

Evolutionary Fabrication: The Co-Evolution of Form and Formation

A Dissertation

Presented to

The Faculty of the Graduate School of Arts and Sciences

Brandeis University

Michtom School of Computer Science

Jordan Pollack, Advisor

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

by

John Rieffel

May, 2006

This dissertation, directed and approved by John Rieffel's committee, has been accepted and approved by the Graduate Faculty of Brandeis University in partial fulfillment of the requirements for the degree of:

DOCTOR OF PHILOSOPHY

Adam B. Jaffe, Dean of Arts and Sciences

Dissertation Committee:

Jordan Pollack, Chair

Harry Mairson

Timothy J. Hickey

Neil Gershenfeld

©Copyright by

John Rieffel

2006

in memoriam

John Arden Bretz (1917-2006)

Mary Ethel Robbins (1917-2006)

Acknowledgments

No dissertation emerges in a vacuum, and this particular one wouldn't have even left the ground were it not for the enormous web of support, both intellectual and personal, that surrounds me.

To begin with, I would like to express my enormous gratitude to all of the members of the DEMO Lab with whom I have had the pleasure of collaborating over the past five years. The lab is a terrific incubator of knowledge; most of the ideas in this dissertation grew out of discussions, sometimes heated but always friendly, held within those walls. Particular thanks to my close colleague Shivakumar Viswanathan, *il miglior fabbro*, who, with his encyclopedic knowledge of the relevant literature, has been both my strongest champion and most stern critic. Of course, the credit for assembling all of these incredible minds under one roof is due to Jordan Pollack, who has a keen eye for interesting people and important ideas. Jordan's ability to instantly zero in on the most important result of every experiment, coupled with his ability to grasp the broader implications of my work, served to focus and ground my efforts over the years.

I would also like to thank each of the members of my committee. Harry Mairson has been an terrific mentor and teacher over the years. Leading recitation for his introductory Scheme class has been one of the highlights of my time at Brandeis. Tim Hickey gladly joined the committee on relatively short notice and yet provided

significant insight and advice. Neil Gershenfeld rearranged a very busy schedule to attend my defense, and provided a much-needed engineering perspective to this dissertation. Combined, my committee provided vital feedback and posed numerous challenging questions, serving to keep me on my toes throughout the entire process.

I am also blessed by a large and extremely supportive extended family, all of whom have been both patient and generous with me. My greatest gratitude is reserved for Emily, my wife and my closest friend, who makes everything possible.

Abstract

Evolutionary Fabrication: The Co-Evolution of Form and Formation

A dissertation presented to the Faculty of
the Graduate School of Arts and Sciences of
Brandeis University, Waltham, Massachusetts

by John Rieffel

Evolutionary Design has been used to automatically generate a wide variety of novel and creative objects such as circuits, robots, and satellite antennae. And yet, despite the availability of sophisticated rapid prototyping machines capable of printing objects out of plastic, metal, and even circuitry, relatively few of these evolved designs have been physically manufactured in the real world.

We argue that the cause of this paucity of physical artifacts lies in the “design first, build later” philosophy of contemporary Evolutionary Design. By only specifying the *form* of an object, this approach leaves unanswered the vital question of *formation*. As evolved forms become more complex, their formation becomes increasingly difficult for both humans and computers to discover. As a consequence, there is a growing *Fabrication Gap* between the complexity of objects which we can evolve and those which we can manufacture.

The alternative proposed here is to use Artificial Ontogenies, a computational method inspired by the biological processes of growth, in order to directly evolve the *formation* of objects. We introduce *Evolutionary Fabrication*, the direct evolution of assembly instructions *within* a simulated manufacturing system, and show that this approach is capable of injecting the novelty and creativity associated with evolutionary approaches into the realm of fabrication, generating not just novel objects, but novel means of assembling those objects as well.

Ultimately, the evolution of form and formation become fully intertwined when the language of assembly itself becomes subject to evolution, capable of discovering increasingly large sub-assemblies and adding them to its vocabulary. Through this co-evolution of form and formation, Evolutionary Fabrication discovers both *how* to build objects and *what to build them out of*. In this manner, Evolutionary Fabrication is capable of designing and assembling scalably complex objects in a hierarchical manner, even in the presence of error during assembly.

Via this co-evolution of form and formation, Evolutionary Fabrication circumvents the Fabrication Gap, leading the way to systems which can move from broad specification to complete artifact without the need for further human intervention. This budding field of Fully Automated Design and Manufacture will have an impact on realms ranging from product design to planetary exploration.

Preface

When I first arrived at the DEMO Lab in the Fall 2001 I was given the task of assembling a variety of the “GenoBots” created by my colleague Greg Hornby [38]. A few of the simpler robots were relatively easy to assemble, and produced some spectacular results - the most notable being the “QuadraBot” that walked in straight lines on four legs, each actuated by a single oscillating servo motor.

Unfortunately, as the designs became more complex, the robots in turn became more difficult to build. The brittle plastic parts broke easily, and the actuators struggled to move ever larger masses. More frustrating, however, was the process of looking at one of Greg’s simulations and figuring out *how* exactly to build the shape on the screen. This was my first glimpse of the “Fabrication Gap”: the notion that there is a tremendous amount of missing knowledge between the blueprint of an object and the process required to physically manufacture it.

And so, my motivation for this work stems from the same question that many Ph.D. candidates ask themselves: “Can’t I get a computer to do this for me?” This, in the broadest sense, is the subject of this dissertation: how we can use Evolutionary Algorithms not just to *design* complex objects, but to *assemble* them as well.

Contents

Abstract	vii
Preface	ix
1 Introduction	1
1.1 The Fabrication Gap	2
1.2 Bridging the Reality Gap	4
1.3 Evolutionary Fabrication	4
1.4 Assembly as Ontogeny	6
1.5 Development and Noise	7
1.6 The Growth of Complexity	7
1.7 Summary	9
1.8 Preview	10
2 Foundations	12
2.1 The Evolution of Form	13
2.2 Manufacturing Evolved Designs	17
2.3 The Fabrication Gap	18
2.4 The Evolution of Formation	25
2.5 Evolutionary Fabrication	34
2.6 Engineering without Engineers	36
2.7 Summary	38
3 Framework	40
3.1 System Description	41
3.2 A Brief Example	45
3.3 Summary	48
4 Evolutionary Fabrication for Design	50
4.1 Filled Volume Fitness	51
4.2 Shaded Volume Fitness	53
4.3 The Emergence of Novel Assembly Methods	55
4.4 Summary	64

5	Evolutionary Fabrication in Uncertain Environments	65
5.1	Evolving Reliability without Tests	66
5.2	Evolving for Scalable Complexity	71
5.3	Modular Evolutionary Fabrication	78
5.4	Modular Assembly in Noisy Environments	81
5.5	Scalable Modular Assembly	86
5.6	Summary	89
6	Discussion and Conclusion	92
6.1	Novelty and Invention	92
6.2	Assembly vs. Disassembly	94
6.3	Hierarchy and Noise	95
6.4	Measures of Complexity and Scale	101
6.5	Embodied Evolutionary Fabrication	102
6.6	Conclusion	105
A	Example Hierarchies	108

List of Tables

3.1	Parameterized Assembly Instructions	43
3.2	An Example Assembly Plan	43
4.1	Structures generated with the “filled” fitness function.	52
4.2	Structures Evolved for “shadow” fitness.	54
4.3	Comparison of two environments	57
4.4	Evolved structures.	63
5.1	Visualization of improving yields.	73
5.2	Example modules and their associated assembly plans	78
5.3	Hierarchical assembly of modules from Table 5.2	79
A.1	Example Hierarchical Assemblies of Modules at 0.1% noise	109

List of Figures

1.1	An example of novel dynamic assembly from Section 4.3	10
1.2	A robust hierarchical modular assembly from Section 5.3	11
2.1	Pablo Funes' evolved tree	19
2.2	One of Greg Hornby's evolved GenoBots	19
2.3	Intertwined Rings: an unbuildable design.	23
2.4	Consequences of noisy development	31
3.1	An illustration of the assembly process	45
3.2	The Goal Arch	46
3.3	Evolved assembly plan building the goal arch.	48
4.1	Illustration of the "filled" fitness function.	51
4.2	Assembly of an arch	54
4.3	A novel evolved assembly method	55
4.4	Another novel evolved assembly process.	55
4.5	Illustration of two assembly environments.	57
4.6	Another dynamic assembly sequence.	58
4.7	Another dynamic assembly sequence	58
4.8	An extreme example of dynamic assembly.	58
4.9	Fitnesses and Dynamic Assembly Contribution	60
4.10	Fitness Comparison Between Environments	60
4.11	Fitness Contribution from Dynamic Assembly	61
4.12	Average Fitness per Brick	62
4.13	Average fitness per instruction.	62
5.1	Illustrating the effects of noise on development.	66
5.2	The goal arch.	68
5.3	A sample of phenotypes from the same genotype.	69
5.4	Robust Assembly	72
5.5	Robust Assembly	72
5.6	Robust Assembly	72
5.7	The process of module rejection and discovery.	80
5.8	Illustration of endosymbiotic module acquisition.	82

5.9	Properties of a good module	82
5.10	Illustration of Shaded Fitness Function.	83
5.11	A demonstration of the effects of noisy development.	85
5.12	Performance comparison of modular and non-modular evolution.	85
5.13	An Example Hierarchical Assembly of Modules at 0.1% noise. Further examples are provided in Appendix A	87
5.14	An Example Hierarchical Assembly of Modules at 1.0% noise. Further examples are provided in Appendix A	87
5.15	Illustration of Larger Environment and “Leafy” Fitness.	88
5.16	Evolutionary Fitness of the “Leafy” Fitness Function	90
5.17	Assembly of a Larger Structure	90
6.1	A non-reversible toppling motion	93
6.2	An extreme example of dynamic assembly	94
6.3	Recapitulating Simon’s Parable.	96
6.4	A robust hierarchical modular assembly from Section 5.3	97
6.5	As Noise Increases Reuse Increases	98
6.6	Comparing Modular Hierarchies	99
6.7	Distinctions between Modular Hierarchies	100
6.8	Mass per Instruction	103
6.9	Mass per Instruction	103
A.1	An Example Hierarchical Assembly of Modules at 0.1% noise	108
A.2	An Example Hierarchical Assembly of Modules at 0.1% noise	110
A.3	An Example Hierarchical Assembly of Modules at 0.1% noise	111
A.4	An Example Hierarchical Assembly of Modules at 0.1% noise	112
A.5	An Example Hierarchical Assembly of Modules at 0.1% noise	112
A.6	An Example Hierarchical Assembly of Modules at 1.0% noise	113
A.7	An Example Hierarchical Assembly of Modules at 1.0% noise	114
A.8	An Example Hierarchical Assembly of Modules at 1.0% noise	114
A.9	An Example Hierarchical Assembly of Modules at 1.0% noise	115
A.10	An Example Hierarchical Assembly of Modules at 1.0% noise	116
A.11	An Example Hierarchical Assembly of Modules at 5.0% noise	117
A.12	An Example Hierarchical Assembly of Modules at 5.0% noise	118
A.13	An Example Hierarchical Assembly of Modules at 5.0% noise	119
A.14	An Example Hierarchical Assembly of Modules at 5.0% noise	120
A.15	An Example Hierarchical Assembly of Modules at 5.0% noise	121

Chapter 1

Introduction

Imagine a table-top machine, akin to the fabled “Star Trek” replicator, able to create any object it is asked for. Now imagine that same machine endowed with creativity, capable of inventing entirely new products as well: an “automated invention machine” [51]. Next, consider a fleet of these machines landing on the surface of Mars, where each proceeds to take measure of its surroundings, gather locally available raw materials, and then produce a series of robotic rovers, each uniquely suited to the terrain in which it landed. Long the domain of science fiction, machines capable of fully automated design and assembly are coming closer to reality, and have the potential to revolutionize personal and industrial manufacturing.

At first glance, fully automated design and assembly seems very close indeed. On the side of automated design we have the progress of Evolutionary Algorithms in the realm of design, which over the past twenty years, have produced a wide variety of objects ranging from tables [38, 29] and trusses [66] to entire robots [75]. Evolutionary Design has demonstrated the capacity for automated creativity unfettered by human bias. Indeed, the literature is ripe with unexpectedly novel, elegant, and “human competitive” solutions to design problems [40, 2, 51, 76, 64].

And on the side of automated manufacturing we have the recent advances in rapid prototyping and industrial robotics. State of the art bench-top systems, often dubbed “Santa Claus Machines”, are now capable of manufacturing relatively sophisticated objects out of a wide range of materials such as plastic, metal and ceramic. Cutting edge research units are even able to print batteries, wiring, and entire circuits [63, 16]. On a larger scale, massive industrial robots can now assemble entire cars.

This dissertation introduces *Evolutionary Fabrication*: a means by which we can merge the fields of Evolutionary Design and Automated Manufacture, and in that manner realize the full automation of design and assembly.

1.1 The Fabrication Gap

Unfortunately it soon becomes clear that the full automation of design and assembly is not as simple as feeding the product of Evolutionary Design directly into a Rapid Prototyping machine. Several evolved designs have been manufactured in the real world [30, 29, 75, 38, 28], perhaps the most notable being Lohn *et al.*’s evolved antenna [59], due to be launched into space aboard a Low Earth Orbit satellite this year. And yet, despite being automatically designed, they were all *manually* assembled.

This need for further human intervention arises because there is a knowledge gap between the process of design and the process of manufacture. How, exactly, does the rapid prototyping machine know *how* to print an object? How does the industrial robot know how to build the car? Conventionally, the end result of Evolutionary Design is a descriptive model of the evolved object: a blueprint. Rapid prototyping machines and industrial robots, on the other hand, require a specific set of instructions in order to perform their task: an *assembly plan*. Consider a blueprint as a photograph of a cheese souffle in a cookbook. While the photo may describe in great detail, and

in full color, what the finished soufflé should look like, it contains no information on *how* to cook it. In order to actually prepare the dish, a cook needs the recipe on the facing page. Similarly, before they can be assembled, evolved blueprints must be translated into a set of explicit instructions for the assembly mechanism. This crucial missing knowledge between the description of an object and the assembly of that object is the *Fabrication Gap*.

There are, of course, a few ways of automatically transforming a blueprint into an assembly process. Rapid prototyping machines, for instance, accept 3-D CAD files as input, and can reduce these models into a series of small horizontal slices, which they then print, layer by layer. Hornby was able to assemble his evolved tables and parts of his evolved robots in this manner [38]. On a broader scale, the field of engineering devoted to determining how to build a given object by inferring a sequence of assembly instructions is known as *Assembly Sequencing*. Often, in order to reduce the complexity of the task, conventional approaches to assembly sequencing make a number of strong assumptions about the process of assembly, for instance that it is *monotone* (once two parts are assembled they stay together) and *two-handed* (that each stage of assembly joins exactly two sub-assemblies) [32, 33]. Operating under these assumptions (and provided the object *can* be assembled in the first place) the method works, although the task of finding an optimal, or even near-optimal assembly plan has been proven to be NP-complete [45].

This raises a more fundamental question: how do we know that designs we have evolved are *buildable* at all? One approach is to constrain the representation used by evolutionary design in such a manner that it only produces buildable objects. Funes, for instance, was able to evolve large LEGO by using a representation which used the bricks themselves as primitives, pruning the evolved program tree in cases where mutation and crossover created an impossible structure [30, 29]. Funes' definition of

buildable is narrow, however, requiring only that the finished physical object behave identically to the simulated evolved object. As we'll see in Chapter 2, even these simple structures were not easy to assemble manually, much less automatically.

1.2 Bridging the Reality Gap

If there is another clear lesson to be learned from those evolved objects which have been successfully manufactured in the real world, it is the importance of realistic simulation in ensuring that the behavior of an evolved object corresponds to that of its physical counterpart [14, 42]. Funes used a type of finite element analysis to model the forces between LEGO bricks [29]. Lipson's GOLEM robots [75] and Hornby's tables and GenoBots [38] were all evolved with quasi-static kinematics simulators.

Realistically simulating the behavior of an evolved object helps to ensure that the real object will behave as expected *once built*. It does not, however, provide any guarantees as to how the object will behave *as it is assembled* in the real world. If we translate this notion of the importance of realistic simulation from the realm of behavior into the realm of assembly, then it stands to reason that the best way to ensure that an object can be assembled in the real world is to realistically simulate the entire process of its assembly.

1.3 Evolutionary Fabrication

This begs the question: if, in order to automate assembly, evolved blueprints must be translated into assembly procedures, and if a good representation and good simulation are both contingent upon detailed knowledge of the assembly process, why not skip the middle man and evolve those assembly plans directly?

A parallel exists here between the “design first, build later” approach to evolutionary design, and the classical “build first, program later” approach to robotics. In that scheme, engineers invent complex robotic systems, and only later try to find a controller capable of operating it. This has been described as a “chicken and egg problem” [31]: the evolution of robotic control assumes a fixed morphology, and the evolution of robotic morphology assumes a fixed and functional controller. Of course, biology doesn’t first “discover” an animal’s body, and only later its brain, rather both evolve in tandem. Inspired by those biological processes, modern robotics has met with considerable success by co-evolving morphology and control [73, 91, 53, 61, 17, 26].

Similarly, nature does not draw increasingly complex “blueprints” of the systems it wants to build and then subsequently try to assemble them. Rather, the form of a species and the *formation* of that species are intertwined. If we have learned the lesson of body-brain co-evolution, that morphology and control must arise together, why then are we recapitulating these mistakes in the design domain by first evolving complex objects and subsequently attempting to build them?

If Evolutionary Algorithms are indeed an “automated invention machine” [51], why not allow them to to invent *how* to build rather than what to build? Would this, in principle, allow evolution to invent not just novel objects, but novel means of assembling those objects as well? This is the central question that this dissertation seeks to answer.

Approaching Fully Automated Design and Manufacture from this perspective requires a new formulation of Evolutionary Design, one that replaces *descriptive* blueprints with *prescriptive* assembly plans. In this approach, the formation of an object can no longer be taken for granted; we must realistically simulate not only the behavior of a finished object, but its entire assembly as well.

In this dissertation we introduce *Evolutionary Fabrication*: a model of fully automated design and assembly which operates under this novel approach. In this model, we simulate an entire assembly mechanism and directly evolve the instructions for it. Through this evolution of *formation*, each result is, by its very nature, both automatically designed and automatically assembled. Ultimately, the evolution of form and formation become fully intertwined when the language of assembly itself becomes subject to evolution, capable of discovering increasingly large sub-assemblies and adding them to its vocabulary. In this co-evolutionary approach, Evolutionary Fabrication is able to simultaneously discover how to build large, complex objects and *what to build them out of*.

1.4 Assembly as Ontogeny

Approaching evolutionary design through assembly requires tools and perspectives different from those used in traditional Evolutionary Design. It is worth observing that as objects are assembled, piece by piece, they *grow* over time, developing slowly from initial components into a finished product. In fact, this process of growth is an *ontogeny* of sorts.

This analogy to the biological processes of growth and development can be extremely useful. Artificial systems inspired by these organic processes fall under the rubric of *Artificial Ontogeny* [52, 94]. Artificial Ontogenies are gaining popularity as a means of Evolutionary Design, and have been demonstrated to have a number of significant advantages including compactness and scalability [38], implicit modularity [13, 39], and high adaptivity and evolvability [96]. The relevant question in the context of Evolutionary Fabrication is whether these qualities can transfer into a system which explicitly models the physical assembly of an object.

1.5 Development and Noise

Developmental approaches to design are not without their drawbacks. For starters, there is the overhead associated with realistically simulating an object's complete development. Furthermore, as an object or an organism develops, each step of its growth can be considered a different phenotype - who is to say that the "final" result is the best [97]?

More problematic for the purposes of Evolutionary Fabrication are the deleterious effects of noise and error, which can induce a one-to-many relationship between genotype and phenotype: a single genotype may develop into an entire distribution of phenotypes, each with a corresponding fitness [81]. In biology, the interaction between ontogeny and environment is a cornerstone of the field of Developmental System Theory [69, 54]. The matter of developmental noise has lately begun to attract attention in developmentally-inspired artificial systems [110, 99, 98, 81, 83, 82].

Any system such as ours, which hopes to realistically model physical assembly, must therefore address the issue of how to *reliably* produce results despite environmental noise. In this work we show how even simple *ballistic* assembly process, that is, one without the means to measure intermediate progress, can nonetheless learn to reliably assemble a goal structure.

1.6 The Growth of Complexity

It is one thing to say that we can use Evolutionary Fabrication to evolve how to build certain things. It is quite another to say that the process will scale to increasingly large objects, particularly under the presence of noise. Work on Artificial Ontogenies has shown how developmental approaches can produce scalably large and complex *representations* [38, 13, 94], but if the assembly system which interprets those repre-

sentations is limited to a single sized building block, such as a LEGO brick, then the assembly of larger objects becomes increasingly difficult, particularly as small errors during assembly begin to accumulate.

In his seminal essay “The Architecture of Complexity”, Herbert Simon [90] argues that the evolution of complex forms from primitive elements is highly contingent upon hierarchical, modular assembly. Using his famous parable of two watch-makers, Simon demonstrates how modularity insulates systems from the effects of error during assembly, and greatly increases the rate at which large complex forms can emerge.

This reasoning translates readily into the realm of assembly and ontogeny: in order to build larger, more complex objects, development must discover increasingly larger components with which to build them. In the context of Evolutionary Fabrication, this means that our system must be capable of discovering increasingly large, increasingly complex, and reliably attainable *sub-assemblies* and incorporate them into the language of representation as new building blocks .

Several models of modular acquisition in Evolutionary Algorithms exist [50, 4, 86, 21, 101]. Most, however, assume that the same genotypic sequence will have the same phenotypic consequence across different contexts. They can therefore be stymied by developmental representations such as ours, in which the phenotypic consequence of a genetic sequence is highly contingent upon its context. To better illustrate this notion of context sensitivity, let us once again return to our earlier example of a *souffle* recipe. The set of instructions (a genotypic module) which produces whipped egg whites (a phenotypic module) in a souffle recipe would produce a mess (if anything at all) if they occurred later in the recipe or, for that matter, in an omelet recipe.

The challenge of modular acquisition in such *context dependent* artificial ontogenies therefore lies in figuring out a process by which the *meaning* of a favorable genetic sequence, rather than the *syntax*, can be preserved. The means by which

we overcome achieve this is *symbiogenesis* - the process by which one organism, the *symbiont*, is completely absorbed by its host. In the endosymbiotic model of modular encapsulation, complete organisms, not just specific portions of their phenotype, are used to form modules.

1.7 Summary

Broadly, the aim of this work is to foster an entirely new approach to the “automated invention machine”, one which replaces the evolution of *form* with the evolution of *formation*. Doing so, we argue, avoids the informational “Fabrication Gap” between the *descriptive* blueprints produced by conventional approaches and the *prescriptive* instructions needed to build them.

More concretely, our central contribution lies in introducing *Evolutionary Fabrication*, the direct genetic programming of an assembly mechanism, as a suitable alternative and a promising approach to the full automation of design and assembly. We claim that by evolving *how to build*, not only can it produce *buildable* objects, but it is capable injecting the novelty and creativity seen in conventional evolutionary design into the realm of fabrication, discovering not only novel designs, but novel means of assembling those designs as well. Further, we show how Evolutionary Fabrication has mechanisms, both implicit and explicit, which allow it build scalably large complex structures, in the presence of noise and error during assembly.

Combined, these contributions lay the crucial theoretical and empirical foundations for our ultimate goal of creating a real-world, embodied Evolutionary Fabrication system.

1.8 Preview

Chapter 2 lays the foundation for our work. It will describe the field of Evolutionary Design in more detail, discussing related work and paying particular attention to case studies of the few examples of physically embodied results. This will lay out the motivation behind the creation of Evolutionary Fabrication as a new paradigm for fully automated design and assembly.

Chapter 3 describes our experimental framework for exploring the capabilities and limits of Evolutionary Fabrication (EvoFab), and provides a brief example of the process by evolving the assembly plan of a specified goal structure.

Chapter 4 demonstrates the ability of our Evolutionary Fabrication framework to perform open-ended design tasks, learning to build structures given only a broad specification. Section 4.3 demonstrates the emergence of novel means of assembly in Evolutionary Fabrication (see Figure 1.1), and explores the causes of those phenomena.

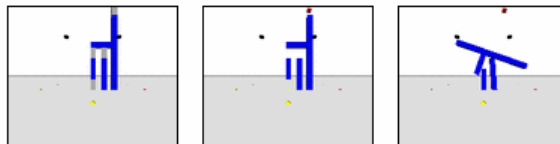


Figure 1.1: An example of novel dynamic assembly from Section 4.3

Chapter 5 explores the effects of noise and error during Evolutionary Fabrication. Section 5.1 demonstrates the ability of EvoFab to discover how to reliably build a goal structure in the presence of noise, even without the means to measure intermediate results. Section 5.2 demonstrates how Evolutionary Fabrication can reliably build large, complex structures in noisy environments through the discovery of hierarchical modular assembly (see Figure 1.2) .

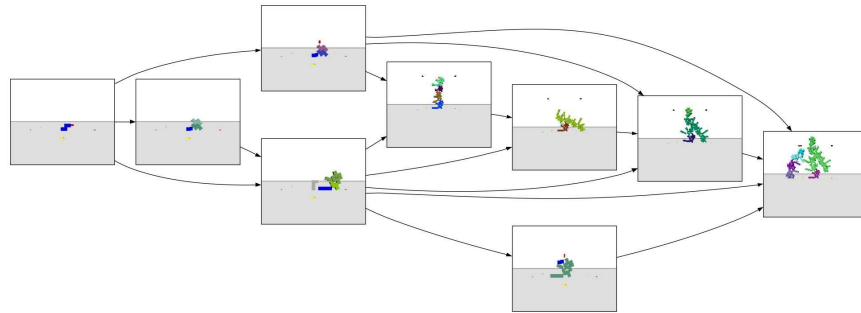


Figure 1.2: A robust hierarchical modular assembly from Section 5.3

Chapter 6 discusses in more detail the themes and implications of Evolutionary Fabrication, and sketches out the future of fully automated design and assembly, sharing some ideas on how such a system can be implemented in the real world. Finally, **Section 6.6** summarizes and concludes this dissertation.

Chapter 2

Foundations

This chapter will lay out the central themes of this dissertation. We begin by introducing the Evolution of Form, the branch of Evolutionary Design which seeks to automatically create complex and novel objects. We then explore in greater detail more recent efforts to physically manufacture these evolved designs. As we'll see, this transition from evolved design to physical object is far from seamless. These complex objects, designed automatically and without human effort, subsequently require significant human intervention to physically manufacture. As evolved designs become increasingly complex, a *Fabrication Gap* opens up between those objects which we can evolve and those which we can actually manufacture.

A central claim of this chapter is that the root cause of this difficulty lies in the use of *blueprints* to describe evolved objects. By only specifying *what* to build, blueprints leave unanswered the equally important question of *how* to build it. Absent this information, the complexity of those blueprints we can evolve quickly out-paces the complexity of those blueprints we can determine how to manufacture, thus, the Fabrication Gap.

We argue that the solution to this gap begins with the evolution of *formation*

rather than form. Artificial Ontogenies, a type of Evolutionary Computation inspired by biological growth, are a natural method of modeling formation, and allow us to evolve *how* to build rather than simply *what* to build.

Of course, while their prescriptive nature may help avoid the Fabrication Gap, developmental representations are not without their own burdens. Error and noise during development can significantly complicate the task of evolution. When subjected to noise, a developmental genotype is capable of growing into an entire range of phenotypes, each with a corresponding fitness. As the scale of development increases, so do the deleterious effects of noise.

The best way to combat noise while evolving the assembly of increasingly complex objects is through hierarchical, modular assembly. In essence the key lies in finding something, *anything* that you can reliably build in the presence of noise, and then adding that robust object as a new primitive in your assembly process. It is through such *endosymbiotic encapsulation* that the evolution of form and the evolution of formation become fully intertwined.

We end the chapter by introducing Evolutionary Fabrication as the Artificial Ontogeny-based marriage of Rapid Prototyping and Evolutionary Design, which will lead to the full automation of design *and* assembly.

2.1 The Evolution of Form

Generally speaking, Evolutionary Algorithms (EAs) are a form of population-based informed random search inspired by biological evolution. The most popular types of EA are Genetic Algorithms (GAs) [67], developed circa 1973, Evolution Strategies (ESs) [8] circa 1973, and Genetic Programming (GP) [50], circa 1992.

Loosely speaking, the appeal of Evolutionary Algorithms lies in trying to harness

some of the creative energy demonstrated by the only process known to have successfully generated complex intelligence: biological evolution of life on earth. The reasoning goes that what is good for the biological goose is good for the computational gander. More specifically, Evolutionary Algorithms have the ability to exploit the underlying structure of large search spaces in order to arrive at unique solutions. While certainly no *panacea* (see, in particular, Wolpert and Macready's No Free Lunch [107]), they are often capable of arriving at novel solutions to difficult problems.

