Computer Creativity in the Automatic Design of Robots

Jordan B. Pollack, Gregory S. Hornby, Hod Lipson* and Pablo Funes

18th July 2002

DEMO Laboratory
Computer Science Dept.,
Brandeis University,
Waltham, MA 02454, USA,
www.demo.cs.brandeis.edu
pollack,hornby,funes@cs.brandeis.edu
*School of Mechanical and Aerospace Engineering,
Cornell University,
Ithaca, NY 14853, USA
hod.lipson@cornell.edu

Abstract

The central issue addressed by this work is the ability to automatically design robots with complex morphologies and a tightly adapted control system at low cost. Inspired by nature, automatic design is achieved by using an artificial co-evolutionary process to discover the body and brain of artificial life forms simultaneously through interaction with a simulated reality. Through the use of rapid manufacturing, these evolved designs can be transferred from virtual to true reality. The artificial evolution process embedded in realistic physical simulation can create simple designs, often recognizable from the history of biology or engineering. This paper provides a brief review of three generations of these robots, from automatically designed LEGO structures, through the "GOLEM" project of electromechanical systems, to new modular designs which make use of a generative, DNA-like, representation.

1 Introduction

Traditionally, robots are designed on a hardware first, software last basis: mechanical and electrical engineers design complex articulated bodies with state-of-the-art sensors, actuators and multiple degrees of freedom; the next task is simply to “write the software”. But people have drastically underestimated the difficulty in writing control software and this, in addition to the high costs of designing and building the hardware, has led their development to a stasis [1]. Modern commercial robots perform only simple and highly repetitive manufacturing tasks with very little decision, if any, by the on-board software [2]. The central issue addressed by our work is the ability to design robots of more complex structure and more on-board intelligence at a lower cost. We suggest that this can be achieved only when robot design and construction are fully automatic.

In nature, the software we are talking about is the process of evolution. The body and brain of a creature are tightly coupled, the fruit of a long series of small mutual adaptations – like the chicken and the egg, neither one was designed first. There is never a situation in which the hardware has no software, or where a growth or mutation, beyond the adaptive ability of the brain, survives. Autonomous robots, like living creatures, require a highly sophisticated correspondence between brain, body and environment. Using natural systems as inspiration, we use evolutionary algorithms and Lindenmayer systems (L-systems) [3] to co-evolve both the brain and the body, simultaneously and continuously, from a simple controllable mechanism to one of sufficient complexity for a particular specialized task; and, create a representation scheme that allows for the hierarchical construction of more complex components from previously defined ones for the evolution of modular designs. Although we were not the first to propose brain/body coevolution [4–6], we are the first to have gone from computer simulation to reality.

The results of our work are predictive of a future in which humans are engaged in more complex and artistic forms of creativity, while the lower level details of design and interactions between components are managed by computers. In this paper, we begin to see that computer software, properly situated, can create functional forms, and further, that there are representational techniques which enable these systems to begin to scale to more complex forms in which evolved components are replicated and reused in hierarchical patterns [7, 8]. We have been working for some time on refining our understanding of the dynamics of coevolutionary learning, which, if successful, could lead to more productive self-organization of complex and artificially engineered systems [9,10]. Even so, there remains artistic choices in the construction media, the design of fitness criteria, and the selection of evolved designs to be translated into the real world.

2 Evolution algorithms

Evolutionary algorithms (EAs), a technique inspired by biological evolution [11], have been used to generate computer images [12], shapes and objects [13, 14], creature controllers [15–17] and creature morphologies [18]. An evolutionary algorithm works by keeping a population of candidate solutions, and refining the population with respect to a given "fitness function," which can provide an absolute measurement of the quality of any candidate. Through successive iterations, the EA replaces poor quality members of the population with individuals generated by applying variation to higher quality members of the population. The fitness function is an automatic way for the researcher or designer to specify their goals.

One form of evolutionary algorithms is coevolution, in which a population is not ranked by an absolute fitness function, but using a relative measure such as a comparison between different candidates. Coevolution, when successful, dynamically creates a series of learning environments each slightly more complex than the last, and a series of learners which are tuned to adapt in those environments. Sims' work [19] on body-brain coevolution, and the more recent Framsticks simulator [20], demonstrated that the neural controllers and simulated bodies could be coevolved. The goal of our research in coevolutionary robotics is to replicate and extend results from virtual simulations like these to the reality of computer designed and constructed special-purpose machines that can adapt to real environments.

