Automated Assembly as Situated Development:

Using Artificial Ontogenies to Evolve Buildable 3-D Objects

John Rieffel
DEMO Lab, Brandeis University
Waltham, MA
jrieffel@cs.brandeis.edu

Jordan Pollack
DEMO Lab, Brandeis University
Waltham, MA
pollack@cs.brandeis.edu

ABSTRACT

Artificial Ontogenies, which are inspired by biological development, have been used to automatically generate a wide array of novel objects, some of which have recently been manufactured in the real world. The majority of these evolved designs have been evaluated in simulation as completed objects, with no attention paid to how, or even if, they can be realistically built. As a consequence, significant human effort is required to transfer the designs to the real world. One way to reduce human involvement in this regard is to evolve *how* to build rather than *what* to build, by using *prescriptive* rather than descriptive representations. In the context of Artificial Ontogenies, this requires what we call *Situated Development*, in which an object's development occurs in the same environment as its final evaluation. Not only does this produce sufficient information on how to build evolved designs, but it also ensures that only buildable designs are evolved. In this paper we explore the consequences of Situated Development, and demonstrate how it can be incorporated into Artificial Ontogenies in order to generate *buildable* objects, which can be sequentially assembled in a realistic 3-D physics environment.

Categories and Subject Descriptors

I.2.9 [**Artificial Intelligence**]: Robotics

General Terms

Design,Algorithms

Keywords

Artificial Ontogeny, Evolutionary Design, Assembly, Fabrication

1. INTRODUCTION

Evolutionary Algorithms have been used to automatically design a wide array of novel and interesting structures and robots. Few of these evolved designs, however, have been transferred to reality. One significant obstacle in this regard is the amount of human effort required to interpret designs evolved in simulation and then assemble their physical counterparts in the real world. This effort is often compounded by the fact that the evolved designs only specified the appearance of the final object, and carried no information on how (or even *if*) the object could be realistically assembled. To reduce human involvement, it would be better if Evolutionary Design evolved *how* to build rather than simply *what* to build, and if it only produced designs where were known to be *buildable*. Essential to this, as we discuss, are two things: first is to use *prescriptive*, rather than *descriptive* representations of designed objects, and second is to simulate and evaluate objects as they are assembled, not just when they are completed.

A simple form of prescriptive representation is the *assembly plan* - a linear set of sequential instructions to an assembly mechanism. The process by which a manufacturing mechanism transforms an assembly plan into a physical structure can bear considerable similarity to the biological processes of development and growth. Artificial Ontogenies, a form of Evolutionary Design inspired by such biological mechanisms, are therefore a natural choice for exploring issues of assembly and buildability [1].

In fact, Artificial Ontogenies have been used to produce quite a few recent evolved designs [16]. However, the majority of these systems only used Artificial Ontogenies as an intermediate step towards producing a final, descriptive representation, and took the actual process of development for granted - either by treating it as an instantaneous process which could produce a final structure *in toto*, or by allowing development occur *in utero*, that is, in some environment that was considerably simpler than the one in which the completed design was evaluated.

An alternative is what we call *Situated Development* - Artificial Ontogenies in which an object's development occurs in the same environment that the fully grown phenotype is evaluated in. The best recent example of Situated Development is Bongard's Gene-Regulatory Networks [1], which slowly "grew" a robotic morphology piece-by-piece in a realistic physics environment. Bongard used biological growth as a model, whereas our interest is in a development process which is more analogous to mechanical assembly. Our conjecture is that evolving prescriptive representations in a

[1]Since we are using biological growth as a metaphor for physical assembly, we use the terms *development* and *assembly* interchangeably.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
GECCO'05, June 25–29, 2005, Washington, DC, USA.
Copyright 2005 ACM 1-59593-010-8/05/0006 ...$5.00.

Situated Development environment should ensure that sufficient assembly information is acquired, and that only buildable designs are generated.

In this paper we briefly review some difficulties encountered in transferring evolved designs into the real world. We then discuss *buildability*, and explore the consequences of adding it as a constraint to evolutionary design. Finally we demonstrate the use of Situated Development within Artificial Ontogenies to generate objects which are buildable in a realistic 3-D environment, and show how Situated Development can lead to novel and unanticipated modes of assembly.

