Evolving Assembly Plans for Fully Automated Design and Assembly

John Rieffel
jrieffel@cs.brandeis.edu
(781) 736-3366

Jordan Pollack
pollack@cs.brandeis.edu
DEMO Lab, Brandeis University
415 South St Waltham, MA 02454

Abstract

Evolutionary Design has demonstrated great potential to automatically generate a wide array of novel, interesting, and human-competitive designs. Few of these evolved designs, however, have in turn been physically manufacture. This is due largely to the fact that most evolved designs only specify what *to build, and carry no information on how, or even if, a designed object can be assembled in the real world. When the goal is a physical object, rather than a mere schematic, substantial further effort, most often human-level, is subsequently required to develop a physical assembly process. Evolution of such descriptive representations therefore stands as an obstacle to the full automation of both design and assembly. In this paper we describe an alternative, the evolution of* prescriptive *representations, which offers to remove human effort from the design-and-assembly loop.*

1 Introduction

Evolutionary Algorithms (EAs) have recently been used to design, without human involvement, a wide array of novel and interesting objects. Yet very few of these evolved designs have been transferred from simulation to reality – the two most notable examples perhaps being Lohn *et al*'s evolved antenna [9], due to be launched into space aboard a Low Earth Orbit satellite, and Pollack *et al*'s evolved robots [11].

Despite being designed without human input, none of the physically-manifested evolved designs were in turn automatically assembled. Rather, they required significant human effort and insight to transfer them from simulation to the real world. Funes' LEGO structures [4], for instance, had to be assembled on a horizontal surface and then slowly tilted into place. Lohn *et al*'s evolved antennas had to be expertly bent and soldered into place by hand, with extreme care taken to preserve the precise measurements specified by the evolved design. Our interest is in removing human effort from the assembly of, not just the design of, evolved objects. other words to fully automate both design and assembly. Crucial to this Fully Automated Design and Assembly are two things: evolving *how* to build rather than *what* to build, and simulating the entire process of an object's assembly.

2 Blueprints and Assembly Sequence Planning

The majority of Evolutionary Design systems have produced, as their product, a *descriptive* representation of the designed object, such as a blueprint. When simple schematics are the end-goal of Evolutionary Design, then such descriptive representations are sufficient - and many impressive human-competitive results have emerged from such systems, as demonstrated by recent winners of the human-competitive competition at GECCO 2005 such as Lohn*et al*'s antennas [9], and Lipson's straight-line drawer [8].

When, however, the aim is to produce a physical object, then the evolved descriptive representations require significant further effort, most often human level, to generate the actual object. This is due to the fact that descriptive representations describe *what* to build, but contain no information about *how* to build - just as looking at a photograph of a meal provides little insight into how to cook it. The further effort therefore lies in inferring the actual assembly process required to create the physical object.

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 studied, and it has been proven to be NP-complete in the general case [7].

3 Direct Evolution of Assembly Plans

The closest that the field of Evolutionary Design has come to Fully Automated Design and Assembly is Hornby's evolved tables [5]. His voxel-based representations were converted into STL - a CAD format which was then interpreted by a rapid-prototyping machine, which in turn printed the object out of plastic. While approaches such as this may reduce human involvement, they don't eliminate it - and may in fact only defer it. Significant human effort was necessary to create the means by which the rapid prototyping machine could translate an STL file into a series of commands to the print-head. More importantly, such a solution is *brittle*, if the entire machine were tilted at a slight angle, the printed object would no longer resemble the original design, and an entirely new human-designed STL-to-printer translation system would be needed. We are interested, therefore, in more dynamic and adaptive methods.

This begs 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. Unlike blueprints, prescriptive representations can provide step-by-step instructions on how to build an evolved design. In their simplest form, prescriptive representations are merely assembly plans: sequential, ballistic [1] , sets of instructions to an assembly mechanism, which when executed results in the construction of an object.

As linear sets of instructions, a key advantage of assembly plans is that they can be directly interpreted by an assembly mechanism, without the need for further human intervention, and so they allow for the full automation of assembly.

