).
- Clark, A. J., X. Tan, and P. K. McKinley (2015b). "Evolutionary multiobjective design of a flexible caudal fin for robotic fish". In: *Bioinspiration & Biomimetics* 10.6, p. 65006. ISSN: 1748-3190. DOI: [10.1088/1748-3190/10/6/065006](https://doi.org/10.1088/1748-3190/10/6/065006).
- Clune, J. and H. Lipson (2011). "Evolving 3D objects with a generative encoding inspired by developmental biology". In: *ACM SIGEVOlution* 5.4, pp. 2–12. ISSN: 19318499. DOI: [10.1145/2078245.2078246](https://doi.org/10.1145/2078245.2078246).
- Coevoet, E, A Escande, and C Duriez (2017). "Optimization-Based Inverse Model of Soft Robots With Contact Handling". In: *IEEE Robotics and Automation Letters* 2.3, pp. 1413–1419. ISSN: 2377-3766. DOI: [10.1109/LRA.2017.2669367](https://doi.org/10.1109/LRA.2017.2669367).
- Cook, O. F. (1906). "Factors of Species-Formation". In: *Science* 23.587, pp. 506–507. ISSN: 0036-8075. DOI: [10.1126/science.23.587.506](https://doi.org/10.1126/science.23.587.506).
- Corucci, F., M. Calisti, H. Hauser, and C. Laschi (2015). "Evolutionary discovery of self-stabilized dynamic gaits for a soft underwater legged robot". In: *Proceedings of the 17th International Conference on Advanced Robotics, ICAR 2015*, pp. 337–344. DOI: [10.1109/ICAR.2015.7251477](https://doi.org/10.1109/ICAR.2015.7251477).
- Corucci, F., N. Cheney, H. Lipson, C. Laschi, and J. C. Bongard (2016). "Material properties affect evolution's ability to exploit morphological computation in growing soft-bodied creatures". In: *Proceedings of the Artificial Life Conference 2016 (ALIFE XV)*.
- Cournède, P. H., A. Mathieu, F. Houllier, D. Barthélémy, and P. De Reffye (2008). "Computing competition for light in the GREENLAB model of plant growth: A contribution to the study of the effects of density on resource acquisition and architectural development". In: *Annals of Botany* 101.8, pp. 1207–1219. ISSN: 03057364. DOI: [10.1093/aob/mcm272](https://doi.org/10.1093/aob/mcm272).
- Cully, A., J. Clune, D. Tarapore, and J. Mouret (2015). "Robots that can adapt like animals". In: *Nature* 521, pp. 503–507. DOI: [10.1038/nature14422](https://doi.org/10.1038/nature14422).
- Curet, O. M., N. A. Patankar, G. V. Lauder, and M. A. MacIver (2011a). "Aquatic manoeuvring with counter-propagating waves: a novel locomotive strategy". In: *Journal of The Royal Society Interface* 8.60, pp. 1041–1050. ISSN: 1742-5689, 1742-5662. DOI: [10.1098/rsif.2010.0493](https://doi.org/10.1098/rsif.2010.0493).

- Curet, O. M., N. A. Patankar, G. V. Lauder, and M. A. MacIver (2011b). "Mechanical properties of a bio-inspired robotic knifefish with an undulatory propulsor". In: *Bioinspiration & Biomimetics* 6.2, p. 26004. ISSN: 1748-3190. DOI: 10.1088/1748-3182/6/2/026004.
- Darwin, C. (1845). *Journal of Researches into the Natural History and Geology of the Countries Visited during the Voyage of H.M.S. Beagle Round the World*. Vol. 3, p. 235. ISBN: 1619492814. DOI: 10.1038/129439a0. arXiv: arXiv:1011.1669v3.
- Darwin, C. R. (1872). *On the Origin of Species by means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*. 6th ed. John Murray. ISBN: 9780882709192. DOI: 10.1038/005318a0. arXiv: arXiv:1011.1669v3.
- Dawkins, R. (1978). "Replicator Selection and the Extended Phenotype". In: *Zeitschrift für Tierpsychologie* 47.1, pp. 61–76. ISSN: 14390310. DOI: 10.1111/j.1439-0310.1978.tb01823.x.
- Dawkins, R. (1986). *The Blind Watchmaker: Why the Evidence of Evolution Reveals a Universe without Design*. Norton & Company, Inc. ISBN: 0-393-31570-3.
- Dawkins, R. (1988). "Evolution of Evolvability". In: *Artificial Life, SFI Studies in the Sciences of Complexity* 4, pp. 201–220.
- De Beer, G. (1940). *Embryos and ancestors*. Oxford At The Clarendon Press., p. 136.
- De Jong, K. (2006). *Evolutionary Computation: A Unified Approach*, p. 272. ISBN: 0262041944.
- Dehal, P. and J. L. Boore (2005). "Two rounds of whole genome duplication in the ancestral vertebrate". In: *PLoS Biology* 3.10. ISSN: 15449173. DOI: 10.1371/journal.pbio.0030314.
- Desbordes, S. C. (2003). "The glypican Dally-like is required for Hedgehog signalling in the embryonic epidermis of *Drosophila*". In: *Development* 130.25, pp. 6245–6255. ISSN: 0950-1991. DOI: 10.1242/dev.00874.
- Doursat, R., H. Sayama, and O. Michel (2013). "A review of morphogenetic engineering". In: *Natural Computing* 12.4, pp. 517–535. ISSN: 15677818. DOI: 10.1007/s11047-013-9398-1.
- Eggenberger-Hotz, P. (1997). "Evolving Morphologies of Simulated 3d Organisms Based on Differential Gene Expression". In: *Proceedings of the 4th European Conference on Artificial Life (ECAL97)*, pp. 205–213. ISBN: 0-262-58157-4. DOI: 10.1007/s13398-014-0173-7.2.
- Ehleringer, J. R. and I Forseth (1980). "Solar tracking by plants." In: *Science* 210.4474, pp. 1094–1098. ISSN: 0036-8075. DOI: 10.1126/science.210.4474.1094.
- Eiben, A. E., R Hinterding, and Z Michalewicz (1999). "Parameter control in evolutionary algorithms". In: *Ieee Transactions on Evolutionary Computation* 3.2, pp. 124–141. ISSN: 1089-778X. DOI: 10.1109/4235.771166.
- Eiben, A. E., S. Kernbach, and E. Haasdijk (2012). "Embodied artificial evolution: Artificial evolutionary systems in the 21st Century". In: *Evolutionary Intelligence* 5.4, pp. 261–272. ISSN: 18645909. DOI: 10.1007/s12065-012-0071-x.