Evolutionary Algorithms can be used to evolve any number of things: from a simple string of bits, to the configuration of a Field Programmable Gate Array (FPGA), to the complete brain and body of a robot. And yet, while the products of evolution may be different, the fundamentals remain the same. The basic unit of manipulation is the *genotype*, which can be changed via genetic operators such as mutation and crossover. The genotype serves as an encoding of the *phenotype* (although in simpler EAs the two are the same), representing a candidate solution to the problem at hand, which can then be *evaluated* for *fitness*. Populations of solutions are bred from generation to generation using the Darwinian principle of selection of the fittest, until a desired result is achieved. A more thorough treatment of EAs can be found in [67].

Since the term "design" is used loosely in the field, it is important to be clear about what we mean by "design" in the context of this dissertation. In his book "Evolutionary Design by Computers", Peter Bentley [7], makes an effort to distinguish *Creative* Evolutionary Design from what he calls Evolutionary Design Optimization. In this view, the former seeks to create new designs out of whole cloth, whereas the latter seeks to optimize various aspects of a pre-existing design. This notion of *creativity* is important. The hope is that Evolutionary Algorithms can arrive at designs which possess unanticipated novelty, equaling or even surpassing human

designs. Such “human-competitive” results include sorting networks [40], photonic crystals [76], optical lens systems [2] and quantum Fourier transforms [64]. Indeed John Koza, one of the creators of Genetic Programming has dubbed GP “an automated invention machine” [51], explaining that its capacity for novelty arises because it is not bound by human reasoning and logic.

Here, we are specifically interested in what we call the *Evolution of Form*, for instance of satellite antennae [59], robot morphology [10, 75], or of entire buildings [88]. What separates the Evolution of Form from other types of evolutionary design is that the end result has some tangible shape, rather than simply being an arrangement of bits. This distinction becomes clear when the time comes to turn the evolved objects into physical artifacts. For many real-world evolutionary designs, such as the FPGAs used in Evolvable Hardware, the transfer from simulation to reality is relatively easy, because the designs take the form of pure information, which can be effortlessly and automatically transferred to their physical counterparts. By contrast, an evolved robot must be crafted, servo motors attached, and batteries charged before it can walk. Likewise, an evolved satellite antenna cannot be launched into space until it has been physically formed and attached to its host.

The evolution of form carries with it the promise of entirely new designs of physical objects. If, however, we want to bring these novel objects into reality we must face the prospect of *manufacturing*, and with it the corresponding *Fabrication Gap*.

2.1.1 Case Study: Sims Creatures

One the earliest and most frequently cited examples of the use of Evolutionary Algorithms to design an object (as opposed to a bit string or program tree) is Karl Sims’ seminal work on virtual creatures [91, 92]. Although more than ten years old, Sims’ work has had far-ranging consequences and provided several key insights which have

had a significant effect upon the entire field of Evolutionary Design, which has, in a sense, been playing catch-up ever since.

First among his contributions is the evolution of robot morphology as well as control. EAs had already been used to evolve controllers for a variety of robots with *fixed* morphologies. Sims' crucial contribution in this regard was in using an encoding which allowed him to evolve the entire physical structure of his robots alongside their neural controllers. By co-evolving morphologies with their controllers, rather than using some *a priori* shape, the evolutionary system was able to generate virtual creatures whose bodies were tightly coupled to their chosen fitness function. Creatures evolved for swimming both looked and behaved differently than those evolved for walking or jumping.

This leads to his second significant choice - the use of a developmental representation. Rather than the canonical bit-string, his genetic encodings were variable-size directed graphs which, when interpreted, "grew" into the specified creature. Recursive connections allowed for a measure of modularity and reuse in the representation. As a result, compact graphs could be used to generate relatively large and complex bodies which exhibited symmetry and modular re-use.

Third, his creatures were evolved within a virtual physics environment. By situating his creatures in a world with gravity, friction, and collisions, they were able to develop surprisingly "life-like" behaviors.

Finally, inspired by Hillis work on parasites [37] and Angeline's work on competition [3], Sims used a competitive co-evolutionary fitness scheme, in which evolved creatures were pitted against each other, rather than a static fitness function.

Although none of his choices were particularly unique, it was this combination - the use of developmental representations to "grow" a robot's body and brain, the evaluation of that robot in a realistic environment, and competitive co-evolution,

which led to the emergence of novel and interesting behaviors, which have since become benchmarks for evolutionary design. There has been no shortage of work in Evolutionary Design since Sims, and yet almost everything carries echoes of that seminal work.

The vast majority of evolved forms since Sims have been of purely virtual objects. Beginning in 1996, Peter Bentley began exploring the evolution of form - first with lenses [6] and later with tables [5]. Other examples of evolved fixed forms include Eggenberger's cellular-based 3-D forms [25], Parker's towers [71], and Jacob's trees [41]. Several researchers have used Genetic Algorithms to evolve architectures [87, 18, 88]. Quite a few others have followed Sims' lead and evolved virtual creatures, such as Bongard's agents [13, 11] and Komosinski's "Framsticks" [48]. More recently, Gondarenko *et al.* evolved a simulated photonic structure capable of high photonic confinement [76].

2.2 Manufacturing Evolved Designs

Our interest, however, is in automatically designing and manufacturing *physical* objects. As such, we have the most to learn from those examples of evolved designs which have been subsequently manufactured in the real world. Funes [29] was among the first to bring evolved designs into the real world with his EvoCAD work, which evolved LEGO structures in a force-based simulator. By accurately modeling the forces between elements, his system was able to create unique structures in simulation which could subsequently be built in the real world. Figure 2.1 contains an example of one of his evolved blueprints, and the corresponding physical object. Following Funes' work, Regli *et al.* used graph grammars to represent the assembly of LEGO structures [72, 49], and were able to evolve walls, pillars, and staircases.

Shortly after Funes' work, Hod Lipson's GOLEM project [75] created printable, controllable robotic forms. Like Sims and Funes, he relied upon a realistic physical simulator, and like Sims, he co-evolved simple physical building blocks with simple neural controllers. The designs were printed on a rapid prototyping machine, motors and batteries were snapped into place, and the resulting physical robots were able to locomote on a flat surface.

Frutiger *et al.* [28] evolved the morphology and controller for a monkey-like swinging robot inside of a physics-based simulation, and then iteratively transferred that result onto a physical prototype over the course of several weeks.

Later, Hornby [38] used a grammar-based system to evolve tables and mobile robots. The developed genotype consisted of instructions to a LOGO-like turtle which then "drew" the structures out of voxels in simulation. Although early results were transferred by hand into CAD before printing on a 3-D printer, Hornby's later designs created CAD files automatically. Like Lipson's GOLEM, once the bodies were printed, final assembly, including the addition motors and wiring, was performed by hand. Figure 2.2 contains an example of one of the evolved GenoBots on the left, and the corresponding physical robot on the right.

The most significant recent result of real-world evolved design is probably Lohn *et al.*'s work on Evolved Antennas [59] - one of which is due to be launched into space aboard a Low Earth Orbit satellite. These designs were generated by L-systems in a manner similar to Hornby's work, and tested in an antenna simulator before being assembled by hand.

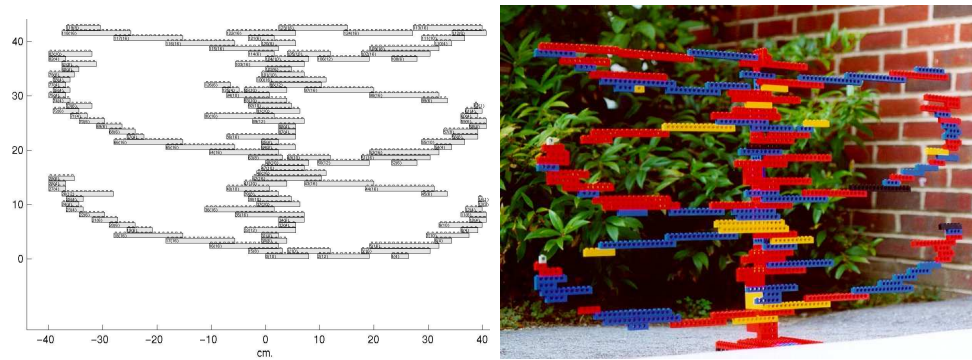


Figure 2.1: Pablo Funes' evolved tree from [29], reprinted with permission from the author. The evolved blueprint is on the left, and the corresponding physical result on the right.

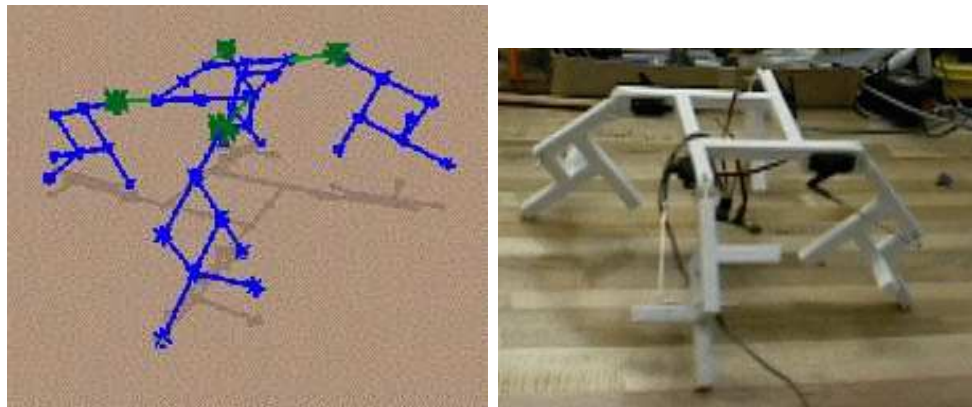


Figure 2.2: One of Greg Hornby's evolved GenoBots[38], reprinted with permission from the author. The evolved blueprint is on the left, and the corresponding physical result on the right.

2.3 The Fabrication Gap

While the above physical objects may have been automatically *designed*, their manufacture was far from automatic. In each case, when it came time for the assembly of those designs, they were all built by hand, in a manner which required significant human interaction. For Funes' LEGO trees, even with blueprints which explicitly described the placement of each piece, hand assembly remained a difficult task (this is well illustrated by the Scientific American Frontiers episode ¹ in which the host, Alan Alda, attempts to build one of the designs) [29]. Frutiger's evolved monkey required significant tweaking before the physical counterpart behaved like the evolved model [28]. And Lohn *et al.*'s satellite antenna had to be meticulously soldered and bent by hand, with care to preserve the precise angles specified by the evolved design [59].

In the examples above, human intervention, rather than being removed altogether, has simply been shifted from the design phase to the manufacturing phase. And yet, if the success and novelty of Evolutionary Design is due to the fact that, as Koza put it, "it does not travel along the well-trod paths of previous human thinking" [51], why are we subsequently injecting that human logic and bias back into the realm of assembly? The need for subsequent human effort to move from evolved design to manufactured object is due to what we call the "Fabrication Gap". Manufacturing processes, either human or automated, require as input some *prescriptive* set of instructions on *how* to build. Conventional Evolutionary Design, on the other hand, produces purely *descriptive* representation of objects. If we seek to remove human effort from the process completely, to fully automate both design and assembly, then we must find a way to automatically produce prescriptive representations of assembly.

¹<http://www.pbs.org/saf/1103/segments/1103-3.htm>

2.3.1 Descriptive Representations and the Assembly Inference Problem

To humans, descriptive blueprints seem like a natural way to represent an evolved object. After all, architects use blueprints to design buildings, and engineers use those same blueprints to build them.

This evolution of blueprints is enough, provided that the end goal of an Evolutionary Design system is a virtual representation of the object. If, however, one is seeking to *manufacture* those evolved forms, then blueprints alone are insufficient for the task. A vast amount of expert knowledge lies between the blueprint of a building and the building itself. Nothing in the plans of a house, for instance, suggests that the foundation must be built before the roof is. As a more concrete example, consider Funes' evolved blueprint of a tree on the left hand side of Figure 2.1. Nothing about the design suggests which brick, or even which branch, should be placed first. Should the trunk be built first, and then the branches added, or perhaps vice versa? Nothing about it, certainly, suggests that the easiest way to build it, as it turned out, is by building it horizontally on a flat surface and tilting it into place.

One notable exception to the reliance on purely descriptive representation is Regli *et al.*'s LEGO structures [72, 49]. Their work utilized conceptual graphs as “assembly representations”, explicitly describing the physical relationship and connections between each component in the evolved structure. However, this semantic representation was relatively high-level, and described only the state of the complete object, not its manufacturing process.

To explain the descriptive weakness of descriptive representations, consider a blueprint as a photograph of a cheese souffle in a cookbook. While the photo may describe in great detail what the final result should look like, it contains no information

on *how* to cook it. In order to actually prepare the dish, the recipe on the facing page is required. Before they can be manufactured, therefore, evolved blueprints must be translated into a set of assembly instructions, just like the photograph of the soufflé. This task can either be performed by human minds with their vast wealth of insight and common sense knowledge, or computationally.

Although the process of determining an assembly sequence may come readily to humans, it is often much harder to solve computationally. Rapid Prototyping Machines approach this by accepting 3-D CAD files as input and reducing these models into a series of small horizontal slices, which they then print, layer by layer. Hornby was able to automate the assembly of his evolved tables in this manner [38].

In the field of engineering, the task of inferring a sequence of assembly instructions given a particular structure *a priori* is known as *Assembly Sequencing* [105, 109]. Often, in order to reduce the complexity of the task, conventional approaches to assembly sequencing make a number of strong assumptions about the process of assembly, for instance that it is *monotone* (once two parts are assembled they stay together) and *two-handed* (that each stage of assembly joins exactly two sub-assemblies) [32, 33, 104]. When operating under these assumptions the method works (provided the object *can* be assembled in the first place), however the task of finding an optimal, or even near-optimal assembly plan has been proven to be NP-complete [45].

Assembly Sequencing often involves the much easier inverse problem of *disassembly planning* - that is, removing parts from an object one at a time until it has been reduced to its basic components. Doing so, however, makes the critical assumption that every stage of assembly is both *reversible* and *symmetric*. Of course, anyone who has taken apart a home appliance and then put it back together, only to be left with a remaining mysterious screw, knows that assembly and disassembly are rarely symmetric, reversible processes in the real world. In later chapters we will

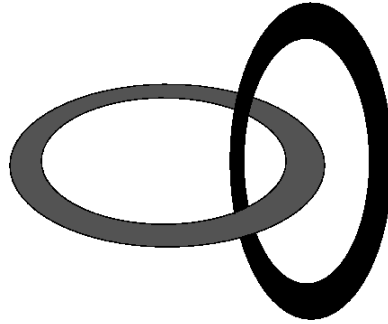


Figure 2.3: Intertwined Rings: an example of an unbuildable design. If the primitive set contains rings, and the design specifies their position and rotation, then it may evolve interlocking rings. Any assembly process which begins with separate, solid rings, however, will never be able to build this structure.

give examples of evolved assembly plans which employ clearly *non-reversible* actions during assembly.

2.3.2 Buildability

The evolution of descriptive blueprints also runs the risk of generating designs which are completely *unbuildable* by the assembly process. If, for instance, the primitives of the design system includes solid rings an evolved blueprint might specify a design which involves two interlocking rings (Figure 2.3). Any real-world assembly process which begins with solid and separate rings, would be incapable of intertwining them in this manner. Most approaches to the evolution of form therefore place further constraints on the language of representation. Funes [30], Regli [72], and Parker [71] for instance, produce objects which can be built out of LEGO by using the bricks themselves as the basic component of design. Funes' method was careful to prune evolved representational trees to prevent them from producing structures with overlapping bricks. Similarly, Bentley's early work on the evolution of solid objects allowed the generation of "impossible" objects, which were then corrected during the mapping from genotype to phenotype [5].

2.3.3 Simulation and The Reality Gap

If there is another clear lesson to be learned from those evolved objects which have been successfully manufactured in the real world, it is the importance of realistic simulation in ensuring that the behavior of an evolved object corresponds to that of its physical counterpart [14, 42].

Each of the physically embodied evolutionary designs above relied upon a realistic physics environment to evaluate their designs. Funes used a custom variety of finite element analysis to model the forces between LEGO bricks and added a 20% safety margin to ensure his structures could be transferred successfully. [29]. Lipson *et al.* used a quasi-static motion simulator to model the behavior of their GOLEM robots [75], as did Hornby with his evolved tables and robots [38]. Both Linden's [56, 57] and Lohn's [59] antennae were evolved inside of electromagnetic simulators.

Realistically simulating the behavior of an evolved object helps to ensure that the real object will behave as expected *once built*. It does not, however, provide any guarantees as to how the object will behave *as it is assembled* in the real world. If we translate this notion of the importance of realistic simulation from the realm of behavior into the realm of assembly, then it stands to reason that the best way to ensure that an object can be assembled in the real world is to realistically simulate the entire process of its assembly.

2.3.4 Summary

In summary, the Fabrication Gap between that which we can *design* automatically and that which we can *build* automatically is due in large part the way that contemporary Evolutionary Design is performed. The blueprints evolved by conventional approaches are purely *descriptive* representations of design, and leave unanswered the question of

how to build the evolved object. Methods of automatically determining the assembly of a given blueprint become impossible as the complexity of designs increases. Finally, evolving *what* to build rather than *how* to build also runs the risk of discovering objects which are impossible to build at all, such as intertwined rings.

One workaround is to incorporate the constraints of your expected assembly mechanism into the design process through carefully crafted representations and accurate physical models. This is the approach taken by conventional *post hoc* methods of crossing the Fabrication Gap, such as slicing and Assembly Sequencing. And yet for sufficiently complex design spaces and assembly processes, it may be impossible to exhaustively enumerate all limitations and constraints. Furthermore, to over-constrain a design process is to run the risk of crippling the creativity that is so essential to evolutionary design.

Indeed, if one is going to put the effort of injecting knowledge of a assembly system into the design space, why not instead simply simulate the entire assembly mechanism, and evolve assembly plans instead of blueprints? Objects produced in this manner are by their very nature buildable, and are produced in tandem with the information necessary to build them.

2.4 The Evolution of Formation

A central observation of this dissertation is that mechanical assembly is an *ontogeny*. As objects are assembled, piece by piece, they *grow* over time, developing slowly from initial components into a finished product. It seems natural, therefore, to cast our gaze to biology for inspiration on evolving *how* to “grow” objects.

Of particular interest to us is Evolutionary Developmental Biology, or Evo-Devo, which studies the relationship between biological development and evolution [80, 34].

The link between the evolution of form and the processes of formation date back to the German biologist Ernst Haeckel who illustrated striking similarities between the embryologies of distinct vertebrates. In his “Fundamental Biogenetic Law”, more commonly referred to as “recapitulation theory”, the stages of development (ontogeny) of an organism recreate, in condensed form, the evolutionary history (phylogeny) of an organism.

While strict recapitulation has been discredited (Haeckel “fudged” several of his drawings to reinforce his point [34]), the echoes of the Biogenetic Law abound in modern evolutionary developmental biology. There are clear instances of *homology* in comparative embryology, in which distinct species which share an evolutionary past often share developmental traits [34, 93].

One of the strongest examples of homology, and one of the most important discoveries in comparative developmental biology, is the set of *homeobox* genes, which play an important role in laying out the spatial structure of a developing embryo. First discovered in *drosophila*, homeobox genes have been discovered in a number of vertebrates and insects. Mutations to homeobox genes can have wide-ranging effects on the morphology of an organism, affecting everything from patterning to the placement of limbs [80].

Slack *et al.* use the existence of homologies such as the homeobox gene to argue for the existence of the “zootype”: a developmental stage common to all animals, despite high variation in developmental processes both before and after it. They argue that this “phylotypic stage” has been conserved during evolution because this is the point at which the basic body plan, *bauplan*, common to all species is laid out and at which development is most brittle [93, 36].

This conservation of the phylotypic stage during evolution and speciation results in what is referred to as the “phylotypic hourglass”: different species vary greatly

in their respective developments both prior and subsequent to the phylotypic stage, but vary significantly less during the phylotypic stage [36, 94]. The largest source of variation during the phylotypic stage is *heterochrony*: changes in the developmental timing or sequence of events [79, 78]

Developmental features such as the homeobox genes and the *bauplan* also point to a level of modularity in developmental systems, which we discuss at more length in Sections 2.4.5 and 5.2.

From the perspective of Evolutionary Design, the most valuable aspect of biological development is its ability to generate enormously complex systems from a (relatively) compact genetic representation. As Stanley and Miikkulainen point out, there are 30 thousand active genes in the human genome, which manage to produce 100 trillion neural connections in the human brain. Moreover, biology manages to reliably accomplish this generative feat through the essentially stochastic processes of biochemistry [94]. Our interest is in harnessing this efficiency and robustness for the purposes of automating artificial manufacturing processes.

2.4.1 Artificial Ontogenies in Design

Artificial developmental systems which use biological growth and development as metaphors for physical assembly fall under a variety of names - Artificial Embryogeny [94], Artificial Embryology [20], Computational Embryology [52] and Artificial Ontogeny [10, 13] . Since the term *ontogeny* is the broadest of the terms, encompassing the entire course of biological development from conception to final form, we will use *Artificial Ontogeny* to collectively describe these developmental representations.

In their *Taxonomy* [94], Stanley and Miikkulainen divide Evolutionary Designs based on Artificial Ontogenies into two groups: cellular-based and grammar-based.

Cellular Approaches

Cellular approaches to Artificial Ontogeny take their inspiration from the biochemical processes of cellular life. Dellaert and Beer use a developmental model in which an “egg” slowly grows into a multicellular robot [23, 24]. Bongard and Pfeifer [10, 13] use a model of Gene Regulatory Networks (GRNs) to evolve the body and brain of robotic agents. Similarly, Eggenberger used Differential Gene Expression to evolve 3-D shapes and objects [25] and De Garis [20] evolves both 2-D and 3-D shapes using an “Artificial Embryo”. Bentley and Kumar use a variety of different “embryogenies” to evolve target 2-D shapes. On a broader scale, Bonabeau *et al.* use a multi-agent “stigmergic” system, much like swarming ants, to evolve 3-D architectures [9].

While cellular approaches have produced several interesting results, their treatment of ontogeny is too abstract to readily lend itself to the description of automated manufacture at the scale we are interested in.

Grammatical Approaches

Grammatical approaches to Artificial Ontogeny instead rely on the more abstract rules of artificial grammars to model biological growth. Generally, grammatical approaches use a series of rewrite rules to transform a short initial S-expression into a larger string which represents the desired object. Coates uses Genetic Programming and Lindenmayer systems (L-systems) to evolve both 2-D and 3-D shapes [15]. In [96], Toussaint uses L-systems to evolve 3-D plants in OpenGL. Hornby’s GenoBots, which we discussed above, also used L-systems to evolve ruled to “draw” 3-D voxels representing robot morphology. Lohn’s antennae were evolved in a similar fashion [59].

As we discuss at more length below, from our perspective of automating assembly, the major advantage of grammatical approaches is their ability to produce linear

strings of instructions which, if phrased correctly, can be used to explicitly describe the *process* of an object’s assembly.

2.4.2 Benefits of Artificial Ontogenies

Unlike traditional evolutionary computation, Artificial Ontogeny treats the genotype as an *indirect*, or *procedural* encoding of the phenotype. The genotype is decoded and transformed into a phenotype by means of some developmental process. This abstraction layer between genotype and phenotype allows for quite a bit of flexibility during evolution. To begin with, Artificial Ontogenies, much like biological ones, allow for a compact representation of a solution. Small changes in a genotype can have large consequences on the fully developed phenotype. Hornby, for instance, was able to show how a single change in his L-system representation of a table produced co-ordinated changes on all four legs [38].

Furthermore, developmental systems are capable of a high degree of both explicit and implicit modularity, allowing for highly structured hierarchical organization [13, 39]. These results mirror the development of high-level repeated structure and symmetry in both plants and animals, such as those in which *Hox* genes play a role. Moreover, developmental approaches allow for a level of redundancy: multiple genotypes can map to the same phenotype. Toussaint has demonstrated how “neutral” mutations between genotypes which produce the same phenotype can allow developmental systems to adapt and improve their evolvability [96].

2.4.3 Development of Representation vs. Representation of Development

While these development approaches model biological growth at an abstract level, they do not necessarily lend themselves readily to the task of automated assembly. To begin with, the final result of each of the ontogenies reviewed above is still, a purely descriptive representation of the evolved object. Although these models can be very detailed, they nevertheless remain, essentially, three-dimensional blueprints, thereby negating their utility for automated manufacture.

The L-systems used by both Hornby [38] and Toussaint [96] point in a promising direction, in the sense that they are used to generate strings of OpenGL instructions which are then interpreted to “draw” 3-D objects out of voxels. Such drawing, however, ultimately bears little resemblance to *physical* assembly.

As we have discussed, the Fabrication Gap is due to traditional Evolutionary Design’s reliance on descriptive blueprints. While Artificial Ontogenies offer an alternative by allowing us to evolve *how* to build, they are of little use if, in the end, they are only used to produce blueprints. If our goal is to automate both design and assembly, then the end result of our process should instead be an *explicit* set of instructions which lend themselves to automatic interpreted by a manufacturing system.

Furthermore, most Artificial Ontogenies used for design take the actual assembly process for granted, either by allowing virtual structures to appear *ex nihilo* - that is, out of thin air - or else *in utero* - in a very simplified environment, significantly less complex than the real world environment in which their physical counterparts are to be assembled. Hornby’s tables, for instance, were not subject to gravity as they were assembled, only when they were completed [38].

One exception to this lack of “embodied” development is Bongard’s Gene-Regulatory Networks [10], which slowly “grew” a robotic morphology piece-by-piece in a realistic physics environment. Bongard’s cellular approach, however, has the same descriptive weakness as other cellular approaches in that it does not easily lend itself to interpretation by an *external* assembly mechanism.

Applying the lessons about the importance of realistic simulation (which we reviewed in Section 2.3.3) to the realm of manufacture leads us to assert that the best way to ensure that evolved objects can be built is to simulate their entire assembly *in situ*, that is in an environment that closely resembles the physical environment in which they will be ultimately assembled.

2.4.4 Noise and Development

Developmental representations are not without their drawbacks. In particular, stochastic effects which lead to error and noise during development can significantly complicate the task of evolution. When subjected to noise during development, a genotype is capable of developing into an entire range of phenotypes, each with a corresponding fitness (Figure 2.4) Determining which, if any, is the phenotype most representative of the originating genotype is a difficult, and in some cases, entirely misleading task [97].

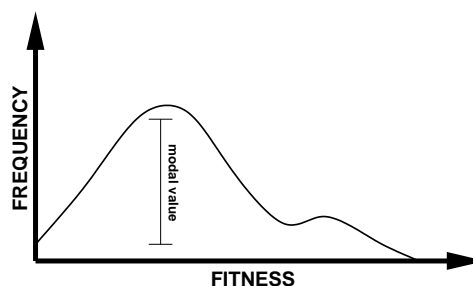


Figure 2.4: Under the presence of developmental noise, each genotype develops into an entire **range** of phenotypes, each with an associated fitness.

In biology, the interaction between ontogeny and environment is a cornerstone of the field of Developmental System Theory [69, 54]. Lewontin points out that the same phenotypic trait, for example eye size in *drosophila*, can be affected by both mutation and environment [54]. He goes on to note that “small events at the level of thermal noise acting during cell division and differentiation have large effects on the final developmental outcome” [55].

The matter of noise and error during development has only lately begun to attract attention in developmentally-inspired artificial systems. Yilmaz and Wu recently explored the relation between genetic redundancy and developmental noise [110]. Viswanathan [98] has studied the impact of stochastic development on assembly , and has demonstrated the ability of adaptive processes which measure the state and progress of the system to achieve higher reliability than purely ballistic processes.

Any approach such as ours, which hopes to successfully assemble complex objects in the physical world, must therefore be sure to address its ability to overcome noise and error during during the manufacturing process.

2.4.5 Adaptive Representation

One outstanding challenge in the open-ended evolution of formation lies in *scalable complexity* - that is, how to build increasingly large, increasingly complex objects in a managed fashion. As Herbert Simon argues in his seminal essay “The Architecture of Complexity”, hierarchical, modular assembly is crucial for the evolution of large complex forms [90]. A popular example of modularity in biological systems is the *eyeless* gene in *drosophila* which, when mis-expressed, causes complete eyes to sprout on the wings, legs, and antennae of the flies [35]. Wagner and Altenberg [100] persuasively argue that the evolvability of a system is highly contingent upon its ability to adapt its representation by discovering and incorporating evolutionary modules.

The value of modularity lies in coupling functionally related portions of the genotype while simultaneously decoupling unrelated portions. Changes to a representational module have few side effects in the remaining genome, and changes outside a module have few effects upon the module.

From the perspective of the Evolution of Formation, this means that the language of assembly must itself be mutable and adaptive, capable of discovering and using new modules over the course of evolution. Several models for adaptive representation exist, the most common of which fall under the rubric of Hierarchical Genetic Programming (HGP) where encapsulated modules become new primitives in the language [77, 50, 4, 86, 21]. While they vary in their details, each of these models of modular encapsulation involve incorporating *genotypic* sequences, thereby protecting them from the deleterious effects of mutation and crossover, and then adding them to the language of representation. As such, encapsulated modules are simply shorthand for the genetic sequence they represent - one can be substituted for the other without consequence.

