Crossing the Fabrication Gap: Evolving Assembly Plans to Build 3-D Objects

John Rieffel
DEMO Lab, Brandeis University
Waltham, MA 02454
jrieffel@cs.brandeis.edu

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

Abstract- Evolutionary Computation has demonstrated the ability to design novel and interesting objects. Such objects are increasingly being assembled in the physical world, albeit with some difficultly. An obstacle to this assembly is that most evolved designs are *descriptive* representations: they specify *what* to build, but carry no information on *how* to build it. Inferring a corresponding assembly sequence for such an object is a complex task for any but the most trivial designs. We offer an alternative solution to this spectre of the *Fabrication Gap*, namely the direct evolution of assembly sequences. As we show, such methods not only lead to the evolution of *buildable* objects, but also lead to the emergence of novel means of assembly as well.

1 Introduction

Beginning with Karl Sims' seminal work on evolved robots (1994), evolutionary computation has gained increasing popularity as a means of automatically creating novel designs of objects (Lohn et al., 2005) (Hornby, 2003), structures (Funes, 2001) (Toussaint, 2003), and robots (Komosiński and Ulatowski, 1999) (Pollack et al., 2001) (Ventrella, 1994).

Recently, these evolved objects are beginning to be assembled in the physical world. One of the earliest examples of this physical assembly is Funes' LEGO structures (2001), which were evolved in a physically realistic simulation which calculated joint strength between elements, and then built by hand. Later, Hornby used parametric L-Systems to produce both tables and mobile robots (2003). Most recently, Lohn *et al* at NASA have created an antenna, due to be launched into low earth orbit aboard a satellite (2005).

In none of these cases, however, was the transfer from simulated to physical object in any sense automatic. Rather, significant human effort was required to assemble the evolved object in the real world. Funes' LEGO tree, for instance, was assembled horizontally and then slowly tilted into place. Lohn *et al*'s evolved antennas had to be expertly bent and soldered into place, with extreme care taken to preserve the precise measurements specified by the evolved design.

A large source of this added effort is due to the fact that most evolved designs are *descriptive* representations, and as such specify *what* to build, but carry no information on *how* to build the specified object. The task of subsequently inferring an assembly sequence for the evolved object usually falls on human shoulders, thereby adding human involvement back into what was until then an automatic process. As we discuss below in Section 2, inferring an assembly sequence for a predetermined object automatically, without human intervention, is provably quite difficult.

We refer to this gap in knowledge and effort between evolved design and assembly process as the *Fabrication Gap* - an echo of Jakobi's Reality Gap (Jakobi et al., 1995) between simulated and real-world robotics. Our interest is in removing this human effort and bridging this Fabrication Gap, with the goal of producing a system capable of Fully Automated Design and Assembly.

The closest that the field of Evolutionary Design has come to such automation is Hornby's evolved tables (2001). In that work, the evolved voxel-based representations were converted into STL - a CAD format which could be parsed by a rapid-prototyping machine. While this may reduce human involvement, it doesn't eliminate it - and in fact, may 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 printhead. More importantly, such a solution is *brittle* - if one of the servo-motors were faulty, or if the entire machine were tilted at a slight angle, the printed object would no longer resemble the original design. We are interested, therefore, in more dynamic and adaptive methods.

This example raises the question: rather than rely on some brittle translation between descriptive representation and assembly process, why not evolve rapid prototyping machine instructions directly? Doing so allows for the evolution of *how* to build rather than merely *what* to build. This can be accomplished via the use of *prescriptive* representations, such as *assembly plans*, which describe an object's assembly rather than its final specification. Better yet, not only does this provide sufficient information on how to build an evolved object, but it also provides a method of automating assembly, provided that the language of evolved assembly plans is directly interpretable by some compatible manufacturing system.

Of course, accomplishing this requires the high-fidelity simulation of an object's assembly. Physically realistic simulations are not new to Evolutionary Design - in fact, even Sims' robots were evolved in a physical simulation (1994). Most such Evolutionary Design systems (for instance (Komosiński and Ulatowski, 1999; Ventrella, 1994; Hornby, 2003; Pollack et al., 2001)), however, only simulate the behavior of a complete object, not its corresponding assembly process.