- Eiben, A. E. (2014). "Grand Challenges for Evolutionary Robotics". In: *Frontiers in Robotics and AI* 1, June, pp. 1423–1451. ISSN: 2296-9144. DOI: 10.3389/frobt.2014.00004.
- Eiben, A. E., N. Bredeche, M. Hoogendoorn, J. Stradner, J. Timmis, a. M. Tyrrell, and A. F. T. Winfield (2013). "The Triangle of Life: Evolving Robots in Real-time and Real-space". In: *Advances in Artificial Life, ECAL 2013*, pp. 1056–1063. DOI: 10.7551/978-0-262-31709-2-ch157.
- Eigen, M. and P. Schuster (1977). "A principle of natural self-organization - Part A: Emergence of the hypercycle". In: *Naturwissenschaften* 64.11, pp. 541–565. ISSN: 00281042. DOI: 10.1007/BF00450633.
- Faiña, A., F. Orjales, F. Bellas, and R. Duro (2011). "First Steps towards a Heterogeneous Modular Robotic Architecture for Intelligent Industrial Operation". In: *Workshop on Reconfigurable Modular Robotics at the IEEE/RSJ International Conference on Intelligent Robots and Systems - IROS'11*. IEEE/RSJ.
- Faiña, A., F. Bellas, F. López-Peña, and R. J. Duro (2013). "EDHMoR: Evolutionary designer of heterogeneous modular robots". In: *Engineering Applications of Artificial Intelligence* 26.10, pp. 2408–2423. ISSN: 09521976. DOI: 10.1016/j.engappai.2013.09.009.
- Faiña, A., L. T. Jacobsen, and S. Risi (2017). "Automating the Incremental Evolution of Controllers for Physical Robots". In: *Artificial Life* 23.2, pp. 142–168. ISSN: 1064-5462. DOI: 10.1162/ARTL_a_00226.
- Falster, D. S. and M. Westoby (2003). "Leaf size and angle vary widely across species: What consequences for light interception?" In: *New Phytologist* 158.3, pp. 509–525. ISSN: 0028646X. DOI: 10.1046/j.1469-8137.2003.00765.x.
- Floreano, D. and F. Mondada (1994). "Automatic creation of an autonomous agent: Genetic evolution of a neural-network driven robot". In: *From animals to animats* JANUARY, pp. 421–430. ISSN: 1098-6596. DOI: 10.1.1.51.6256. arXiv: arXiv:1011.1669v3.
- Floreano, D. and J. Urzelai (2001). "Evolution of plastic control networks". In: *Autonomous Robots* 11.3, pp. 311–317. ISSN: 09295593. DOI: 10.1023/A:1012459627968.
- Floreano, D. and C. Mattiussi (2008). *Bio-Inspired Artificial Intelligence*, p. 659. ISBN: 9780262062718. DOI: 10.1007/s10710-010-9104-3.
- Fogel, L. J., A. J. Owens, and M. J. Walsh (1966). *Artificial Intelligence through Simulated Evolution*. John Wiley & Sons, p. 170.
- Fortin, F.-A., F.-M. De Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné (2012). "{DEAP}: Evolutionary Algorithms Made Easy". In: *Journal of Machine Learning Research* 13, pp. 2171–2175.

- Goldsmith, T. C. (2008). "Aging, evolvability, and the individual benefit requirement; medical implications of aging theory controversies". In: *Journal of Theoretical Biology* 252.4, pp. 764–768. ISSN: 00225193. DOI: 10.1016/j.jtbi.2008.02.035.
- Goldsmith, T. C. (2014). *The Evolution of Aging*. 3rd ed. Crownsville: Azinet Press, p. 200. ISBN: 978-0978870904.
- Goldsmith, T. C. (2016). "Aging is programmed! (A response to Kowald-Kirkwood "Can aging be programmed? A critical literature review")". In: *Aging Cell*, p. 7. ISSN: 14749726. DOI: 10.1111/ace1.12510.
- Graf, J. and W. Banzhaf (1996). "Interactive Evolution for Simulated Natural Evolution". In: *Artificial Evolution*. Vol. 1063, pp. 259–272. ISBN: 3540611088. DOI: doi:10.1007/3-540-61108-8_43.
- Graham, J. M. and L. W. Wilcox (2013). *Plant Biology: Pearson International Edition*. 2nd ed. Essex: Pearson Education Limited, pp. 1–680. ISBN: 9781292042497.
- Greenman, J., O. Holland, I. Kelly, K. Kendall, D. McFarland, and C. Melhuish (2003). "Towards robot autonomy in the natural world: A robot in predator's clothing". In: *Mechatronics* 13.3, pp. 195–228. ISSN: 09574158. DOI: 10.1016/S0957-4158(01)00045-9.
- Guarente, L. and C. Kenyon (2000). "Genetic pathways that regulate ageing in model organisms". In: *Nature* 408.6809, pp. 255–262. ISSN: 00280836. DOI: 10.1038/35041700.
- Guettas, C., F. Cherif, T. Breton, and Y. Duthen (2014). "Cooperative co-evolution of configuration and control for modular robots". In: *International Conference on Multimedia Computing and Systems (ICMCS)*, pp. 26–31. DOI: 10.1109/ICMCS.2014.6911138.
- Habel, R., A. Kusternig, and M. Wimmer (2009). "Physically Guided Animation of Trees". In: *Computer Graphics Forum* 28.2, pp. 523–532. ISSN: 01677055. DOI: 10.1111/j.1467-8659.2009.01391.x.
- Hamilton, W. (1964). "The genetical evolution of social behaviour. I". In: *Journal of Theoretical Biology* 7.1, pp. 1–16. ISSN: 00225193. DOI: 10.1016/0022-5193(64)90038-4.
- Hansen, N and A Ostermeier (1997). "Convergence Properties of Evolution Strategies with the Derandomized Covariance Matrix Adaptation: The CMA-ES (1997)". In: *5th European Congress on Intelligent Techniques and Soft Computing*, pp. 650–654.
- Hansen, N. and A. Ostermeier (2001). "Completely Derandomized Self-Adaptation in Evolution Strategies". In: *Evol. Comput.* 9.2, pp. 159–195. ISSN: 1063-6560. DOI: 10.1162/106365601750190398.
- Hansen, N., S. D. Müller, and P. Koumoutsakos (2003). "Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix

- Adaptation (CMA-ES)". In: *Evol. Comput.* 11.1, pp. 1–18. ISSN: 1063-6560. DOI: 10.1162/106365603321828970.
- Harvey, I., P. Husbands, D. Cliff, and Others (1992). *Issues in evolutionary robotics*. Vol. 2. July 1992, pp. 364–373. ISBN: 0-262-63149-0. DOI: 10.1016/S0921-8890(96)00067-X.
- Haubold, B. and T. Wiehe (2006). *Introduction to Computational Biology: An Evolutionary Approach*. 1st ed. Birkhäuser Basel, pp. XIV, 328. ISBN: 978-3-7643-6700-8. DOI: 10.1007/3-7643-7387-3.
- Heinrich, M. K., M. Wahby, M. D. Soorati, D. N. Hofstadler, P. Zahadat, P. Ayres, K. Støy, and H. Hamann (2016). "Self-Organized Construction with Continuous Building Material: Higher Flexibility based on Braided Structures". In: *1st International Workshop on Self-Organising Construction (SOCO), SASO 2016 and ICCAC 2016*. Ausburg.
- Herrera, M, A Miller, and J Nishimura (2016). "Altruistic aging: The evolutionary dynamics balancing longevity and evolvability." In: *Mathematical Biosciences and Engineering* 13, pp. 455–465. DOI: 10.3934/mbe.2017028.
- Hiller, J and H Lipson (2012). "Automatic Design and Manufacture of Soft Robots". In: *IEEE Transactions on Robotics* 28.2, pp. 457–466. ISSN: 1552-3098. DOI: 10.1109/TRO.2011.2172702.
- Hinton, G. E. and S. J. Nowlan (1987). "How Learning Can Guide Evolution". In: *Complex Systems* 1, pp. 495–502. ISSN: 08944393. DOI: 10.1177/089443938900700128.
- Hofbauer, J. and K. Sigmund (2003). *Evolutionary game dynamics*. DOI: 10.1090/S0273-0979-03-00988-1.
- Hogeweg, P. (1988). "Cellular automata as a paradigm for ecological modeling". In: *Applied Mathematics and Computation* 27.1, pp. 81–100. ISSN: 00963003. DOI: 10.1016/0096-3003(88)90100-2.
- Holland, J. H. (1975). "Adaptation in Natural and Artificial Systems". In: *Ann Arbor MI University of Michigan Press Ann Arbor*, p. 183. ISSN: 10834419. DOI: 10.1137/1018105. arXiv: 0262082136.
- Holland, J. H. (1992). "Genetic Algorithms". In: *Scientific American* 267.1, pp. 66–72. ISSN: 0036-8733. DOI: 10.1038/scientificamerican0792-66.
- Holland, J. H. (2012). *Signals and Boundaries: Building Blocks for Complex Adaptive Systems*, p. 308. ISBN: 2011052776.
- Hornby, G. S. (2006). "ALPS: The Age-Layered Population Structure for Reducing the Problem of Premature Convergence". In: *Proceedings of the 8th annual conference on Genetic and evolutionary computation - GECCO'06*, pp. 815–822. ISBN: 1595931864. DOI: doi:10.1145/1143997.1144142.
- Hornby, G. S. (2009). "The age-layered population structure (ALPS) evolutionary algorithm". In: *Proceedings of the 11th Annual conference on Genetic and evolutionary computation - GECCO'09*. ISBN: 9781605585055.