Since we are using developmental representations to model the actual physical assembly of an object, such purely genotypic encapsulation is insufficient for our purposes. Because of their prescriptive nature, developmental representations display a measure of context dependency: the same sequence of operations can have vastly different results depending on where in the process it occurs.

The challenge of modular acquisition in developmental representations, then, lies in preserving not the syntax, but rather the *meaning* of a desired phenotypic result. Chapter 5 discusses these issues in more detail, and introduces an alternative model of encapsulation in which complete structures, not specific portions of their genotype, are used to form modules.

2.5 Evolutionary Fabrication

The lesson drawn from recent efforts at bringing evolved designs in the real world is clear. The Fabrication Gap is caused in large part by the evolution of purely descriptive *blueprints* which leave unanswered the question of *how* to build the evolved object. Furthermore, these approaches are capable of designing objects whose assembly is extremely difficult to discover, and, indeed, objects which are not in fact buildable at all. *Post-hoc* attempts at discovering an object's assembly through slicing or Assembly Sequencing make generalizations about the assembly process. Although these constraints make the task of inference easier, they also lock the assembly process itself into these modes. There is no purely mechanical reason why rapid prototyping machines must print objects with successive accretive layers, nor any reason that compound objects must be assembled by a monotone, two-handed process. As we will see in later chapters, directly evolving *how* to build removes these assumptions, and unleashes more novel ways of assembling objects.

The solution then, is to directly evolve the process of an object's assembly. Artificial Ontogenies, inspired by biological growth, provide the best framework to accomplish this. But Artificial Ontogenies are insufficient for describing assembly unless they explicitly model an object's manufacturing process, and can be explicitly be interpreted by a specific assembly mechanism.

Evolutionary Fabrication, the marriage of Evolutionary Design, Automated Manufacture, and Artificial Ontogeny, offers to replace the *evolution of form* with the *evolution of formation*. By directly evolving assembly plans, and by simulating the entire process of assembly *within* a manufacturing system, it can simultaneously evolve *what* to build and *how* to build it, thus avoiding the Fabrication Gap.

2.5.1 Formal Model

Formally, we can consider Evolutionary Fabrication to consist of several coupled aspects. First is the Assembly Mechanism \mathcal{M} which accepts instructions from a set \mathcal{L} . Every finite assembly procedure $\alpha \in \mathcal{L}^*$ when executed by \mathcal{M} , produces a structure $s \in \mathcal{S}$, where \mathcal{S} is the set of all structures buildable by \mathcal{M} with measurable properties which we are seeking to optimize. Evolutionary Fabrication can then treat \mathcal{L}^* as the genotypic search space and \mathcal{S} as the phenotypic space. Implicit in the description of \mathcal{M} are the aspects of the larger environment which may affect assembly, such as friction and stochastic noise. In this model, the only constraints placed on assembly are those that due specifically to the particular nature of the assembly mechanism \mathcal{M} . This differs from top-down approaches like Assembly Sequencing, which by contrast limit the size of \mathcal{L}^* by imposing constraints which are not intrinsic to \mathcal{M} , such as monotonicity and reversibility. Having fewer constraints on assembly methods corresponds to a larger space of assembly procedures to search, but also corresponds to more *possibilities* of assembly. The extent to which Evolutionary Fabrication is able to arrive at novel, or qualitatively different solutions to assembly, is therefore based upon the extent to which the constraints imposed by top-down approaches are disjoint from the actual constraints of the assembly mechanism. By coupling search to a *specific* language of assembly for a *specific* assembly mechanism, we also lose the generality which arrives from the assumptions of top-down methods. On the other hand, it is this very specificity which allows for the immediate automation of any assembly plan we find, without any subsequent need for fine tuning.

Chapter 3 will introduce a framework composed of an assembly mechanism, a language of assembly, and an evolutionary algorithm which will enable us to explore this process of Evolutionary Fabrication.

2.5.2 Alternatives to Evolutionary Fabrication

There are of course some alternatives to Evolutionary Fabrication. A considerable amount of research is being poured into nano- and meso-scale self-assembly, with promising results [103]. This level of assembly however is limited by size and energy: proteins can self-assemble, but automobiles cannot. Modular, reconfiguring, and self-reproducing robotics systems such as Rus's Crystalline Robots [89], Yim's snake-like robots [111], Chirikjian's Metamorphic Robots [70] and Lipson's self-reproducing structures [113] offer the possibility of reconfigurable morphologies, but are confined to relatively complex modular units which have been meticulously designed and assembled by hand.

In many senses, Evolutionary Fabrication rests in between these two scales, capable of creating objects on one hand much larger than those created by nano and micro-scale assembly, and on the other hand producing large complex objects without the need for hand-designed modular units.

2.6 Engineering without Engineers

While this dissertation is most firmly rooted in the realms of Artificial Life and Evolutionary Design, care must be taken to properly situate Evolutionary Fabrication in the context of modern engineering methods. We must emphasize, for instance, that we are not trying to address any particular weakness in top-down methods of automatically manufacturing a specific, single, solid 3-D shape. There are several well tested and heavily used algorithms for generating tool paths from CAD models for both additive and subtractive manufacturing processes. Rapid prototyping machines, for instance, use an $O(n \log n)$ algorithm to determine the looping path of the print head as it follows the contours of a horizontal slice through the object [47, 65]. Like

Assembly Sequencing, however, these methods are predicated upon having the design of a *buildable* product in the first place.

Where does the buildable design come from? As Wilson [105] points out, there are two human minds involved in the traditional task of product creation: a designer who specifies the shape and properties of the *finished* product, and a manufacturing engineer, who tries to find a way to manufacture the design. When a product is deemed too difficult to manufacture, the manufacturing engineer sends the product back to the designer for a re-design. This decoupling of design and manufacture allows each engineer to specialize in his particular field, but, particularly as products and their associated manufacture become increasingly complex, the process requires multiple passes through the design-manufacture loop, which adds to the ultimate cost of a finished product.

Stemming from the broader engineering discipline of Design-for-Assembly (DFA), modern approaches seek to reduce this human effort by integrating product design with assembly planning [105, 46, 43, 44, 112]. Kim, for instance, uses a model which automatically generates possible assembly sequences for CAD models as they are iteratively designed and modified by a user [46]. Each of these methods still has a human in the loop, responsible for the actual *design* of the object. Nor are they guaranteed to always produce *feasible* assembly procedures, and so often further heuristics and expert-level knowledge-based systems are required in order to determine unrealistic and impossible assembly operations [1]. All of these tools, in fact, are meant to ease communication between the design engineer and the manufacturing engineer, not supplant either of them.

In this dissertation we are not seeking to use Evolutionary Fabrication to replicate, recapitulate, or compete with these proven human-based engineering methods and results, but rather seeking to explore how “mindless” systems, *absent* human knowledge,

can arrive and complex buildable objects. After all, there is no “Intelligent Factory Foreman” supervising biological development [74]. How do natural systems, given access to the extremely complex assembly mechanism of biological growth, but with no “knowledge” *per se* of its limits and capabilities, learn to build complex objects? How, furthermore, can they modify the very nature of their assembly process, in order to build increasingly large and increasingly complex objects? Broadly speaking, Evolutionary Fabrication hopes to reproduce, to some degree, this “engineering without engineers” performed by biological life. We make no claims that results produced via Evolutionary Fabrication are in any sense *better* than knowledge-based engineering results, only that they are qualitatively *different*, and this difference arises from the relative absence of human knowledge in the system.

2.7 Summary

Evolutionary Algorithms are capable of generating novel and exciting designs for objects. These designs are of little utility, however, until they can be manufactured in the real world. Present approaches to Evolutionary Design, which rely on the evolution of descriptive blueprints, are stymied by the Fabrication Gap.

Artificial Ontogenies can be used to represent the *formation* of an object rather than just its *form*, and have already been proven to be quite useful for design tasks. In order to be useful for our purposes of fully automated manufacture, however, they must clearly and explicitly describe and simulate the entire process of assembly.

Evolutionary Fabrication addresses this by placing the evolutionary algorithm *within* a rapid prototyping machine, and by directly evolving the instructions for that machine. In the following chapter we lay out the Framework for a system capable of doing exactly this.

As will be demonstrated by subsequent chapters, not only does this co-evolution of form and formation avoid the Fabrication Gap by producing specific assembly plans for evolved objects, but it is capable of the same *creativity* that has been demonstrated by other Evolutionary Design systems – discovering not just novel objects, but novel means of assembling those objects as well.

Chapter 3

Framework

In the previous chapter we introduced the Fabrication Gap caused by the evolution of purely descriptive *blueprints*, which leave unanswered the question of *how* to build the evolved object. Evolutionary Design systems based upon blueprints are therefore capable of designing objects whose assembly is extremely difficult to discover, and, in the extreme, *unbuildable* objects. The solution to the Fabrication Gap lies in *Evolutionary Fabrication*: co-evolving form with formation by simultaneously evolving objects and the processes which create those objects.

We present in this chapter an elementary framework for implementing Evolutionary Fabrication. By modeling an assembly mechanism, and directly evolving instructions *within* that mechanism, we can evolve *how* to build rather than *what* to build. We will then use this framework to show that Evolutionary Fabrication is capable of generating not just novel objects but *novel ways of assembling those objects*.

3.1 System Description

The purpose of the framework described here is to create a unifying experimental structure with which to explore the capabilities and possibilities of Evolutionary Fabrication. As we discussed in Section 2.5.1, Evolutionary Fabrication is contingent on a *specific* assembly mechanism and a *specific* language of assembly. This specificity is what distinguishes Evolutionary Fabrication from more abstract top-down approaches, and what allows for the immediate *automation* of the results we produce.

To that end, the framework described here is a model of an assembly mechanism grounded in a realistic physical context, and evolution unfolds within it. Instead of the blueprints used by traditional Evolutionary Design, the genotypes of our system are *assembly plans*: linear sets of instructions to the assembly mechanism. As the assembly plan is interpreted, the structure grows and, as such, this process of assembly is an *Artificial Ontogeny*.

One important aspect of the system described below is its *ballistic* nature. That is, the machine described is incapable of measuring the intermediate results of its actions, it can only measure the final result. This simplifies the model significantly (and as we discuss below, measurement is a double-edged sword.) While we make no claim that this lack of measurement is in any sense *necessary*, we will show that such ballistic Evolutionary Fabrication is capable of robust and reliable behavior beyond what might be expected, even in the presence of noise and error during assembly.

We are also using a very simple approach by performing evolution directly on these linear assembly plans. Certainly, more complex and indirect means of generating these assembly plans exist, such as the grammatical approaches used by Hornby [38] and Toussaint [96]. Doing so, however, might obscure the nature or origin of any results. While we make no claims that this direct evolution is the optimal way to evolve

assembly plans, it is certainly the best to illustrate our process. As we mentioned in Chapter 2 we are interested in the *representation of development*, not the development of representation.

While small changes are made to the framework for different experiments in later chapters, we describe here the basic components.

3.1.1 Design

Although many contemporary rapid prototyping machines work via material deposition, extruded plastic behaves like a coiling rope, which can be difficult to simulate. Instead we rely upon a simpler pick-and-place model which deals with discrete brick elements. None of the techniques used, nor the results which arise, however, are in any sense unique to this model. We will argue that similar, though of course not identical, phenomena should be seen in a real system.

The physics of our framework is based upon the Open Dynamics Engine (ODE)¹ the widely used open-source physics engine, which provides high-performance simulations of 3D rigid body dynamics. Doing so provides us with suitably complex dynamics, and the ability to model gravity, collisions and friction.

3.1.2 Assembly Plans as Genotypes

The print-head of the system behaves like a LOGO turtle, capable of movement in the X-Z plane, and of depositing 2x1x1 bricks in the environment. Turtle-based systems have been used in a variety of Evolutionary Design tasks [38, 96, 41].

When strung into a sequence, commands to the turtle (see Table 3.1) form an *assembly plan*, as shown in Table 3.2. Commands which would cause the turtle to

¹www.ode.org

move outside the target area, or place a brick where a brick already exists, are ignored.

Table 3.1: Parameterized Assembly Instructions

Instruction	Parameters
(M)ove	+2, +1, -1, -2
(R)otate	+90, -90, +180
(P)ut Brick	(a)head, to (r)ight, to (left), (b)ehind
Put (S)caffolding	(a)head, to (r)ight, to (left), (b)ehind
(T)ake Brick	(none)

Table 3.2: An Example Assembly Plan

$P(a)P(a)R(-90)M(+2)M(+2)R(+90)S(a)S(a)R(180)M(+2)M(+2)R(-90)P(a)P(a)$
--

You will note the absence of any means of measuring the intermediate results of assembly. As we discussed above, we are interested in pushing the limits of ballistic assembly before exploring more “informed” methods. There are compelling reasons for this simplicity. To begin with, adding to the set of primitives increases the search space of the algorithm considerably. Moreover, measurement is, by nature, costly in terms of time and resources. Additionally, most rapid prototyping machines are only capable of detecting the most urgent of errors, such as a block in the print head - none of the current models, for instance, carry digital cameras which measure progress, allowing them to abort a job if something unexpected happens. Indeed, how would they know whether or not some phenomenon is normal or not? In later chapters (specifically section 4.3) we will demonstrate means of assembly which definitely fall outside of the norm of behavior for rapid prototyping machines, and yet are highly desirable.

3.1.3 Material

The turtle is capable of placing two kinds of bricks: permanent ones (shown as blue in color frames, or black in grey scale frames), and temporary ones (shown in gray),

which are removed once the assembly is completed. This aspect is based on a feature of modern rapid prototyping machines, such as a current model made by Stratasys², which can lay thin water-soluble support structures. In Chapter 5 we will show how Evolutionary Fabrication can discover scaffolding when it is not provided as a component of the system, by placing and removing intermediate structural elements.

3.1.4 Assembly as Ontogeny

As the print head reads instructions from the evolved assembly plan and deposits material, a structure grows. This process of assembly is, in the truest sense, an artificial ontogeny.

The interpretation of evolved assembly plans falls into three ontogenic stages, as shown in Figure 3.1. In the first, the turtle interprets the assembly plan, moving and placing bricks as directed. In this stage, each brick is a separate component in the environment, subject to gravity and interactions (such as collisions) with other objects. Once assembly is complete and the structure is stable, the scaffolding is removed and adjacent bricks are glued together (but not to the floor). Finally, once the scaffolding is gone, the final structure is allowed to come to a rest before being evaluated.

3.1.5 Algorithmic Details

Our framework uses a pareto-based multi-objective optimization algorithm [27, 19]. This provides some flexibility in fitness functions, by allowing us to use different criteria (such as length and mass) as different objectives. Using length as a separate objective is particularly useful in variable-length representations such as ours, by

²www.stratasys.com

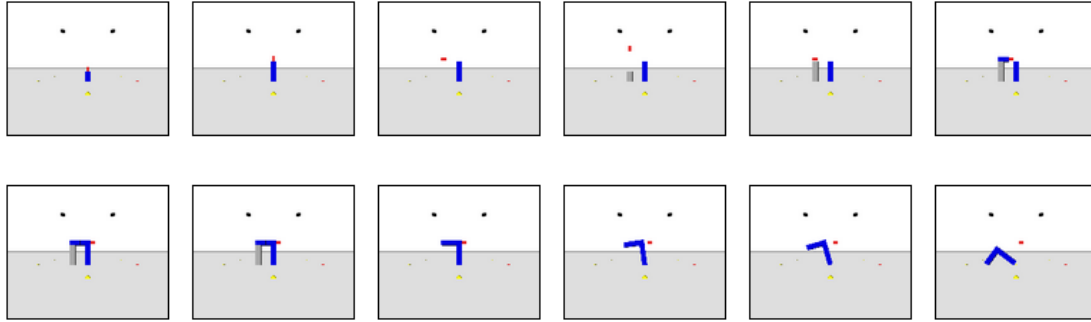


Figure 3.1: The execution of the assembly plan shown in Table 3.2. Frames are ordered left to right, top to bottom. Assembly has three stages. In the first (Frames 1-8), both permanent bricks (shown in blue) and temporary bricks (grey) are placed. In the second, adjacent permanent bricks are glued together and scaffolding is removed (Frame 9). Finally, the remaining structure settles (Frames 10-12).

helping to reduce bloat [22]. After each generation is evaluated, the N non-dominated individuals (i.e. pareto front) are selected as parents, and N new individuals generated using two-point crossover (60%) and mutation (2% per locus). In order to limit population sizes, duplicate genotypes were rejected, and duplicate objective values were limited using crowding [62], with a limit of 3 individuals per bin. Variations from this are noted in each experiment. In most experiments, the initial population was created with 30 random genotypes, each with a random length between 8 and 40 instructions.

It is worth emphasizing, however, that none of the claims of this work are at all contingent upon the details of any specific type of algorithm. The only important aspect is that by using assembly plans as genotypes, we are evolving *how to build*.

3.2 Brief Example: Evolving a Goal Structure

While the full power of Evolutionary Design lies in open ended creation of form, there is often the need to determine whether a pre-determined structure is at all

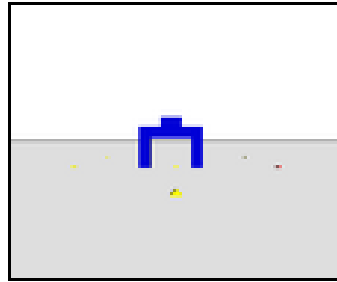


Figure 3.2: The Goal Arch. Each leg consists of two vertical bricks, whereas the center section consists of three horizontal bricks. Note, therefore, that the center bricks are not resting on top of the legs, but are instead cantilevered off their side - as a consequence, until the glue phase they cannot remain in place without scaffolding.

buildable. In this context, the goal is to find a suitably efficient assembly plan which, when interpreted by the automated assembly process, results in the goal structure. This is, in a sense, automating the previously human task of inferring a *descriptive* representation's *prescriptive* counterpart, or reverse-engineering an object's assembly.

We can begin, therefore, by evolving assembly plans capable of building a pre-determined goal structure, in this case an arch (Figure 3.2), largely because of the anticipated level of difficulty, and also for historical reasons - namely its presence as an example of a hard problem in Winston's seminal textbook on Machine Learning [106].

In order to compare each resulting structure to the goal structure, a bitmap was generated by sampling the central region in the X-Z plane, at a sub-brick resolution. This bitmap was then compared to a corresponding bitmap of the goal structure.

The specific objectives used were as follows. In each case smaller values are considered more fit. Similar objectives are used in most of the experiments in this dissertation, and so we will elaborate on them here.

- **Length:** (Total length of the assembly plan) All things being equal, the shorter an assembly plan is, the better. Since real-world rapid prototyping machines can be slow, the more efficient we are in this regard, the better.
- **Mass:** (Number of bricks in the entire world, not just the sample region.) The material deposited by rapid prototyping machines can be rather expensive. Parsimony in conserving the total amount of material should be rewarded.
- **Missing Material:** the total number of bricks missing from goal structure. A value of zero corresponds to having all of the necessary bricks in place.
- **Error:** the total number of “wrong” bricks. A value of zero corresponds to precisely replicating the goal structure without either missing or extraneous material.

3.2.1 Results

Figure 3.3 shows animation frames from a representative evolved solution. Discovered after roughly 2000 generations and with a length of 22 instructions, it is able to perfectly generate the goal structure.

This efficiency is due largely to the novel placement of the vertical scaffolding used to hold up the center section of the arch. Each vertical scaffolding brick is placed directly under the center of mass of the brick it supports. This placement location exists between two of the discrete print-head positions, and so could not have been placed directly. Rather, it is dropped horizontally onto the leg sections and subsequently topples vertically into its final location. In fact, if it had been placed directly into one of the adjacent positions, it would not have been under the supported brick’s center of mass, and the supported brick might have tilted sideways.

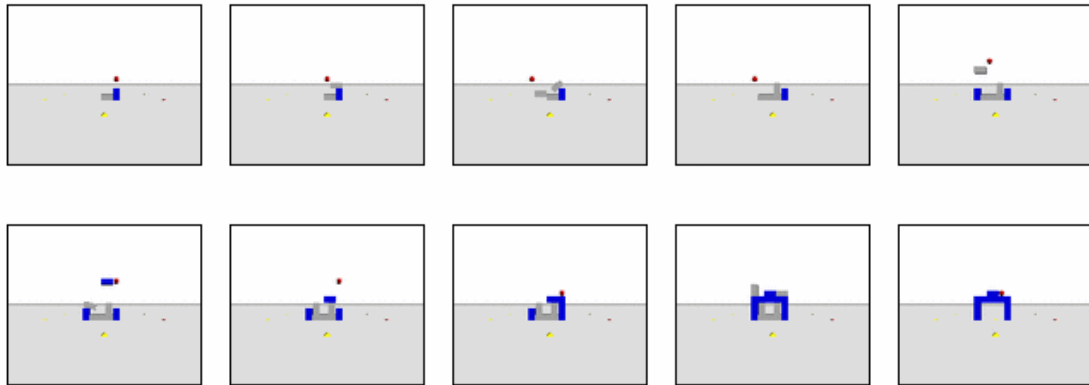


Figure 3.3: Building the Goal Arch. Note how the horizontal scaffolding placed in frame 3 tumbles into a vertical position to support the top of the arch. This is repeated with the piece of scaffolding placed in frame 5.

This novelty in assembly is driven largely by the length objective used in our algorithm above. Since, all other things being equal, a shorter assembly plan is more fit than a longer assembly plan, there is an evolutionary incentive for a certain succinctness. The mass criterion also plays a role, encouraging parsimony in resources used.

3.3 Summary

We have laid out here the framework for an Evolutionary Fabrication system. This system is essentially an evolutionary algorithm the genotypes of which are assembly procedures for an assembly mechanism. Hence the marriage of automated design and automated assembly. Armed with this framework, we can now illustrate how Evolutionary Fabrication is capable of evolving not just novel objects, but novel means of assembling those objects as well.

We have shown intimations of this in the simple example in Section 3.2, in which we evolved an assembly plan to build a pre-determined goal structure. The evolved

assembly plan exploits the dynamics of the assembly mechanism in order to produce the goal structure in an efficient manner. This penchant for efficiency in both time and material could be well employed in modern rapid prototyping machines, which tend to produce a surplus of expensive supporting material in order to print objects.

This novelty and efficiency, which we will see much more of in the following sections, is where the true benefit of Evolutionary Fabrication lies.

Chapter 4

Evolutionary Fabrication for Design

In the previous chapter we presented our model of Evolutionary Fabrication, and demonstrated its ability to discover the assembly plan for a goal structure. While evolving assembly plans for pre-determined structures is important, the true promise of Evolutionary Fabrication lies in creating interesting objects given only a broad specification. The antennae evolved by Lohn *et al.* [59] for instance, were evolved to achieve high gain at specific bandwidth while fitting into a relatively small bounding box, and Sims' creatures were rated on the horizontal distance they traveled [92, 91].

Although the fitness functions we use below are simple, they serve the purpose of demonstrating that Evolutionary Fabrication is capable of generating novel, unique and human-competitive results as it evolves the *process* of assembly.

4.1 Filled Volume Fitness

Consider first a simple “filled” fitness function, which measures the amount of filled volume within the bounding box - that is, space occupied either by bricks or covered from above by bricks. The union of the grey and black areas in Figure 4.1 demonstrates this concept.



Figure 4.1: Illustration of the “filled” and “shade” fitness function. The structure itself is black. The filled fitness measures union of both back and grey regions, the shade fitness measures only the grey area.

Just as in Section 3.2, in order to measure the fitness of a completed structure, a bitmap was generated by sampling the central region in the X-Z plane, at a sub-brick resolution. The maximum height of each column of the bitmap can then be summed to measure the coverage.

The length and mass objectives are retained from the experiment in Section 3.2, and the fitness function above replaces the two goal-based objectives from 3.2.

- **Fitness:** (maximizing) the amount of shaded area including structure.
- **Length:** (minimizing) the shorter the assembly plan the better.
- **Mass:** (minimizing) the less material required, the better.

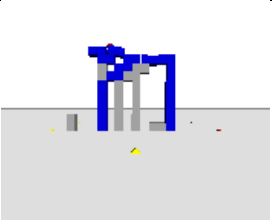
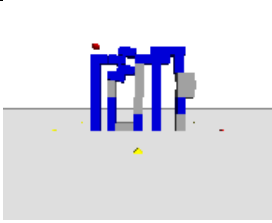
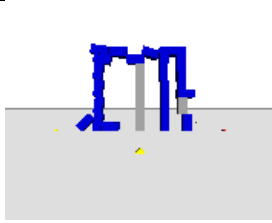
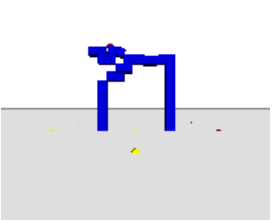
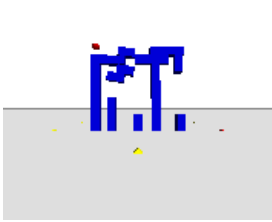
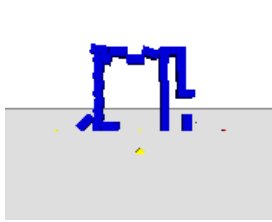
Here, since we have no goal structure in mind, the roles of the *length* and *mass* criteria become much more significant. As the saying goes, time is money, and the faster an object can be assembled the faster it can be used for its desired function. Moreover, the materials used by rapid prototyping machines are notoriously expen-

sive, and anything which can reduce the amount of material used during assembly is welcome.

4.1.1 Results

Table 4.1 shows representative structures and fitness values generated by the “filled” fitness function, both before and after scaffolding has been removed. Evolved structures fill the target area well, but tend to have a large number of extraneous structural appendages (consider the “foot” on the right hand side of the bottom-most figure). Since the structures are rewarded for the amount of the target area that they fill, regardless of whether the space is occupied or not, there is little incentive to remove extraneous bricks or to use temporary scaffolding bricks.

Table 4.1: Structures generated with the “filled” fitness function. The images in the top row show the structures before the scaffolding (grey bricks) has been removed, and the images below right show the final stable structure with the scaffolding removed.

Scaffolding			
Final			
Fill	89%	93%	95

4.2 Shaded Volume Fitness

Consider next a “shaded area” fitness function, which measures the total amount of open volume beneath a structure, as illustrated by Figure 4.1.

The motivation in using this more space-conscious fitness function lies in exploring the system’s ability to generate objects which conserve building material. This fitness function also bears some similarity to those used for Funes LEGO trees [29] and Toussaint’s plants [96]. The specific objectives used were identical to the section above, save for this fitness function which measures “shaded” volume as opposed to “filled” volume.

4.2.1 Results

Table 4.2 contains representative structures built by assembly plans which resulted from several runs of the system. It shows the structures before and after the glue/melt phase, and lists their fill percentage as well as the length of the assembly plan that produced them. As is evident, all of the evolved structures have near-optimal fitness. For comparison, our best hand-built structure, shown on the far right hand side, while it has maximal fitness, required 34 bricks, more than the maximum 25 allowed for evolved assembly plans, and was 65 instructions long.

Figure 4.2 shows the assembly of the arch in the fourth column of Table 4.2. The assembly of the arch relies on a single central column of scaffolding, as opposed to the multiple columns required in the other arches shown. This single central column is largely effective because it widens to two brick widths along the top, and is therefore able to support two bricks above it. The lower leftwards “spur” on the central scaffolding column plays two important roles: first, by being horizontal it allows the column to be nine bricks high, leaving room for a tenth row of permanent bricks

at the maximum allowable height (the two upper inwards spurs on the legs of the structure serve a similar function.) Secondly, it counterweights the upper horizontal scaffolding bricks, which allow the column to support two permanent bricks.

This process of assembly above is another example of Evolutionary Fabrication’s ability to arrive at short and efficient assembly plans.

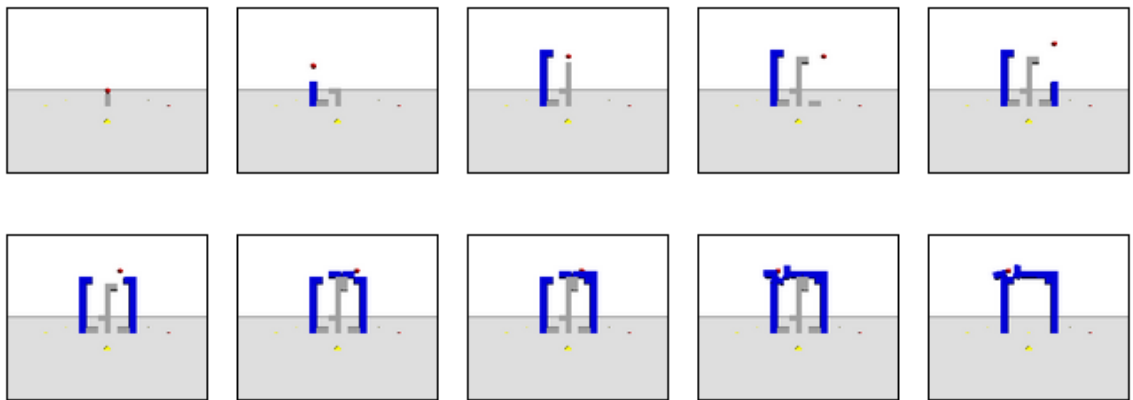
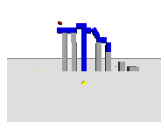
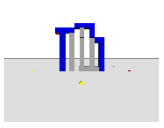
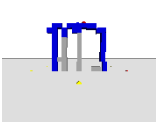
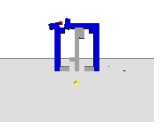
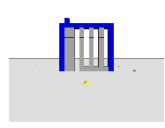
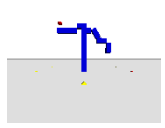
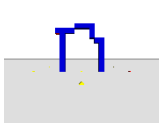
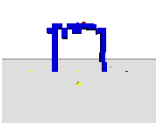
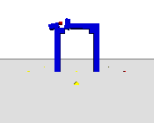
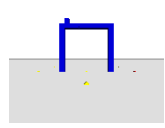


Figure 4.2: Assembly of the arch in the fourth column of Table 4.2

Table 4.2: Structures Evolved for “shadow” fitness. The top-row images show the structure before scaffolding (grey) is removed, and the bottom images show the final, stable structure. The structure on the far right was hand-built.