Prescriptive representations are in fact not uncommon *intermediate* representations in Evolutionary Design. Both Hornby's tables [6] and Toussaint's virtual plants [14] for instance, were "built" by a LOGO-like turtle interpreting a linear set of instructions to place OpenGL voxels in simulation space. In neither case, however, was this process meant to be analogous to an actual physical assembly process.

Evolving for automated assembly requires that Evolutionary Design simulate an object's entire assembly process, not just its final behavior. In this work we use Artificial Ontogenies, a form of Evolutionary Design that takes its inspiration from biological processes of growth and development, as a tool for exploring these issues of automated assembly. Unlike the majority of Artificial Ontogenies, however, we do not take the process of assembly, nor the environment in which it occurs, for granted. Rather, we use what we describe as *Situated Development*: Artificial Ontogenies in which an object's assembly is subject to the same environment as its evaluation as a complete object. 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. Like Hornby and Toussaint's work above, however, this process was not meant to be analogous to physical assembly.

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

The situated development environment which we use, based upon a realistic 3-D dynamics engine, is constructed so that it is akin to an automated manufacturing process, and therefore provides a framework for exploring fully automated design and manufacture.

4 A Framework for Simulating Fully Automated Design and Assembly

We have discussed how prescriptive representations evolved within the context of Situated Development allow for the evolution of *buildable* objects, and can lead to the full automation of physical assembly. We now introduce a framework we have created for simulating the process of fully automated design and assembly, and discuss results obtained from experiments run within that framework.

4.1 Earlier Work: Assembly Plans for Simple 2-D Structures

This is in many ways an extension of our earlier work on evolving assembly plans within stochastic development environments [12, 13]. In that work, our environment consisted of a discrete 10x10 grid with "tetris"-like physics. The goal of those experiments was to evolve assembly plans capable of building a goal structure - an arch - with high reliability, despite the the noisy environment. Our evolved assembly plans were able to overcome the stochastic physics and achieve high yields by placing what we called *ontogenic scaffolding*: intermediate structural elements that supported the goal structure during assembly, but were removed before final evaluation. The important result of this work was the demonstration that mechanisms such as ontogenic scaffolding could emerge in ballistic assembly plans, using only evolutionary-scale feedback.

4.2 A Situated Development Environment

While an important result, our 2-D, grid-based, tetris-physics assembly environment was far from realistic. In this paper we are more interested in more realistically simulating an assembly mechanism, and on the effects of such an environment on the evolution of assembly plans.

In the context of this environment, we begin by evolving assembly plans for a goal structure (again, an arch). We then remove the need for an explicit goal structure, and

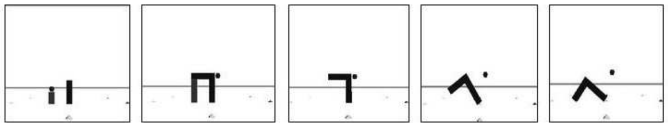


Figure 1. Assembly has three stages. In the first, both permanent and temporary bricks are placed. In the second, adjacent permanent bricks are glued together, and scaffolding is removed. Finally, the remaining structure settles.

evolve assembly plans for structures with more implicit design goals.

Our simulated assembly system is based upon the Open Dynamics Engine (ODE) [2] 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 or pick-and-place arm, which is 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.

Since our earlier work demonstrated the ability of similar systems to discover scaffolding using only evolutionary-scale feedback [12, 13], we chose to allow for the *explicit* placement of scaffolding. The machine is therefore 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. Modern models of rapid prototyping machines have similar capabilities.

The assembly process falls into three stages (Figure 1). In the first, the machine interprets the assembly plan, moving around and placing bricks as directed. Although each assembly instruction is executed at discrete time intervals, the assembly space dynamics are continuously evaluated. 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.

[2]`www.ode.org`

Table 1. 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
Put (S)caffolding	(a)head, to (r)ight, to (l)eft, (b)ehind
(T)ake Brick	(none)

5 Experiments

Two sets of Evolutionary Multi-Objective Optimization (EMOO) [2] experiments were run in this Situated Assembly system. In the first, we evolve an assembly plan to build a pre-defined arch goal structure, similar to the one evolved in our earlier work [12] (except of course, for being three-dimensional). Secondly, we evolve assembly plans for structures with an implicit goal.