- Hornby, G. S., H. Lipson, and J. B. Pollack (2003). "Generative representations for the automated design of modular physical robots". In: *IEEE Transactions on Robotics and Automation* 19.4, pp. 703–719. ISSN: 1042296X. DOI: 10.1109/TRA.2003.814502.
- Hornby, G. and J. Pollack (2001). "The advantages of generative grammatical encodings for physical design". In: *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)* 1, pp. 600–607. DOI: 10.1109/CEC.2001.934446.
- Ieropoulos, I., C. Melhuish, and J. Greenman (2003). "Artificial Metabolism: Towards True Energetic Autonomy in Artificial Life". In: *Advances in Artificial Life*. Springer Berlin Heidelberg, pp. 792–799. DOI: 10.1007/978-3-540-39432-7_85.
- Ijspeert, A. J. (2008). "Central pattern generators for locomotion control in animals and robots: A review". In: *Neural Networks* 21.4, pp. 642–653. ISSN: 08936080. DOI: 10.1016/j.neunet.2008.03.014.
- Jakobi, N. (1998). "Minimal Simulations For Evolutionary Robotics". PhD thesis, p. 116.
- Jakobi, N., P. Husbands, and I. Harvey (1995). "Noise and the reality gap: The use of simulation in evolutionary robotics". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 929, pp. 704–720. ISBN: 3540594965. DOI: 10.1007/3-540-59496-5_337.
- Kamimura, A., H. Kurokawa, E. Yoshida, S. Murata, K. Tomita, and S. Kokaji (2005). "Automatic locomotion design and experiments for a modular robotic system". In: *IEEE/ASME Transactions on Mechatronics* 10.3, pp. 314–325. ISSN: 10834435. DOI: 10.1109/TMECH.2005.848299.
- Kirkwood, T. B. L. (1977). "Evolution of aging". In: *Nature* 270, pp. 301–304. ISSN: 00280836. DOI: 10.1038/270301a0.
- König, L. and H. Schmeck (2009). "A completely evolvable genotype-phenotype mapping for evolutionary robotics". In: *SASO 2009 - 3rd IEEE International Conference on Self-Adaptive and Self-Organizing Systems*, pp. 175–185. ISBN: 9780769537948. DOI: 10.1109/SASO.2009.20.
- Kowald, A. and T. B. Kirkwood (2016). "Can aging be programmed? A critical literature review". In: *Aging Cell* 15.6, pp. 986–998. ISSN: 14749726. DOI: 10.1111/ace1.12510.
- Kriegman, S., N. Cheney, F. Corucci, and J. C. Bongard (2017). "A Minimal Developmental Model Can Increase Evolvability in Soft Robots". In: *Proceedings of the Genetic and Evolutionary Computation Conference - GECCO'17*, pp. 131–138. DOI: 10.1145/3071178.3071296.
- Kuhlemeier, C. (2007). "Phyllotaxis". In: *Trends in Plant Science* 12.4, pp. 143–150. ISSN: 13601385. DOI: 10.1016/j.tplants.2007.03.004.

- Kutschera, U. and W. R. Briggs (2016). *Phototropic solar tracking in sunflower plants: An integrative perspective*. DOI: [10.1093/aob/mcv141](https://doi.org/10.1093/aob/mcv141).
- Kyryakov, P., A. Gomez-Perez, A. Glebov, N. Asbah, L. Bruno, C. Meunier, T. Iouk, and V. I. Titorenko (2016). "Empirical verification of evolutionary theories of aging". In: *Aging* 8.10, pp. 2568–2589. ISSN: 19454589. DOI: [10.18632/aging.101090](https://doi.org/10.18632/aging.101090).
- Laschi, C., M. Cianchetti, B. Mazzolai, L. Margheri, M. Follador, and P. Dario (2012). "Soft robot arm inspired by the octopus". In: *Advanced Robotics* 26.7, pp. 709–727. ISSN: 01691864. DOI: [10.1163/156855312X626343](https://doi.org/10.1163/156855312X626343).
- Lehman, J. and K. O. Stanley (2011). "Novelty search and the problem with objectives". In: *Genetic Programming Theory and Practice IX*, pp. 37–56. DOI: [doi: 10.1007/978-1-4614-1770-5_3](https://doi.org/10.1007/978-1-4614-1770-5_3).
- Lehman, J. (2012). "Evolution through the Search for Novelty". PhD thesis. University of Central Florida.
- Lehman, J. (2015). "Enhancing Divergent Search through Extinction Events". In: *Proceedings of the Genetic and Evolutionary Computation Conference - GECCO'15*, pp. 951–958. DOI: [10.1145/2739480.2754668](https://doi.org/10.1145/2739480.2754668).
- Lehman, J. and K. O. Stanley (2008). "Exploiting Open-Endedness to Solve Problems Through the Search for Novelty". In: *Artificial Life XI*, pp. 329–336.
- Lehman, J., R. Miikkulainen, and G. Sun (2015). "Extinction events can accelerate evolution". In: *PLoS ONE* 10.8. ISSN: 19326203. DOI: [10.1371/journal.pone.0132886](https://doi.org/10.1371/journal.pone.0132886).
- Lessin, D., D. Fussell, and R. Miikkulainen (2013). "Open-ended behavioral complexity for evolved virtual creatures". In: *Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference - GECCO '13*, p. 335. DOI: [10.1145/2463372.2463411](https://doi.org/10.1145/2463372.2463411).
- Leung, Y., Y. Gao, and Z. B. Xu (1997). *Degree of population diversity - A perspective on premature convergence in genetic algorithms and its Markov chain analysis*. DOI: [10.1109/72.623217](https://doi.org/10.1109/72.623217).
- Levy, G., N. Neshet, L. Zullo, and H. Binyamin (2017). "Motor Control in Soft-Bodied Animals: The Octopus". In: *The Oxford Handbook of Invertebrate Neurobiology*. Ed. by J. H. Byrne. DOI: [10.1093/oxfordhb/9780190456757.013.36](https://doi.org/10.1093/oxfordhb/9780190456757.013.36).
- Lin, K., H. Hsin, N. Libina, and C. Kenyon (2001). "Regulation of the *Caenorhabditis elegans* longevity protein DAF-16 by insulin/IGF-1 and germline signaling". In: *Nature Genetics* 28.2, pp. 139–145. ISSN: 10614036. DOI: [10.1038/88850](https://doi.org/10.1038/88850).
- Lindenmayer, A. (1968a). "Mathematical models for cellular interactions in development. I. Filaments with one-sided inputs." In: *Journal of theoretical biology* 18.3, pp. 280–299. ISSN: 00225193. DOI: [10.1016/0022-5193\(68\)90079-9](https://doi.org/10.1016/0022-5193(68)90079-9).