Scaffolding					
Final					
Fitness	84%	80%	95%	90%	(hand-built)
Length	30	54	58	43	65

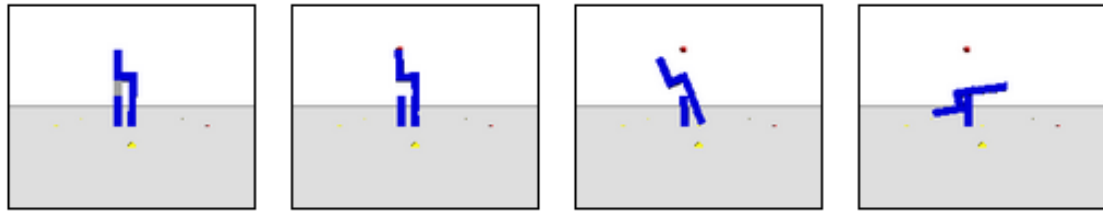


Figure 4.3: A novel assembly method. Once the assembly is complete (Frame 1), scaffolding is removed and remaining bricks glued together (Frame 2), the larger section topples onto the smaller section, balancing there to form a T. This resulting shape has significantly higher fitness (49%) than the original structure (10%)(Frame 1)

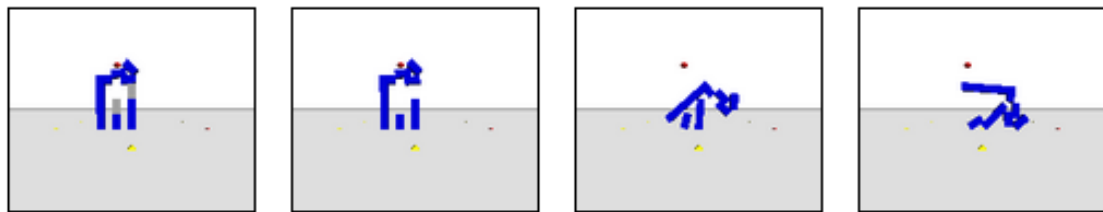


Figure 4.4: Another novel assembly process. The original structure has a fitness of only 22%. Once scaffolding is removed and remaining bricks glued, the leftmost portion tumbles rightward, and the smaller segments below are knocked sideways, ultimately serving to prop up the larger shape. This final structure has a fitness of 52%)

4.3 The Emergence of Novel Assembly Methods

In the setup above, evolved structures will occasionally be unstable once scaffolding is removed, causing the structure to tumble into a final, structurally distinct shape, often with higher fitness. Figures 4.3 and 4.4 provide an example of this phenomenon.

This phenomenon of “dynamic assembly”, is an interesting exploitation of the system as we designed it, and is a preliminary example of the novel types of assembly processes that can arise from evolving assembly plans directly in a realistic environment. Of course, this raises the question of whether there is any particular advantage

to such novel assembly, or whether it is a mere curiosity. We can seek to address this question by exploring the following issues: What are the evolutionary incentives of dynamic assembly? Are solutions which use dynamic assembly any more *fit* than those without? Are they any more efficient, either in terms of time or in terms of material?

4.3.1 Measuring Novel Assembly

This novel dynamic assembly of structure can be measured by calculating a structure's fitness immediately after scaffolding is removed, and comparing this value to the structure's final fitness once stabilized. This difference corresponds to the amount of fitness contributed by dynamic assembly.

$$f_{dynamic} = f_{final} - f_{initial} \quad (4.1)$$

Similarly, as a measurement of efficiency in terms of time and material, we can calculate the fitness-per-instruction and fitness-per-brick of each solution by dividing each solution's fitness by its assembly plan length and mass.

Armed with this means of measuring dynamic assembly and efficiency, we can compare two slightly different environments. The first environment, which we call Setup A, is the original environment from our earlier experiments. In the second, we made a small change: Rather than limiting the turtle to the same box that fitness was evaluated over, it was allowed to range over a larger, 200×200 box (labeled B Figure 4.5). Otherwise, all of the objectives, and the 25 brick limit, remain unchanged. This slight adjustment allows the turtle to place bricks outside of the fitness box - which then fall into it during the final settle phase of development. Table 4.3 summarizes the differences between environments.

Table 4.3: Comparison of Setup A and Setup B

	Setup A	Setup B
# runs	22	19
Turtle Range	100×100	200×200
Fitness Range	100×100	100×100

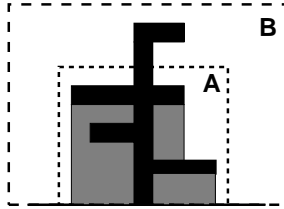


Figure 4.5: Illustration of the two Assembly Environments. In each case the fitness function is measured over the smaller box (A) and within that box the gray regions under the black structure is considered “shaded”. Note that the uppermost overhang does not contribute any shade, because it exists outside of the fitness bounds. In the first environment, the turtle is limited to the same box (A) as the fitness measure, whereas in the second (Section 4.3.1), the turtle may range in the larger box (B).

4.3.2 Results

Looking back at Table 4.2, which contains representative results from our earlier experiments, it can be seen that the evolved structures tend towards stable arch-like structures with two legs.

By comparison, Table 4.4 shows representative structures evolved in Setup B (black spheres have been placed in the upper corners of the box over which fitness is measured). Unlike Setup A, which produced relatively stable arch-like structures, this small change results in a majority of “T”-like structures, all of which are assembled dynamically. Figures 4.6 and 4.8 show detailed frames of the process of dynamic assembly for each structure in Table 4.4

Figure 4.9 provides a direct comparison of the contribution from dynamic assembly across the two regimes. Data were generated by averaging the values of the best individual in each generation across runs. Figure 4.10 provides more detail of the

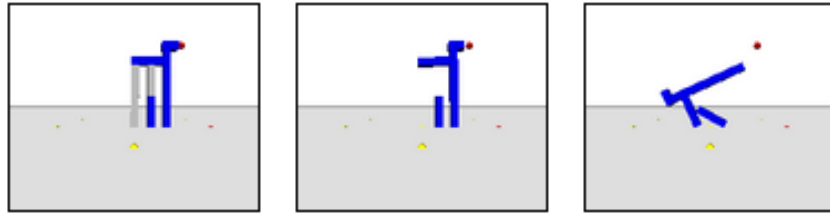


Figure 4.6: Another dynamic assembly sequence from Setup B. This is 22 instructions long, with a final mass of 10. Initial fitness is 31%, final fitness is 47%.

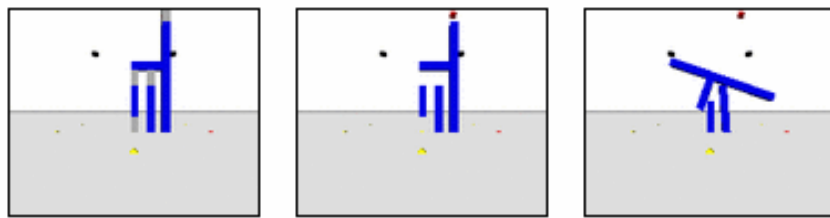


Figure 4.7: Another dynamic assembly sequence from Setup B. 20 instructions, 14 mass. Initial fitness: 27%, Final Fitness: 62%. The black spheres mark the upper corners of the fitness bounds.

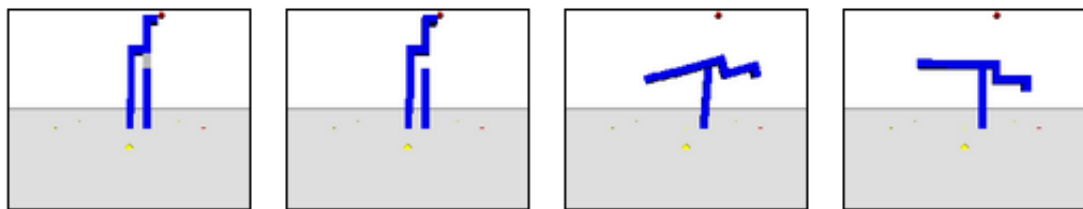


Figure 4.8: The most extreme example of dynamic assembly from Setup B. With only 17 instructions, and a mass of 13. Initial fitness is only 0.4%. Final fitness is 80%. The black spheres mark the upper corners of the fitness bounds. The near-zero initial fitness is due to the fact that the overhanging brick is outside of the fitness bounds, and therefore contributing no shade.

progress of evolution across both environments. It is worth observing that both environments produce equally fit results, even though the larger environment corresponds to a larger search space. Figure 4.11 shows the fitness contribution due to dynamic assembly between the two environments, as measured by Equation 4.1. It is interesting to note that in the larger environment sometimes dynamic assembly accounts for almost the entirety of an structure's fitness (such as, for instance, the last structure in Figure 4.8) whereas in the smaller environment, a structure's fitness will occasionally *decrease* after settling.

Figures 4.12 and 4.13 respectively compare the average value of the fitness per brick and fitness per assembly plan instruction for the best individual of each generation. Although the difference is slight, the fitness per instruction is higher in the larger environment than it is in the smaller environment. In Figure 4.12, the slightly lower fitness per brick of of the larger suggests that some material bloat occurs in the slightly larger environment. This slight bloat in material could be due to the fact that extraneous bricks outside of the fitness range (Box A in Figure 4.1) do not have a deleterious effect upon fitness the way that extra bricks within the box do. Still, these bricks may serve a useful function, for instance as a counterbalance during dynamic assembly.

These differences in mass and length are more significant when the size of the larger environment is taken into account. Larger margins outside the fitness box give the turtle a wider area to roam in, and more room in which to place bricks. This corresponds to a significantly larger search space of assembly plans. Yet while equally fit assembly plans involve a few more bricks in this environment, they are in fact *shorter* than those evolved in the smaller environment. This leads us to believe that Evolutionary Fabrication exploits dynamical assembly in order to produce assembly plans that are more efficient in *time*, at the expense of efficiency in *material*.

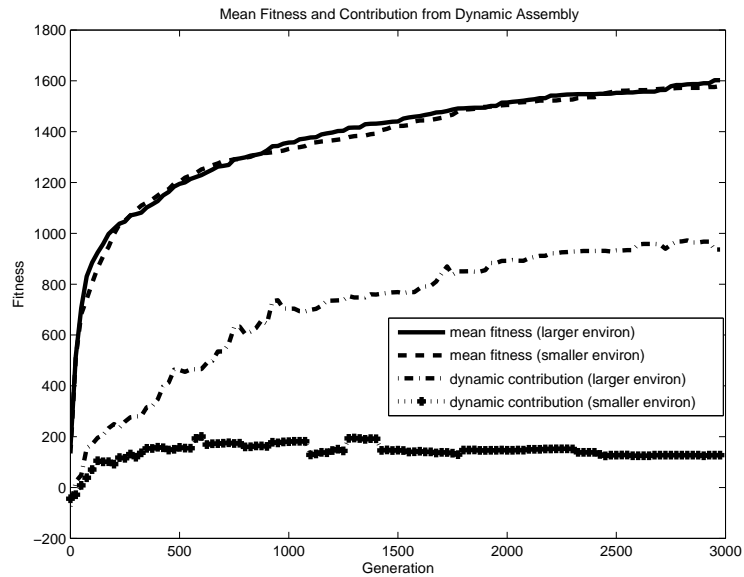


Figure 4.9: Comparison of fitnesses and fitness contribution from dynamic assembly between the larger and smaller environment. Data is averaged across 22 runs for the smaller, and 19 runs for the larger. Although maximal fitness is equivalent across both regimes, Setup B contains significantly more examples of dynamic assembly

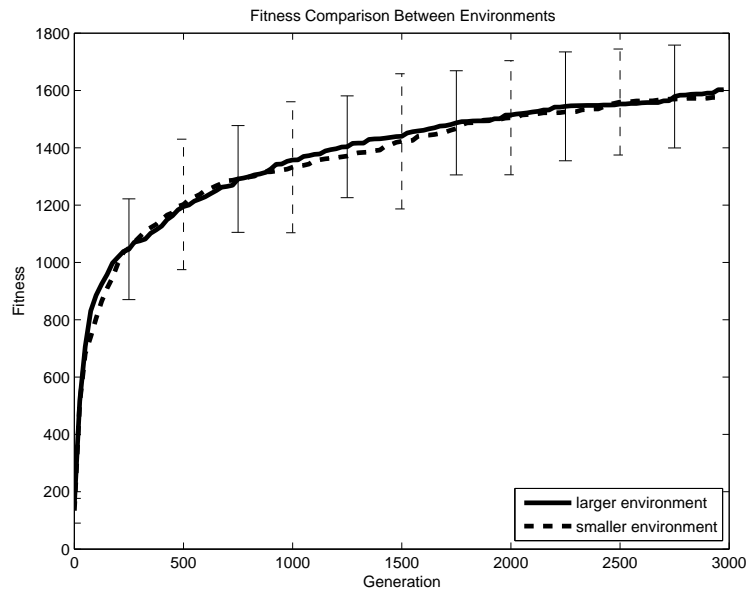


Figure 4.10: A comparison of the progress of evolution between the two environments, with error bars. Despite having a larger search space, progress in the larger environment is essentially identical to that in the smaller environment.

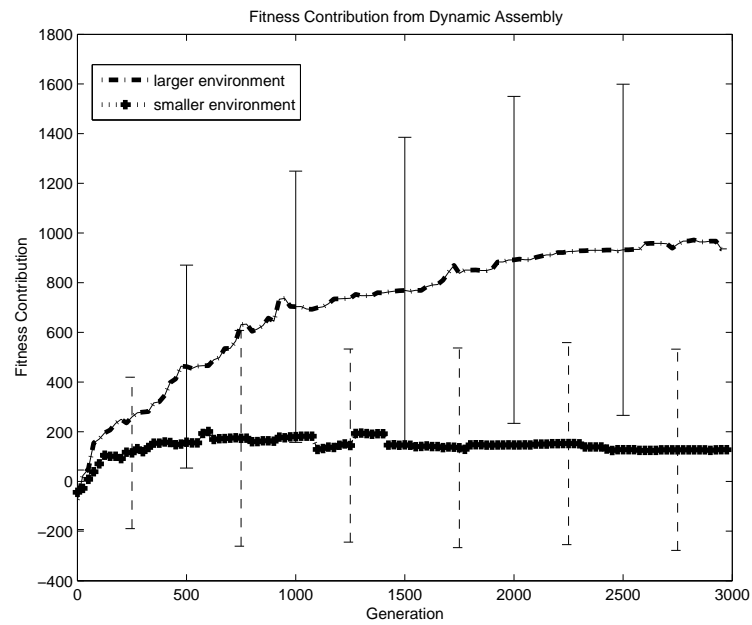


Figure 4.11: A comparison of the fitness contribution due to dynamic assembly, with error bars. There is a much higher incidence of dynamic assembly in the larger environment. In the smaller environment solutions are sometimes actually *worse* after they settle.

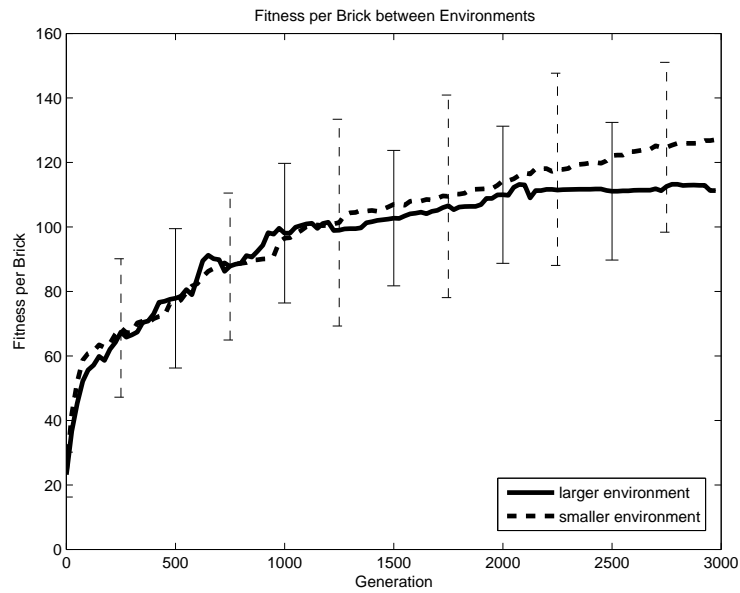


Figure 4.12: Average Fitness Per Brick between environments. The difference is slight, but equally fit structures tend to contain fewer bricks in the smaller environment.

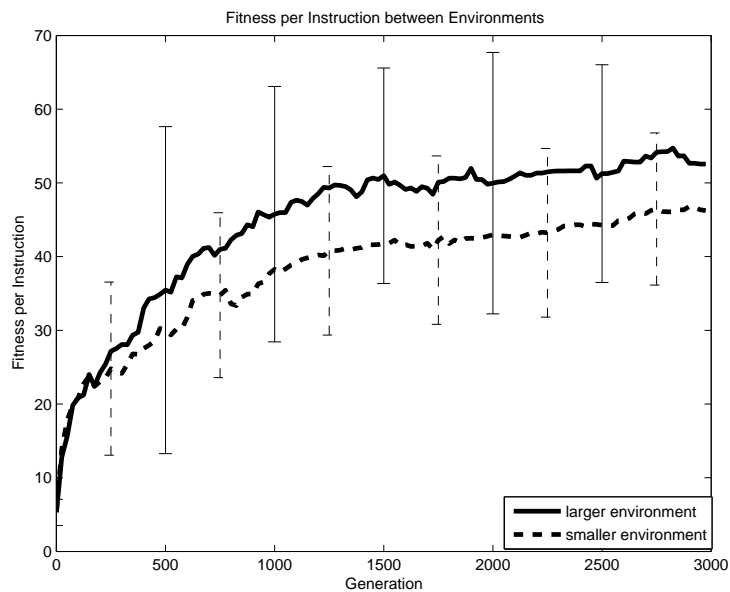
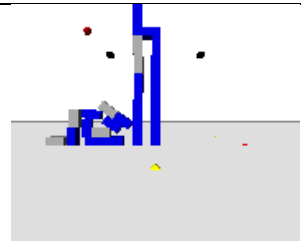
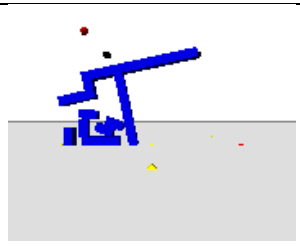
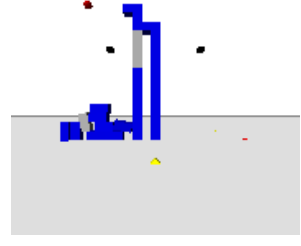
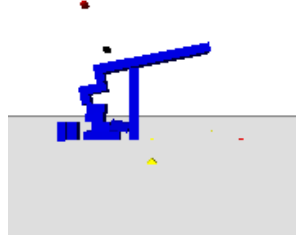
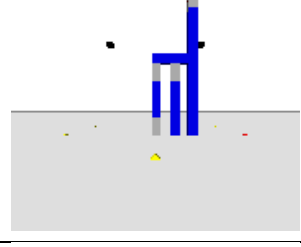
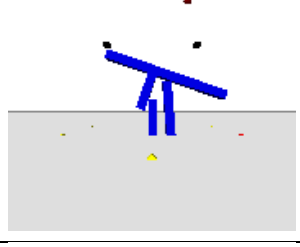
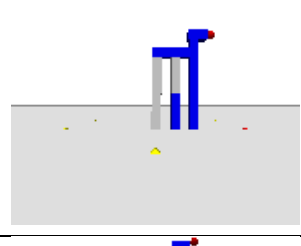
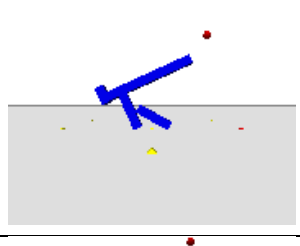
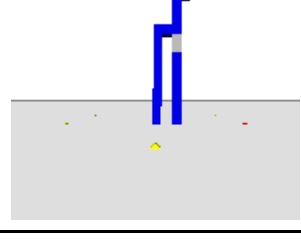
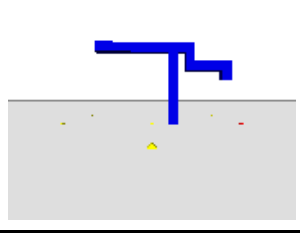


Figure 4.13: Average Fitness Per Assembly Plan Instruction between environments. Equally fit structures tend to be slightly shorter in the larger environment, despite the wider range available to the turtle.

Table 4.4: Structures Evolved in Setup B before (left) and after (right) the grey scaffolding is removed. Black spheres have been placed in the upper corners of the box over which fitness is evaluated. The larger sphere is the location of the turtle

	Initial		Final
5%		84%	
2%		81%	
27%		62%	
31%		47%	
0.4%		80%	

4.4 Summary

In this chapter we have demonstrated the ability of Evolutionary Fabrication to perform open-ended design tasks. In Section 4.1 the goal was to fill a bounding region, and in Section 4.2 the goal was to maximize the covered volume of the bounding region. In each case, Evolutionary Fabrication is able to arrive at near-optimal, and human-competitive solutions. While these fitness functions were both rather simple, they serve the purpose of demonstrating that Evolutionary Fabrication can discover how to build structures given only relatively loose specifications.

Moreover, as shown in Section 4.3, an added benefit of this evolution of assembly plans arises from Evolutionary Fabrication’s ability to design not just novel structures, but novel means of assembling those structures as well. The most obvious of these novel assembly methods is what we have termed “dynamic assembly”. Dynamic Assembly describes an evolved assembly process which builds multiple unstable sub-assemblies which, once scaffolding is removed, topple into a final shape.

The novelty of dynamic assembly extends beyond just the irreversibility of the process. Our framework above is limited to placing bricks one at a time, and so lacks any formal ability for modular assembly of larger components. Nonetheless, the assembly plans evolved above have discovered how to construct two separate modules, and then join them in the final phase of assembly. We’ve shown that that in each case the process used in creating the final structure is more efficient than a purely sequential process.

In the following section we will explicitly add modularity to our framework, and demonstrate the ability of Evolutionary Fabrication to evolve increasingly large objects, even in the presence of noise and error during assembly.

Chapter 5

Evolutionary Fabrication in Uncertain Environments

“The organism is determined neither by its genes nor by its environment nor even by the interaction between them, but bears a significant mark of random processes...” Lewontin, *The Triple Helix*, (p.38)

“Unlike the adult organism or larva, the embryo seems to be rather privileged...Its primary function is to develop reliably and this reliability is the main feature on which selection will act.” Wolpert, *The evolutionary origin of development: cycles, patterning, privilege, and continuity* [108].

In the previous chapter we established that Evolutionary Fabrication is able to generate solutions to open-ended design tasks and that by evolving assembly plans directly it is able to discover novel means of assembly. The next important questions to ask are whether Evolutionary Fabrication can cope with noise and error during assembly, and whether the process can scale to increasingly larger and more complex domains.

As we discussed in Section 2.4.4, developmental representations are not without their drawbacks. In particular, stochastic effects which lead to error and noise during development can significantly complicate the task of evolution. When subjected to noise during development, a genotype is capable of developing into an entire range of phenotypes, each with a corresponding fitness (Figure 5.1).

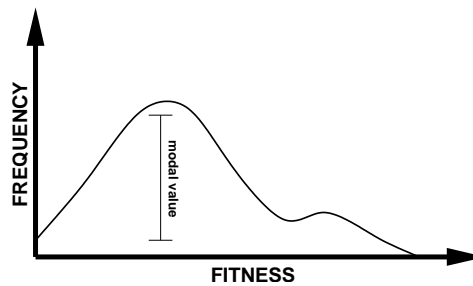


Figure 5.1: Under the presence of developmental noise, each genotype develops into an entire **range** of phenotypes, each with an associated fitness.

This has particular relevance to Evolutionary Fabrication in the real world. Unlike simulation, real world assembly environments are subject to the effects of friction and random perturbation, which can have a dramatic effect on results. Considering these effects, how can Evolutionary Fabrication learn to reliably build specified objects in the presence of noise and error during assembly? How can increasingly large, increasingly complex objects be built in these environments? This chapter seeks to address those questions.

5.1 Evolving Reliability without Tests

It is important to note that the Framework described in Chapter 3 has no means of measuring the intermediate results of assembly: any error during assembly cannot be noticed until assembly is complete and the final product can be observed. While we make no claims that such *ballistic* behavior is better than more observant models, it

is worth pushing the limits of these simpler models. Measurement and testing are time consuming and expensive, and any increase in the set of assembly primitives has a dramatic effect upon the size of the evolutionary search space. As we'll see, even absent measurement, Evolutionary Fabrication is remarkably capable of reliably generating large complex objects in the presence of noise and error during assembly.

The first question we seek to answer is whether, absent any means to test intermediate solution, evolutionary fabrication can discover how to reliably build a goal structure. Since the presence of noise complications matters significantly, it is worth exploring the issue first in a somewhat simpler context.

5.1.1 A Simpler Framework

Consider a simplified version of the Evolutionary Fabrication Framework in which the print-head turtle ranges over a discrete 100x100 grid and the bricks are now 10×20 rectangles subject to a simple “tetris-like” physics much simpler than ODE.

The assembly plan genotype remains largely the same - containing instructions such as forward, rotate, put brick and take brick. There are no “scaffolding” bricks in this model. Consequently, the assembly process is also simplified somewhat - each assembly instruction is interpreted in turn and, when assembly is complete, the resulting structure is evaluated.

Rather than an open ended fitness function, we are now trying to develop the assembly process for a pre-determined structure - the arch shown in Figure 5.2.

Noise Model

A noisy development environment can be induced by allowing vertical bricks to topple to either side 50% of the time, and for cantilevered bricks to topple 50% of the time.

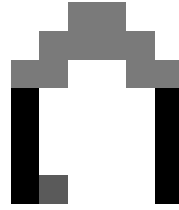


Figure 5.2: The goal arch for the simplified noisy development model. Vertical bricks are black and horizontal bricks are grey. The turtle is the dark grey square next to the left leg.

5.1.2 Evolving without Noise

In the absence of noise evolution quickly discovers the most efficient means of building the goal arch. Of course, if we then take that “naive” assembly plan and execute it in the noisy environment, it performs dismally. Figure 5.3 provides a sample of the distribution of phenotypes that arise in this situation.

5.1.3 Measuring Reliability

If, as Figure 5.3 demonstrates, noise during assembly induces such a wide distribution of phenotypes, then the challenge lies in finding ways to *reliably* build the goal structure. One way of measuring the reliability of an assembly plan in a noisy environment is to execute it multiple times and see how often, if at all, it can produce the goal structure. This provides the notion of “yield”. In the experiments below, we build each genotype 50 times, gathering statistical properties of the results to use as fitness objectives.

The specific objectives, shown below, are similar to those used for evolving the goal arch in the earlier section, with the addition of the distribution-measuring aspects such as the “yield” metric.

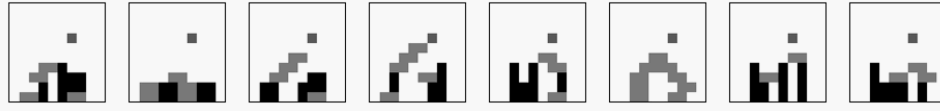


Figure 5.3: A sample of the wide distribution of phenotypes that arise when a single assembly plan is assembled in a noisy environment.

- **Length:** shorter is better.
- **Mass:** attempting to minimize the number of bricks in the structure.
- **Missing Material:** Number of bits missing from goal structure
 - best result, average result, and yield (perfect structure).
- **Error:** Number of “wrong” bits - i.e. either extraneous or missing bits.
 - best result, average result, and yield (perfect structure).

5.1.4 Emergence of Ontogenic Scaffolding

The system described above is typically able to generate assembly plans with yields above 70%. Figures 5.4, 5.5, and 5.6 contain animation frames from a typical result. This particular result described below is 82 instructions long, and achieves 70% yield.