We are working on coevolutionary algorithms to develop control programs operating realistic physical device simulators, both commercial-off-the-shelf and our own custom simulators, where we finish the evolution inside real embodied robots. We are ultimately interested in obtaining electro-mechanical structures that have complex morphology and place more degrees of freedom under control than anything that has ever been manually designed. We also expect these machines will have low enough engineering costs, because they employ minimal human design labor, to be produced in small quantity.

It is not feasible that controllers for complete structures could be evolved (in simulation or otherwise) without first evolving controllers for simpler constructions. Compared to the traditional form of evolutionary robotics [21–25], in which controllers are serially downloaded into a given piece of hardware, it is relatively easy to explore the space of body constructions in simulation. Realistic simulation is also crucial for providing a rich and nonlinear universe. However, while simulation creates the ability to explore the space of constructions far faster than real-world building and evaluation could, there remains the problem of transfer to real constructions and scaling to the high complexities used for real-world designs.

3 Results

The fundamental method of our work is evolution inside simulation, but in simulations more and more realistic so that the resulting blueprints are not simply visually believable, as in Sims' work, but also buildable, either manually or automatically. Our first results involved automatically creating structures with a large number of parts that could be transferred from simulation to the real world. In the second generation we evolved automatically buildable dynamic machines that are nearly autonomous in both their design and manufacture, using rapid prototyping technology. The third generation begins to address scaling, by handling complex structures through modularity. Even with these three demonstrations, we feel the work is at a very early stage, with major issues only beginning to be addressed, such as the integration of sensors and automating the feedback from "live" interactions.

3.1 Generation 1: Legobots

Our first step towards fully evolved creatures was evolving static, LEGO structures in simulation that could then be built in reality. A simulator for evolving such structures needs to satisfy the following constraints:

- Representation —- it should cover a universal space of mechanisms.
- Conservative — because simulation is never perfect, it should preserve a margin of safety.
- Efficient — it should be quicker to test in simulation than through physical production and test.
- Buildable — results should be convertible from a simulation to a real object.

The simulator we constructed considers the union between two bricks as a rigid joint between the centers of mass of each one, located at the center of the actual area of contact between them. This joint has a measurable torque capacity: more than a certain amount of force applied at a certain distance from the joint will break the two bricks apart. The fundamental assumption of our model is the idealization of the union of two LEGO bricks as a rotational joint with limited capacity.

Using this simulator, in conjunction with a simple evolutionary algorithm, we evolved complex LEGO structures, which were then manually constructed [26–28]. Our system reliably builds structures that meet fitness goals, exploiting physical properties implicit in the simulation. Building the results of the evolutionary simulation (by hand) demonstrated the power and possibility of fully automated design: the long bridge of figure 1 shows that our simple system discovered the cantilever, while the weight-carrying crane shows it discovered the basic triangular support.

3.2 Generation 2: Genetically Organized Lifelike Electromechanics (GOLEM)

The LEGO machines, with computer generated blueprints and manual construction, demonstrated that the interaction between simulated physics and evolution leads to a primitive form of discovery which can be transferred into reality. The next goal was to add motion to our designs, and address the issue of manufacture. While LEGO kits have motion components, the design space is very broad and difficult to model, and no robot can match the manual dexterity of a 10 year old human in assembly. To achieve actuated, automatically manufactured robots, we started with a whole new process in which robot morphology was constrained to be buildable by a commercial off the shelf rapid prototyping machine[1] allowing us to evolve bodies and controllers in simulation that were essentially able to transfer automatically into reality [29].

These robots are comprised of fixed bars, ball-joints and linear actuators controlled by sigmoidal neurons. The configuration of the bodies are constrained to be buildable out of thermoplastic using our rapid prototyping machine. The entire configuration is evolved for a particular task and selected individuals are printed pre-assembled (except motors), later to be recycled into different forms. In doing so, we establish for the first time a complete physical evolution cycle. In this project, the evolutionary design approach assumes two main principles: (a) to minimize inductive bias, we must strive to use the lowest level building blocks possible, and (b) we coevolve the body and the control, so that that they stimulate and constrain each other. We use arbitrary networks of linear actuators and bars for the morphology, and arbitrary networks of sigmoidal neurons for the control. Evolution is simulated starting with a soup of disconnected elements and continues over hundreds of generations of hundreds of machines, until creatures that are sufficiently proficient at the given task emerge. The simulator used in this research is based on quasi-static motion. The basic principle is that motion is broken down into a series of statically-stable frames solved independently. While quasi-static motion cannot describe high-momentum behavior such as jumping, it can accurately and rapidly simulate low-momentum motion. This kind of motion