- Lindenmayer, A. (1968b). "Mathematical Models for Cellular Interactions in Development II. Simple and Branching Filaments with Two-sided Inputs". In: *Biol* 18, pp. 300–3. ISSN: 00225193. DOI: [10.1016/0022-5193\(68\)90080-5](https://doi.org/10.1016/0022-5193(68)90080-5).
- Lindenmayer, A. and H. Jürgensen (1992). "Grammars of Development: Discrete-State Models for Growth, Differentiation, and Gene Expression in Modular Organisms". In: *Lindenmayer Systems: Impacts on Theoretical Computer Science, Computer Graphics, and Developmental Biology*. Ed. by G. Rozenberg and A. Salomaa. Berlin: Springer Berlin Heidelberg. Chap. 1, pp. 3–21.
- Lipson, H and J. B. Pollack (2000). "Automatic design and manufacture of robotic lifeforms." In: *Nature* 406.6799, pp. 974–978. ISSN: 0028-0836. DOI: [10.1038/35023115](https://doi.org/10.1038/35023115).
- Lohn, J. D., D. S. Linden, G. S. Hornby, W. F. Kraus, A. Rodríguez-Arroyo, and S. E. Seufert (2003). "Evolutionary design of an X-band antenna for NASA's Space Technology 5 mission". In: *Proceedings - NASA/DoD Conference on Evolvable Hardware, EH*. Vol. 2003-Janua, pp. 155–163. ISBN: 0769519776. DOI: [10.1109/EH.2003.1217660](https://doi.org/10.1109/EH.2003.1217660).
- Long, J. (2012). *Darwin's Devices: What Evolving Robots Can Teach Us About the History of Life and the Future of Technology*. New York: Basic Books. ISBN: 978-0465021413.
- Low, K. H. (2009). "Modelling and parametric study of modular undulating fin rays for fish robots". In: *Mechanism and Machine Theory*. Special Issue on Bio-Inspired Mechanism Engineering 44.3, pp. 615–632. ISSN: 0094-114X. DOI: [10.1016/j.mechmachtheory.2008.11.009](https://doi.org/10.1016/j.mechmachtheory.2008.11.009).
- Low, K. H. and A Willy (2006). "Biomimetic Motion Planning of an Undulating Robotic Fish Fin". In: *Journal of Vibration and Control* 12.12, pp. 1337–1359. ISSN: 1077-5463. DOI: [10.1177/1077546306070597](https://doi.org/10.1177/1077546306070597).
- MacIver, M. A., E Fontaine, and J. W. Burdick (2004). "Designing future underwater vehicles: principles and mechanisms of the weakly electric fish". In: *IEEE Journal of Oceanic Engineering* 29.3, pp. 651–659. ISSN: 0364-9059. DOI: [10.1109/JOE.2004.833210](https://doi.org/10.1109/JOE.2004.833210).
- Mahfoud, S. (1995). "Niching methods for genetic algorithms". PhD thesis. University of Illinois at Urbana-Champaign, p. 251.
- Marbach, D. and A. Ijspeert (2005). "Online optimization of modular robot locomotion". In: *IEEE International Conference Mechatronics and Automation* 1.July, pp. 248–253. DOI: [10.1109/ICMA.2005.1626555](https://doi.org/10.1109/ICMA.2005.1626555).
- Mayr, E. (1994). "Recapitulation Reinterpreted: The Somatic Program". In: *The Quarterly Review of Biology* 69.2, pp. 223–232.
- Medawar, P. B. (1952). *An unsolved problem of biology*. DOI: [10.1016/S0140-6736\(00\)99799-X](https://doi.org/10.1016/S0140-6736(00)99799-X).
- Merks, R. M. H. and M. A. Guravage (2013). "Building simulation models of developing plant organs using VirtualLeaf." In: *Plant Organogenesis*. Vol. 959,

- pp. 333–52. ISBN: 978-1-62703-221-6. DOI: 10.1007/978-1-62703-221-6_23.
- Mitchell, M., S. Forrest, and J. H. Holland (1991). “The Royal Road for Genetic Algorithms: Fitness Landscapes and GA Performance”. In: *Proceedings of the First European Conference on Artificial Life 1*, pp. 245–254. DOI: 10.1.1.49.212.
- Mitteldorf, J. and A. C. R. Martins (2014). “Programmed Life Span in the Context of Evolvability”. In: *The American Naturalist* 184.3, pp. 289–302. ISSN: 0003-0147. DOI: 10.1086/677387.
- Moreno, R., C. Liu, A. Faina, H. Hernandez, and J. Gomez (2017). “The EMERGE modular robot, an open platform for quick testing of evolved robot morphologies”. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion - GECCO’17*, pp. 71–72. DOI: 10.1145/3067695.3075616.
- Morse, G., S. Risi, C. R. Snyder, and K. O. Stanley (2013). “Single-unit pattern generators for quadruped locomotion”. In: *Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference - GECCO ’13*, p. 719. DOI: 10.1145/2463372.2463461.
- Mosadegh, B., P. Polygerinos, C. Keplinger, S. Wennstedt, R. F. Shepherd, U. Gupta, J. Shim, K. Bertoldi, C. J. Walsh, and G. M. Whitesides (2014). “Pneumatic Networks for Soft Robotics that Actuate Rapidly”. In: *Advanced Functional Materials* 24.15, pp. 2163–2170. ISSN: 1616-3028. DOI: 10.1002/adfm.201303288.
- Mouret, J.-B. and J. Clune (2015). “Illuminating search spaces by mapping elites”. In: 521.7553, p. 9993.
- Newell, A. C. and P. D. Shipman (2005). “Plants and fibonacci”. In: *Journal of Statistical Physics* 121.December, pp. 937–968. ISSN: 00224715. DOI: 10.1007/s10955-005-8665-7. arXiv: 29044450664.
- Nicotra, A. B., A. Leigh, K. Boyce, C. S. Jones, K. J. Niklas, D. L. Royer, and H. Tsukaya (2011). “The evolution and functional significance of leaf shape in the angiosperms”. In: *Functional Plant Biology* 38.Gates 1980, pp. 535–552. ISSN: 1445-4408. DOI: 10.1071/FP11057.
- Nolfi, S. and D. Floreano (1994). “How to Evolve Autonomous Robots: Different Approaches in Evolutionary Robotics”. In: *Artificial life IV*. Vol. 1997, pp. 190–197. ISBN: 0-262-52190-3.
- Nolfi, S. and D. Floreano (2000). *Evolutionary Robotics: The Biology, intelligence, and Technology of Self-Organizing Machines*. London: MIT Press. ISBN: 978-0-262-64056-5.
- Noth, A., M. W. Engel, and R. Siegwart (2006). “Flying solo and solar to mars”. In: *IEEE Robotics and Automation Magazine* 13.3, pp. 44–52. ISSN: 10709932. DOI: 10.1109/MRA.2006.1678138.