5.1 Method

In each case, the genotypes of the system consisted of assembly plans: sequential set of parameterized instructions to the assembly system. Table 1 lists the instructions used.

The evolutionary framework used was based upon Multi-Objective Optimization (MOO) [2]. Since the scope of this paper is about the use of Artificial Ontogenies to simulate Fully Automated Design and Assembly, not on the advantages of any particular algorithm, we refer the reader to our earlier work [12] for details of the system used. In each experiment, the initial population was created with 30 random genotypes, with randomized 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), were added to the population. In order to limit population sizes, duplicate genotypes were rejected, and duplicate objective values were limited using crowding [10], with a limit of 3 individuals per bin.

5.2 Evolving Assembly Plans for an Explicit Goal

While the full power of Evolutionary Design lies in open ended design, there is often the need to determine whether a pre-determined structure is at all *buildable* by a specified automated assembly mechanism. In this context, the goal is to find a suitably efficient assembly plan which, when interpreted by that mechanism, results in the goal structure. This is, in a sense, automating the task of Assembly Sequence Planning from the bottom-up. We began, therefore, by evolving assembly plans capable of building a predetermined goal structure, as in our earlier work. For the sake of consistency we once again chose an arch (shown in the last frame of Figure 2).

In order to compare each resulting structure to the goal structure, a simple bitmap was generated by sampling the central $10b \times 10b$ region (where b is the width of a brick) in the X-Z plane, at a resolution of $.2b$. 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): length of genotype, mass of phenotype, number of squares missing from the goal structure, and number of "wrong" (i.e. either extraneous or missing) squares. These objectives are identical to the ones used to evolve the goal arch in our earlier 2-D grid-based work.

The first two objectives exist for more than just deterrence against bloat, per [3]. Rather, they also reward assembly plans for efficiency in terms of time (the length objective) and in terms of material (the mass objective). Physical prototyping machines are slow, and require expensive material - therefore any reduction in print time or print material is highly valuable.

Figure 2 shows animation frames from a representative evolved solution. (Full color images of all results, as well as animations, are available at the author's web page [3]). Discovered after roughly 2000 generations and with a length of 22 instructions, it is able to perfectly generate the goal structure. By comparison, we were unable to produce a hand-built assembly plan shorter than 29 instructions.

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 printhead 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 wouldn't have been under the supported brick's center of mass, and the supported brick might have tilted sideways.

5.3 Evolving Assembly Plans with for Implicit Goals

Having evolved assembly plans for a pre-determined goal structure, we now evolved assembly plans for structures with implicit goals. This allows for more open-ended design, and follows the lead of several earlier evolved designs.

[3]www.cs.brandeis.edu/˜jrieffel/situated-development/

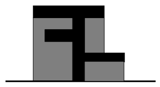


Figure 3. Shading Fitness Function. Only the gray region is shaded by the black structure.

We chose a simple "shaded area" fitness function, which measures the total amount of open volume beneath a structure. First a bitmap of the resulting structure is created in the same fashion as above. Then, every region (x,z) of the bitmap which is empty and underneath a filled region $(x, z') where (z' > z)$, is considered shaded. Figure 3 illustrates this. This fitness function is similar to those used for Funes LEGO trees [4] and Toussaint's OpenGL plants [14]

The length and mass objectives are retained from the experiment is Section 5.2, and the fitness function above replaces the two goal-based objectives from 5.2.

Table 2 shows representative evolved assembly plans. The table 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. For comparison, our best hand-built structure, shown on the far right hand side, while perfect, required 34 bricks, more than the maximum 25 allowed for evolved assembly plans, and was 65 instructions long.

As is suggested by the array of shapes in Table 2, the majority of structures evolved with the implicit design goal were arch-shaped rather than tree-shaped, even though tree shapes, since they have only one supporting leg, can shade more area. It is worth exploring this further: the best explanation is that although trees are more fit, they are less stable in the face of change. Any addition or removal of a brick along a branch would unbalance it, causing it to tip over after the glue/melt phase. Any change on one branch would need to have a corresponding change along the other branch in order to retain balance. Arches, on the other hand, are much more stable in the face of perturbation: any addition of a brick along the top surface would have no effect on the balance of the structure. Furthermore, any removal of a central brick might allow the two sides to fall toward each other and meet. Therefore, arches present a larger, if slightly less optimal, evolutionary target than trees do.