One of the fews examples of a system which *does* simu-

late assembly (or rather, growth) is Bongard's work (Bongard and Pfeifer, 2003), which used a Gene-Regulatory Network-based (GRN) Artificial Ontogeny to grow agents starting from a single "cellular" unit. However, while this GRN system proved to be quite effective for design of morphology, it does not easily present itself as a means of describing automated manufacture. Since our interest is in producing assembly sequences which can be automatically interpreted, we will instead directly evolve simple assembly sequences containing instructions to a manufacturing system such as a rapid prototyping machine.

In this paper we describe our framework for simulating such a system. As we show, not only does Evolutionary Design in such a system produce assembly processes for novel objects, but it also produces novel *means of assembly* as well. Although recent papers of ours have mentioned this emergence of novelty in passing, here we provide a method of measuring it, and provide a quantitative comparison of the phenomenon across two different environments. This analysis provides insight into how such novelty can be encouraged and harnessed, in order to bridge the Fabrication Gap.

2 Assembly Sequence Planning

As we mentioned , most Evolved Designs which were subsequently assembled in the real world have required significant human effort. The source of this effort is largely in the form of figuring out how to build, in a sequential manner, an object resembling the evolved *descriptive* representation.

In the field of engineering, the task of inferring a sequence of assembly instructions given a particular structure *a priori* is known as *Assembly Sequence Planning*, or *Assembly Sequencing* and is a rather richly studied topic. Although the process of determining an assembly sequence may come readily to humans, it often much harder to computationally solve, and has in fact been demonstrated to be NP-complete in the general case (Kavraki et al., 1993).

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 parts one at a time. Doing so, however, makes the critical assumption that every stage of assembly is both *reversible* and *symmetric*. And even within those constraints, the problem is demonstrably rather complex (Goldwasser et al., 1996; Kavraki et al., 1993). Needless to say, without those constraints descriptive Evolutionary Design Systems are capable of generating designs of objects whose assembly sequence is, optimistically, extremely difficult if not impossible to produce automatically.

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.

3 Direct Evolution of Assembly Plans

As we discussed in our introduction, faced with such difficulties in inferring an assembly sequence from an evolved descriptive representation, it might be better to avoid the problem entirely by directly evolving assembly sequences. In this realm, the language of assembly no longer needs to be constrained to reversible and symmetric operations, and as a consequence, the domain of *buildable* evolved structures is greatly expanded - because they no longer need to be *unbuildable* as well.

It is this last aspect that we would particularly like to explore the consequences of in this paper. By directly evolving assembly sequences, and unconstrained to reversible and symmetric assembly methods, Evolutionary Design can arrive at not just novel objects, but novel means of assembling those objects. Of course, there is no "free lunch" in this process. By allowing evolutionary design to range over the entire space of assembly sequences, we greatly increase the search domain, and other methods may be needed to intelligently constrain the search. As a final advantage, however, consider that evolved assembly sequences can be directly interpreted by a compatible manufacturing system, and so allow for fully automated design *and* assembly, offering the prospect of removing the human from the loop entirely.

4 A Framework for Exploring Assembly Sequence Design

Our goal in this paper is to simulate a system which is capable of Fully Automated Design and Assembly by means of the evolution of assembly plans in a realistic assembly environment. Such a system is analogous to a rapid prototyping machine capable of executing assembly plans, coupled to an evolutionary algorithm capable of generating new plans. Our framework for exploring these issues 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.

Assembly is performed by a LOGO-like turtle, acting as a print head, capable of movement in the X-Z plane, and of depositing 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 recent work has demonstrated the ability of similar systems to discover scaffolding implicitly during the course of evolution (Rieffel and Pollack, 2004a; Rieffel and Pollack, 2004b), here we allow for the *explicit* placement of scaffolding. The turtle is capable of placing both 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 analogous to similar features of modern rapid prototyping machines, which can lay thin water-soluble support structures that are dissolved once manufacture is complete.

[1] www.ode.org

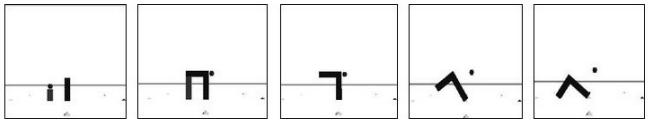


Figure 1: Assembly has three stages. In the first, both permanent and temporary bricks are placed. In the second, adjacent permanent bricks are glued together, and scaffolding is removed. Finally, the remaining structure settles.