- Novella, S. (2008). "Suboptimal Optics: Vision Problems as Scars of Evolutionary History". In: *Evolution: Education and Outreach* 1.4, pp. 493–497. ISSN: 1936-6426. DOI: 10.1007/s12052-008-0092-1.
- Nowak, M. A. (2006). *Evolutionary Dynamics: Exploring the Equations of Life*. Harvard University Press. ISBN: 978-0674023383.
- Nusbaum, T. J. and M. R. Rose (1994). "Aging in *Drosophila*". In: *Comparative Biochemistry and Physiology Part A: Physiology* 109.1, pp. 33–38. DOI: 10.1016/0300-9629(94)90309-3.
- O'Dor, R. K. and D. M. Webber (1986). "The constraints on cephalopods: why squid aren't fish". In: *Canadian Journal of Zoology* 64.8, pp. 1591–1605. ISSN: 0008-4301. DOI: 10.1139/z86-241.
- Paley, W. (1802). *Theology or Evidences of the Existence and Attributes of the Deity*. London.
- Pfeifer, R. and F. Iida (2005). "Morphological computation: Connecting body, brain and environment". In: *Japanese Scientific Monthly* 58.2, pp. 48–54. DOI: 10.1007/978-3-642-00616-6_5.
- Pfeifer, R. and J. C. Bongard (2006). *How the Body Shapes the Way We Think: A New View of Intelligence*. Cambridge: The MIT Press, p. 424. ISBN: 0262162393.
- Philamore, H., J. Rossiter, and I. Ieropoulos (2015). "An Energetically-Autonomous Robotic Tadpole with Single Membrane Stomach and Tail". In: *Conference on Biomimetic and Biohybrid Systems*. Barcelona, pp. 366–378. DOI: 10.1007/978-3-319-22979-9_37.
- Pollack, J. B., H. Lipson, S. Ficici, P. Funes, G. Hornby, and R. a. Watson (2000). "Evolutionary Techniques in Physical Robotics". In: *Evolvable Systems: From Biology to Hardware* 1801.Moravec 1999, pp. 175–186. ISSN: 16113349. DOI: 10.1007/3-540-46406-9_18.
- Prasad, K. and P. Dhonukshe (2013). "Polar Auxin Transport". In: 17, pp. 25–45. DOI: 10.1007/978-3-642-35299-7.
- Prinsloo, G. and R. Dobson (2015). *Solar Tracking: High precision solar position algorithms, programs, software and source-code for computing the solar vector, solar coordinates & sun angles in Microprocessor, PLC, Arduino, PIC and PC-based sun tracking devices or dynamic sun following har*, pp. 1–542. ISBN: 978-0-620-61576-1. DOI: :10.13140/RG.2.1.4265.6329/1.
- Prusinkiewicz, P. (2004). "Self-Similarity in Plants: Integrating Mathematical and Biological Perspectives". In: *Thinking in Patterns - Fractals and Related Phenomena in Nature*, pp. 103–118. DOI: 10.1142/9789812702746_0008.
- Prusinkiewicz, P. and A. Lindenmayer (1990). *The algorithmic beauty of plants*, p. 228. ISBN: 0387972978. DOI: 10.1016/S0168-9452(96)04526-8.
- Prusinkiewicz, P. and A. Runions (2012). "Computational models of plant development and form." In: *The New phytologist* 193.3, pp. 549–569. ISSN: 1469-8137. DOI: 10.1111/j.1469-8137.2011.04009.x. arXiv: 84855766776.

- Rechenberg, I. (1973). "Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution". PhD thesis, p. 170.
- Reece, J. B., L. A. Urry, M. L. Cain, S. A. Wasserman, P. V. Minorsky, and R. B. Jackson (2010). *Campbell Biology*, p. 1472. ISBN: 10: 0321739752. DOI: [10.1007/s13398-014-0173-7.2](https://doi.org/10.1007/s13398-014-0173-7.2).
- Reisinger, J., K. O. Stanley, and R. Miikkulainen (2005). "Towards an empirical measure of evolvability". In: *Proceedings of the 2005 workshops on Genetic and evolutionary computation - GECCO '05*, p. 257. ISBN: 0123456999. DOI: [10.1145/1102256.1102315](https://doi.org/10.1145/1102256.1102315).
- Rieffel, J., D. Knox, S. Smith, and B. Trimmer (2013). "Growing and Evolving Soft Robots". In: *Artificial Life* 20.1, pp. 143–162. ISSN: 1064-5462. DOI: [10.1162/ARTL_a_00101](https://doi.org/10.1162/ARTL_a_00101).
- Rieffel, J., J.-B. Mouret, N. Bredeche, and E. Haasdijk (2017). "Introduction to the Evolution of Physical Systems Special Issue". In: *Artificial Life* 23.2, pp. 119–123. ISSN: 10645462. DOI: [10.1162/ARTL_e_00232](https://doi.org/10.1162/ARTL_e_00232).
- Risi, S., J. Lehman, and K. O. Stanley (2010). "Evolving the Placement and Density of Neurons in the HyperNEAT Substrate". In: *GECCO '10 Proceedings of the 12th annual conference on Genetic and evolutionary computation Gecco*, pp. 563–570. ISSN: 9781450300728. DOI: [10.1145/1830483.1830589](https://doi.org/10.1145/1830483.1830589).
- Rohmer, E., S. P. N. Singh, and M. Freese (2013). "V-REP: A versatile and scalable robot simulation framework". In: *IEEE International Conference on Intelligent Robots and Systems*. Tokyo, pp. 1321–1326. ISBN: 9781467363587. DOI: [10.1109/IROS.2013.6696520](https://doi.org/10.1109/IROS.2013.6696520).
- Runions, A., R. S. Smith, and P. Prusinkiewicz (2014). *Computational Models of Auxin-Driven Development*. ISBN: 978-3-7091-1526-8. DOI: [10.1007/978-3-7091-1526-8_15](https://doi.org/10.1007/978-3-7091-1526-8_15).
- Rus, D. and M. T. Tolley (2015). "Design, fabrication and control of soft robots". In: *Nature* 521.7553, pp. 467–475. DOI: [10.1038/nature14543](https://doi.org/10.1038/nature14543).
- Russell, S. J. and P. Norvig (2010). *Artificial Intelligence: A Modern Approach*, p. 1132. ISBN: 0137903952. DOI: [10.1017/S0269888900007724](https://doi.org/10.1017/S0269888900007724).
- Sadeghi, A., A. Tonazzini, L. Popova, and B. Mazzolai (2014). "A novel growing device inspired by plant root soil penetration behaviors". In: *PLoS ONE* 9.2. ISSN: 19326203. DOI: [10.1371/journal.pone.0090139](https://doi.org/10.1371/journal.pone.0090139).
- Sadeghi, A., A. Mondini, E. Del Dottore, V. Mattoli, L. Beccai, S. Taccola, C. Lucarotti, M. Totaro, and B. Mazzolai (2017). "A plant-inspired robot with soft differential bending capabilities". In: *Bioinspiration & Biomimetics* 12. ISSN: 17483190. DOI: [10.1088/1748-3190/12/1/015001](https://doi.org/10.1088/1748-3190/12/1/015001).
- Santos, M., E. Szathmáry, and J. F. Fontanari (2015). "Phenotypic plasticity, the baldwin effect, and the speeding up of evolution: The computational roots of an illusion". In: *Journal of Theoretical Biology* 371, pp. 127–136. ISSN: 10958541. DOI: [10.1016/j.jtbi.2015.02.012](https://doi.org/10.1016/j.jtbi.2015.02.012). arXiv: 1411.6843.