1.1 Evolutionary Design

Using Evolutionary Algorithms to automatically generate novel designs dates back at least as far as Karl Sims' seminal work on virtual creatures [17]. Since then, Evolutionary Design has generated numerous novel and interesting objects including LEGO structures [5], robots [14] [11] [19] , virtual plants [18] and antennas for satellites [12].

Thanks in large part to automated manufacture and rapid prototyping machines, evolved designs are increasingly being produced in the physical world. The two most notable examples perhaps being Pollack *et al*'s evolved robots, and Lohn *et al*'s [12] evolved antenna, which is due to be launched into space aboard a Low Earth Orbit satellite. Despite being designed with minimal human input, all of these evolved designs required significant human effort and insight to transfer them from simulation to the real world. Many of Funes' LEGO structures [5], for instance, had to be assembled by hand on a horizontal surface, and then slowly tilted into the vertical plane. Lohn *et al*'s evolved antennas, had to be carefully hand-bent and soldered into shape, with extra care taken to preserve the precise measurements specified by the design. ([16] provides a more comprehensive review of evolved designs which have been built in the real world).

In most cases, the need for human involvement in transferring these designs was due to the fact that the final evolved design was purely *descriptive*. Descriptive representations, such as blueprints, specify what an object should look like, but contain no information about how to actually build it, just as looking at a photograph of a meal provides little insight into how to prepare it. Consequently, human involvement is required in figuring out how to assemble an object matching the evolved description. Thus, while the evolution of descriptive representations removes human effort from the design task, it fails to remove human effort from the assembly task - and may in fact increase it.

While the process of determining an assembly sequence for a given blueprint may come readily to humans, it is much harder to solve computationally. In fact, there is an entire field of engineering, *Assembly Sequence Planning*, which studies this very task. The complexity of Assembly Sequence Planning has been well analyzed, and it has been proven to be NP-complete in the general case [10]. Computational approaches to sequence planning for a given object usually involve the much easier inverse problem of *disassembly planning* - that is, exploring all the ways of removing subcomponents of a complete object one at a time [6]. Doing so, however, makes the critical assumption that every stage of assembly is both *reversible* and *symmetric* [6]. Anyone who has taken apart a home appliance and then put it back together, only to be left with a solitary remaining mysterious screw, knows that assembly and disassembly are rarely symmetric, reversible processes in the real world.

The closest that the field of Evolutionary Design has come to the goal of automating both the design and the assembly of evolved objects is Hornby's evolved tables [7], in which his voxel-based representations were converted automatically converted into a CAD format which was then interpreted by a rapid-prototyping machine, which in turn printed the object out of plastic.

This approach doesn't entirely eliminate human involvement though: human effort was necessary to create the means by which the rapid prototyping machine could translate a CAD file into a series of commands to the printer. Moreover if the machine were tilted at a slight angle, the same series of commands would produce a drastically different result, and a new CAD-to-printer translation system would be needed. We are interested, therefore, in more dynamic and adaptive methods.

This raises the question: rather than rely on some brittle translation between descriptive representation and assembly process, why not evolve rapid prototyping machine instructions directly? Such evolution of *prescriptive* representations allows us to design the *process* of assembly rather than the subject of assembly. One simple type of prescriptive representation is the *assembly plan*. Unlike blueprints, assembly plans provide step-by-step instructions on *how* to build something. In its simplest form, an assembly plan is a sequential, ballistic [2] , set of instructions to an assembly mechanism, which when executed results in the construction of an object. Hornby's tables and robots [8] for instance, were generated by a linear set of instructions to a LOGO-like turtle which "drew" OpenGL voxels in 3-D space.

1.2 Situated Development

We would like to suggest that producing representations which can describe *buildable* objects, that is, objects producible by an actual physical assembly process, requires that objects be subject to physics over the entire course of their assembly. When this occurs in the context of an Artificial Ontogeny, we call this process *Situated Development*.