The evolved assembly plan is able to achieve this reliability by means of *ontogenic scaffolding* - structural elements that are functionally necessary for reliable assembly, but that do not exist in the finished product. Once all of the elements of the final structure have been placed, this scaffolding is removed.

Consider the frames in Figures 5.4, 5.5, and 5.6 below. The evolved assembly plan first places horizontal bricks to the left and the right of what will become the first leg of the structure. Their presence guarantees that the leg will stay in place. The plan then places the first and second vertical bricks - both parts of the goal structure.

Note the redundant instruction in the sixth frame of Figure 5.4 - although it appears extraneous in this particular sequence, it proves useful in situations where the first attempt at laying the second brick fails: in which case the fallen brick ends up acting as scaffolding for the subsequent attempt.

In the following frames of Figure 5.4, the evolved plan proceeds to lay scaffolding for what will be the leftmost leg and the leftmost cantilever of the arch.

The assembly continues in Figure 5.5 as the plan continues to lay bricks that are simultaneously scaffolding for the leftmost cantilever and for the left leg of the arch. Once scaffolding has been laid on both sides, both vertical bricks of the left leg are placed. By the final frames of Figure 5.5, all of the bricks of the final structure are in place.

All that remains is for the print head to remove the scaffolding, as it does in Figure 5.6.

A meaningful way to visualize results which contain a distribution of phenotypes corresponding to a single genotype is to build a composite image by averaging the results of multiple runs of that genotype. The center column of Table 5.1 compares composites of a naive solution (which was evolved in the absence of noisy assembly and then assembled under noise) with evolved solutions which achieved 11%, 27% and 59% and 75% yield. As can be seen, as evolution progresses, the composite image increasingly looks like the final goal structure.

In the context of noisy development in which genotypes are rewarded for their yield percentage of a final structure, such as ours, one can consider the role of evolution as learning to shift phenotypic fitness distributions, rather than individual fitnesses, towards the optimal. This is borne out by the distribution column in Table 5.1. Each figure is a histogram which shows the distribution of phenotype fitness over the same 100 builds used to generate the composite images. As shown, as the yield increases,

the distribution tightens and shifts towards the optimal.

5.2 Evolving for Scalable Complexity

One outstanding challenge in the open-ended evolution of formation lies in *scalable complexity* - that is, how to build increasingly large, increasingly complex objects in a managed fashion. Crucial to this is the process of hierarchical, modular assembly.

Hierarchy and modularity play an important role in biology - not just in the organization of organisms, but also in their development. A popular example of representational modularity in nature is the “*eyeless*” gene in *Drosophila* which, when mis-expressed, causes physiologically complete eyes to sprout in unexpected places [35]. The importance of representational modularity lies in its ability to couple functionally related portions of the genotype while simultaneously decoupling them from functionally unrelated portions [100]. Changes to a representational module have few side effects in the remaining genome, and changes outside a module have few effects upon the module. The question arises, how can such modularity *evolve*?

In his landmark essay “The Architecture of Complexity”, Herbert Simon [90] begins to answer this question, making the case for the evolutionary necessity of hierarchical modularity through his parable of the two watchmakers, Tempus and Hora. Tempus builds his watches incrementally, piece by piece, and when interrupted, he puts down the watch he is working on, which then falls apart into its constituent pieces. Hora, on the other hand, first combines pieces into separate small modules, and then combines those modules together into a final watch. Consequently, he only loses the particular module that he is working on, and so is significantly more likely to build a complete watch than Tempus is. Simon uses this story to claim that increasingly complex forms are nearly impossible to evolve without such hierarchical composition

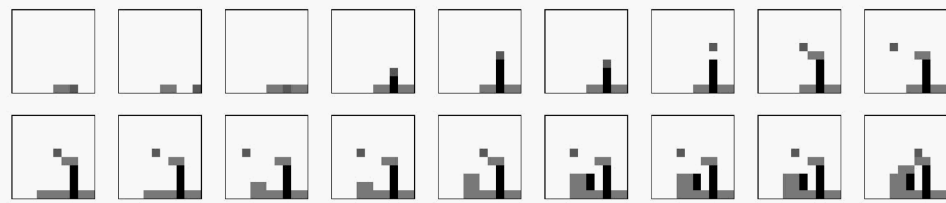


Figure 5.4: Robust Assembly Plan Steps 1-18: In the first steps, the builder lays scaffolding (frames are read left to right, top to bottom)

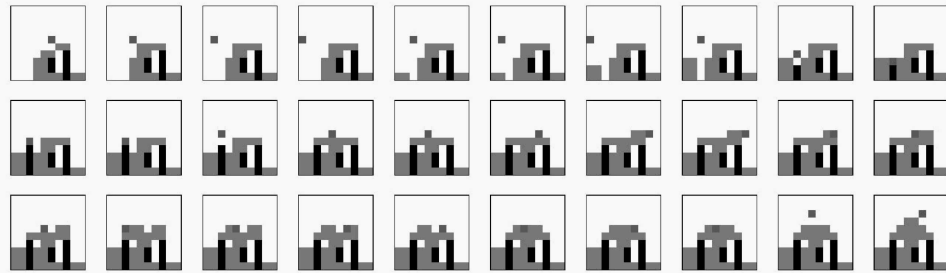

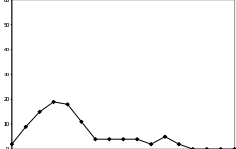

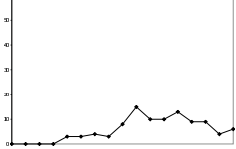

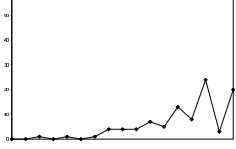

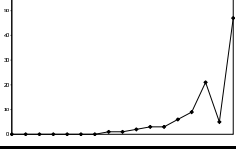

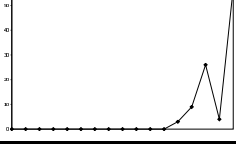


Figure 5.5: Robust Assembly Plan Frames 19-49: more scaffolding is lain and the arch is completed



Figure 5.6: Robust Assembly Plan Frames 50-80: scaffolding is removed

Table 5.1: Visualization of the improving yield of evolved assembly plans. The middle column contains composite results created by averaging 100 builds - darker squares represent locations more likely to contain a brick. The right hand column shows the histogram distribution of phenotype fitness across those same 100 builds - the horizontal axis represents increasing fitness, with maximal fitness, meaning perfect assembly, on the extreme right. Scales between histograms are identical

% Yield	Composite	Distribution
Naive		
11%		
27%		
39%		
75%		

of *stable* modules. In his own words, “The time required for the evolution of a complex form from simple elements depends critically on the numbers and distribution of potential intermediate stable forms.”

Many researchers have used this parable as inspiration for exploring adaptive representations and the relationship between modularity and evolvability. In this context the stability of a form is interpreted as evolutionary stability - that is, insulation from potentially deleterious effects of mutation and crossover. As such, the evolvability of adaptive representations comes from their ability to dynamically generate modules in the process of search [100].

Our interest here, however, is in a different perspective on the story - that of the relationship between modularity and noise in developmental representations, which take their cue from the biological processes of ontogeny and growth. Since they seek to model “biological assembly” (albeit of systems quite distinct from watches), these systems lend themselves to a rather straightforward interpretation of Simon’s parable. Like watchmakers, developmental systems seek to assemble complex forms from primitive constituent parts, and perform their tasks under the risk of interruption. While for the watchmakers interruption during assembly took the form of a telephone call, in developmental systems the role is performed by error and noise during ontogeny. Like watchmakers, in order to generate complex objects, adaptive developmental representations require modularity and, much like Hora’s subcomponents, in order to be useful, these developmental modules must be stable and reliably producible in the presence of noisy assembly.

If stochastic development imposes one-to-many relation on the genotype-phenotype map, such that each genotype can grow into an entire distribution of phenotypes, then another criterion for modular acquisition arises: that of developmental *stability*.

An important aspect of the utility of modules is their reusability. If, therefore, a

module is to be generated and reused multiple times, then it stands to reason that multiple copies of the module should exhibit low developmental variance. Consider Figure 5.9. A genotype which, under noise, develops into a wide or multi-modal distribution of phenotypes, such as the one of the left hand side of the figure, is not ontogenically stable, and so would not make a very reliable module. A more ontogenically stable genotype, on the other hand, typically develops into a tight, unimodal distribution with low variance, such as the one on the right hand side of the figure, meaning that it will generate near-identical copies, and so makes for a more suitable module.

5.2.1 Modularity and Hierarchy in Representation

The importance of representational modularity lies in its ability to couple functionally related portions of the genotype while simultaneously decoupling them from functionally unrelated portions. Changes to a representational module have few side effects in the remaining genome, and changes outside a module have few effects upon the module. Wagner and Altenberg [100] persuasively argue that the evolvability of a system is highly contingent upon its representation's ability to adapt by discovering and incorporating evolutionary modules.

Several models of representational modularity in evolutionary computation exist. Many systems, such as Hornby's L-systems [39] and Bongard's Gene Regulatory Networks [11] feature representations which are implicitly modular.

Of those which provide for the *explicit* encapsulation of modular components, the most common fall under the rubric of Hierarchical Genetic Programming (HGP), where encapsulated modules become new primitives in the language. Koza [50] developed Automatically Defined Functions (ADFs), in which sub-functions are allowed to evolve their own function and terminal sets. Angeline expanded upon ADFs with

module acquisition (MA), which co-evolve a representational genetic “library” of encapsulated primitives which are universally available to evolving programs[4]. Subsequently, Rosca and Ballard introduced Adaptive Representation through Learning (ARL), which replaced the randomness inherent in modular acquisition in ADFs and MA with a “usefulness” heuristic based upon fitness contribution and activation frequency of subtrees [86].

More recently, de Jong co-evolved a representation and its corresponding population of genotypes [21]. Candidate modules were chosen by finding the most frequent pair of alleles in the current population and, drawing from Watson’s work on symbiotic composition [101], were added to the language as primitives only if their fitness contribution was at least as good as the fitness contribution of all other possible pairs in a randomly chosen context.

While they vary in their details, each of these models of modular encapsulation involve incorporating *genotypic* sequences, thereby protecting them from the deleterious effects of mutation and crossover, and then adding them to the language of representation. As such, encapsulated modules are simply shorthand for the genetic sequence they represent - one can be substituted for the other without consequence. Below, we will motivate a system of encapsulation which, by contrast, involves incorporating *phenotypic* results into the language or representation.

5.2.2 Why Endosymbiosis?

Because of their prescriptive nature, developmental representations display a measure of context sensitivity: the same sequence of operations can have vastly different results depending on where in the process it occurs. Luke and Spector [60], for instance, discuss how the procedural nature of cellular encodings makes them particularly sensitive to crossover. This same “execution order dependence” exists in the assembly

plans used by Evolutionary Fabrication. Consider a developmental representation as a *recipe*. The set of instructions which produce egg whites in a souffle recipe (separate yolks and whites, place whites in a bowl, whip into soft peaks) would produce a mess (if anything at all) if they occurred later in the recipe or, for that matter, in an omelette recipe.

To make matters worse, in developmental systems a contiguous portion of the phenotype which we might want to modularize may have been produced by disjoint portions of the genotype. Similarly, a contiguous portion of the genotype may produce disjoint phenotypic results.

These factors can therefore stymie adaptive models such as HGP, which generate modules by extracting and compressing favorable genetic sequences. A genetic sequence which produces a favorable trait in one context will not necessarily preserve that result when transferred to another context. Furthermore, favorable phenotypic traits may not be attributable to modularizable portions of the genotype, and modularizable portions of the genotype may not produce useful phenotypic modules.

The challenge of modular acquisition in developmental representations, then, lies in preserving not the syntax, but rather the *meaning* of a desired phenotypic result. *Endosymbiosis*, the encapsulation of an entire organism by its host, is the model which we propose for this.

In our model of endosymbiotic encapsulation (See Section 5.3), complete organisms, not just specific portions of their phenotype, are used to form modules. As such, endosymbionts become *precompiled* phenotypes, and join the set of primitives available to the representation. To continue the metaphor of the recipe, endosymbiosis is analogous to the parallel creation of a *sous chef*, who specializes in producing that one particular higher-order ingredient, such as stiffened egg whites.

When referenced during the course of development, it is the phenotypic result –

the complete endosymbiont – rather than the genetic sequence responsible for creating the endosymbiont, that is used by the developmental process. In this manner, the meaning, rather than merely the syntax, of a module is preserved, and can be applied consistently across contexts.

5.3 Modular Evolutionary Fabrication

As we described above, our hope is that this endosymbiotic model of modular encapsulation will allow Evolutionary Fabrication to evolve how to build increasingly large, increasingly complex objects, even in the presence of noise during assembly which otherwise severely affects results. Adding this modularity to our Evolutionary Fabrication Framework is rather simple, as described below.

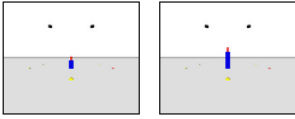
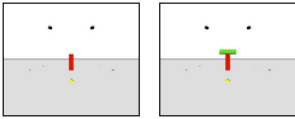
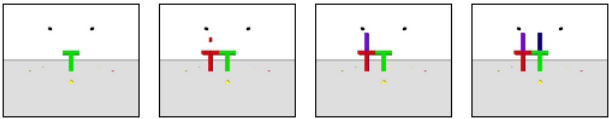
New Operators

The “put” command is modified to take an argument - a unique identifier corresponding to an object in the library of encapsulated modules. Initially, the only objects available to the “put” command are primitive $2x1x1$ bricks. As new modules are encapsulated, they are inserted into the library as new objects and can be referenced by the “put” command (for instance `put(brick)` or `put(module15)`.) As the modular library grows, the mutation operator selects from any of the modules currently available. Table 5.2 contains examples of what these new assembly plans look like, and Table 5.3 shows what the execution of these plans would produce.

Table 5.2: Example modules and their associated assembly plans

mod1	$P(a), P(a)$
mod2	$M(+2), P(\mathbf{mod1}), M(+2), R(+90), P(\mathbf{mod1})$
mod3	$M(+2), M(+1), P(\mathbf{mod2}), R(-90), M(+2), M(+2), R(+90), P(\mathbf{mod2}), \dots$ $\dots M(+2), M(+2), P(\mathbf{mod1}), R(+90), M(+2), M(+2), R(-90), P(\mathbf{mod1})$

Table 5.3: Hierarchical assembly of modules from Table 5.2

mod1	
mod2	
mod3	

Noise Model

Noise is injected into the system using a “shaky hand” model. When instructed to place an object, the turtle puts anywhere within some range Δx around its current position, with uniform probability. Noise settings were given as percentages of a brick’s width.

Evaluation

To measure the effect of developmental noise on evaluated genotypes, each assembly plan was interpreted 10 times, and average values over each objective were used for selection.

5.3.1 Module Discovery

The key feature of this new framework is the ability to discover new modules *during the process of evolution* and then add them to the set of primitives available to evolving assembly plans, as shown in the schematic of Figure 5.8.

Every 10 generations, unused modules are removed from the library and new

candidate modules are discovered and added to the library, as shown in Figure 5.7. We describe the details of the process below.

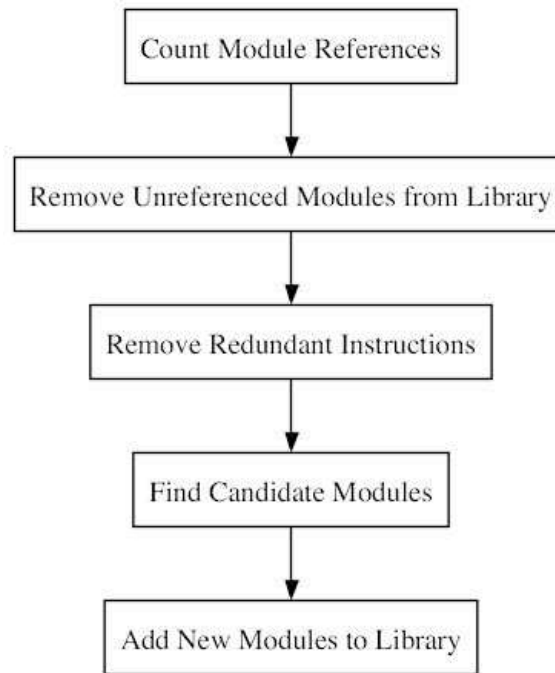


Figure 5.7: The process of module rejection and discovery.

Module Selection

Candidate endosymbiotic modules were selected from the phenotypes of the evolving population every 10 generations. The criteria for selection are as follows:

- **Fitness:** the selected module must be a member of the current pareto front. That is to say it must have some intrinsic fitness beyond these other criteria.
- **Unity:** the selected module must only consist of a single piece. Obviously if an assembly plan results in a structure with two distinct pieces, then it cannot be usefully encapsulated as a context-independent building block.

- **Reliability:** the assembly plan which produces the candidate module must have sufficiently low variance in fitness (see Figure 5.9.)

If a phenotype matches the criteria then it is added to the module library as a whole object, becoming available as an argument to the “put” instruction.

Module Rejection

The *evolutionary viability* of the modules is determined by their reference count in the population of evolving assembly plans. Care must be taken in order to prevent spurious modules, those which are referenced by assembly plans but have no consequence (for instance those that are ignored because their placement would intersect with an existing object) from causing “bloat” in the set of primitives. Candidate modules are therefore tested for *necessity*. Each member of the pareto front which contains a reference to a module is built twice - once with and once without the referent module. If the results are the same then the module reference is permanently removed from the assembly plan. This is repeated for each module to which an assembly plan refers.

Once those unnecessary module references have been removed from the population of assembly plans, the reference count of each module in the library is measured. Whenever a module’s reference count in the evolving population drops to zero it is deemed irrelevant, and removed from the object library.

5.4 Modular Assembly in Noisy Environments

In these following experiments we determine the usefulness of this model of modular encapsulation, and more importantly, its ability to cope with noise during assembly.

The design task is again to create a structure which maximizes the total open volume beneath it, thereby rewarding structures which both maximize height and

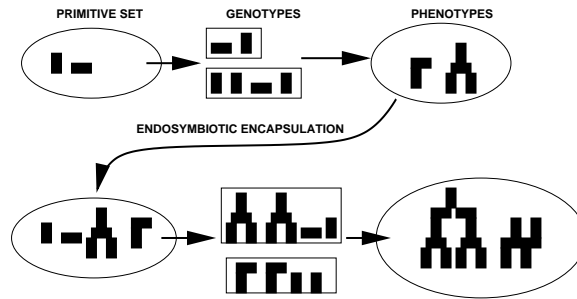


Figure 5.8: In the endosymbiotic model of module acquisition, only complete phenotypes, rather than genetic samples, are added to the set of primitives.

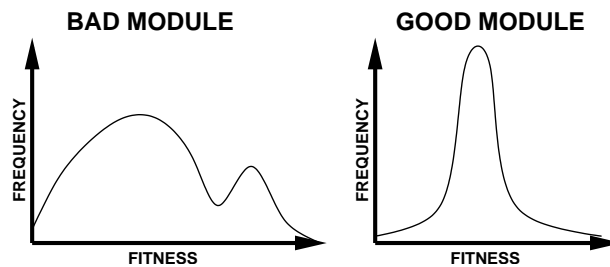


Figure 5.9: Properties of a useful module. Under the presence of developmental noise, each genotype develops into an entire **range** of phenotypes, each with an associated fitness. A bad module is one which exhibits high variance or multimodality across multiple builds; a good module will be unimodal with very low variance.

maximize the number of empty spaces beneath them, per Figure 5.10.



Figure 5.10: Illustration of Shaded Fitness Function. The structure itself is black, and the gray region is considered “shaded”.

- Length Of Assembly Plan (minimizing)
- Mass of Objects in Environment (minimizing)
- Shaded Area (maximizing)

To measure the effect of developmental noise on evaluated genotypes, each assembly plan was interpreted 10 times, and average values over each objective were used for selection.

5.4.1 Results

Three sets of experiments were run for 1000 generations each, with noise set to 0.1%, 1.0% and then 2.5% of a brick width. For comparison, parallel setups without modular acquisition were run at noise levels of 0, 0.1 and 1.0%. Figure 5.11 demonstrates the deleterious effects of noise on non-modular development. In the absence of noise, evolution proceeds fairly well. But even with relatively modest noise, at a level of 0.1%, average fitnesses are half of what they are in the noiseless case. As noise is increased to 1.0%, performance drops even further.

Figure 5.12 shows a comparison in performance for modular assembly across a range of noise values. Not surprisingly, the modular setups reach near-optimal fitness rather quickly, and outperform the non-modular ones in Figure 5.11. Furthermore,

there is very little difference in performance for modular assembly across the range of noises. Interestingly, the modular noisy evolution shown in Figure 5.12, across the entire range of noise values even outperforms non-noisy non-modular evolution in Figure 5.11. We discuss this further below.

Figures 5.13 and 5.4.1 contain representative assembly trees for some of the evolved objects. Appendix A contains several more examples. These trees provide some insight into the processes by which evolved assembly plans were able to hierarchically assemble the objects.

Interestingly, modular assembly outperforms non-modular assembly, even in the absence of noise. Consider that non-modular assembly must place structures brick by brick, and so is in general limited to incremental improvements in fitness over the course of evolution. The strength of modular assembly, on the other hand, lies in its ability to add larger sub-assemblies to its vocabulary, and then place them as a single unit, thus enabling faster progress.

This discovery of increasingly large sub-assemblies affects not only the *speed* of evolution, but also the *type* of structure that is evolved. As we first noted in [84], non-modular assembly plans in a non-stochastic environment tend to generate arches, even though tree-shapes are a more optimal solution. Our conjecture at the time was that this was due to the difficulty in building balanced trees brick by brick: both matching branches of the tree must be discovered in parallel, and most mutations to a balanced tree would unbalance it. Arches, on the other hand, are more evolutionarily stable, because they are supported on two legs, and can be discovered by a process which first creates a filled arch and then slowly learns to empty out the middle portion.

A key observation, therefore, is that the majority of structures evolved in this noisy environment with modular assembly plans are trees rather than arches. This is because the adaptive representation is able to generate larger, multi-brick modules,

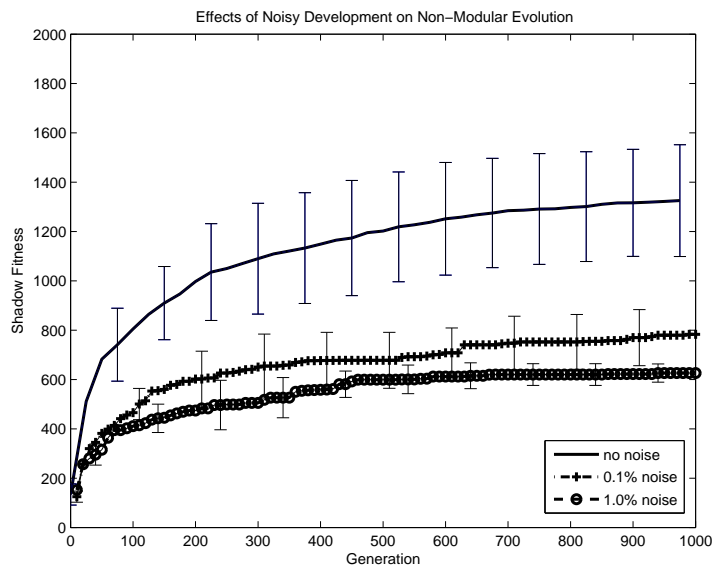


Figure 5.11: A demonstration of the effects of noisy development. Without noise, non-modular evolution proceeds well. Even with relatively low noise (0.1% of brick width), however, runs do significantly worse. As noise increases, performance decreases. Averaged across 22 runs (noiseless) and 10 runs (noisy), with error bars.

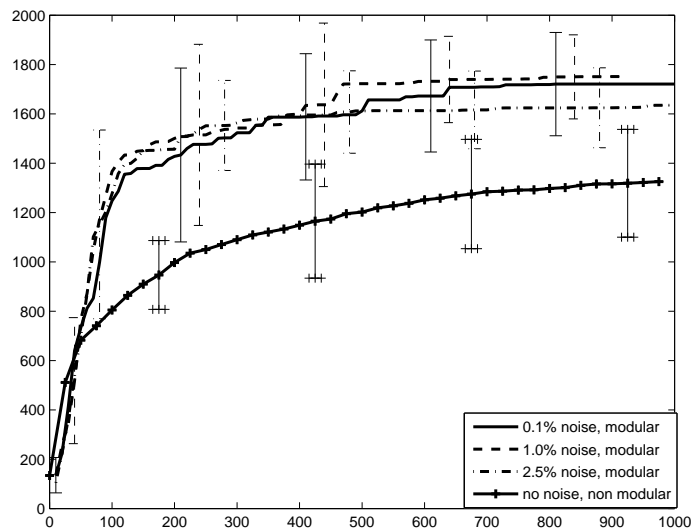


Figure 5.12: Across a range of noise levels, not only does modular evolution outperform non-modular noisy evolution, but it also outperforms non-modular, non-noise evolution. Averaged across 10 runs of each setup, with error bars.

and then place them as a single, balanced unit atop a column. As can be seen, in every case it is a single module which forms both branches of the tree.

It could even be argued that several of the assembly trees shown in the tables also exhibit a form of *exaptation* in the evolutionary process. The tree shown in Figure 5.13 is an excellent example. Because symbiogenetic modules are selected for, among other things, their presence on the pareto front, they often have a measurable fitness when encapsulated as modules. When they are used in hierarchical assembly, however, instead of being placed in a manner which takes advantage of this inherent fitness (for instance, by placing them in parallel to form an arch), they are rotated and placed sideways atop a newly formed trunk. As such they serve a new function - for instance as a branch instead of a trunk, and in that role they are able to contribute more fitness than they do alone.

5.5 Scalable Modular Assembly

The above experiments establish first that error during assembly has a deleterious effect on the progress of Evolutionary Fabrication. Secondly, they show that our modular extension to Evolutionary Fabrication is able to overcome that noise by a form of “bootstrapping”, by finding small reliable sub-assemblies and then incorporating them as new primitives in the assembly process.

To determine whether these methods scale to larger environments and more complex fitness functions, we next quadruple the size of the assembly environment, as shown in Figure 5.15. Merely making the environment bigger would only result in larger tree shapes, but not necessarily in more complex hierarchies of assembly. In the following experiment, we therefore replace the “shade” fitness function with one that measures the “leafiness” of a structure. We calculate this by taking vertical slices

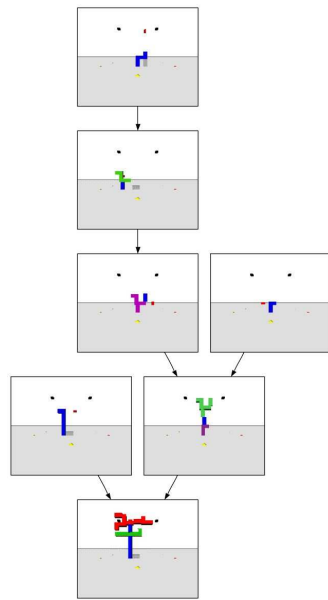


Figure 5.13: An Example Hierarchical Assembly of Modules at 0.1% noise. Further examples are provided in Appendix A

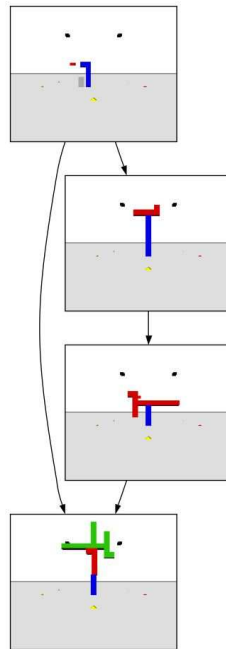


Figure 5.14: An Example Hierarchical Assembly of Modules at 1.0% noise. Further examples are provided in Appendix A

along the width of the structure and counting the number of surfaces we cross along the way, as illustrated in Figure 5.15.

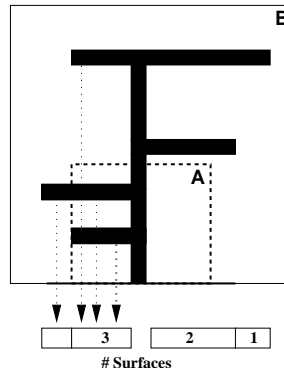


Figure 5.15: Illustration of Larger Environment and “Leafy” Fitness. The larger environment (Box B) is four times as large as the original (Box A). “Leafiness” is measured as the number of surfaces encountered along a vertical slice.

The fitness objectives are listed below:

- Length Of Assembly Plan (minimizing)
- Mass of Objects in Environment (minimizing)
- Leafiness (maximizing)

The optimal structure is no longer a “T”, therefore, but a series of tightly packed horizontal branches. Because of the interplay with the remaining mass objective, and the exaptative nature of our modular acquisition, the structures we will see will be less regular and more open than this.