- Saunders, F., E. Golden, R. D. White, and J. Rife (2011). "Experimental verification of soft-robot gaits evolved using a lumped dynamic model". In: *Robotica* 29.6, pp. 823–830. DOI: [10.1017/S0263574711000014](https://doi.org/10.1017/S0263574711000014).
- Schmidt, M. D. and H. Lipson (2011). "Age-Fitness Pareto Optimization". In: *Genetic Programming Theory and Practice VIII*. Vol. 8, pp. 129–146. ISBN: 978-1-4419-7746-5. DOI: [10.1007/978-1-4419-7747-2_8](https://doi.org/10.1007/978-1-4419-7747-2_8).
- Schuster, P. and J. Swetina (1988). "Stationary mutant distributions and evolutionary optimization". In: *Bulletin of Mathematical Biology* 50.6, pp. 635–660. ISSN: 00928240. DOI: [10.1007/BF02460094](https://doi.org/10.1007/BF02460094).
- Secretan, J., N. Beato, D. B. D'Ambrosio, A. Rodriguez, A. Campbell, J. T. Folsom-Kovarik, and K. O. Stanley (2011). "Picbreeder: a case study in collaborative evolutionary exploration of design space." In: *Evolutionary computation* 19.3, pp. 373–403. ISSN: 1063-6560. DOI: [10.1162/EVCO_a_00030](https://doi.org/10.1162/EVCO_a_00030).
- Sfakiotakis, M., J. Fasoulas, M. M. Kavoussanos, and M. Arapis (2015). "Experimental investigation and propulsion control for a bio-inspired robotic undulatory fin". In: *Robotica* 33.5, pp. 1062–1084. ISSN: 0263-5747, 1469-8668. DOI: [10.1017/S0263574714002926](https://doi.org/10.1017/S0263574714002926).
- Shefferson, R., O. Jones, and R. Salguero-Gómez (2017). *The Evolution of Senescence in the Tree of Life*. Cambridge University Press. ISBN: 9781107078505.
- Siahmansouri, M., A. Ghanbari, and M. M. S. Fakhrabadi (2011). "Design, Implementation and Control of a Fish Robot with Undulating Fins". In: *International Journal of Advanced Robotic Systems* 8.5, p. 60. ISSN: 1729-8814. DOI: [10.5772/50898](https://doi.org/10.5772/50898).
- Silver, D., J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. Van Den Driessche, T. Graepel, and D. Hassabis (2017). "Mastering the game of Go without human knowledge". In: *Nature* 550.7676, pp. 354–359. ISSN: 14764687. DOI: [10.1038/nature24270](https://doi.org/10.1038/nature24270). arXiv: [1610.00633](https://arxiv.org/abs/1610.00633).
- Sims, K (1992). "Interactive evolution of dynamical systems". In: *Proceedings of the First European Conference on Artificial Life*. Paris, pp. 171–178.
- Sims, K. (1994a). "Evolving 3D Morphology and Behavior by Competition". In: *Artificial Life* 1.4, pp. 353–372. ISSN: 1064-5462. DOI: [10.1162/artl.1994.1.4.353](https://doi.org/10.1162/artl.1994.1.4.353).
- Sims, K. (1994b). "Evolving virtual creatures". In: *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*. July, pp. 15–22. ISBN: 0897916670. DOI: [10.1145/192161.192167](https://doi.org/10.1145/192161.192167).
- Sinibaldi, E., A. Argiolas, G. L. Puleo, and B. Mazzolai (2014). "Another lesson from plants: The forward osmosis-based actuator". In: *PLoS ONE* 9.7. ISSN: 19326203. DOI: [10.1371/journal.pone.0102461](https://doi.org/10.1371/journal.pone.0102461).
- Smith, J. M. (1970). "Natural Selection and the Concept of a Protein Space". In: *Nature* 225. DOI: [10.1038/225563a0](https://doi.org/10.1038/225563a0).

- Smith, J. M. (1987). "When learning guides evolution". In: *Nature* 329, pp. 761–762. DOI: 10.1038/329761a0.
- Song, K., E. Yeom, and S. J. Lee (2014). "Real-time imaging of pulvinus bending in *Mimosa pudica*." In: *Scientific reports* 4, p. 6466. ISSN: 2045-2322. DOI: 10.1038/srep06466.
- Sproewitz, A., R. Moeckel, J. Maye, and A. J. Ijspeert (2008). "Learning to Move in Modular Robots using Central Pattern Generators and Online Optimization". In: *The International Journal of Robotics Research* 27.3-4, pp. 423–443. ISSN: 0278-3649. DOI: 10.1177/0278364907088401.
- Stanley, K. O. (2007). "Compositional pattern producing networks: A novel abstraction of development". In: *Genetic Programming and Evolvable Machines* 8.2, pp. 131–162. ISSN: 13892576. DOI: 10.1007/s10710-007-9028-8.
- Stanley, K. O. and R. Miikkulainen (2002a). "Efficient Evolution of Neural Network Topologies". In: *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on Evolutionary Computation*, pp. 1757–1762. DOI: 10.1109/CEC.2002.1004508.
- Stanley, K. O. and R. Miikkulainen (2002b). "Evolving neural networks through augmenting topologies." In: *Evolutionary Computation* 10.2, pp. 99–127. ISSN: 1063-6560. DOI: 10.1162/106365602320169811.
- Stanley, K. O., D. B. D'Ambrosio, and J. Gauci (2009). "A hypercube-based encoding for evolving large-scale neural networks." In: *Artificial Life* 15.2, pp. 185–212. ISSN: 1064-5462. DOI: 10.1162/artl.2009.15.2.15202.
- Stanley, K. O. and J. Lehman (2015). *Why Greatness Cannot Be Planned*.
- Still, S., K. Hepp, and R. J. Douglas (2006). "Neuromorphic walking gait control". In: *IEEE Transactions on Neural Networks* 17.2, pp. 496–508. ISSN: 10459227. DOI: 10.1109/TNN.2005.863454.
- Støy, K., W. M. Shen, and P. M. Will (2002). "Using role-based control to produce locomotion in chain-type self-reconfigurable robots". In: *IEEE/ASME Transactions on Mechatronics* 7.4, pp. 410–417. ISSN: 10834435. DOI: 10.1109/TMECH.2002.806223.
- Stoy, K., D. Brandt, and D. J. Christensen (2010). *Self-Reconfigurable Robots: An Introduction*. Vol. 18, pp. 237–240. ISBN: 978-0262013710.
- Syswerda, G. (1991). "A Study of Reproduction in Generational and Steady State Genetic Algorithms". In: *Foundations of Genetic Algorithms* 1, pp. 94–101. DOI: 10.1016/b978-0-08-050684-5.50009-4.
- Tinbergen, N. (1963). "On aims and methods of Ethology". In: *Zeitschrift für Tierpsychologie* 20.4, pp. 410–433. ISSN: 14390310. DOI: 10.1111/j.1439-0310.1963.tb01161.x.
- Toth, G. and C. Szigeti (2016). "The historical ecological footprint: From over-population to over-consumption". In: *Ecological Indicators* 60, pp. 283–291. ISSN: 1470160X. DOI: 10.1016/j.ecolind.2015.06.040.