Our simulated assembly falls into three stages (Figure 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, adjacent permanent bricks are glued together (but not to the floor), and then scaffolding elements are removed. Finally, once the scaffolding is gone, the final structure is allowed to come to a rest before being evaluated.

4.1 Set Up

The genotypes of our Evolutionary Algorithm were assembly plans, consisting of a sequential set of parameterized instructions to the situated development system described above.

Rather than using a single fitness function, we used a simple Evolutionary Multi Objective Optimization (EMOO) Algorithm (Coello, 1999) over a set of objectives, described in more detail below. Initial population size was 30 individuals, each with a randomized length of 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 (Mahfoud, 1995), with a limit of 3 individuals per bin.

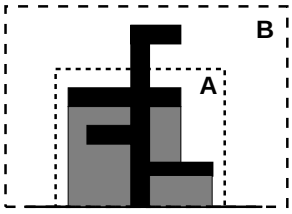


Figure 2: Illustration of the Development Environment. In both setups, the fitness function is measured over the smaller box (A). Within that box structures are rewarded for maximizing the "shaded" gray regions under the black structure. Note that the uppermost overhang does not contribute any shade, because it exists outside of the fitness bounds. In Setup A, the range of the turtle is limited to the smaller box (A). In Setup B, the range of the turtle extends to the outer box (B).

Our design task was to create a structure which maximizes the total open volume beneath a structure, thereby encouraging structures which both maximize height and maximize the number of empty spaces beneath the structure.(Figure 2) In order to measure the "shaded" fitness of each structure, a bitmap was generated by sampling box A in Figure 2 - the central 100×100 region (bricks are $10 \times 10 \times 20$) in the X-Z plane.

The specific objectives of our EMOO where therefore:

- Length of Assembly Plan. (minimizing)
- Mass - number of bricks in the entire world, not just the sample region. (minimizing)
- Number of shaded bits. (maximizing)

The first two objectives exist for more than just deterrence against bloat, per (De Jong et al., 2001). 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.

4.2 Early Examples Novel Assembly in Evolutionary Design

In earlier work (Rieffel and Pollack, 2005), we used an environment in which the range of the turtle was limited to the same box as fitness was measured in (box A in the Figure 2).

An interesting phenomenon that we noticed is that occasionally, evolved structures would be unstable once scaffolding was removed. This instability then caused the structure to tumble into a final, structurally distinct shape, often with higher fitness. Figure 3 provides an example of this phenomenon.

This phenomenon of "dynamic assembly", as we call it, is an interesting exploitation of the system as we designed it, and is a preliminary example of the novel types of assembly process that can arise from evolving assembly plans directly in a realistic environment.

From the perspective of Assembly Sequence Planning (Section 2), this process of assembly via toppling is a good example of a non-reversible assembly sequence - the structure's final location is a result of the assembly's interaction with its environment, rather than purely a direct result of an assembly instruction. Any purely sequential disassembly sequence of the arch would not be able to produce a matching inverse of the action.

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 answer this 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 solutions which use dynamic assembly any more efficient, either in terms of time (assembly plan length) or in terms of material (mass of final structure).

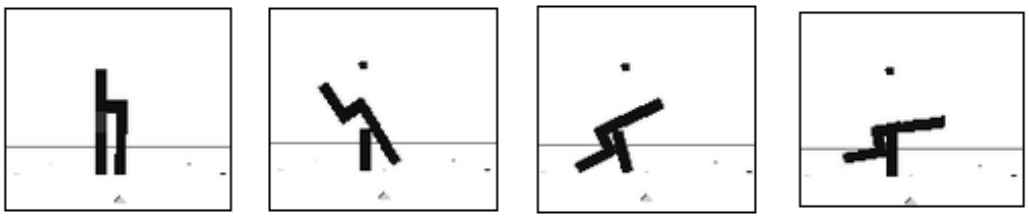


Figure 3: A novel dynamic assembly method discovered in Setup A (Section 4.2) . Once the assembly is complete (Frame 1), scaffolding is removed and remaining bricks glued together (Frame 2), the larger section topples onto the smaller section, balancing there to form a T. This resulting shape has significantly higher fitness (49%) than the original structure (10%)(Frame 1)

4.3 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.

$$f_{dynamic} = f_{final} - f_{initial} \quad (1)$$

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.