In his PhD thesis, which produced some of the first real world objects based on evolved designs [5], Funes lists *buildability* as a necessary property of evolved structures. In his words, buildability means that "results should be convertible from a simulation to a real object". We would like to elaborate on that definition, in the context of automated assembly and Artificial Ontogenies. In that sense, to be buildable, an object should be able to be produced by a situated, sequential, and stable process. We describe those terms as follows:

- *Situatedness*: Development never occurs in complete isolation. While developmental environments (e.g. a mammalian womb or chip manufacturing clean room) may serve to attenuate harsh external environmental factors, they are always subject to the stochastic nature of physics and chemistry. In the extreme case, an object is built in the exact same environment in which it is evaluated when complete. In such *fully situated* development, each stage of assembly is subject to the same physics and environment as the final object. One of the few examples of situated growth in

[2] That is to say, without any ability to test intermediate results, or alter their behavior mid-assembly

evolutionary design is Bongard's work [1], which uses a Gene-Regulatory Network-based Artificial Ontogeny to grow agents starting from a single structural unit.

- *Sequentiality*: Unlike voxel-based designs, which can simply appear *in toto*, the assembly of physical objects is always, on some level, of a sequential nature. Even the most hierarchical and parallel construction technique cannot ignore the issue of *precedence*. For instance, placement of later structural components is predicated on the prior placement of some supporting structure. Hornby's tables [8], for instance, were specified by a linear set of instructions to a "turtle" which drew OpenGL voxels.

- *Stability*: Since every stage of assembly is subject to physics, components can interact and interfere with one another when they are placed. Issues of balance, torque, and material strength therefore come into play. Often, mechanisms such as scaffolding must be utilized to ensure that fragile components of the system remain stable and stationary. Our recent work [15], for instance, explored how assembly plans could reliably build stable structures in a stochastic assembly environment.

When combined, Situatedness, Sequentiality, and Stability lead to the primary difficulty in imposing buildability as a constraint: sensitivity to error. Since the result of every action in a sequential assembly is predicated on the result of the previous action, any error in the early stages of assembly can have a compounding effect upon the subsequent stages.

Recent research has begun to explore how simulating assembly can affect Evolutionary Design. In particular, in earlier work of ours [15, 16] we used a two-dimensional development environment, with a"tetris"-like physics to evolve assembly plans capable of building a predefined goal arch. Their evolved assembly plans were able to reliably build the goal arch despite a stochastic noise model, which could knock over structural elements. The key to the assembly plans' robustness was the use of what they called *ontogenic scaffolding* - the placement of intermediate structural elements that were removed once the structure was completed.

While a useful proof of concept, our earlier work relied upon an extremely simple discrete grid-based environment. For instance, once a structural element was placed it remained "glued" in place, and couldn't be perturbed or knocked over by later actions of the assembly. In this work, we are interested in a richer and more realistic simulation of assembly, in which a continuous environment allows for complex interactions between structural elements (involving mass, torque, momentum, etc), which play a larger role in development.

2. A FRAMEWORK FOR EVOLVING BUILDABLE 3-D OBJECTS

Our Situated Development environment is based upon the Open Dynamics Engine (ODE) [3] the widely used open-source physics engine, which provides high-performance simulations of 3D rigid body dynamics.

Assembly is performed by a LOGO-like turtle, acting as a print head, capable of movement in the X-Z plane, and of depositing 2x1x1 bricks in the environment. When strung into a sequence, commands to the turtle (move, rotate, put brick, take brick) form an *assembly plan*. Commands which would cause the turtle to move outside the target area, or place a brick where a brick already exists, are ignored. The speed of an ODE simulation is heavily influenced by the number of objects being simulated. Consequently, the maximum number of objects placed by any assembly plan was limited to 25.

Table 2: Parameterized Assembly Instructions

Instruction	Parameters
(M)ove	+2, +1, -1, -2
(Rotate	+90, -90, +180
(P)ut Brick	(a)head, to (r)ight, to (left), (b)ehind
(P)ut Scaffolding	(a)head, to (r)ight, to (left), (b)ehind
(T)ake Brick	(none)

Since our recent work demonstrated the ability of similar systems to "discover" scaffolding implicitly during the course of evolution [15, 16], we chose to allow for the *explicit* placement of scaffolding. The turtle is capable of placing two kinds of bricks: permanent ones (shown as black in the animation 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 Stratasys [4], which can lay thin water-soluble support structures that are dissolved once manufacture is complete.