5.5.1 Results

Three sets of experiments were run, at noise levels of 0.1% (11 runs), 1% (8 runs) and 5%(10 runs). Figure 5.16 shows the progression of evolution across the three noise levels, with error bars. As can be seen, even with noise at 5% of a brick width,

modular evolution is still able to progress to produce highly fit individuals – in fact, at its best it does better than the average 1.0% run, and better than the worst 0.1% run. This leads us to believe that higher noise has a *retarding* effect upon progress, in the sense that it takes evolution longer to find reliable modules, but less so a *limiting effect*: by allowing the 5% run to continue we see that it catches up to the 1.0% run within the next 100 generations. Figure 5.17 contains sample assembly hierarchies evolved at the 5% noise level. Further examples are shown in Appendix A.

5.6 Summary

This section has demonstrated Evolutionary Fabrication’s ability to overcome noise and error during assembly. Even without the ability to explicitly place meltable scaffolding, evolved assembly plans can nonetheless learn to reliably build a goal structure in the face of error by placing *ontogenic scaffolding* - temporary elements that are essential for the reliable assembly of the goal structure, but which are removed before final evaluation.

Section 5.3 introduced a form of modular encapsulation based upon phenotypic symbiogenesis which addresses the context dependency of developmental representations. Structures which exhibit *high reliability* in the face of noisy development were chosen as candidate modules, and added as a whole, to the set of primitives. Not only does this method of modular assembly overcome the deleterious effects of noisy development in an artificial ontogeny across a range of noise levels. Interestingly, even at the highest level of noise it also outperforms non-modular methods evolved without noise.

The strength of this method lies in a form of *developmental bootstrapping* - small subcomponents are composed into larger *stable* modules available to the represen-

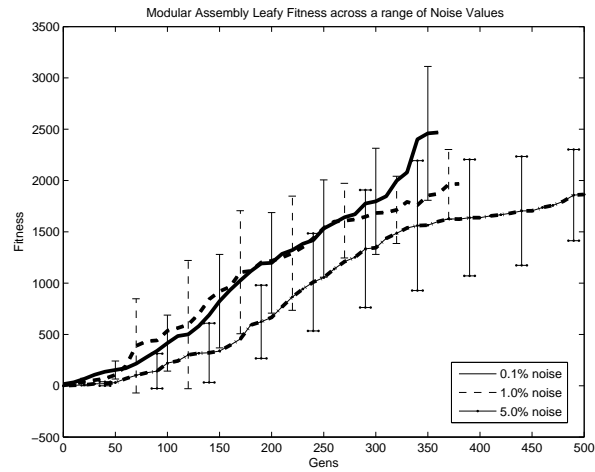


Figure 5.16: Evolutionary Fitness of the “Leafy” Fitness Function across a wider range of noise values. Higher noise has more of a *retarding* effect than it does a *limiting* effect.

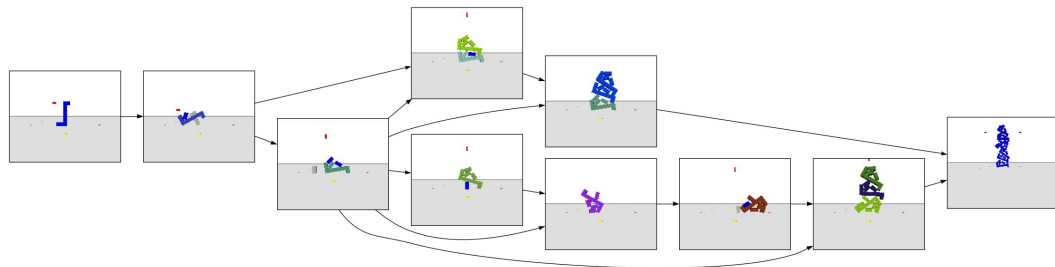


Figure 5.17: An example of hierarchical assembly of larger structures at 5% noise. Note the increased size and complexity of the assemblies. Further examples are provided in Appendix A

tation, and in that manner multi-tier hierarchically composed assembly methods emerge. As the size of the sub-assemblies increases, Evolutionary Fabrication is able to make incrementally larger structures with relatively fewer instructions.

Chapter 6

Discussion and Conclusion

At this point we have introduced Evolutionary Fabrication as an alternative to blueprint-based Evolutionary Design. Further, we have shown how Evolutionary Fabrication, while allowing for the full automation of design *and* assembly, remains endowed with the same level of creativity and innovation found in more traditional, blueprint-based approaches. By directly evolving assembly plans, EvoFab is capable not only of finding efficient means of manufacturing objects, but of discovering entirely novel and unexpected ways of doing so. We can now discuss some of the broader themes and implications of these results.

6.1 Novelty and Invention

What do we mean, exactly, when we say that Evolutionary Fabrication allows for the “emergence” of “novel” means of assembly? In their work, Ronald *et al.* warn, “We do not think...that emergence should be diagnosed whenever the unexpected intrudes into the visual field of the experimenter” [85]. Inspired by the Turing Test, they provide an observer-based test for emergence based on the notion of “surprise”: that

is the extent to which the connection between local rules of interaction and observed global behavior is *non-obvious*. This meshes well with Koza’s notion of *inventiveness* in [51], wherein he cites one of the US Patent Office’s criteria for a new invention: that it should be non-obvious “to a person having ordinary skill in the art to which said subject matter pertains”. Like the test for emergence, inventiveness is contingent upon impressing an external observer with an unexpected and novel phenomenon that seems to exceed the properties of the system.

A clear example of novel assembly occurs in the assembly of the goal arch from Section 3.2, reproduced below in Figure 6.1. The scaffolding brick placed in frame 1 tumbles from its horizontal placement into a vertical position below. By falling into position in this manner, the scaffolding ends up directly under the center of mass of the brick it will support. This final location is actually between two of the discrete print-head positions, and so the scaffolding could not have been placed directly. If it had instead been directly placed by the print head the into an adjacent position the brick it supports might have tilted sideways.

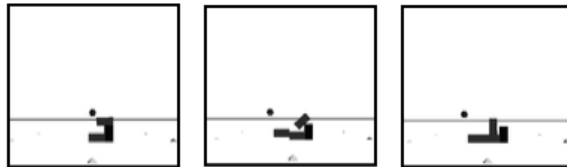


Figure 6.1: A non-reversible toppling motion. Note how the horizontal scaffolding placed in frame 1 tumbles into a vertical position.

The “dynamic assembly” observed in Section 4.3 is yet another striking example of a phenomenon which meets the criteria for novelty: watching those assembly processes, particularly the one shown in Figure 6.2 never fails to elicit a certain *frisson* of surprise. The non-obviousness of the phenomenon can be attributed to the fact that even though Evolutionary Fabrication is limited to placing bricks one at a time, it nonetheless discovers how to dynamically assemble two or more multi-brick sub-

assemblies by exploiting the settling phase of the construction process. This emergent, modular, meta-assembly *transcends* the brick-by-brick language of assembly.

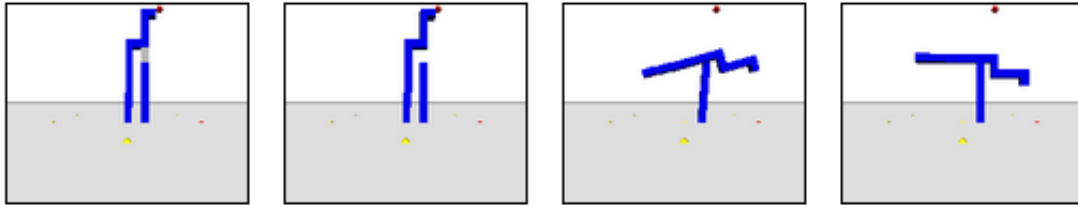


Figure 6.2: An extreme example of dynamic assembly

Each of these phenomena of unusual assembly meet the criteria of emergence and inventiveness, and serve as a strong example of the ability of Evolutionary Fabrication to inject novelty into the realm of assembly. While we do not claim that these exact phenomena would arise in a the real world, we conjecture that equally novel and interesting methods of assembly would emerge in a physically embodied system, just as they have in Evolutionary Robotics [68, 102] and Evolvable Hardware [95].

6.2 Assembly vs. Disassembly

The emergence of phenomena such as dynamic assembly also demonstrates the comparative advantage of directly evolving assembly over *post hoc* methods of determining assembly from a blueprint, such as Assembly Sequencing. Recall how Assembly Sequencing approaches simplify their task by assuming that assembly is a monotone, two handed process [33, 45]. Operating under these assumptions, Assembly Sequencing can discover an object's disassembly, and then reverse the steps to arrive at its assembly.

And yet, the the assembly shown in Figure 6.1 and the dynamic assemblies in Section 4.3 are clearly *not* reversible, even though the assembly mechanism is both

monotone and two-handed. The structure shown in 6.2 may have been assembled brick-by-brick, but it certainly cannot be un-built brick by brick.

When assembly sequencing makes these assumptions about assembly, it does so in order to make the problem more tractable. Searching for an assembly sequence from the bottom up, as we do in Evolutionary Fabrication, opens up a much larger search space. As we've shown, the benefit of this tradeoff is the ability to discover unexpected, efficient, and novel means of assembly.

6.3 Hierarchy and Noise

Simon argues that “hierarchic systems will evolve far more quickly than non-hierarchic systems of comparable size” [90]. Using the parable of the error-prone watchmakers Tempus and Hora, he provides a quantitative analysis of the difficulty of assembly under noise: if each watch contains 1000 parts, and the probability of an error during assembly is p , then Tempus, who assembles his watches in a linear, non-modular fashion, has a $(1-p)^{1000}$ chance to create a complete watch. Hora, on the other hand, builds 10-piece sub-assemblies, each of which then has a $(1-p)^{10}$ of interruption, and when interrupted only loses the assembly he is working on rather than the complete watch. With $p = 0.1$ Hora will successfully complete his watch about 4000 times more often than Tempus.

Our results comparing modular assembly to non-modular assembly in Section 5.3 recapitulate this analysis in the context of Evolutionary Fabrication. As shown by Figures 5.11 and 5.12, the hierarchical system far outperforms the non-hierarchical system, even with only minimal noise. Figure 6.3 compares the rates of hierarchical and non-hierarchical evolution at 0.1% noise.

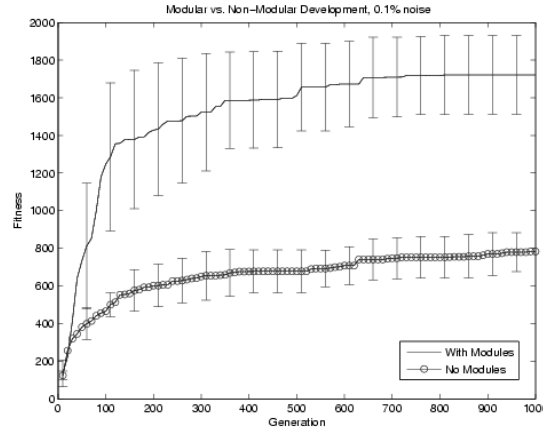


Figure 6.3: Our results comparing the progress of modular vs. non-modular evolutionary systems recapitulates Simon: *“hierarchical systems will evolve far more quickly than non-hierarchical systems of comparable size”* [90]

6.3.1 Measuring Modularity

We can also investigate how varying levels of noise affect the *type* of hierarchies which emerge. Given the assembly plans for a structure and its corresponding modules we can generate a directed graph of the hierarchical assembly process by creating an adjacency matrix. (In fact, all of the graphs of hierarchical assembly shown in this work were produced automatically in this manner). We can then measure properties of the graph, such as the depth of the hierarchy, the number of modules (nodes), and the amount of reuse (edges). Using these measurements we can compare relative values between hierarchies evolved at differing noise levels.

Consider for instance the the matrix below, corresponding to the graph shown in Figure 6.4. The value at (i, j) corresponds to the number of times $module_i$ is used in $module_j$. In this hierarchy there are four nodes, four edges, and the depth is four.

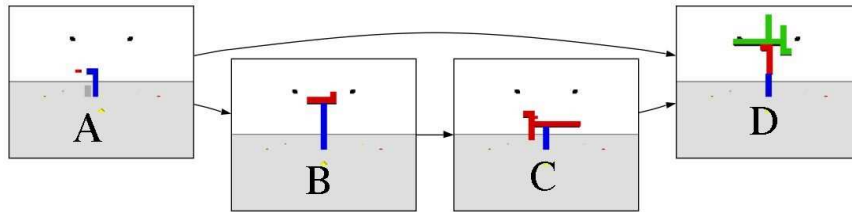


Figure 6.4: A robust hierarchical modular assembly from Section 5.3

	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>
<i>A</i>	0	0	0	0
<i>B</i>	1	0	0	0
<i>C</i>	0	1	0	0
<i>D</i>	1	0	1	0

Using these metrics we can compare the properties of the hierarchies which emerged in the “leafy” fitness function at 0.1% noise with those that arose at 5.0% noise. In each survey we measure the properties of all structures with fitness between 1200 and 1800. Taking into account the retarding effect that increased noise level has on the progress of evolution, in order to compare structures of equivalent fitnesses the 0.1% noise hierarchies were sampled at generation 280, and the 5.0% hierarchies were sampled at generation 500. The survey consists of 109 individuals across 10 runs at 0.1% noise and 212 individuals across 11 runs at 5.0% noise. In the bar graphs below, measurements for each hierarchy were normalized by the overall fitness of the hierarchy.

A two-sample T-test shows that the distributions of fitnesses between the two surveys are equivalent ($p = 0.778$), as are the normalized depths ($p = 0.9724$) and total number of edges ($p = 0.5743$). The distributions of *nodes* between the two samples, however is quite distinct ($p = 3.771e^{-8}$) - with the samples from the noisier environment having significantly fewer nodes. These data are presented in the bar

graphs in Figures 6.6 and 6.7.

This above analysis shows a key quantitative difference between the hierarchies that emerge at the different noise levels. While the fitness, depths and edge counts in both sets of hierarchies remain comparable, those hierarchies which emerge in higher noise have fewer nodes overall. The number of nodes in the hierarchy correspond to the number of *distinct* modules that are found by evolution. It makes sense that the hierarchies that emerged in the higher noise regime have fewer modules: as noise levels increase, the number of reliably buildable sub-assemblies decreases. And yet, while the number of nodes is different, the total number of edges between remains the same. In other words, the relative number of edges *per module* increases as noise increases.

This leads us to conclude that *increased levels of noise lead to increased levels of modular reuse* in our evolved hierarchical assemblies (Figure 6.5).

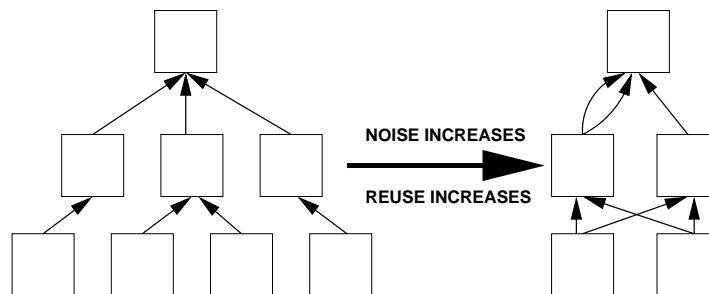


Figure 6.5: As the noise in the developmental system increases, the height of the trees remains essentially the same, as does the total number of edges. The number of modules, however, decreases, meaning that there are more edges *per module*. In other words, *increased levels of noise lead to increased levels of modular reuse* in our evolved hierarchical assemblies.

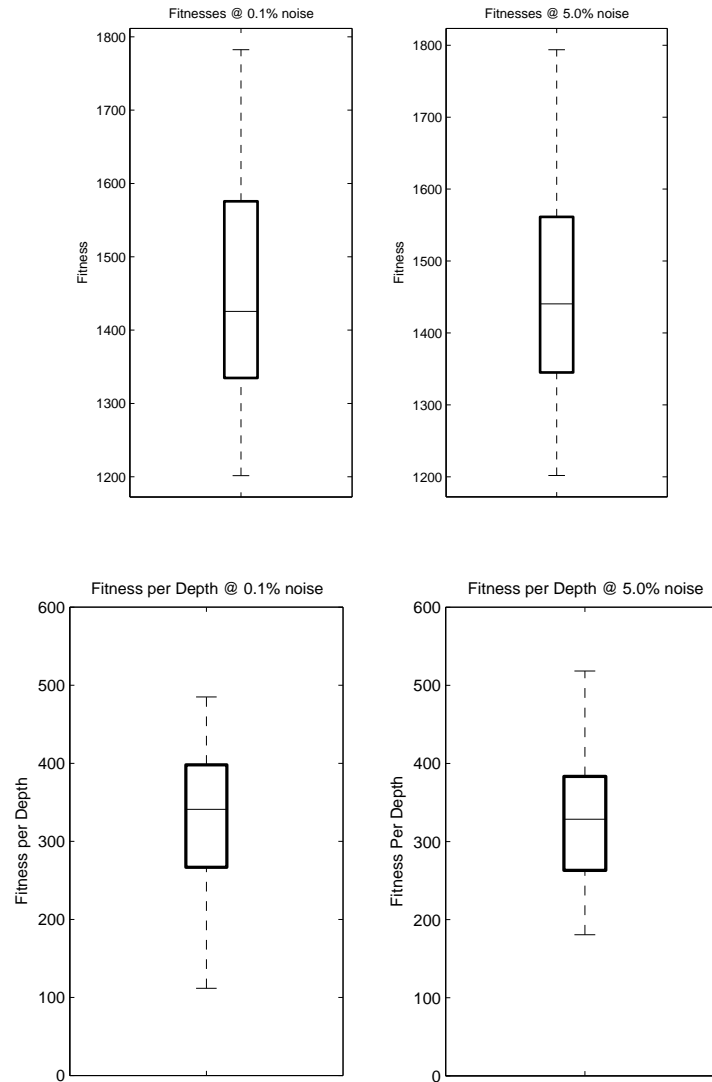


Figure 6.6: Comparing the effects on noise on the shape of the hierarchical assemblies which emerge. Increased noise does not seem to affect either the average fitness or the average depth of evolved hierarchies. The distribution of fitnesses sampled at 0.1% noise are indistinct from those sampled at 5.0% noise ($p = 0.7778$). The distributions of normalized depth between the two samples are also similar ($p = 0.9724$).

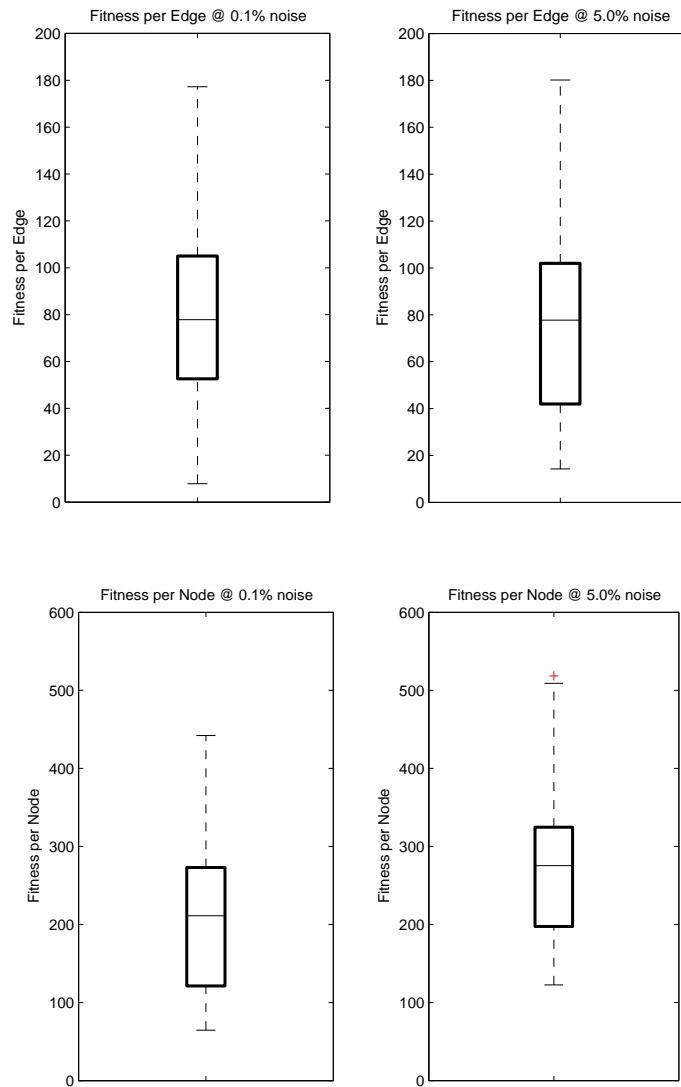


Figure 6.7: Comparing the effects on noise on the shape of the hierarchical assemblies which emerge. As noise increases, the amount of modular *reuse* increases. While the distribution of normalized edges in each sample is comparable ($p = 0.5743$), those hierarchies evolved in the higher noise regime have significantly fewer nodes ($p = 3.771e^{-8}$).

6.4 Measures of Complexity and Scale

Because the aim of Evolutionary Fabrication is to *assemble* things rather than merely design them, the *complexity* of a structure has less to do with the size of the structure or the total number of bricks, and more to do with the the complexity of the *process* required to build it. By this measure, the trees seen in our evolution for “shade” contain fewer bricks than equally fit arches, and yet require considerably more effort to assemble, because of the need for scaffolding to produce balanced branches.

Building on this notion of complexity of process, and borrowing from the information-theoretic notion of *Kolmogorov complexity*, we can provide a more formal, if preliminary, notion of structural complexity. Returning briefly to the abstract model of Evolutionary Fabrication provided in Section 2.5.1 we can say that the complexity of a structure s vis a vis an Assembly Mechanism \mathcal{M} and a language of assembly \mathcal{L} is *the minimum length of the assembly plan $a \in \mathcal{L}^*$ which, when interpreted by the mechanism \mathcal{M} produces the structure s* . This corresponds well with the measures of complexity for assembly sequences provided by Goldwasser, such as the total number of steps, and the number of non-two-handed steps [33].

The question of how Evolutionary Fabrication scales to increasingly large and complex structures rests, therefore, on its ability to evolve increasingly large and complex *processes* of assembly. Non-modular assembly, for instance, is limited in its scalability because an increase in the complexity of an assembly process can only be accomplished by a corresponding increase in the *length* of the assembly plan. Modular assembly, on the other hand, allows for a large degree of *compression*. Every module has associated with it a corresponding assembly procedure, and so a single modular reference in an evolving assembly plan increases the compressed length of the assembly plan by one instruction, but corresponds to a significantly

larger uncompressed set of instructions.

The measure of *compression* of a modular representation can be provided by calculating the mass per compressed instruction (MPI) of a modular assembly. Non-modular assembly is uncompressed, and so we would expect to see a relatively constant MPI over the course of evolution: the only way to place more bricks is to add a corresponding number of instructions. In modular assembly, by contrast, we should expect to see a significantly higher MPI, in that a single instruction can result in the placement of a multi-brick module.

Figure 6.8, which measures the MPI of both methods over the course of the first 300 generations confirms this. The MPI of non-modular assembly remains flat, while the MPI of modular assembly increases significantly over the same time period. This effect is repeated in the larger environment used of the “leafy” fitness in Section 5.5, as shown by Figure 6.9.

It is important to observe in these charts not only that modular representations are relatively compressed, but that their measure of compression *increases* over the course of evolution. This means not only that the scalability of modular assembly is higher than that of non-modular assembly, but that the *rate* of scalability increases over the course of evolution: as hierarchical assemblies of modules evolve, each module contains an increasing number of bricks.

6.5 Embodied Evolutionary Fabrication

We have shown how, in principle and if phrased correctly, Evolutionary Fabrication allows for the full automation of both design and assembly. We have yet to demonstrate a physical Evolutionary Fabrication system. The reasons for the delay are clear: rapid prototyping machines are expensive to own and to operate, and until this

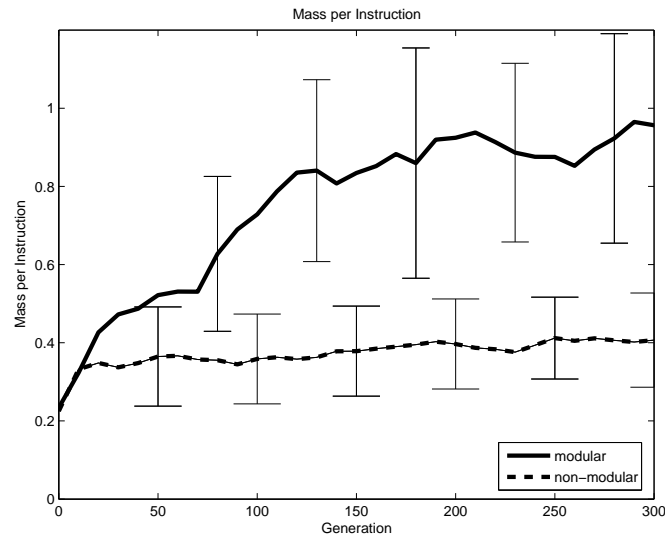


Figure 6.8: Mass per instruction (MPI) in modular and non-modular evolution. MPI is a metric of the *compression*. While non-modular assembly has relatively flat MPI, modular assembly is able to place increasingly more bricks per instruction.

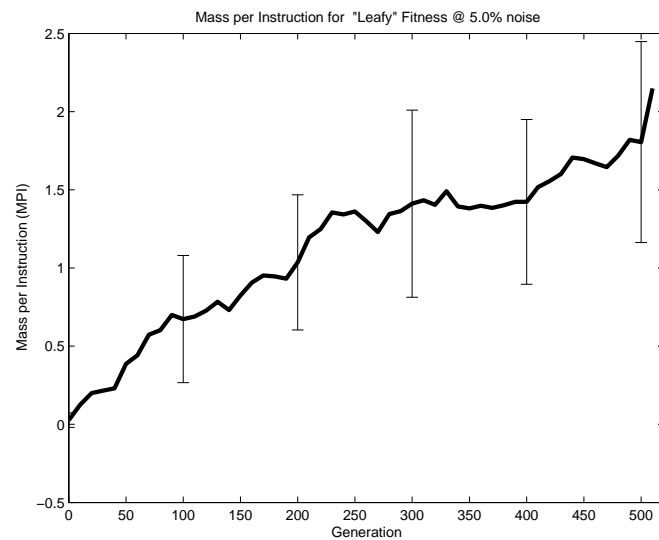


Figure 6.9: Mass per instruction (MPI) in the larger, “leafy” fitness function. Not only is MPI higher than non-modular assembly, it *increases* over the course of evolution.

body of research we lacked any foundational theory or reassurance that the scheme would even work in simulation.

How might we go about accomplishing “embodied” Evolutionary Fabrication? Two obvious approaches exist: the first is to take a cue from Watson *et al.*’s work on Embodied Evolution [102], wherein we may remove the notion of simulation entirely by fully embedding both design and assembly in the real world. Their results indicate that real-world evolution can often exploit physical phenomena to arrive at solutions manifestly different, and more optimal, than those found in simulation. This recapitulates in a robotic context Thompson’s work on the “Silicon Evolution” of an FPGA, in which his physically evolved circuit exploited the analog characteristics of the device to arrive at a solution that could not be produced in simulation. Outside of the realm of Evolvable Hardware, there are few other examples of fully-embodied evolution [58].

Realistically speaking, a fully embodied approach to Evolutionary Fabrication would not be practical. The costs of building and evaluating thousands of generations of assembly plans directly on a rapid prototyping machine would be prohibitively expensive and prohibitively slow.

Evolving in simulation, however, raises the spectre of Jakobi’s “Reality Gap”. How can we be certain that physical assembly will behave the same as simulated assembly? Evolutionary Robotics accomplishes this by ensuring critical elements of reality are accounted for in simulation [42]. While this approach works for evolving controllers for robots with fixed sensor morphologies in fixed environments, it presents a challenge as robots scale and environments become more variable. Furthermore, realistically modeling requires *a priori* knowledge of the operating environment, which may not be available.

Scaling simulations to larger and unknown environments requires *adaptive* simu-

lations. One approach, first proposed by Brooks in 1992 [14], is to co-evolve physical robots and their simulators, fine-tuning the simulation over the course of time. Bongard and Lipson's recent work uses a genetic algorithm to co-evolve a robotic controller and the parameters of an ODE-based simulation to compensate for unanticipated morphological changes in the robot, offers promising results [12].

An adaptive simulation of a manufacturing process, one which evolves in tandem with its physical counterpart, is therefore the best way to realize a physical Evolutionary Fabrication System. Assembly plans could be speedily evolved within simulation, and promising results could in turn be printed by the actual machine. Such an Embodied Evolutionary Fabrication machine would be the first real example of Fully Automated Design and Assembly, capable of inventing *and building* novel solutions to design problems at the touch of a button.

6.6 Conclusion

The advent of state of the art rapid prototyping machines, capable of producing three-dimensional objects out of plastic, ceramic, and metal, replete with circuitry and power sources, promises to revolutionize the realms of personal and industrial manufacturing. In parallel, Evolutionary Design has been creating, without human intervention, a wide range of novel and human-competitive solutions to challenging design problems.

Up until now, however, there has been no satisfactory means of joining the two fields to establish fully automated design and assembly. As we have shown, progress has been shackled by its dependence on blueprints which evolve *form*, but leave unanswered the vital question of *formation*. As a consequence there is growing *Fabrication Gap* between evolved designs and the processes required to build them.