4.4 Comparing Assembly Environments

Armed with this means of measuring dynamic assembly and efficiency, we ran two sets of experiments in slightly different environments. The first environment (Setup A) was identical 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 2). 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 1 summarizes the differences between environments.

Table 1: Comparison of Setup A and Setup B

	Setup A	Setup B
# runs	11	16
Turtle Range	200×100	100×100
Fitness Range	100×200	100×100

4.4.1 Results

Table 2 contains representative results derived from Setup A. The figures on the left hand side show the structure before scaffolding was removed, and the figures on the right hand side show the final, stable structure.(Full color images of all results, as well as animations, are available at the author's web page [2]). As can be seen, structures tend towards stable arch-like structures with two legs.

Table 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). As can be inferred from variety of structures shown in the table, unlike the setup in Setup A, which produced relatively stable arch-like structures, this small change results in a significantly number of structures which are assembled dynamically.

Figure 4 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.Table 3 summarizes the values in Figure 4, and provides some further comparisons. Although both regimes produce equally fit structures, the contribution of dynamic assembly towards fitness in Setup B is significant, whereas solutions in Setup A contain very little dynamic assembly. This suggests that a relatively small change in the environment can significantly change both the type of of structure, and the means of assembly.

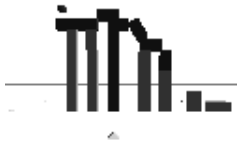
Figures 5 and 6 respectively compare the average value of the fitness per brick and fitness per assembly plan instruction for the best individual of each generation. In each case the values are close, and variance is rather high.

In Figure 5, the slightly lower fitness per brick of Setup B 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 2) do not have a deleterious effect upon fitness the way that extra bricks within the box do - and yet may serve a useful function, for instance as a counterbalance during dynamic assembly.

On the other hand, the fitness per assembly plan instruction (Figure 6) is slightly higher for Setup B. This suggests that while solutions evolved in the larger environment may not be more efficient in terms of material, they gain efficiency in time. In other words, assembly plans in Setup B tend to be shorter than equivalently fit assembly plans evolved in Setup A. This efficiency can easily be attributed to the novel methods of dynamic assembly that are discov-

[2] www.cs.brandeis.edu/~jrieffel/situated-development/

Table 2: Structures Evolved in Setup A. The left-hand images show the structure before scaffolding (grey) is removed, and the right-hand images show the final, stable structure. The small black sphere shows the location of the turtle.

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

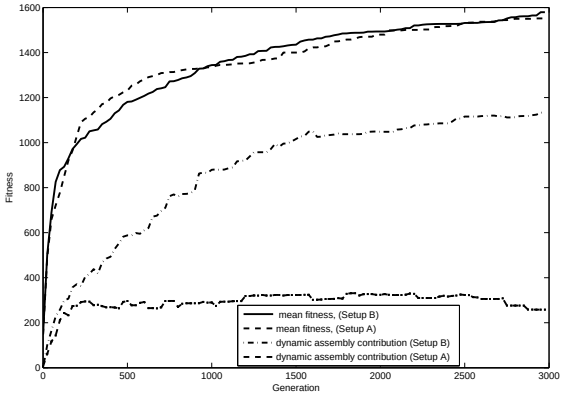


Figure 4: Comparison of fitnesses and fitness contribution from dynamic assembly between Setup A and Setup B. Data is averaged across 11 runs for Setup A, and 16 runs for Setup B. Although maximal fitness is equivalent across both regimes, Setup B contains significantly more examples of dynamic assembly

Table 3: Comparison of values between Setup A and B

	Setup A		Setup B	
Mean Vals at Generation 3000				
		σ		σ
f_{final}	1552	205	1579	174
$f_{dynamic}$	258	306	1136	539
$f_{final}/brick$	138	25	111	24
$f_{final}/instr$	48	11	54	16