[1]We use a Stratasys machine which "prints" layer after of layer of plastic in response to a computer generated 3D model. The largest buildable piece fits inside an 8 by 8 by 12 inch volume.

is sufficiently rich for the purpose of the experiment and, moreover, it is simple to induce in reality since all real-time control issues are eliminated. Several evolution runs were carried out for the task of locomotion. Fitness was awarded to machines according to the absolute average distance traveled over a specified period of neural activation. The evolved robots exhibited various methods of locomotion, including crawling, ratcheting and some forms of pedalism (figure 2). These forms and mechanisms often appear to be "designed" and to take advantage of our engineering vocabulary; however, they emerge from the interaction of evolution and the simulation of the potential robotic bodies and their brains.

Selected robots are then replicated into reality: their bodies are first fleshed to accommodate motors and joints, and then copied into material using rapid prototyping technology. A temperature-controlled print head extrudes thermoplastic material layer by layer, so that the arbitrarily evolved morphology emerges preassembled as a solid three-dimensional structure without tooling or human intervention. Motors are then snapped in (manually), and the evolved neural network is activated (figure 3). The robots then perform in reality much as they did in simulation.

3.3 Generation 3: Generative Representations

While the GOLEM project validated our approach to automatic design and manufacture, the machines which were produced are obviously fairly simple compared to the kinds of robots buildable by teams of human engineers. In fact most work in automatic design of engineering products, using techniques inspired by biological evolution [30–32], suffers the same criticism.

Our third generation starts to address the issue of whether evolutionary automatic design techniques can attain the higher level of complexity necessary for practical engineering projects. Ideally an automated design system would start with a library of basic parts and would iteratively create new, more complex components, from ones already in the library. Once a component is specified, the system would be able to use the component throughout the design, as well as for creating even more complex components, in the same way as the initial starting set of basic parts. To achieve this, we have developed a kind of *generative representation*, which allows for the reuse of elements in a design. A generative representation involves an encoding of candidate designs not in a direct and primitive form, but more abstractly, using something like a computer program or a grammar [33, 34].

Again we found inspiration in natural systems and chose Lindenmayer systems (L-systems) as the basis for the generative representation in which we encode designs. L-systems are a grammatical rewriting system introduced by Lindenmayer in 1968 [35] to model the biological development of multicellular organisms. An L-system consists of a set of rules for rewriting characters in strings, with rules applied in parallel to all characters in the string just as cell divisions happen in parallel in multicellular organisms. Complex strings are created from simpler strings by iteratively rewriting symbols in the string with other symbols according to the rewriting rules. Rules are of the form:

rule head	condition	successor
$a(n):$	$(n > 2)$	$\rightarrow a(n-2)\ b(n)$
$a(n):$	$(n > 0)$	$\rightarrow c\ a(n-1)$
$b(n):$	$(n > 2)$	$\rightarrow d\ a(n-1)$
$b(n):$	$(n > 0)$	$\rightarrow c\ d\ b(n-1)$

Rewriting consists of matching symbols in the string being processed with a corresponding rule for which both the rule head matches the symbol in the string and the condition part of the rule is satisfied. Symbols which do not satisfy any rewrite rule are not replaced and are copied directly to the next string. Starting from a predetermined symbol, production rules from the L-system are used either until no more apply or for a fixed number of rewriting iterations. For example, using the above set of rules and starting with an initial string consisting of the single symbol $a(4)$, the following sequence of strings are produced:

$$
\begin{gathered}
a(4) \\
a(2)b(4) \\
ca(1)da(3) \\
ca(0)da(1)b(3) \\
ca(0)dca(0)da(2) \\
ca(0)dca(0)dcca(0)
\end{gathered}
$$

The final string of characters is interpreted as an assembly procedure for constructing an object, with each symbol representing a different construction command. Thus a generative encoding is analogous to a computer language, with production rules a type of sub-procedure for specifying complex components. The end result is an evolutionary algorithm optimizing a population of L-systems, each of which is not a blueprint for a design, but an algorithm for creating the blueprint.