6 Conclusion

Prescriptive representations, interpreted in the context of a Situated Development environment, allow for the full automation of both design and assembly. We have described a framework which simulates such Fully Automated Design and Manufacture, and demonstrated the ability of assembly plans evolved in that context to describe novel and efficient

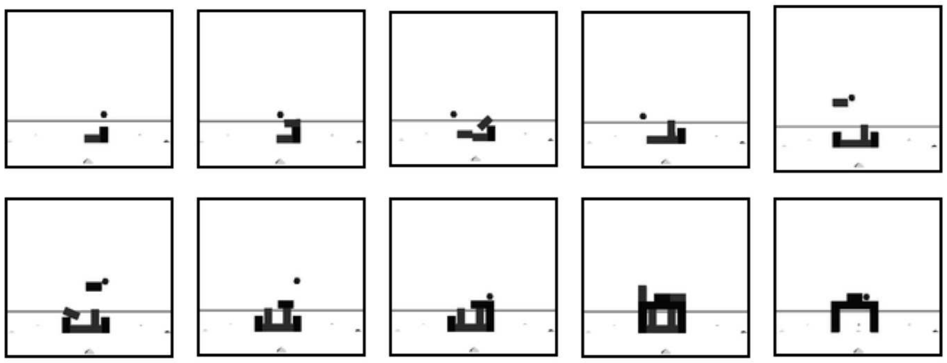
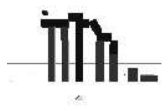
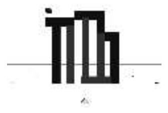
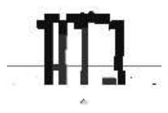
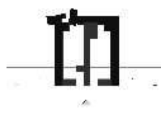
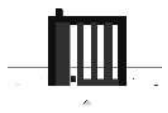
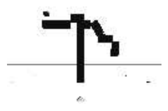
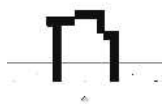
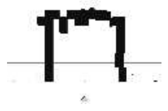
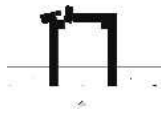
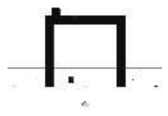


Figure 2. Building the Goal Arch. Note how the horizontal scaffolding placed in frame 3 tumbles into a vertical position to support the top of the arch. This is repeated with the piece of scaffolding placed in frame 5. (frames are read left to right, top to bottom - solid bricks are black, scaffolding is gray, the small sphere is the location of the print-head. The horizontal line is the horizon)

Table 2. Structures Evolved for "shadow" fitness. The top-row images show the structure before scaffolding (gray) is removed, and the bottom images show the final, stable structure. The structures on the far right were hand-built.

With Scaffolding					
Final					
Fitness	84%	80%	95%	90%	100% (hand-built)
Length	30	54	58	43	65

manufacturing processes.

Although our interest is in the physical manifestation of evolved designs, we have, for the time being, limited ourselves to the *simulation* of physical assembly. We are currently investigating approaches to transferring our results to the real world. One approach is to evolve prescriptive representations in simulation space, as we have done, but to use a language of representation that can be directly interpreted by a corresponding physical manufacturing system.

In the long term, our aim is to create a fully adaptive automated design and manufacturing system, capable of changing its manufacturing process to suit local conditions and materials. Currently, planetary rovers are designed on earth, and must be generalized to accommodate a wide array of anticipated landing sites. Imagine, instead, sending an entire fleet of identical rover manufacturing plants to the surface of Mars, each one landing in a different (and unanticipated) environment - some in deep sand, some on the angled face of a crater. Having landed and surveyed its landing site, each could then generate rovers specialized to their particular local environment, all without human involvement.

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] 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.

[3] 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.

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

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

[6] 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.

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

[8] H. Lipson. How to draw a straight line using a GP: Benchmarking evolutionary design against 19th century kinematic synthesis. In M. Keijzer, editor, *Late Breaking Papers at the 2004 Genetic and Evolutionary Computation Conference*, Seattle, Washington, USA, 26 July 2004.

[9] 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.

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

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

[12] 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.

[13] 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.

[14] 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.