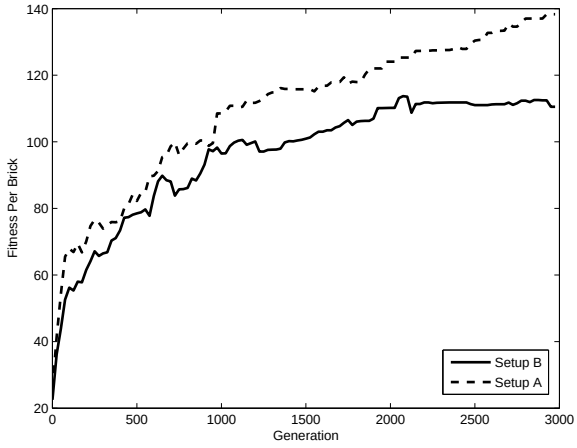


Figure 5: Average Fitness Per Brick between Setup A and Setup B.

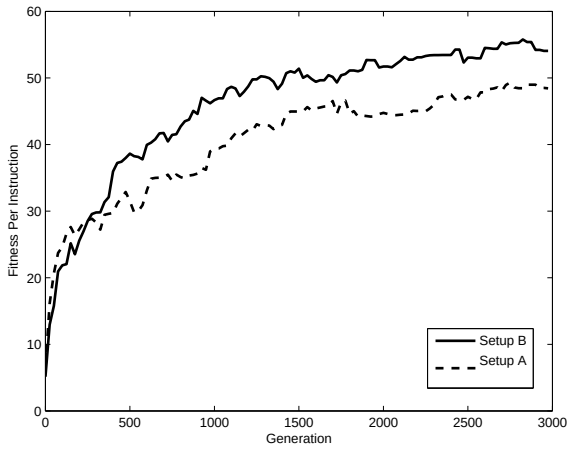


Figure 6: Average Fitness Per Assembly Plan Instruction between Setup A and Setup B.

ered in the larger assembly environment.

Figures 7, 8 and 9 provide illustrations of such dynamic assembly, and demonstrate how this efficiency is accomplished. Of the three, the most interesting is the "T" shape in Figure 9. The initial fitness, immediately after scaffolding has been removed but before the structure has settled is below 1% of maximum. Once scaffolding is removed, however, the left column is unbalanced, and so topples onto the right-hand column - tottering back and forth until coming

to a rest perfectly balanced atop it. An corresponding non-dynamic assembly process would require substantial extra scaffolding in order to create an equivalent structure.

5 Discussion and Conclusion

We began by introducing the notion of the "Fabrication Gap" that can arise as a consequence of evolving descriptive representations of designed objects. This gap, between the specified object and the unknown, *a priori*, assembly sequence which can produce it, is an obstacle to the full automation of both design and assembly, since it often requires substantial human intelligence and insight to cross.

We have presented one way to avoid, rather than cross, this gap - by evolving *prescriptive* assembly plans instead of *descriptive* blueprints, and by accurately simulating the entire process of an object's assembly. Doing so produces a result which is not only buildable, but also one which immediately presents itself for automation, thereby clearing the way for Fully Automated Design and Assembly.

An added benefit of this evolution in *assembly space* rather than design space arises from Evolutionary Algorithms' ability to exploit aspects of their substrate. In our case, this means that Evolutionary Design can discover novel means of assembly, not just novel designs. We've also shown how a small change to the assembly environment results in a significantly higher occurrence of these phenomena.

The most obvious of these novel assembly methods is what we've termed "dynamic assembly", in which the evolved process builds multiple sub-assemblies supported by scaffolding which, once scaffolding is removed, become unstable and topple into a significantly more fit final shape. As we've shown, such dynamic assembly methods are often more efficient in time than corresponding non-dynamic solutions.

At this point we have made no claims about the *robustness* of these novel assembly methods, and all likelihood they are quite brittle. In recent work (2004a), however, we demonstrated the ability of evolved assembly plans to discover robust assembly methods in the face of noise during assembly, albeit in a simpler physics model - and we hope to translate those methods into this richer environment.

As described, these examples of dynamic assembly are highly specific to the environment in which they were evolved. Moreover, even though our interest is in physical assembly, we have presented only simulated results. Any system which evolves solutions in simulation and then hopes to translate them into reality faces the spectre of the so-called"reality gap" (Jakobi et al., 1995). Ultimately we will need high fidelity simulations of assembly mechanisms. One option 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* (Bongard and Lipson, 2004), 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 one promising approach to fine-tuning simulation to match reality.