Our simulated assembly falls into three stages (Table 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.

2.1 Experiments

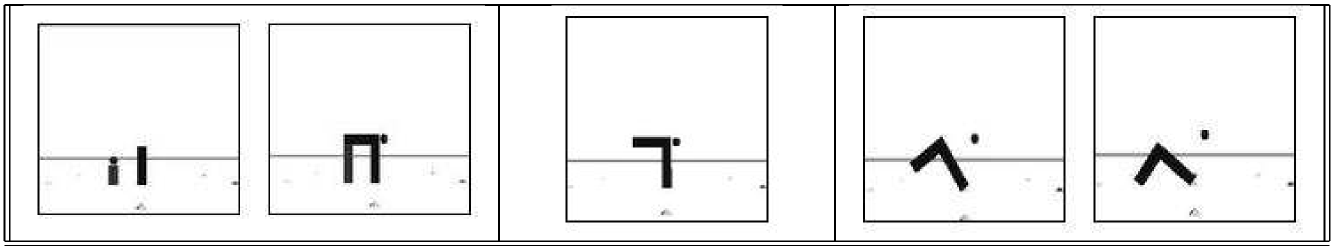
The goal of our experiments was to explore the effects of Situated Development on Evolutionary Design. The genotypes of our Evolutionary Algorithm were assembly plans, consisting of a sequential set of parameterized instructions to the situated development system described above in Section 2. Table 2 lists the instructions used.

Note that the instructions above only allow for *ballistic* assembly - that is, there are no commands which test intermediate results (such as probing for the existence of a brick at a particular location), nor any ability to branch or otherwise alter behavior predicated on such a test. While intermediate tests may ultimately be necessary for the assembly of more complex structures, they have associated costs, in terms of sensors needed to make tests, in terms of complexity of genetic representation (e.g. trees instead of linear sequences), and also in terms of the larger genetic search space caused by an increased number of alleles.

The metaphor of development blurs the traditional GA's notion of phenotype. Instead of appearing *in toto*, the phenotype emerges over the course of development. (Viswanathan refers to this as an *ontogenic trajectory* [20])In this work,

[3] `www.ode.org`

[4] `www.stratasys.com`



Assembly	Glue Bricks,Melt Scaffolding	Settle

Table 1: Assembly has three stages. In the first (Frames 1 and 2), both permanent bricks (shown in black) and temporary bricks (grey) are placed. In the second, adjacent permanent bricks are glued together, and scaffolding is removed (Frame 3). Finally, the remaining structure settles (Frames 4 and 5).

Figure 1: Illustration of Fitness Functions. The structure itself is black, and they gray region is considered "shaded". In the first fitness function, which measures the "coverage" by summing maximum column heights, both the black and grey areas of the figure are counted. The second fitness function, "shading", only measures the gray area.

since our assembly plans are ballistic, and no feedback is provided over the course of assembly, we only treat the final, stable structure as the phenotype, and use this for fitness measurements.

We ran two sets of experiments with related fitness functions. In each case the resulting 3-D structure was "downsampled" into a 2-D bitmap by sampling the central $10\beta \times 10\beta$ region (where β is the width of a brick) in the X-Z plane, at a resolution of $\alpha = 0.2\beta$.

As Figure 1 illustrates, the first fitness function measures the "fill" of the sample region by calculating the sum of the maximum height along each column of the bitmap, regardless of whether a square is occupied by a brick or empty. The maximum possible value was therefore $2500\alpha^2$.

The second fitness function operates similarly, but only measures the total *open* volume beneath a structure, thereby encouraging structures which both maximize height and maximize the number of empty spaces beneath the structure. Specifically, every region (x,z) of the bitmap which is empty and underneath a filled region, was considered shaded area. The optimal value for this fitness function is $2000\alpha^2$:the top row of the target area is 10 squares (5 bricks) wide and 9 squares high (which requires at least 5 bricks to reach, since every brick must take up 2 squares) high. This second fitness function is similar to those used for Funes' LEGO trees [5]. Figure 1 illustrates this both of these fitness functions.