A graphical example of an evolved L-system is shown in figure 4.a, along with the sequence of strings it generates in figure 4.b. In our system we distinguish between symbols that can be rewritten (represented by cubes) and symbols which are later used for constructing the object (represented by spheres). In figure 4.a, a rule is graphically represented by a cube connected to a number of black spheres, each of which is followed by a sequence of symbols. The column of cubes on the left represent rule heads, and the black spheres to which they are connected represent different conditions. The sequence of symbols following each of these black spheres are the successor symbols. In addition to cubes and spheres, triangles are used to represent a repeat operator, which indicates that the symbols following it are to be repeated a given number of times. This L-system is started with the first production rule, and the sequence of strings shown in figure 4.b are the strings of symbols generated after each iteration of parallel replacement. The final string of symbols is used to construct the design.

Using this system for evolving L-systems, different design types can be generated by replacing one set of construction commands with another. So far we have evolved 3D static structures [36] (figure 5), and locomoting mechanisms called genobots [37–39]. In figure 6 we show two simulated genobots, the first of which (a) rolls along by twisting the connections between each pair of rectangle-shaped pieces and the second (b) moves by undulating like a sea-serpent. A third genobot, shown in both simulation and reality in figure 7, moves by using its four legs to walk.

4 Discussion and Conclusion

Can evolutionary and coevolutionary techniques be used in the design of real robots as "artificial lifeforms?" In this paper we have presented three generations of our work, each of which addresses one or more dimensions of the problem. We have overviewed research in use of simulations for handling high part-count static structures that are buildable, dynamic electromechanical systems with complex morphology that can be built automatically, and generative encodings as a means for scaling to complex structures.

The limitations of the work are clearly apparent: these machines do not yet have sensors, and are not really interacting with their environments. Feedback from how robots perform in the real world is not automatically fed-back into the simulations, but requires humans to refine the simulations and constraints on the design system. Finally, there is the question of how complex a simulated system can be before the errors generated by transfer to reality are overwhelming.

We cannot claim immediate solution to these problems. There is work in evolutionary robotics (e.g. by Jakobi [40]) which integrates sensors into the evolutionary mix and we have already demonstrated fixed-morphology robots with sensors learning in simulation [41] and through interactions in the real environment [42]. In our next generations of evolved creatures we expect to see some sensor integration into our system allowing us to coevolve the morphology and controllers of reactive robots.

As for complexity overwhelming the process, it is not yet a problem because the complexity of what can be evolved is still low. It is a primary research topic for us, and we have been studying how coevolution can lead to complex performance in domains like game-playing [43], and the design of complex algorithms like sorting networks [44] and cellular automata [45].

The issue of whether or not this kind of artificial life work will ever be practical and scalable is best related to the history of computer chess. The theory that machines could play a game like chess came in the 1920's. This was followed by the first chess playing computer in the mid 1950's, which played by making random legal moves. While proponents of funding for the new field of AI were over-optimistic, by the end of the century Deep Blue was able to win a tournament against the leading human player, using almost unlimited CPU time and 80 year-old theory [46].

Perhaps the small demonstrations of automatic design will lead – with continued development, and increases in computer speed and simulation fidelity, coupled to increases in basic theory of coevolutionary dynamics and representation schemes – over time, to the point where fully automatic design is taken for granted, much as computer aided design is taken for granted in manufacturing industries today.

Our current research moves towards the overall goal via multiple interacting paths of simulation, theory, building and testing in the real world, and applications. It is a broad, multidisciplinary long-term endeavor, where what we learn in one path aids the others. We believe that such a broad endeavor is the only way to ultimately construct complex autonomous machines which can economically justify their own existence.

Bio's and Acknowledgements

Jordan Pollack is a computer science professor at Brandeis University. He received his Ph.D from University of Illinois in 1987. He directs the DEMO laboratory, which is known for work in evolution, robotics, learning, and complex systems.

Gregory Hornby has just received his Ph.D in computer science from Brandeis University in 2002.

Hod Lipson is a mechanical engineering professor at Cornell University. He received his Ph.D from Technion in 1998, and was a postdoctoral researcher at Brandeis University through 2001.

Pablo Funes received his Ph.D from Brandeis University in 2000, and is now a staff member at Icosystem.

This research was supported over the years in part by the National Science Foundation (NSF), the office of Naval Research (ONR), and by the Defense Ad-

vanced Research Projects Agency (DARPA).