- Turing, a. M. (1952). "The Chemical Basis of Morphogenesis". In: *Society* 237.641, pp. 37–72. ISSN: 0962-8436. DOI: 10.1098/rstb.1952.0012.
- Valiant, L. G. (1984). "A theory of the learnable". In: *Communications of the ACM* 27.11, pp. 1134–1142. ISSN: 00010782. DOI: 10.1145/1968.1972.
- Valiant, L. G. (2009). "Evolvability". In: *Journal of the ACM* 56.1, pp. 1–21. ISSN: 00045411. DOI: 10.1145/1462153.1462156.
- Valladares, F. and R. W. Pearcy (2000). "The role of crown architecture for light harvesting and carbon gain in extreme light environments assessed with a realistic 3-D model". In: *Anales del Jardín Botánico de Madrid* 58.1, pp. 3–16. ISSN: 0211-1322. DOI: 10.3989/ajbm.2000.v58.i1.127.
- Van Batenburg, F. H., A. P. Gulyaev, and C. W. Pleij (1995). "An APL-programmed genetic algorithm for the prediction of RNA secondary structure". In: *Journal of Theoretical Biology* 174.3, pp. 271–277. ISSN: 00225193. DOI: 10.1006/jtbi.1995.0098.
- Vargas, P. A., E. A. D. Paolo, I. Harvey, and P. Husbands (2014). "Chapter 11: Incremental Evolution of an Omni-directional Biped for Rugged Terrain". In: *The Horizons of Evolutionary Robotics*, pp. 237–278.
- Varjosalo, M. and J. Taipale (2008). "Hedgehog: functions and mechanisms". In: *Genes Dev.* 15.22, pp. 2354–72. DOI: 10.1101/gad.1693608.
- Vavak, F. and T. Fogarty (1996a). "A comparative study of steady state and generational genetic algorithms for use in nonstationary environments". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 1143, pp. 297–304. ISBN: 3540617493. DOI: 10.1007/BFb0032791.
- Vavak, F. and T. Fogarty (1996b). "Comparison of steady state and generational genetic algorithms for use in nonstationary environments". In: *Proceedings of IEEE International Conference on Evolutionary Computation*, pp. 192–195. ISBN: 0-7803-2902-3. DOI: 10.1109/ICEC.1996.542359.
- Veenstra, F., A. Struck, and M. Krauledat (2015). "Acquiring Efficient Locomotion in a Simulated Quadruped through Evolving Random and Predefined Neural Networks". In: *12th Biennial International Conference on Artificial Evolution*, pp. 309–316.
- Veenstra, F., A. Faina, K. Stoy, and S. Risi (2016). "Generating Artificial Plant Morphologies for Function and Aesthetics through Evolving L-Systems". In: *Proceedings of the Artificial Life Conference 2016*. MIT Press, pp. 692–699.
- Veenstra, F., A. Faina, S. Risi, and K. Stoy (2017a). "Evolution and Morphogenesis of Simulated Modular Robots: A Comparison Between a Direct and Generative Encoding". In: *Applications of Evolutionary Computation: 20th European Conference, EvoApplications 2017*. Ed. by G. Squillero and K. Sim. Amsterdam: Springer International Publishing, pp. 870–885. DOI: 10.1007/978-3-319-55849-3_56.

- Veenstra, F., C. Metayer, S. Risi, and K. Stoy (2017b). "Toward Energy Autonomy in Heterogeneous Modular Plant-Inspired Robots through Artificial Evolution". In: *Frontiers in Robotics and AI* 4, p. 43. DOI: [10.3389/frobt.2017.00043](https://doi.org/10.3389/frobt.2017.00043).
- Veenstra, F., J. Jørgensen, and S. Risi (2018a). "Evolution of Fin Undulation on a Physical Knifefish-inspired Soft Robot". In: *GECCO '18: Genetic and Evolutionary Computation Conference*. Ed. by J. B. Sartor, T. D'Hondt, and W. D. Meuter. ACM. DOI: [10.1145/3205455.3205583](https://doi.org/10.1145/3205455.3205583).
- Veenstra, F., P. G.d. P. Salaslez, J. Bongard, K. Stoy, and S. Risi (2018b). "Intrinsic Mortality Governs Evolvability". In: *Artificial Life 2018*.
- Vergara, A., Y.-s. Lau, R.-F. Mendoza-Garcia, and J. C. Zagal (2017). "Soft Modular Robotic Cubes: Toward Replicating Morphogenetic Movements of the Embryo". In: *PLoS ONE* 12.1. DOI: [e0169179](https://doi.org/10.1371/journal.pone.0169179).
- Wagner, A. (2008). "Robustness and evolvability: a paradox resolved". In: *Proceedings of the Royal Society B: Biological Sciences* 275.1630, pp. 91–100. ISSN: 0962-8452. DOI: [10.1098/rspb.2007.1137](https://doi.org/10.1098/rspb.2007.1137).
- Wagner, G. P. (1996). "Homologues, Natural Kinds and the Evolution of Modularity". In: *American Zoologist* 36, pp. 36–43. ISSN: 1540-7063. DOI: [10.1093/icb/36.1.36](https://doi.org/10.1093/icb/36.1.36).
- Wagy, M. D. and J. C. Bongard (2015). "Combining Computational and Social Effort for Collaborative Problem Solving". In: *PLoS ONE* 10.11, pp. 1–17. DOI: [10.1371/journal.pone.0142524](https://doi.org/10.1371/journal.pone.0142524).
- Watson, R., G. Hornby, and J. Pollack (1998). "Modeling building-block interdependency". In: *Parallel Problem Solving from Nature — PPSN V 1498*, pp. 97–106. ISSN: 03029743. DOI: [10.1007/BFb0056843](https://doi.org/10.1007/BFb0056843).
- Watson, R. A., S. G. Ficiej, and J. B. Pollack (1999). "Embodied evolution: Embodying an evolutionary algorithm in a population of robots". In: *Proceedings of the 1999 Congress on Evolutionary Computation, CEC 1999*. Vol. 1, pp. 335–342. ISBN: 0-7803-5536-9. DOI: [10.1109/CEC.1999.781944](https://doi.org/10.1109/CEC.1999.781944).
- Weismann, A. (1889). *Essays upon heredity and kindred biological problems*. Ed. by E. B. Poulton, S. Schönland, and A. E. Shipley. Oxford, Clarendon Press, p. 496.
- Werfel, J., D. E. Ingber, and Y. Bar-Yam (2017). "Theory and associated phenomenology for intrinsic mortality arising from natural selection". In: *PLoS ONE* 12.3. ISSN: 19326203. DOI: [10.1371/journal.pone.0173677](https://doi.org/10.1371/journal.pone.0173677).
- Williams, G. C. (1957). "Pleiotropy, natural-selection, and the evolution of senescence". In: *Evolution* 11.4, pp. 398–411. ISSN: 0018-506X. DOI: [10.2307/2406060](https://doi.org/10.2307/2406060).
- Wodinsky, J (1977). "Hormonal inhibition of feeding and death in octopus: control by optic gland secretion." In: *Science (New York, N.Y.)* 198, pp. 948–951. ISSN: 0036-8075. DOI: [10.1126/science.198.4320.948](https://doi.org/10.1126/science.198.4320.948).
- Wolpert, L (1969). "Positional information and the spatial pattern of cellular differentiation." In: *Journal of theoretical biology* 25.1, pp. 1–47. ISSN: 00225193. DOI: [10.1016/S0022-5193\(69\)80016-0](https://doi.org/10.1016/S0022-5193(69)80016-0).