In this dissertation we have proposed a reformulation of the Evolutionary Design task: the *co-evolution of form and formation*, simultaneously evolving *how* to build an object and *what* to build it out of. Artificial Ontogenies, inspired by the biological processes of growth and development, provide us with the tools to accomplish this.

This approach is not without its drawbacks. Most significantly it requires the realistic simulation of an object's entire assembly, rather than only its behavior once complete. The benefits of this approach, arise when evolution is allowed to range through the entire space of assembly methods, discovering not just novel objects, but novel means of assembling those objects.

The specific framework we have introduced to demonstrate this approach is *Evolutionary Fabrication* (EvoFab). In our model, the genotypes are assembly plans: linear, ballistic sets of instructions to an assembly mechanism rapid , and assembly itself unfolds within a simulation of that mechanism. This is, in essence, the direct genetic programming of a rapid prototyping machine.

We have provided several examples of the emergence of novelty, innovation, and efficiency within EvoFab, most notably the dynamical assembly of sub-assemblies in Section 4.3 , a form of meta-assembly which transcends the sequential and monotone nature of the manufacturing process.

We have also demonstrated two methods by means of which Evolutionary Fabrication can cope with the noise and error that arise in real-world assembly. The first is the emergent scaffolding in section 5.1.3, which allows EvoFab to reliably build a goal object in the presence of noise. The second is the open-ended hierarchical modular assembly in Section 5.3, in which the entire language of design evolves by discovering increasingly large, reliably buildable sub-assemblies and incorporating them as new building blocks. This bootstrapping approach allows for the hierarchical assembly of large objects across a wide range of noise levels.

Each of these accomplishments can be considered a cornerstone in the emerging field of *Fully Automated Design and Assembly*. This marriage of Evolutionary Design and Automated Manufacturing holds the promise of revolutionizing a broad array of fields, ranging from product design to planetary exploration. Armed with the progress presented in this dissertation, the next logical step towards Fully Automated Design and Assembly is to create a real-world embodied Evolutionary Fabrication system with which to recreate these results in a more tangible form.