References

[1] H. P. Moravec. Rise of the robots. *Scientific American*, pages 124–135, December 1999.

[2] J. Morrison and T. Nguyen. On-board software for the mars pathfinder microrover. In *Proceedings of the Second IAA International Conference on Low-Cost Planetary Missions*. John Hopkins University Applied Physics Laboratory, April 1996.

[3] P. Prusinkiewicz and A. Lindenmayer. *The Algorithmic Beauty of Plants*. Springer-Verlag, 1990.

[4] D. Cliff and J. Noble. Knowledge-based vision and simple visual machines. *Philosophical Transactions of the Royal Society of London: Series B*, 352:1165–1175, 1997.

[5] W. Lee, J. Hallam, and H. Lund. A hybrid gp/ga approach for co-evolving controllers and robot bodies to achieve fitness-specified tasks. In *Proceedings of IEEE 3rd International Conference on Evolutionary Computation*, pages 384–389. IEEE Press, 1996.

[6] H. Lund, J Hallam, and W. Lee. Evolving robot morphology. In *Proceedings of IEEE Fourth International Conference on Evolutionary Computation*, pages 197–202. IEEE Press, 1997.

[7] P. J. Angeline and J. B. Pollack. Coevolving high-level representations. In *Artificial life III*, pages 55–71. Addison-Wesley, 1994.

[8] G. S. Hornby and J. B. Pollack. Evolving L-systems to generate virtual creatures. *Computers and Graphics*, 25(6):1041–1048, 2001.

[9] H. Juillé and J. B. Pollack. Dynamics of co-evolutionary learning. In *Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, pages 526–534. MIT Press, 1996.

[10] S. Ficici and J. B. Pollack. Challenges in coevolutionary learning: Arms-race dynamics, open-endedness, and mediocre stable states. In Adami, Belew, Kitano, and Taylor, editors, *Proceedings of the Sixth International Conference on Artificial Life*, 1998.

[11] J. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, 1975.

[12] K.Sims. Artificial evolution for computer graphics. In *Computer Graphics (Siggraph '91 proceedings)*, pages 319–328, 1991.

[13] R. Dawkins. *The Blind Watchmaker*. W. W. Norton, New York, 1987.

[14] P. J. Bentley. *Generic Evolutionary Design of Solid Objects using a Genetic Algorithm*. PhD thesis, Division of Computing and Control Systems, School of Engineering, The University of Huddersfield, 1996.

[15] M. van de Panne and E. Fiume. Sensor-actuator Networks. In *SIGGRAPH 93 Conference Proceedings*, Annual Conference Series, pages 335–342, Anaheim, California, August 1993. In *Computer Graphics* Annual Conf. Series, 1993.

[16] L. Gritz and J. K. Hahn. Genetic programming for articulated figure motion. *Journal of Visualization and Computer Animation*, 6(3):129–142, July 1995.

[17] R. Grzeszczuk and D. Terzopoulos. Automated learning of muscle-actuated locomotion through control abstraction. In *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 63–70, Los Angeles, California, August 1995. In *Computer Graphics* Annual Conf. Series, 1995.

[18] K. Sims. Evolving virtual creatures. In *Computer Graphics*. Annual Conference Series, 1994.

[19] K Sims. Evolving 3d morphology and behavior by competition. In R. Brooks and P. Maes, editors, *Proceedings 4th Artificial Life Conference*, pages 28–39. MIT Press, 1994.

[20] M. Komosinski and S. Ulatowski. Framsticks: Towards a simulation of a nature-like world, creatures and evolution. In Jean-Daniel Nicoud Dario Floreano and Francesco Mondada, editors, *Proceedings of 5th European Conference on Artificial Life (ECAL99)*, volume 1674 of *Lecture Notes in Artificial Intelligence*, pages 261–265. Springer-Verlag, 1999.

[21] D. Floreano and F. Mondada. Evolution of homing navigation in a real mobile robot. *IEEE Transactions on Systems, Man, and Cybernetics*, 1996.

[22] D. Cliff, I Harvey, and P. Husbands. Evolution of visual control systems for robot. In M. Srinivisan and S. Venkatesh, editors, *From Living Eyes to Seeing Machines*. Oxford, 1996.

[23] H. Lund. Evolving robot control systems. In Alexander, editor, *Proceedings of 1NWGA*. University of Vaasa, 1995.