In each case, one of the fitness function above was coupled with assembly plan length, and accumulated mass, as deterrents to bloat [4], as objectives for Multi-Objective Optimization (MOO) [3].

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

In each experiment, the initial population was created with 30 random genotypes, each with a length between 8 and 40 instructions. After each generation was evaluated, the N non-dominated individuals (i.e. pareto front) were 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 [13], with a limit of 3 individuals per bin.

2.2 Results

Tables 3 and 4 contain several representative results of our experiments. More detailed and colorful images, as well as full animations of assembly, are available on the author's web page.

Table 3 shows representative structures and fitness values generated by the "fill" 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 similarly shows representative structures generated by the "shadow" fitness function. Unlike the earlier structures generated by the "fill" fitness function, structures are only rewarded for empty volume, and so there is considerably more incentive to remove extraneous bricks and to use temporary scaffolding elements. These structures therefore tend to be sparser and more "open" in the middle.

Figure 2 contains frames captured from the assembly of the tree in the top row of Table 4. After building the central trunk, the assembly plan lays both scaffolding and permanent bricks for the rightmost branch, and then repeats the process on the left hand side. The scaffolding for the branches not only ensure that the branch bricks remain in place, but also keeps the structure stable as subsequent bricks are placed. Once all of the permanent bricks are glued together, and the scaffolding is removed, the tree is balanced and remains stable.

Figure 3 shows the assembly of the arch in the third row of Table 4. 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.

3. DISCUSSION

As suggested by the sample of structures in Tables 3 and 4, the majority of evolved assembly plans tend to build arches, even though a "T" shape of equivalent height would shade more area. We conjecture that the relative lack of tree shapes is due to the fact that they are less stable than arches when perturbed. Consider the tree-shape in the first row of Table 4: any addition or removal of a brick along the top edge would unbalance the structure, and cause it to topple once scaffolding was removed. The arch in the bottom row, by contrast, would remain stable with any added bricks along the top surface, and would probably remain stable if any of the middle bricks of the top surface were removed (the two sides would fall together and meet, producing a structure which shades only slightly less area). Since they are more stable in the face of genotypic perturbation , arches therefore represent a larger and more robust (although slightly less optimal) evolutionary target than Ts.

3.1 Meta Assembly

Figures 4 and 5 show an interesting phenomenon which arises as a consequence of situated development.

In Figure 4, although the initial structure(Frame 1) is not particularly fit (with a fitness value of 10%), after the assembly phase, once scaffolding is removed and the remaining structure is glued together, the larger section is no longer stable, and topples sideways (Frames 2 and 3), ultimately coming to a rest balanced on the smaller section (Frame 4). This resulting "T" shape is much more fit (a fitness of 49%), and is produced more efficiently this way than by using scaffolding to prop the cross-piece of the "T" into place.

Figure 5 shows a similar result. In this case, the original structure has a fitness of 22%. Once the permanent bricks are glued and scaffolding is removed, the larger structure on the left tips to the side, knocking over the smaller structures in the process. Once it comes to a rest, the smaller structures help prop it into this cantilevered shape with a fitness of 52%.

These are both examples of *meta assembly*: capable only of placing bricks one at a time, and so lacking any formal ability for modular assembly of larger components, the assembly plans, evolved in the context of Situated Assembly, have nonetheless "discovered" how to construct two separate modules, and then join them in the final phase of assembly. We conjecture that in each case the process used in creating the final structure is more efficient than a purely sequential process evolved in a simpler environment without gravity or momentum. It is interesting to observe that complex environments, rather than simply complicating and inhibiting sequential assembly, can instead lead to such novel mechanisms.

4. CONCLUSION AND FUTURE WORK

We have demonstrated a means by which Evolutionary Design can more easily be transferred from simulation to reality. Our process is contingent on two things: first, the evolution of *how* to build rather than simply *what* to build, and secondly the simulation of an object's entire assembly. In the context of Artificial Ontogenies, this occurs when *prescriptive representations*, such as assembly plans, are developed in a *situated development* environment.