Appendix A

Example Hierarchies

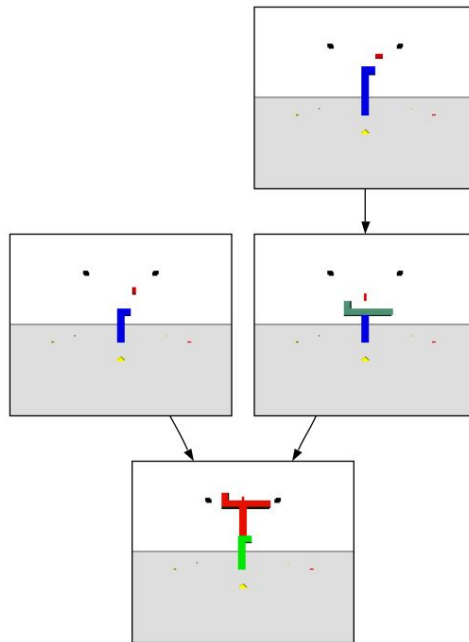


Figure A.1: An Example Hierarchical Assembly of Modules at 0.1% noise

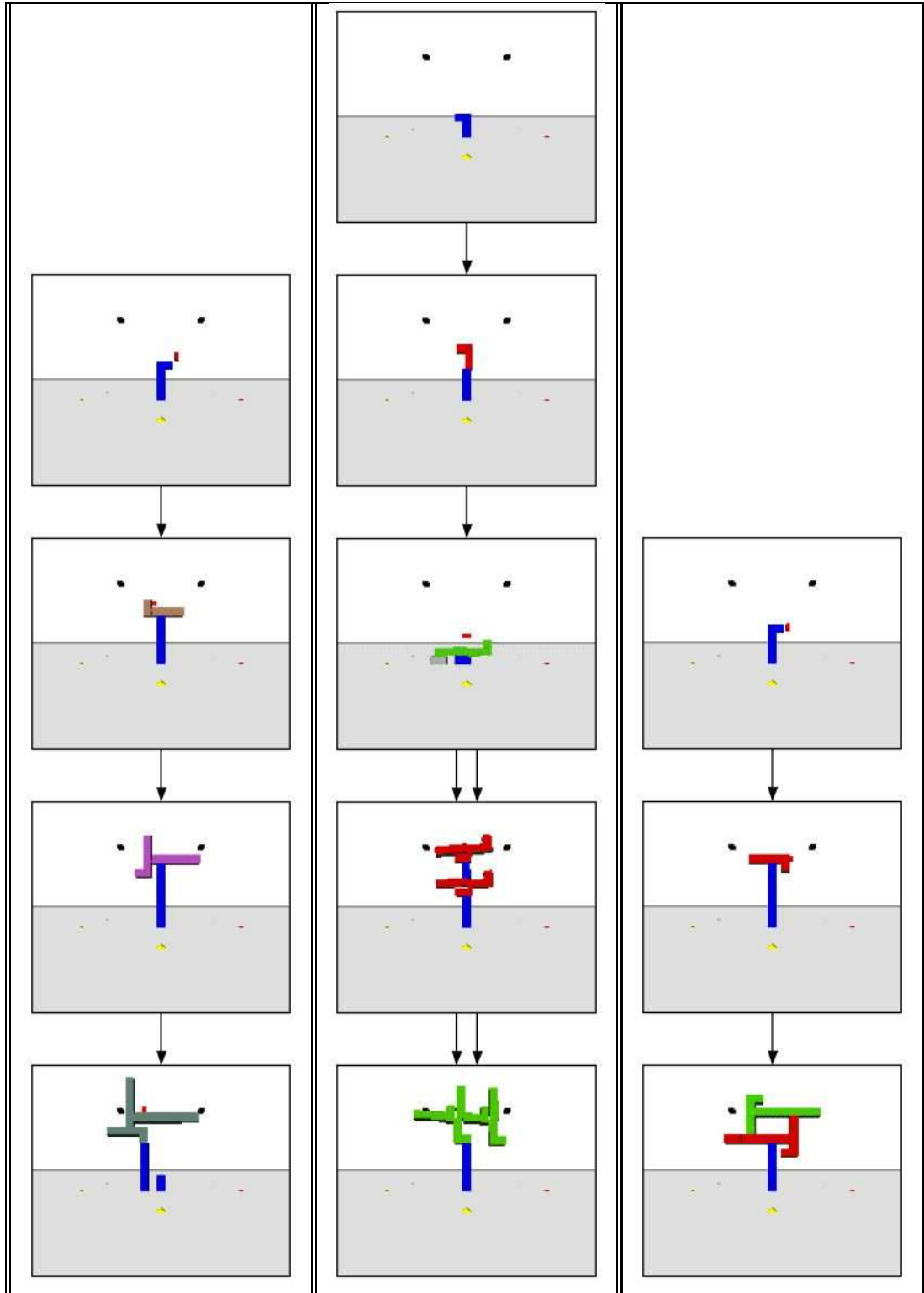


Table A.1: Example Hierarchical Assemblies of Modules at 0.1% noise

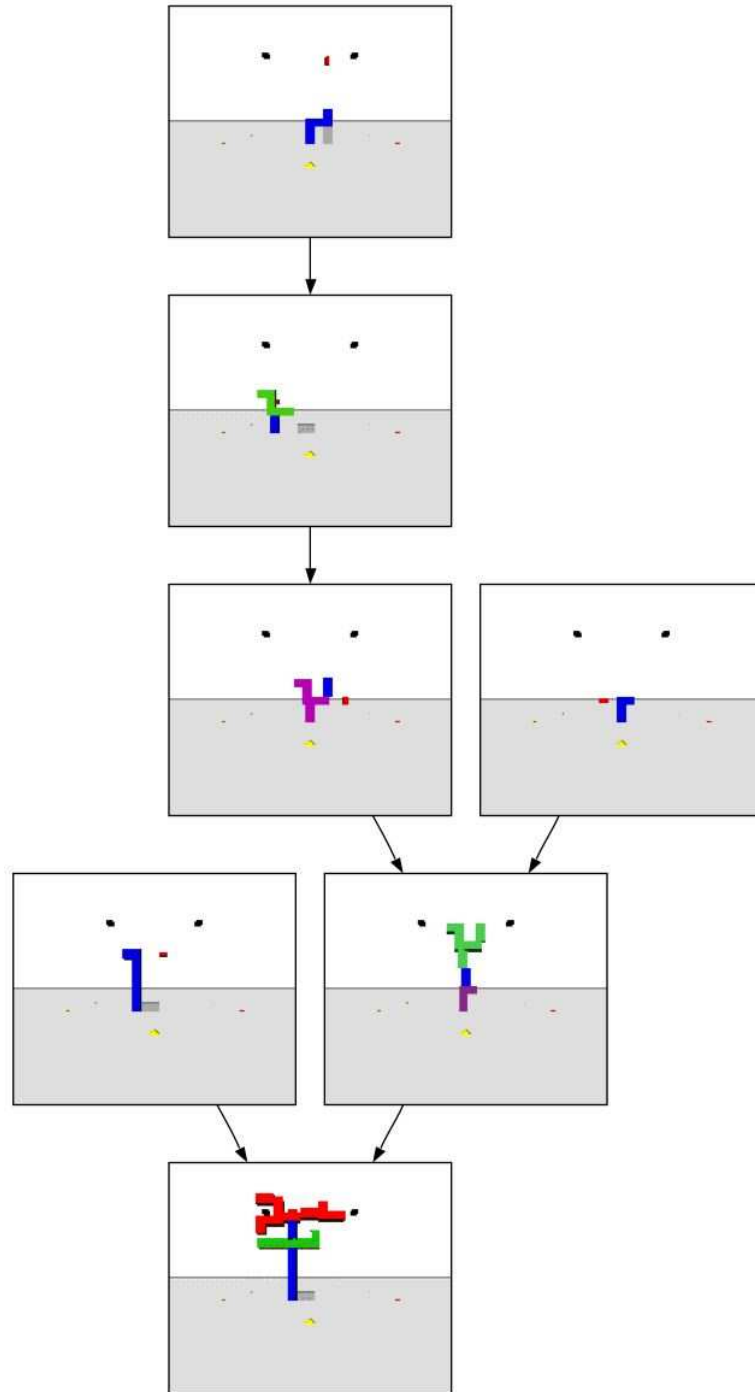


Figure A.2: An Example Hierarchical Assembly of Modules at 0.1% noise

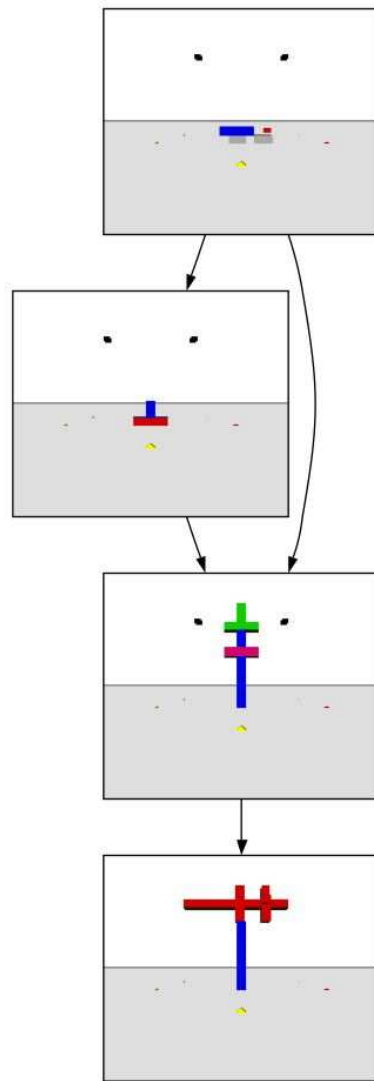


Figure A.3: An Example Hierarchical Assembly of Modules at 0.1% noise

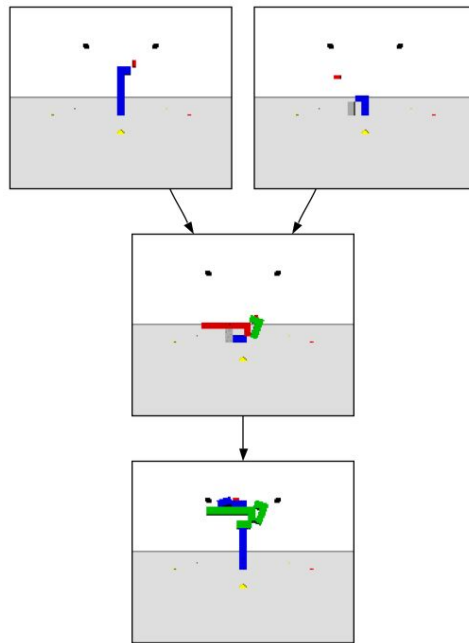


Figure A.4: An Example Hierarchical Assembly of Modules at 0.1% noise

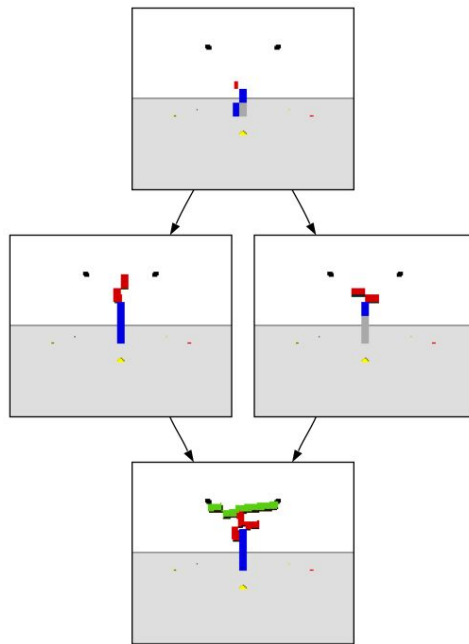


Figure A.5: An Example Hierarchical Assembly of Modules at 0.1% noise

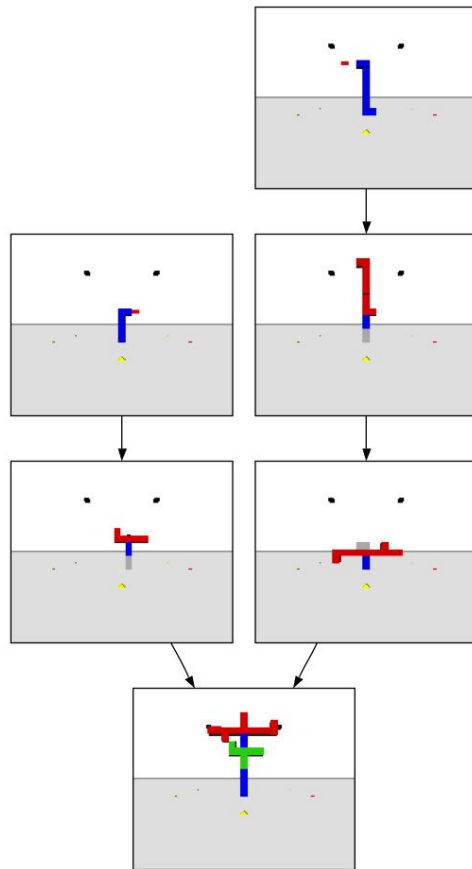


Figure A.6: An Example Hierarchical Assembly of Modules at 1.0% noise

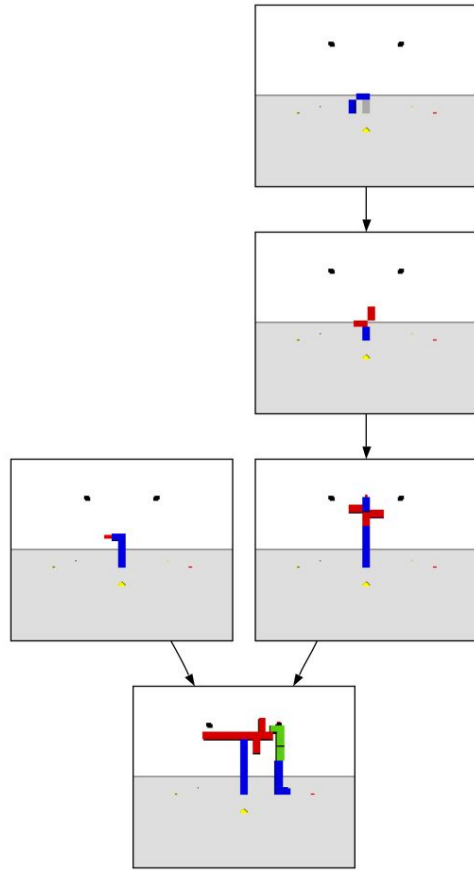


Figure A.7: An Example Hierarchical Assembly of Modules at 1.0% noise

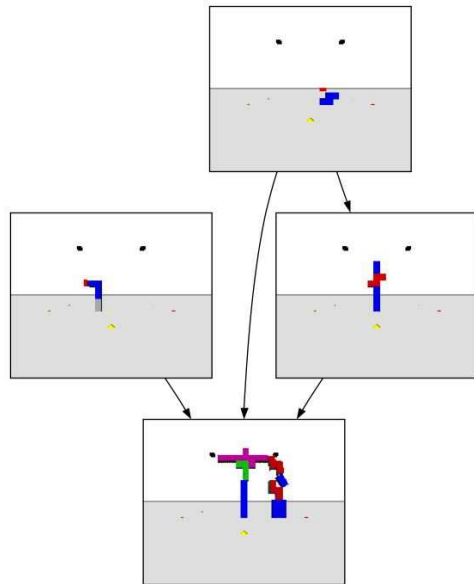


Figure A.8: An Example Hierarchical Assembly of Modules at 1.0% noise

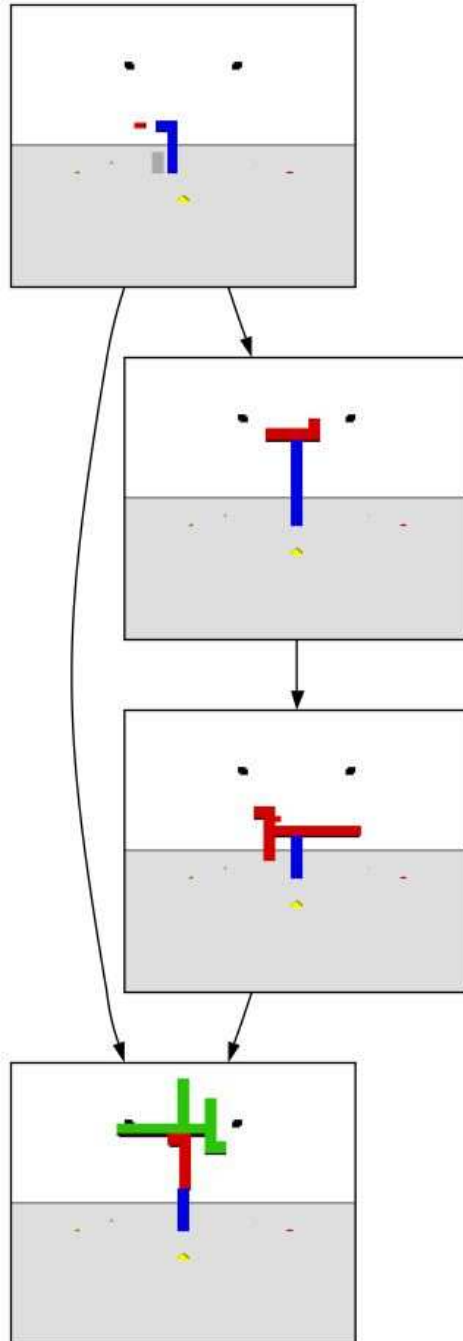


Figure A.9: An Example Hierarchical Assembly of Modules at 1.0% noise

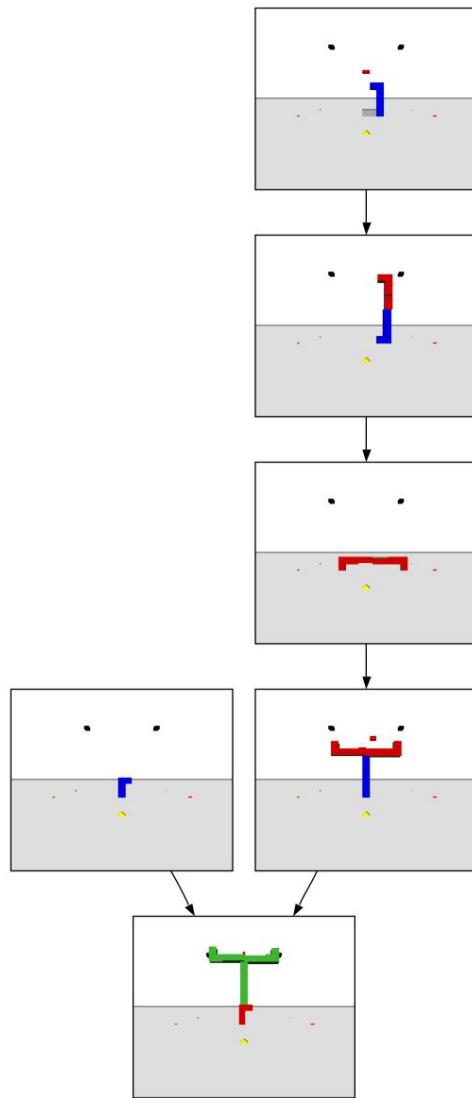


Figure A.10: An Example Hierarchical Assembly of Modules at 1.0% noise

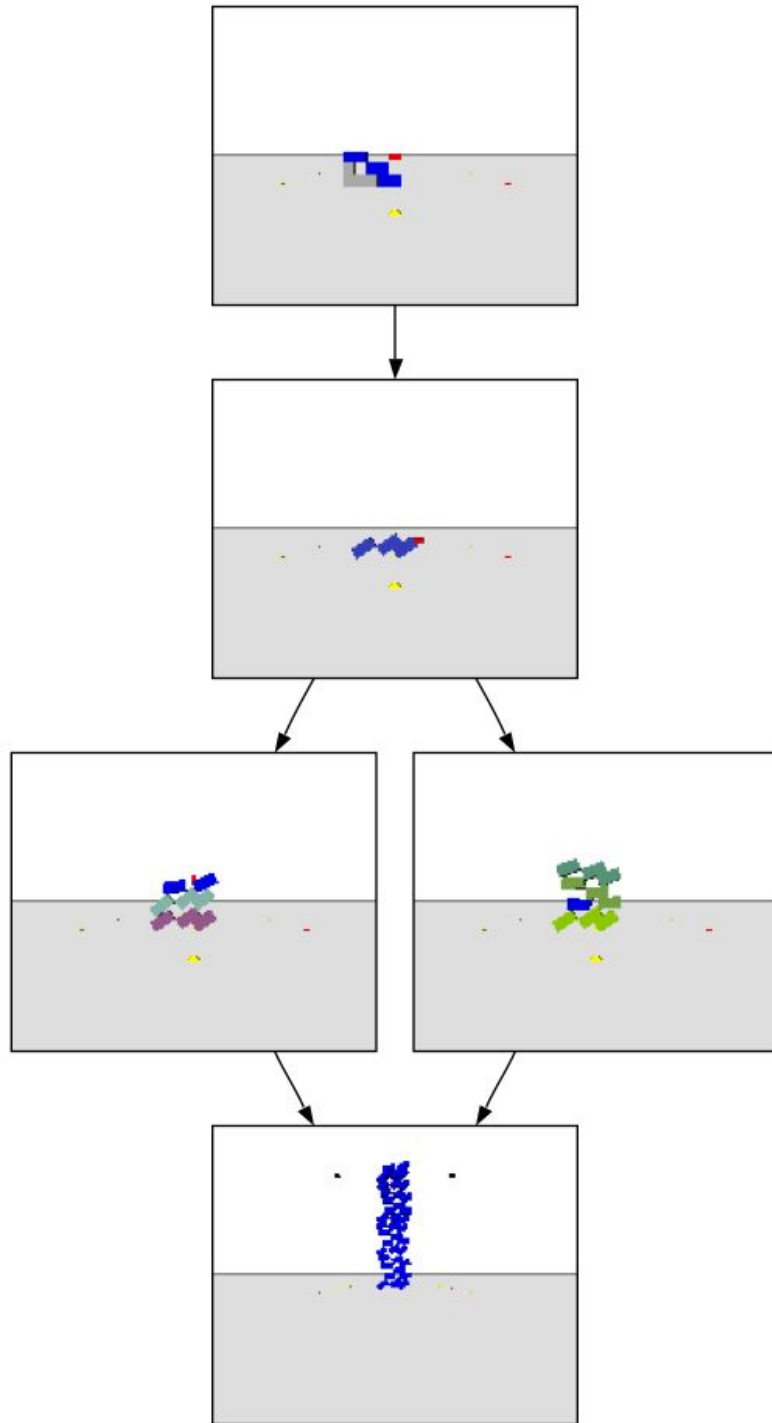


Figure A.11: An Example Hierarchical Assembly of Modules at 5.0% noise

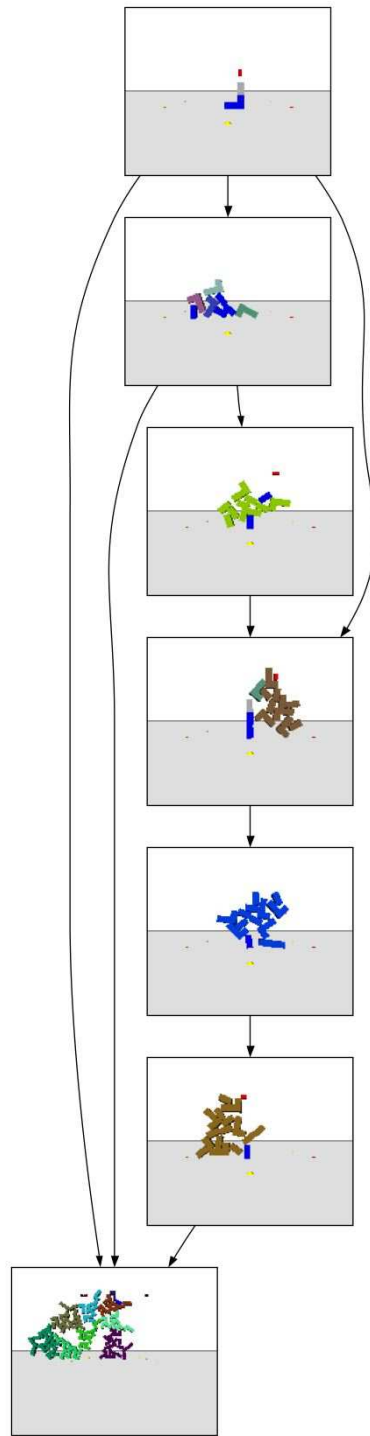


Figure A.12: An Example Hierarchical Assembly of Modules at 5.0% noise

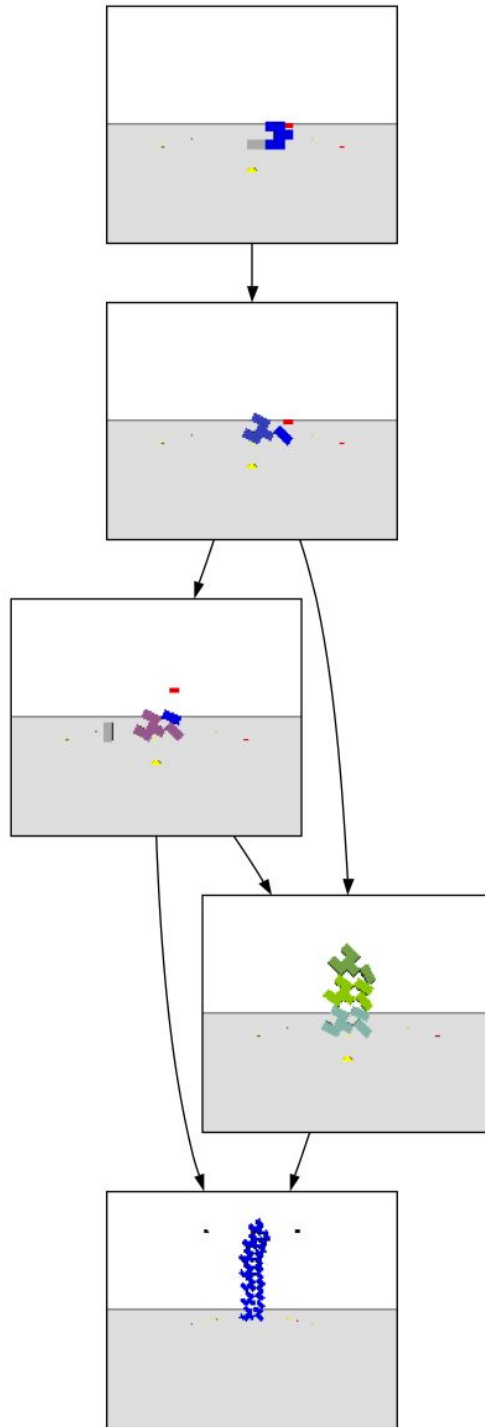


Figure A.13: An Example Hierarchical Assembly of Modules at 5.0% noise

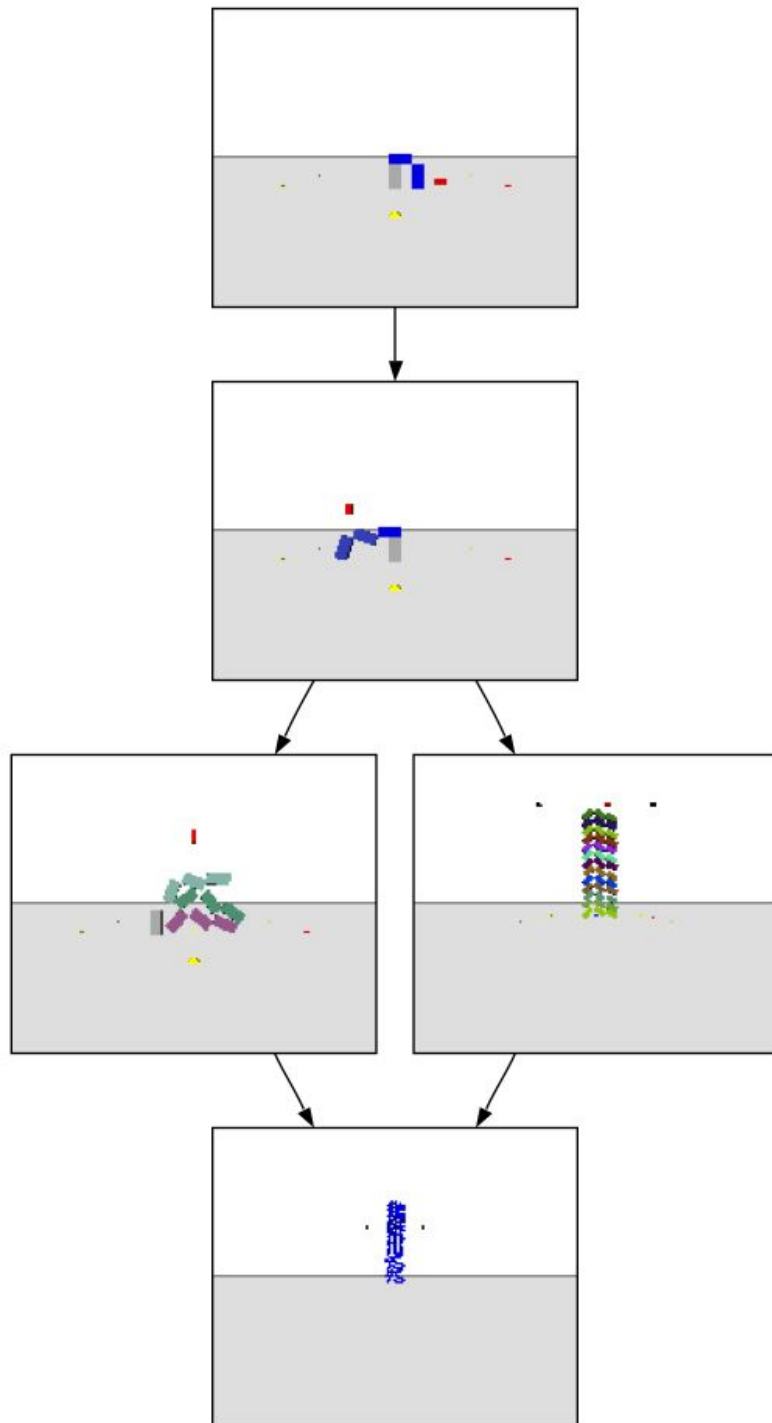


Figure A.14: An Example Hierarchical Assembly of Modules at 5.0% noise

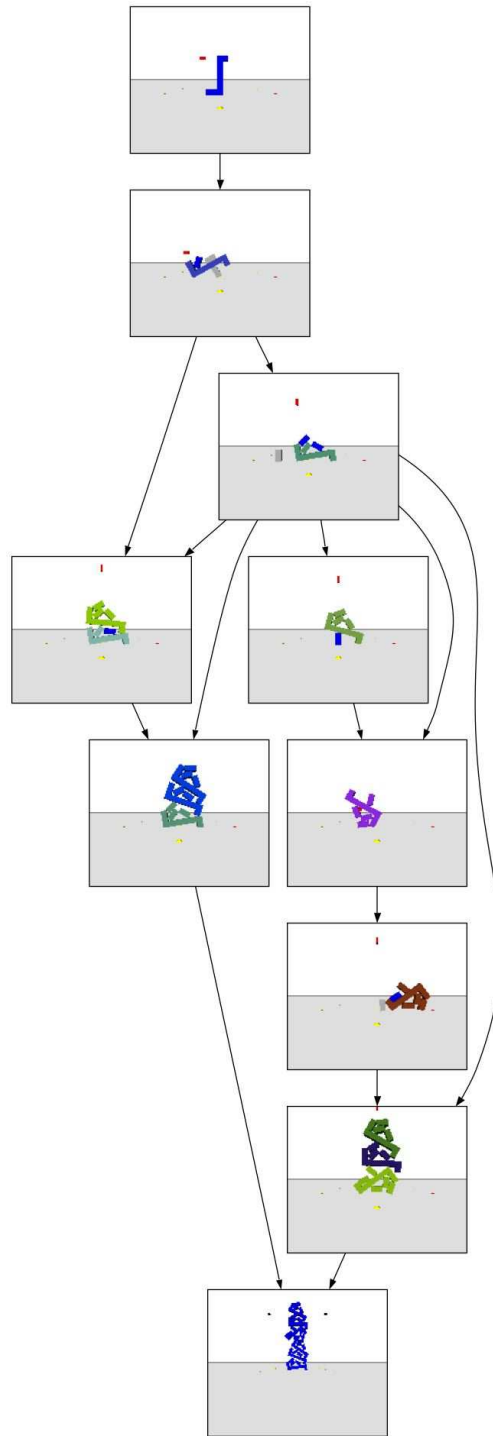


Figure A.15: An Example Hierarchical Assembly of Modules at 5.0% noise

Bibliography

- [1] M. Abrantes and S. Hill. Identifying and explaining infeasible assembly operations. In *International Conference on Manufacturing Automation*, pages 612–617, 1997.
- [2] S. H. Al-Sakran, J. R. Koza, and L. W. Jones. Automated re-invention of a previously patented optical lens system using genetic programming. In M. Keijzer, A. Tettamanzi, P. Collet, J. I. van Hemert, and M. Tomassini, editors, *Proceedings of the 8th European Conference on Genetic Programming*, volume 3447 of *Lecture Notes in Computer Science*, pages 25–37, Lausanne, Switzerland, 2005. Springer.
- [3] P. Angeline and J. Pollack. Competitive environments evolve better solutions for complex tasks. In S. Forrest, editor, *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 264–270. Morgan Kaufmann, 1993.
- [4] P. J. Angeline and J. B. Pollack. Coevolving high-level representations. In C. G. Langton, editor, *Artificial Life III*, volume XVII, pages 55–71, Reading, MA, 15-19 1992 1994. Addison-Wesley.
- [5] P. Bentley. *Modern Heuristic Search Methods*, chapter The Evolution of Solid Object Designs using Genetic Algorithms, pages 199–213. John Wiley and Sons LTD, 1996.
- [6] P. Bentley and P. Wakefield. Conceptual evolutionary design by genetic algorithms. *Engineering Design and Automation Journal*, 3(2):119–131, 1997.
- [7] P. J. Bentley, editor. *Evolutionary Design by Computers*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999.
- [8] H.-G. Beyer and H.-P. Schwefel. Evolution strategies, a comprehensive introduction. *Natural Computing*, 1(1):3–52, 2002.
- [9] E. Bonabeau, S. Guerin, D. Snyers, P. Kuntz, and G. Theraulaz. Three-dimensional architectures grown by simple 'stigmergic' agents. In *Biosystems 56*. Elsevier Science Publishers B. V., 2000.

- [10] J. Bongard and R. Pfeifer. *Morpho-functional Machines: The New Species (Designing Embodied Intelligence)*, chapter Evolving complete agents using artificial ontogeny, pages 237–258. Springer-Verlag, Berlin, 2003.
- [11] J. C. Bongard. Evolving modular genetic regulatory networks. In *Proceedings of the 2002 IEEE Conference on Evolutionary Computation (CEC2002)*, pages 1872–1877, Piscataway, NJ, 2002. IEEE Press.
- [12] J. C. Bongard and H. Lipson. Once More Unto the Breach: Automated Tuning of Robot Simulation using an Inverse Evolutionary Algorithm. In *Proceedings of the Ninth Int. Conference on Artificial Life (ALIFE IX)*, pages 57–62, 2004.
- [13] J. C. Bongard and R. Pfeifer. Repeated structure and dissociation of genotypic and phenotypic complexity in artificial ontogeny. In L. Spector et al., editors, *Proceedings of the 2001 Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 829–836, San Francisco, California, USA, 2001. Morgan Kaufmann.
- [14] R. A. Brooks. *Toward a Practice of Autonomous Systems*, chapter Artificial Life and Real Robots, pages 3–10. MIT Press, 1992.
- [15] T. Broughton, P. Coates, and H. Jackson. *Evolutionary Design by Computers*, chapter Exploring 3D Design Worlds using Lindenmayer Systems and Genetic Programming, pages 323–341. Morgan Kaufmann, 1999.
- [16] J. Canny, J. Risner, and V. Subramanian. Flexonics. In *Proceedings of the Fifth International Workshop on Algorithmic Foundations of Robotics*, Nice, France, 2002.
- [17] D. Cliff and G. F. Miller. Co-evolution of pursuit and evasion II: Simulation methods and results. In P. Maes, M. J. Mataric, J.-A. Meyer, J. B. Pollack, and S. W. Wilson, editors, *From animals to animats 4*, pages 506–515, Cambridge, MA, 1996. MIT Press.
- [18] P. Coates and D. Makris. Genetic programming and spatial morphogenesis. In U. of Sussex, editor, *Proceedings of the 1999 Symposium on Creative Evolutionary Systems*, 1999.
- [19] C. A. C. Coello. An updated survey of evolutionary multiobjective optimization techniques: State of the art and future trends. In P. J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzala, editors, *Proceedings of the Congress on Evolutionary Computation*, volume 1, pages 3–13, Mayflower Hotel, Washington D.C., USA, 6-9 1999. IEEE Press.
- [20] H. de Garis. Artificial embryology: The genetic programming of an artificial embryo. In B. Souček, editor, *Dynamic, Genetic, and Chaotic Programming*, pages 373–393. John Wiley, New York, 1992.

- [21] E. D. de Jong. Representation development from pareto-coevolution. In E. Cantu-Paz, J. Foster, K. Deb, D. Lawrence, R. Roy, U.-M. O'Reilly, H.-G. Beyer, R. Standish, G. Kendall, S. Wilson, M. Harman, J. Wegener, D. Dasgupta, M. A. Potter, A. C. S. N. Jonoska, K. A. Dowsland, and J. F. Miller, editors, *Proceedings of the 2003 Genetic and Evolutionary Computation Conference (GECCO-2003)*, pages 265–276, Heidelberg, 2003. Springer-Verlag.
- [22] E. D. de Jong, R. A. Watson, and J. B. Pollack. Reducing bloat and promoting diversity using multi-objective methods. In L. Spector et al., editors, *Proceedings of the 2001 Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 11–18, San Francisco, California, USA, 2001. Morgan Kaufmann.
- [23] F. Dellaert and R. Beer. Toward an evolvable model of development for autonomous agent synthesis. In *Artificial Life IV, Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, Cambridge, MA, 1994. MIT press.
- [24] F. Dellaert and R. D. Beer. A developmental model for the evolution of complete autonomous agents. In P. Maes, M. Mataric, J. Meyer, J. Pollack, and S. Wilson, editors, *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, pages 393–401. MIT Press, 1996.
- [25] P. Eggenberger. Evolving morphologies of simulated 3d organisms based on differential gene expression. In *Proceedings of the 4th European Conference on Artificial Life*. Springer-Verlag, 1997.
- [26] D. Floreano, S. Nolfi, F. Mondada, M. Patel, V. Honavar, and K. Balakrishnan. Co-Evolution and Ontogenetic Change in Competing Robots. In *Advances in the Evolutionary Synthesis of Intelligent Agents*. MIT Press, Cambridge (MA), 2001.
- [27] C. M. Fonseca and P. J. Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1):1–16, 1995.
- [28] D. Frutiger, J. Bongard, and F. Iida. Iterative product engineering: Evolutionary robot design. In P. Bidaud and F. Amar, editors, *Proceedings of the Fifth International Conference on Climbing and Walking Robots*, pages 619–629. Professional Engineering Publishing, 2002.
- [29] P. Funes. *Evolution of Complexity in Real-World Domains*. PhD thesis, Brandeis University, Dept. of Computer Science, Boston, MA, USA, 2001.
- [30] P. Funes and J. Pollack. *Evolutionary Design by Computers*, chapter Computer Evolution of Buildable Objects, pages 387–403. In Bentley [7], 1999.

- [31] P. Funes and J. B. Pollack. Evolutionary body building: Adaptive physical designs for robots. *Artificial Life*, 4(4):337–357, 1998.
- [32] M. Goldwasser, , J. Latombe, and R. Motwani. Complexity measures for assembly sequences. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 1581–1587, Minneapolis, MN, Apr. 1996.
- [33] M. H. Goldwasser and R. Motwani. Complexity measures for assembly sequences. *International Journal of Computational Geometry and Applications*, 9(4/5):371–417, 1999.
- [34] S. J. Gould. *Ontogeny and phylogeny*. Belknap Press, 1985.
- [35] G. Halder, P. Callaerts, and W. J. Gehring. Induction of ectopic eyes by targeted expression of the eyeless gene in drosophila. *Science*, New Series, Vol. 267(5205):1788–1792, March 1995.
- [36] B. Hall. Phylotypic stage or phantom: is there a highly conserved embryonic stage in vertebrates? *Trends Ecol. Evol.*, 12:461–463, 1997.
- [37] W. D. Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. *Phys. D*, 42(1-3):228–234, 1990.
- [38] G. S. Hornby. *Generative Representations for Evolutionary Design Automation*. PhD thesis, Brandeis University, Dept. of Computer Science, Boston, MA, USA, February 2003.
- [39] G. S. Hornby. Measuring, enabling and comparing modularity, regularity and hierarchy in evolutionary design. In H.-G. Beyer et al., editors, *Proceedings of the 2005 Genetic and Evolutionary Computation Conference (GECCO-2005)*, pages 1729–1736. ACM Press, 2005.
- [40] J. Hugues. Evolution of non-deterministic incremental algorithms as a new approach for search in state spaces. In L. J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, 1995.
- [41] C. Jacob. Genetic l-system programming. In *Parallel Problem Solving from Nature (PPSN III)*, Lecture Notes in Computer Science 866, pages 334–343. Springer-Verlag, 1994.
- [42] N. Jakobi, P. Husbands, and I. Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics. In *Proc. of the Third European Conference on Artificial Life (ECAL'95)*, pages 704–720, Granada, Spain, 1995.
- [43] U. Jayaram, Y. Kim, and S. Jayaram. Reorganizing cad assembly models (as-designed) for manufacturing simulations and planning (as-built). *Journal of Computing and Information Science in Engineering*, 4:98–108, 2004.

- [44] L. Jiang-sheng, Y. Ying-xuue, S. Pahlovy, and L. Jian-guang. A novel data decomposition and information translation method from cad system to virtual assembly application. *International Journal of Advanced Manufacturing Technology*, 28(3), 2006.
- [45] L. E. Kavraki, J.-C. Latombe, and R. H. Wilson. On the complexity of assembly partitioning. *Information Processing Letters*, 48(5):229–235, 1993.
- [46] G. J. Kim, S. Lee, and G. A. Bekey. Interleaving assembly planning and design. *IEEE Transactions on Robotics and Automation*, 12(2):246–251, 1996.
- [47] C. Kirschman and C. Jara-Almonte. A parallel slicing algorithm for solid freeform fabrication processes. In *Solid Freeform Fabrication Conference*, pages 26–33, 1992.
- [48] M. Komosinski and S. Ulatowski. Framsticks: Towards a simulation of a nature-like world, creatures and evolution. In *Proceedings of the 5th European Conference on Artificial Life*. Springer-Verlag, 1999.
- [49] J. Kopena and W. C. Regli. Extensible semantics for representing electromechanical assemblies. In *Proceedings of the 2003 ASME Design Engineering Technical Conferences (DETC'03)*, 2003.
- [50] J. R. Koza. *Genetic Programming: on the Programming of Computers by Means of Natural Selection*. MIT Press: Cambridge, MA, 1992.
- [51] J. R. Koza, M. A. Keane, M. J. Streeter, W. Mydlowec, J. Yu, and G. Lanza. *Genetic Programming IV: Routine Human-Competitive Machine Intelligence*. Kluwer Academic Publishers, 2003.
- [52] S. Kumar and P. J. Bentley. *Advances in evolutionary computing: theory and applications*, chapter Computational embryology: past, present and future, pages 461–477. Springer-Verlag New York, Inc., 2003.
- [53] W.-P. Lee, J. Hallam, and H. H. Lund. A hybrid GP/GA approach for co-evolving controllers and robot bodies to achieve fitness-specified tasks. In *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation*, Nagoya, Japan, 20-22 1996. IEEE Press.
- [54] R. Lewontin. *The Triple Helix*. Harvard University Press, 2000.
- [55] R. Lewontin. *Cycles of Contingency*, chapter Gene, Organism and Environment, pages 59–66. In Oyama et al. [69], 2001.
- [56] D. Linden. *Automated Design and Optimization of Wire Antennas using Genetic Algorithms*. PhD thesis, MIT, 1997.

- [57] D. Linden and E. Altshuler. Automating wire antenna design using genetic algorithms. *Microwave Journal*, 39(3):74–86, 1996.
- [58] H. Lipson, J. C. Bongard, and V. Zykov. Evolving dynamic gaits on a physical robot. In K. D. et al., editor, *Proceedings of the 2004 Genetic and Evolutionary Computation Conference (GECCO-2004)*, volume Late Breaking Papers. Springer-Verlag, 2004.
- [59] J. D. Lohn, G. S. Hornby, and D. S. Linden. An Evolved Antenna for Deployment on NASA’s Space Technology 5 Mission. In U.-M. O’Reilly, R. L. Riolo, T. Yu, and B. Worzel, editors, *Genetic Programming Theory and Practice II*. Kluwer, 2005.
- [60] S. Luke and L. Spector. Evolving graphs and networks with edge encoding: Preliminary report. In J. Koza, editor, *Late Breaking Papers at the Genetic Programming 1996 Conference (GP96)*, pages 117–124. Stanford Bookstore, 1996.
- [61] H. Lund, J. Hallam, and W. Lee. Evolving robot morphology. In *Proceedings of IEEE Fourth International Conference on Evolutionary Computation*. IEEE Press, 1997.
- [62] S. W. Mahfoud. *Niching methods for genetic algorithms*. PhD thesis, University of Illinois at Urbana-Champaign, Urbana, IL, USA, 1995.
- [63] E. Malone and H. Lipson. Functional freeform fabrication for physical artificial life. In *Proceedings of the Ninth International Conference on Artificial Life (ALIFE IX)*, pages 100–105, 2004.
- [64] P. Massey, J. A. Clark, and S. Stepney. Evolution of a human-competitive Quantum Fourier Transform algorithm using genetic programming. In H.-G. Beyer et al., editors, *Proceedings of the 2005 Genetic and Evolutionary Computation Conference (GECCO-2005)*, pages 1657–1664. ACM Press, 2005.
- [65] S. A. McMains. *Geometric Algorithms and Data Representation for Solid Freeform Fabrication*. PhD thesis, University of California, Berkeley, 2000.
- [66] Z. Michalewicz, D. Dasgupta, R. G. L. Riche, and M. Schoenauer. Evolutionary algorithms for constrained engineering problems. *Computers and Industrial Engineering Journal*, 30(2):851–870, 1996.
- [67] M. Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, 1996.
- [68] S. Nolfi, D. Floreano, O. Miglino, and F. Mondada. How to Evolve Autonomous Robots: Different Approaches in Evolutionary Robotics. In *4th International Workshop on Artificial Life*, 1994. R. A. Brooks and P. Maes (eds.).

- [69] S. Oyama, P. E. Griffiths, and R. D. Gray, editors. *Cycles of Contingency*. MIT Press, 2001.
- [70] A. Pamecha, C. Chiang, D. Stein, and G. Chirikjian. Design and implementation of metamorphic robots. In *Proceedings of the 1996 ASME Design Engineering Technical Conference and Computers in Engineering Conference*, 1996.
- [71] G. B. Parker, A. S. Anev, and D. Duzevik. Evolving towers in a 3-dimensional simulated environment. In R. Sarker, R. Reynolds, H. Abbass, K. Tan, B. McKay, and T. G. D. Essam, editors, *Proceedings of the 2003 Congress on Evolutionary Computation (CEC 2003)*, pages 1137–1144. IEEE Press, 2003.
- [72] M. Peysakhov and W. C. Regli. Using assembly representations to enable evolutionary design of lego structures. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 17:155–68, 2003.
- [73] J. Pollack, H. Lipson, P. Funes, S. Ficici, and G. Hornby. Coevolutionary robotics. In *EH '99: Proceedings of the 1st NASA/DOD workshop on Evolvable Hardware*, page 208, Washington, DC, USA, 1999. IEEE Computer Society.
- [74] J. B. Pollack. Mindless intelligence. *IEEE Intelligent Systems*, 21(3), 2006.
- [75] J. B. Pollack, H. Lipson, G. Hornby, and P. Funes. Three generations of automatically designed robots. *Artificial Life*, 7(3):215–223, Summer 2001.
- [76] S. Preble, H. Lipson, and M. Lipson. Two-dimensional photonic crystals designed by evolutionary algorithms. *Applied Physics Letters*, 86, 2005.
- [77] J. Reisinger. An overview of modularity in artificial evolutionary systems. In *Proceedings of Parallel Problem Solving From Nature 2004*, 2004.
- [78] M. K. Richardson. Heterochrony and the phylotypic period. *Dev. Biol.*, 172:412–421, 1995.
- [79] M. K. Richardson, J. Hanken, M. I. Gooneratne, C. Pieau, A. Reynaud, L. Selwood, and G. M. Wright. There is no highly conserved embryonic stage in the vertebrates: implications for current theories of evolution and development. *Anat. Embryol.*, 196:91–106, 1997.
- [80] M. Ridley. *Evolution*. Blackwell Publishers, 2003.
- [81] J. Rieffel and J. Pollack. The Emergence of Ontogenic Scaffolding in a Stochastic Development Environment. In K. D. et al., editor, *Proceedings of the 2004 Genetic and Evolutionary Computation Conference (GECCO-2004)*, pages 804–815, Heidelberg, 2004. Springer-Verlag.

- [82] J. Rieffel and J. Pollack. An endosymbiotic model for modular acquisition in stochastic developmental systems. In *Proceedings of the Tenth International Conference on the Simulation and Synthesis of Living Systems (ALIFE X)*, 2006.
- [83] J. Rieffel and J. B. Pollack. Artificial ontogenies for real world design and assembly. In M. B. et al., editor, *Ninth International Conference on the Simulation and Synthesis of Living Systems (ALIFE9) Workshop: Self-Organization and Development in Artificial and Natural Systems (SODANS)*, pages 37–41. MIT Press, 2004.
- [84] J. Rieffel and J. B. Pollack. Automated assembly as situated development: Using artificial ontogenies to evolve buildable 3-d objects. In H.-G. Beyer et al., editors, *Proceedings of the 2005 Genetic and Evolutionary Computation Conference (GECCO-2005)*, pages 99–106. ACM Press, 2005.
- [85] E. M. A. Ronald, M. Sipper, and M. Capcarrère. Design, observation, surprise! a test of emergence. *Artificial Life*, 5(3):225–239, 1999.
- [86] J. P. Rosca and D. H. Ballard. Discovery of subroutines in genetic programming. In P. J. Angeline and K. E. Kinneer, Jr., editors, *Advances in Genetic Programming 2*, pages 177–202. MIT Press, Cambridge, MA, USA, 1996.
- [87] M. Rosenman. The generation of form using an evolutionary approach. In D. Dasgupta and Z. Michalewicz, editors, *Evolutionary Algorithms in Engineering Applications*, pages 69–86. Springer-Verlag, 1997.
- [88] I. Ross, U. O’Reilly, and P. Testa. Emergent design: Artificial life for architecture design. In *Artificial Life 7 Proceedings*, 2000.
- [89] D. Rus, Z. Butler, K. Kotay, and M. Vona. Self-reconfiguring robots. *Communications of the ACM*, 45(3):39–45, 2002.
- [90] H. A. Simon. The architecture of complexity. *Proceedings of the American Philosophical Society*, 106(6):467–482, December 1962.
- [91] K. Sims. Evolving 3d morphology and behavior by competition. In R. B. . P. Maes, editor, *Artificial Life IV Proceedings*, pages 28–39. MIT Press, 1994.
- [92] K. Sims. Evolving virtual creatures. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 15–22. ACM Press, 1994.
- [93] J. Slack, P. Holland, and C. Graham. The zootype and the phylotypic stage. *Nature*, 361:490–492, 1993.

- [94] K. O. Stanley and R. Miikkulainen. A taxonomy for artificial embryogeny. *Artificial Life*, 9(2):93–130, 2002.
- [95] A. Thompson. Silicon evolution. In J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 444–452, Stanford University, CA, USA, 28–31 1996. MIT Press.
- [96] M. Toussaint. Demonstrating the evolution of complex genetic representations: An evolution of artificial plants. In E. Cantu-Paz, J. Foster, K. Deb, D. Lawrence, R. Roy, U.-M. O’Reilly, H.-G. Beyer, R. Standish, G. Kendall, S. Wilson, M. Harman, J. Wegener, D. Dasgupta, M. A. Potter, A. C. S. N. Jonoska, K. A. Dowland, and J. F. Miller, editors, *Proceedings of the 2003 Genetic and Evolutionary Computation Conference (GECCO-2003)*. Springer-Verlag, 2003.
- [97] S. Viswanathan. How artificial ontogenies can retard evolution. In H.-G. Beyer et al., editors, *Proceedings of the 2005 Genetic and Evolutionary Computation Conference (GECCO-2005)*. ACM Press, 2005.
- [98] S. Viswanathan and J. Pollack. On the robustness achievable with stochastic development processes. In *The 2005 NASA/DoD Conference on Evolvable Hardware*. IEEE Press, 2005.
- [99] S. Viswanathan and J. B. Pollack. Towards an evolutionary-developmental approach for real-world substrates. In M. B. et al., editor, *Ninth International Conference on the Simulation and Synthesis of Living Systems (ALIFE9)*, pages 45–41. MIT Press, 2004.
- [100] G. P. Wagner and L. Altenberg. Complex adaptations and the evolution of evolvability. *Evolution*, 50(3):967–976, 1996.
- [101] R. Watson. *Compositional Evolution: Interdisciplinary Investigations in Evolvability, Modularity, and Symbiosis*. PhD thesis, Brandeis University, Dept. of Computer Science, Boston, MA, USA, 2002.
- [102] R. A. Watson, S. G. Ficici, and J. B. Pollack. Embodied evolution: Embodying an evolutionary algorithm in a population of robots. In P. J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzal, editors, *Proceedings of the Congress on Evolutionary Computation*, volume 1, pages 335–342, Mayflower Hotel, Washington D.C., USA, 6-9 1999. IEEE Press.
- [103] G. M. Whitesides and B. Grzybowski. Self-assembly at all scales (viewpoint). *Science*, 295(5564):2418–2422, March 2002.

- [104] R. Wilson, L. Kavraki, T. Lozano-Perez, and J. Latombe. Two-handed assembly sequencing. *The International Journal of Robotics Research*, 14(4):335–350, 1995.
- [105] R. H. Wilson. *On Geometric Assembly Planning*. PhD thesis, Stanford University, 1994.
- [106] H. P. Winston. *Artificial Intelligence: Third Edition*, chapter Learning By Analyzing Differences, pages 349–364. Addison-Wesley, Reading MA, 1993.
- [107] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, April 1997.
- [108] L. Wolpert. The evolutionary origin of development: cycles, patterning, privilege, and continuity. *Dev. Suppl*, pages 79–84, 1994.
- [109] J. D. Wolter. On the automatic generation of assembly plans. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 1989.
- [110] A. S. Yilmaz and A. S. Wu. Preservation of genetic redundancy in the existence of developmental error and fitness assignment error. In H.-G. Beyer et al., editors, *Proceedings of the 2005 Genetic and Evolutionary Computation Conference (GECCO-2005)*, pages 1317–1324. ACM Press, 2005.
- [111] M. Yim. *Locomotion With A Unit-Modular Reconfigurable Robot*. PhD thesis, Stanford University, 1995.
- [112] Z.-P. Yin, H. Ding, and Y.-L. Xiong. A virtual prototyping approach to generation and evaluation of mechanical assembly sequences. *Proceedings of the I MECH E Part B Journal of Engineering Manufacture*, 218(1):87–102, 2004.
- [113] V. Zykov, E. Mytilinaios, B. Adams, and H. Lipson. Self-reproducing machines. *Nature*, 435(7038):163–164, 2005.