[24] J. C. Gallagher, R. D. Beer, K. S. Espenschield, and R. D. Quinn. Application of evolved locomotion controllers to a hexapod robot. *Robotics and Autonomous Systems*, 19:95–103, 1996.

[25] Y. Kawauchi, M. Inaba, and T. Fukuda. Genetic evolution and self-organization of cellular robotic system. *JSME Int. J. Series C. (Dynamics, Control, Robotics, Design & Manufacturing)*, 38(3):501–509, 1999.

[26] P. Funes and J. B. Pollack. Computer evolution of buildable objects. In Phil Husbands and Inman Harvey, editors, *Fourth European Conference on Artificial Life*, pages 358–367, Cambridge, 1997. MIT Press.

[27] P. Funes and J. B. Pollack. Evolutionary body building: Adaptive physical designs for robots. *Artificial Life*, 4(4):337–357, 1998.

[28] P. Funes and J. B. Pollack. Computer evolution of buildable objects. In P. Bentley, editor, *Evolutionary Design by Computers*, pages 387 – 403. Morgan-Kaufmann, San Francisco, 1999.

[29] H. Lipson and J. B. Pollack. Automatic design and manufacture of robotic lifeforms. *Nature*, 406(6799):974–978, 2000.

[30] C. Kane and M. Schoenauer. Genetic operators for two-dimentional shape optimization. In J.-M. Alliot, E. Lutton, E. Ronald, M. Schoenauer, and D. Snyers, editors, *Artificiale Evolution - EA95*. Springer-Verlag, 1995.

[31] P. Husbands, G. Germy, M. McIlhagga, and R. Ives. Two applications of genetic algorithms to component design. In T. Fogarty, editor, *Evolutionary Computing. LNCS 1143*, pages 50–61. Springer-Verlag, 1996.

[32] P. Bentley, editor. *Evolutionary Design by Computers*. Morgan-Kaufmann, San Francisco, 1999.

[33] J. Koza. *Genetic Programming*. MIT Press, 1992.

[34] F. Gruau. *Neural Network Synthesis Using Cellular Encoding and the Genetic Algorithm.* PhD thesis, Ecole Normale Supérieure de Lyon, 1994.

[35] A. Lindenmayer. Mathematical models for cellular interaction in development. parts I and II. *Journal of Theoretical Biology*, 18:280–299 and 300–315, 1968.

[36] G. S. Hornby and J. B. Pollack. The advantages of generative grammatical encodings for physical design. In *Congress on Evolutionary Computation*, pages 600–607, 2001.

[37] G. S. Hornby, H. Lipson, and J. B. Pollack. Evolution of generative design systems for modular physical robots. In *IEEE International Conference on Robotics and Automation*, pages 4146–4151, 2001.

[38] G. S. Hornby and J. B. Pollack. Body-brain coevolution using l-systems as a generative encoding. In *Genetic and Evolutionary Computation Conference*, pages 868–875, 2001.

[39] G. S. Hornby and J. B. Pollack. Creating high-level components with a generative representation for body-brain evolution. *Artificial Life*, to appear.

[40] N. Jakobi. *Minimal Simulations for Evolutionary Robotics*. PhD thesis, School of Cognitive and Computing Sciences, University of Sussex, May 1998.

[41] G. S. Hornby, S. Takamura, O. Hanagata, M. Fujita, and J. Pollack. Evolution of controllers from a high-level simulator to a high dof robot. In J. Miller, editor, *Evolvable Systems: from biology to hardware; Proc. of the Third Intl. Conf. (ICES 2000)*, Lecture Notes in Computer Science; Vol. 1801, pages 80–89. Springer, 2000.

[42] R. Watson, S. Ficici, and J. B. Pollack. Embodied evolution: Embodying an evolutionary algorithm in a population of robots. In P. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzala, editors, *1999 Congress on Evolutionary Computation*, 1999.

[43] J. B. Pollack and A. D. Blair. Coevolution in the successful learning of backgammon strategy. *Machine Learning*, 32:225–240, 1998.

[44] H. Juillé and J. B. Pollack. Co-evolving intertwined spirals. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 351–358, 1996.

[45] H. Juillé and J. B. Pollack. Coevolving the "ideal trainer": Discovery of cellular automata rules. In Koza, editor, *Proceedings Third Annual Genetic Programming Conference*, July 1998.