The goal of Fully Automated Design and Assembly is, however, a worthwhile one. 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 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.

References

Bongard, J. and Pfeifer, R.: 2003, *Morpho-functional Machines: The New Species (Designing Embodied Intelligence)*, Chapt. Evolving complete agents using artificial ontogeny, pp 237–258, Springer-Verlag, Berlin

Bongard, J. C. and Lipson, H.: 2004, 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)*, pp 57–62

Coello, C. A. C.: 1999, 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 (eds.), *Proceedings of the Congress on Evolutionary Computation*, Vol. 1, pp 3–13, IEEE Press, Mayflower Hotel, Washington D.C., USA

De Jong, E. D., Watson, R. A., and Pollack, J. B.: 2001, 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 (eds.), *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001*, pp 11–18, Morgan Kaufmann Publishers, San Francisco, CA

Funes, P.: 2001, *Evolution of Complexity in Real-World Domains*, Ph.D. thesis, Brandeis University, Dept. of Computer Science, Boston, MA, USA

Goldwasser, M., , Latombe, J., and Motwani, R.: 1996, Complexity measures for assembly sequences, in *Proc. IEEE Int. Conf. on Robotics and Automation*, pp 1581–1587, Minneapolis, MN

Hornby, G. S.: 2003, *Generative Representations for Evolutionary Design Automation*, Ph.D. thesis, Brandeis University, Dept. of Computer Science, Boston, MA, USA

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

Jakobi, N., Husbands, P., and Harvey, I.: 1995, Noise and the reality gap: The use of simulation in evolutionary

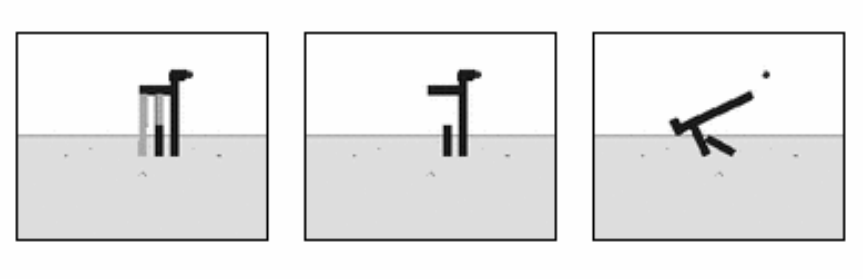


Figure 7: Another dynamic assembly sequence from Setup B. This is 22 instructions long, with a final mass of 10. Initial fitness is 31%, final fitness is 47%, a 52% increase.

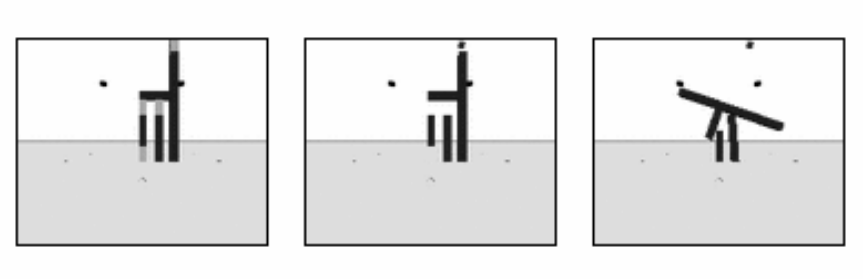


Figure 8: A dynamic assembly sequence from Setup B. 20 instructions, 14 mass. Initial fitness: 27%, Final Fitness: 62%, a 129% increase. The black spheres mark the upper corners of the fitness bounds.