The situated development environment used in this work was significantly more rich and complex than earlier related work. Interactions due to gravity, momentum, and collisions between bodies all complicate the process of assembly. Nonetheless, evolved assembly plans, executed in a sequential manner in this environment were able to build an array of interesting objects. In fact, rather than inhibiting sequential assembly, such a complex environment allows for evolution to discover mechanisms such as *meta assembly*, which can produce structures in a more efficient and interesting manner than might be expected.

In this paper, we've been more interested in how prescriptive representations are *interpreted* - that is, via Situated Development, and have focused less on how they are *generated*. The sequential ballistic assembly plans we use are simple to understand and to interpret, but are not particularly *evolvable*. For instance, the closer a mutation is to the beginning of the assembly plan, the larger its effect on the result is likely to be (imagine inserting a simple "turn right" instruction at the beginning of any of the evolved assembly plans above, as opposed to inserting it near the end). Nor are linear encodings ideal for generating symmetrical or modular shapes. Therefore, although linear assembly plans are a suitable final representation, they may not be the best intermediate representation.

One way to improve upon this is to use some form of generative representation, such as the L-systems. Both Hornby [8] and Toussaint [18] have demonstrated the advantages of L-systems in Evolutionary Design tasks. Using such systems as intermediate representations which then generate linear assembly plans would mitigate the mutational brittleness that simple linear encodings exhibit.

Tying the evolution of prescriptive representations to real-world automated manufacture can lead to the full automation of both simulated design and physical assembly. Several hurdles remain, of course, particularly in bridging the "reality gap" [9] between simulation and reality. Two obvious approaches exist: the first is to take a cue from Watson *et al*'s work on Embodied Evolution [21], and remove the notion of simulation entirely - by fully embedding both design and assembly in the real world. The costs of this, in terms of material and time, can be prohibitive however. The alternative is to allow evolution to modify and fine-tune our simulation environment to better reflect a physical assembly system. Bongard and Lipson's recent work in *adaptive simulation* [2], which 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 the best approach.

Ultimately, our work aims to autonomously generate robotic designs which can then be automatically assembled by an autonomous manufacturing system, all without any human involvement. Imagine, for instance, being able to send 100 identical rover-manufacturing plants to Mars, each of them

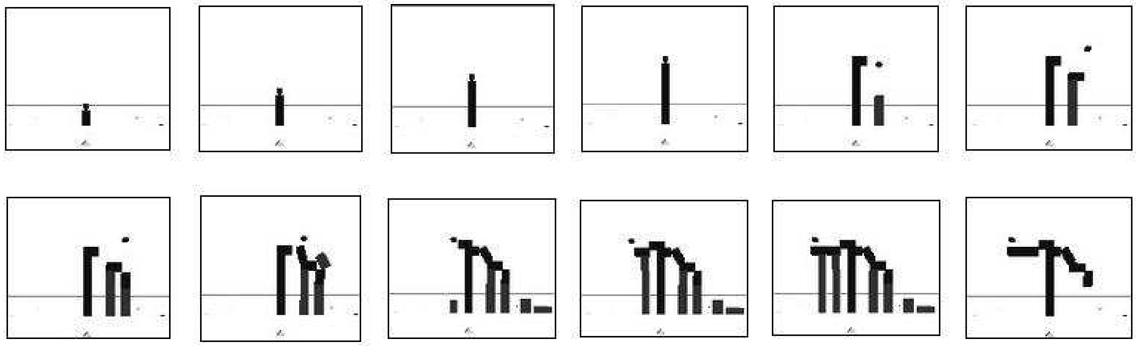


Figure 2: Frames from the assembly of the tree in Table 4.