- Wolpert, L., T. Jessell, P. Lawrence, E. Meyerowitz, E. Robertson, and J. Smith (2007). *Principles of Development*. 3rd ed.
- Wright, S. (1932). "The roles of mutation, inbreeding, crossbreeding and selection in evolution". In: *Proceedings of the Sixth International Congress on Genetics*. Vol. 1, pp. 356–366. ISBN: 9780226910536.
- Wu, S. J. and P. T. Chow (1995). "Steady-state genetic algorithms for discrete optimization of trusses". In: *Computers and Structures* 56.6, pp. 979–991. ISSN: 00457949. DOI: 10.1016/0045-7949(94)00551-D.
- Yang, J. N. (2013). "Viscous populations evolve Altruistic programmed ageing in ability conflict in a changing environment". In: *Evolutionary Ecology Research* 15.5, pp. 527–543.
- Yao, X and Y Liu (1997). "A new evolutionary system for evolving artificial neural networks." In: *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council* 8.3, pp. 694–713. ISSN: 1045-9227. DOI: 10.1109/72.572107.
- Yeom, K. and J. H. Park (2010). "Artificial morphogenesis for arbitrary shape generation of swarms of multi agents". In: *Proceedings 2010 IEEE 5th International Conference on Bio-Inspired Computing: Theories and Applications, BIC-TA 2010*, pp. 509–513. DOI: 10.1109/BICTA.2010.5645177.
- Youngerman, E. D., B. E. Flammang, and G. V. Lauder (2014). "Locomotion of free-swimming ghost knifefish: anal fin kinematics during four behaviors". In: *Zoology (Jena, Germany)* 117.5, pp. 337–348. ISSN: 1873-2720. DOI: 10.1016/j.zool.2014.04.004.
- Zahadat, P., D. Hofstadler, and T. Schmickl (2017). "Vascular morphogenesis controller: A generative model for developing morphology of artificial structures". In: *GECCO 2017 - Proceedings of the 2017 Genetic and Evolutionary Computation Conference*, pp. 163–170. DOI: 10.1145/3071178.3071247.
- Zahadat, P., D. N. Hofstadler, and T. Schmickl (2016). "Vascular Morphogenesis Controller: A Distributed Controller for Growing Artificial Structures". In: *SASO 2016*.
- Zamuda, a and J Brest (2014). "Vectorized procedural models for animated trees reconstruction using differential evolution". In: *Information Sciences* 278, pp. 1–21. ISSN: 0020-0255. DOI: 10.1016/j.ins.2014.04.037.
- Zamuda, A. and J. Brest (2012). "Tree model reconstruction innovization using multi-objective differential evolution". In: *2012 IEEE Congress on Evolutionary Computation*, pp. 1–8. ISBN: 978-1-4673-1509-8. DOI: 10.1109/CEC.2012.6256413.
- Zykov, V., J. Bongard, and H. Lipson (2004). "Evolving dynamic gaits on a physical robot". In: *Proceedings of Genetic and Evolutionary Computation Conference, Late Breaking Paper, GECCO'04*. Vol. 4, pp. 540–549.

Index

- Adaptive radiation, 87
- AFPO (Age-Fitness Pareto Optimization), 70
- Allele, 21
- Altruistic Aging, 53
- Antagonistic pleiotropy, 53

- Baldwin effect, 26
- Biomorphs, 8
- Blind Watchmaker, 2, 8

- Catastrophic forgetting, 87
- Chuang f1, 63
- CMA-ES (Covariance Matrix Adaptation Evolution Strategy), 40, 75, 192
- CPG (Central pattern generator), 95, 130
- CPPN (Compositional Pattern Producing Network), 94

- Development, 23
- Diploidy, 16
- Direct encoding, 129
- Disposable soma, 53
- DNA, 18

- Edhmer (Evolutionary designer of heterogeneous modular robots), 130
- Elitism, 36
- Embodied evolution, 176
- Embodiment, 174
- Energy autonomy, 146
- Error catastrophe, 69, 89
- Evo-devo, 24
- Evolutionary algorithm
 - Extinction events, 53, 73
 - Generational, 35
 - Genetic algorithm, 34
 - Speciation, 73
 - Steady state, 34, 35, 99, 116, 130, 157
- Evolutionary computation, 1, 14
- Evolutionary games, 27
- Evolutionary robotics, 2, 4, 8
- Evolvability, 4, 29
 - definition, 33
- Evolvability of mortality, 86
- Exploration vs exploitation, 127, 129, 145, 205
- Exploration vs. exploitation, 4, 36
- Extended phenotype, 15

- Fiducial marker, 177
- Fitness, 17, 114
 - Absolute, 114
 - Inclusive, 17, 32, 82
 - Landscape, 18
 - Personal, 17, 49, 51

- Gene pool, 16
- Generative encoding, 100, 115, 128, 129, 147, 155
- Genes
 - Homologue, 25
 - Paralogue, 21
- Genetic drift, 22
- Genotype, 23
- Genotype to phenotype mapping, 8, 18, 23, 127, 128

- H-IFF (Hierarchical If-and-Only-If problem), 38, 50, 57

- Hedgehog, 25
Heterochrony, 24
Hill hugging, 61, 73

Knifefish, 183

L-system, 96, 100, 111, 113, 116, 130, 155
 Context sensitive, 101, 118
Light tracking, 165
Longevity, 49
Lotka-Volterra, 28

Maximum parsimony, 3, 30, 33
Modular robot, 146, 149
 Automated assembly, 177
 Simulated, 129
Modular robots, 92
Modularity, 96, 111
Modules
 Cube, 131, 150
 Flower, 150
 Servo, 131, 150
 Solar, 149
Morphogen, 24, 113
Morphological computation, 184
Mortality, 49
 Extrinsic, 51, 85
 Intrinsic, 50, 51, 85
Mutation accumulation, 52
Mutation rate threshold, 20
Mystery of mysteries, 14

NEAT (Neuroevolution of Augmenting Topologies), 94
Neuroevolution, 94, 158
 ENN (Evolving Neural Network), 94

Octopus, 14, 50, 97

Open-loop, 102, 148

Phenotype, 23
Photoreceptors, 31
Phototropism, 148
Phyllotaxis, 114, 148
Phytomorphogenesis, 110
Phytomorphology, 110
Pleiotropy, 23, 53
Premature convergence, 33, 36
Protein sequence, 29

Quasispecies, 20, 56

Reality gap, 4, 176, 181, 184

Schwefel, 75
Selection, 22
Self-similarity, 111
Senescence, 49, 51
 Programmed vs non-programmed, 55
Sequence space, 19
Sessile, 113
Soft robotics, 184
Spatial model, 42, 57, 63
Speciation, 19
Survival of the fittest, 22

Traversability, 20
Trophic structure, 43, 110

Undulation, 185

V-REP (Virtual Robot Experimentation Platform), 93, 98, 128, 150

Watchmaker, viii, 2, 210

Zero-evolvability state, 21, 54, 85