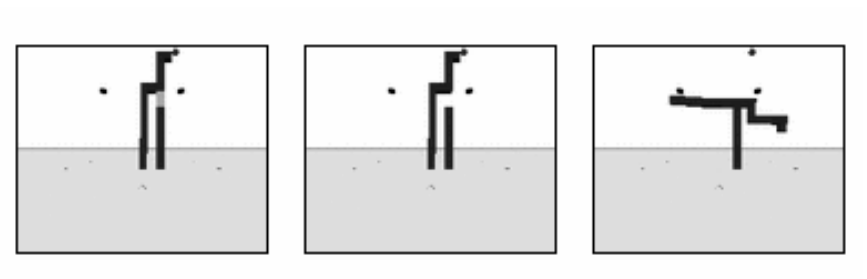


Figure 9: The most extreme example of dynamic assembly from Setup B. With only 17 instructions, and a mass of 13. Initial fitness is 0.4% (no typo), Final fitness is 80%. The black spheres mark the upper corners of the fitness bounds. The near-zero initial fitness is due to the fact that the overhanging brick is outside of the fitness bounds, and therefore contributing no shade.

robotics, in *Proc. of the Third European Conference on Artificial Life (ECAL'95)*, pp 704–720, Granada, Spain

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

Komosiński, M. and Ulatowski, S.: 1999, Framsticks: Towards a simulation of a nature-like world, creatures and evolution, in D. Floreano, J.-D. Nicoud, and F. Mondada (eds.), *Proceedings of the 5th European Conference on Advances in Artificial Life (ECAL-99)*, Vol. 1674 of *LNAI*, pp 261–265, Springer, Berlin

Lohn, J. D., Hornby, G. S., and Linden, D. S.: 2005, 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 (eds.), *Genetic Programming Theory and Practice II*, Kluwer

Mahfoud, S. W.: 1995, *Niching methods for genetic algorithms*, Ph.D. thesis, University of Illinois at Urbana-Champaign, Urbana, IL, USA

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

Rieffel, J. and Pollack, J.: 2004a, The Emergence of Ontogenic Scaffolding in a Stochastic Development Environment, in K. D. et al. (ed.), *Genetic and Evolutionary Computation–GECCO 2004. Proceedings of the Genetic and Evolutionary Computation Conference. Part I*, pp 804–815, Springer-Verlag, Lecture Notes in Computer Science Vol. 3102, Seattle, Washington, USA

Rieffel, J. and Pollack, J. B.: 2004b, Artificial ontogenies for real world design and assembly, in M. B. et al. (ed.), *Ninth International Conference on the Simulation and Synthesis of Living Systems (ALIFE9) Workshop: Self-Organization and Development in Artificial and Natural Systems (SODANS)*, pp 37–41, MIT Press

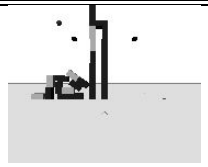
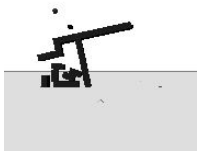
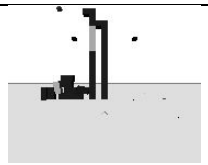
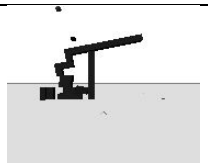
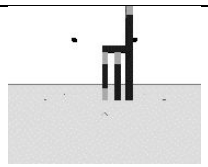
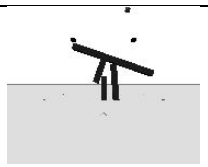
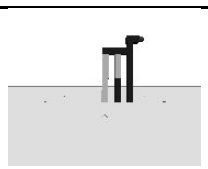
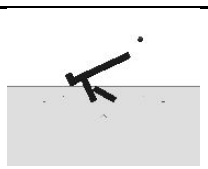
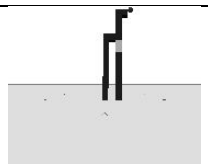
Rieffel, J. and Pollack, J. B.: 2005, Situated development: Using artificial ontogenies to evolve buildable 3-d objects, in *Genetic and Evolutionary Computation–GECCO 2004. Proceedings of the Genetic and Evolutionary Computation Conference. (to appear)*

Sims, K.: 1994, Evolving virtual creatures, in *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pp 15–22, ACM Press

Toussaint, M.: 2003, 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

Ventrella, J.: 1994, Explorations in the emergence of morphology and locomotion behavior in animated characters, in R. A. Brooks and P. Maes (eds.), *Proceedings of the 4th International Workshop on the Synthesis and Simulation of Living Systems* $ArtificialLifeIV$, pp 436–441, MIT Press, Cambridge, MA, USA

Table 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%