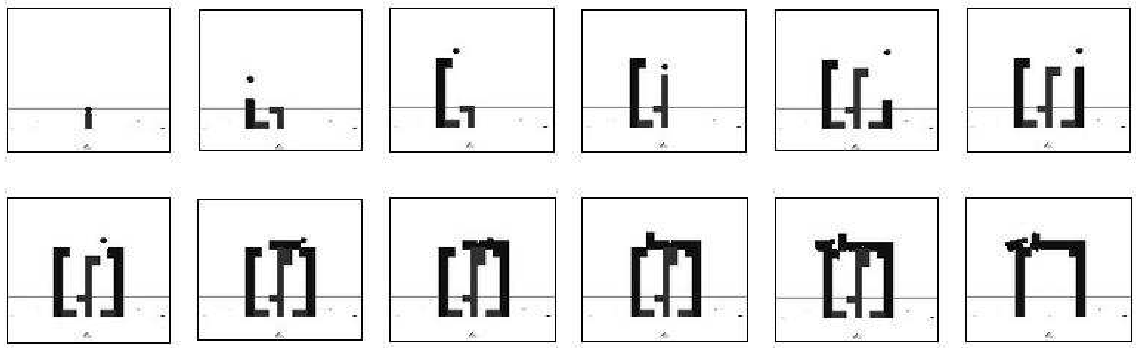


Figure 3: Assembly of the arch in the fourth column of Table 4, evolved with the "shadow" fitness function.

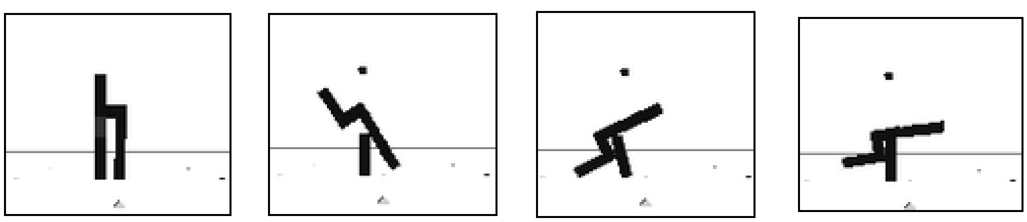


Figure 4: *Meta Assembly*: an interesting consequence of Situated Development. Once the assembly is complete (Frame 1), scaffolding is removed and remaining bricks glued together (Frame 2), the larger section topples onto the smaller section, balancing there to form a T. This resulting shape has significantly higher fitness (49%) than the original structure (10%)(Frame 1)

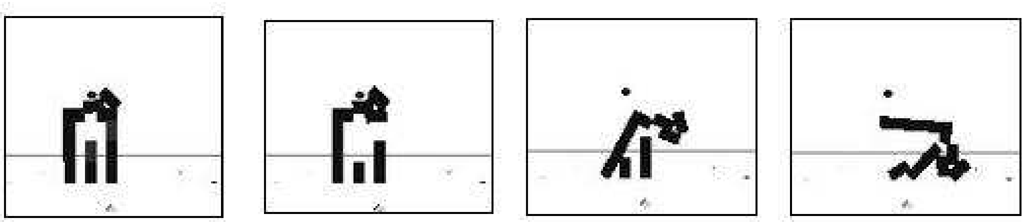


Figure 5: Another Example of *Meta Assembly*. The original structure has a fitness of only 22%. Once scaffolding is removed and remaining bricks glued, the leftmost portion tumbles rightward, and the smaller segments below are knocked sideways, ultimately serving to prop up the larger shape. This final structure has a fitness of 52%)

landing in a different environment - some in craters, some in sandy deserts, etc. And yet each one, once it has surveyed its landing site, could then co-adapt its rover designs and its manufacturing process to local conditions, in order to create mobile rovers closely adapted to its specific environment.

5. REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] E. D. De Jong, R. A. Watson, and J. B. Pollack. Reducing bloat and promoting diversity using multi-objective methods. In L. Spector, E. Goodman, A. Wu, W. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon, and E. Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001*, pages 11–18, San Francisco, CA, 2001. Morgan Kaufmann Publishers.

[5] P. Funes. *Evolution of Complexity in Real-World Domains*. PhD thesis, Brandeis University, Dept. of Computer Science, Boston, MA, USA, 2001.

[6] 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.

[7] G. S. Hornby. *Generative Representations for Evolutionary Design Automation*. PhD thesis, Brandeis University, Dept. of Computer Science, Boston, MA, USA, Feb. 2003.

[8] G. S. Hornby and J. B. Pollack. The advantages of generative grammatical encodings for physical design. In *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*, pages 600–607, COEX, World Trade Center, 159 Samseong-dong, Gangnam-gu, Seoul, Korea, 27-30 2001. IEEE Press.

[9] 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.

[10] L. E. Kavraki, J.-C. Latombe, and R. H. Wilson. On the complexity of assembly partitioning. *Information Processing Letters*, 48(5):229–235, 1993.

[11] M. Komosiński and S. Ulatowski. Framsticks: Towards a simulation of a nature-like world, creatures and evolution. In D. Floreano, J.-D. Nicoud, and F. Mondada, editors, *Proceedings of the 5th European Conference on Advances in Artificial Life (ECAL-99)*, volume 1674 of *LNAI*, pages 261–265, Berlin, Sept. 13–17 1999. Springer.

[12] 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, in press.

[13] S. W. Mahfoud. *Niching methods for genetic algorithms*. PhD thesis, University of Illinois at Urbana-Champaign, Urbana, IL, USA, 1995.

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

[15] J. Rieffel and J. Pollack. The Emergence of Ontogenic Scaffolding in a Stochastic Development Environment. In K. D. et al., editor, *Genetic and Evolutionary Computation–GECCO 2004. Proceedings of the Genetic and Evolutionary Computation Conference. Part I*, pages 804–815, Seattle, Washington, USA, June 2004. Springer-Verlag, Lecture Notes in Computer Science Vol. 3102.

[16] 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.

[17] K. Sims. Evolving virtual creatures. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 15–22. ACM Press, 1994.

[18] M. Toussaint. Demonstrating the evolution of complex genetic representations: An evolution of artificial plants. In *Proceedings of the 2003 Genetic and Evolutionary Computation Conference (GECCO 2003)*. Springer-Verlag, New York, 2003.

[19] J. Ventrella. Explorations in the emergence of morphology and locomotion behavior in animated characters. In R. A. Brooks and P. Maes, editors, *Proceedings of the 4th International Workshop on the Synthesis and Simulation of Living Systems $ArtificialLifeIV$*, pages 436–441, Cambridge, MA, USA, July 1994. MIT Press.

[20] 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.

[21] 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. Zalzala, editors, *Proceedings of the Congress on Evolutionary Computation*, volume 1, pages 335–342, Mayflower Hotel, Washington D.C., USA, 6-9 1999. IEEE Press.

Table 3: Structures generated with the "filled" fitness function. The images on the left show the structures before the scaffolding (grey bricks) has been removed, and the images on the right show the final stable structure with the scaffolding removed.

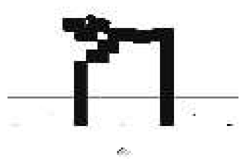
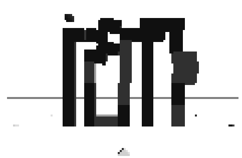
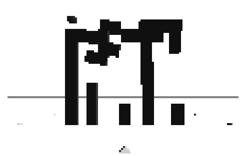
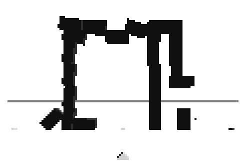
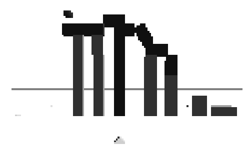
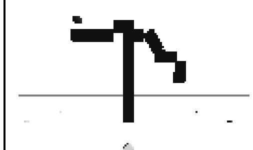
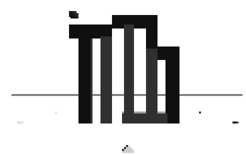
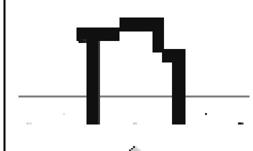
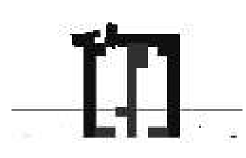
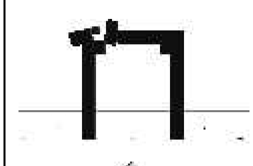
Fill	With Scaffolding	Final
89%		
93%		
95%		

Table 4: Structures Evolved for "shadow" fitness. The left-hand images show the structure before scaffolding (grey) is removed, and the right-hand images show the final, stable structure.

Fill	With Scaffolding	Final
84%		
80%		
90%		
95%