[46] M. Newborn. *Kasparov vs. Deep Blue: Computer Chess Comes of Age*. Springer Verlag, New York, 1996.

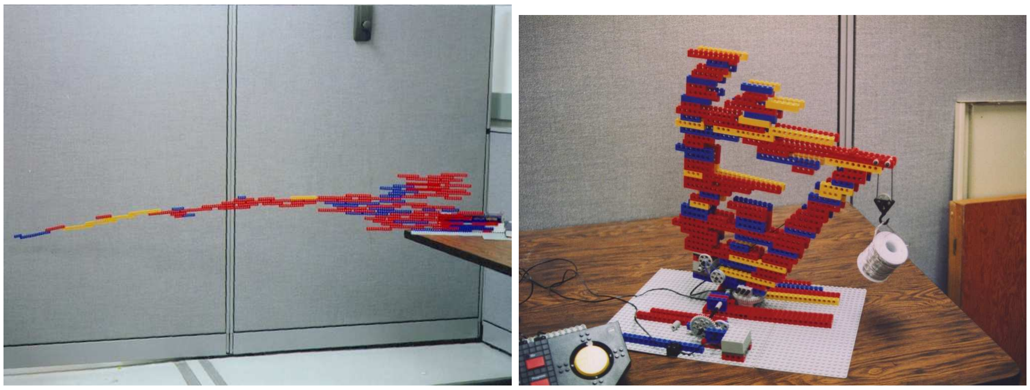
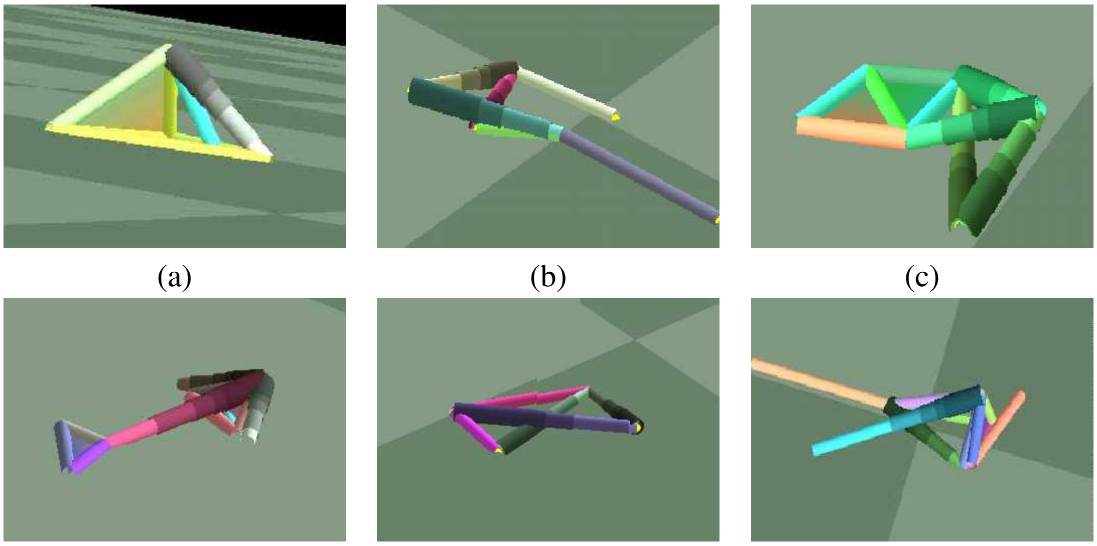


Figure 1: Photographs of the FAD LEGO Bridge (Cantilever) and Crane (Triangle). Photographs copyright Pablo Funes & Jordan Pollack, used by permission.



(a) (b) (c)

(d) (e) (f)

Figure 2: (a) A tetrahedral mechanism that produces hinge-like motion and advances by pushing the central bar against the floor. (b) Bipedalism: the left and right limbs are advanced in alternating thrusts. (c) Moves its two articulated components to produce crab-like sideways motion. (d) While the upper two limbs push, the central body is retracted, and vice versa. (e) This simple mechanism uses the top bar to delicately shift balance from side to side, shifting the friction point to either side as it creates oscillatory motion and advances. (f) This mechanism has an elevated body, from which it pushes an actuator down directly onto the floor to create ratcheting motion. It has a few redundant bars dragged on the floor. Images copyright Hod Lipson & Jordan Pollack, used by permission.

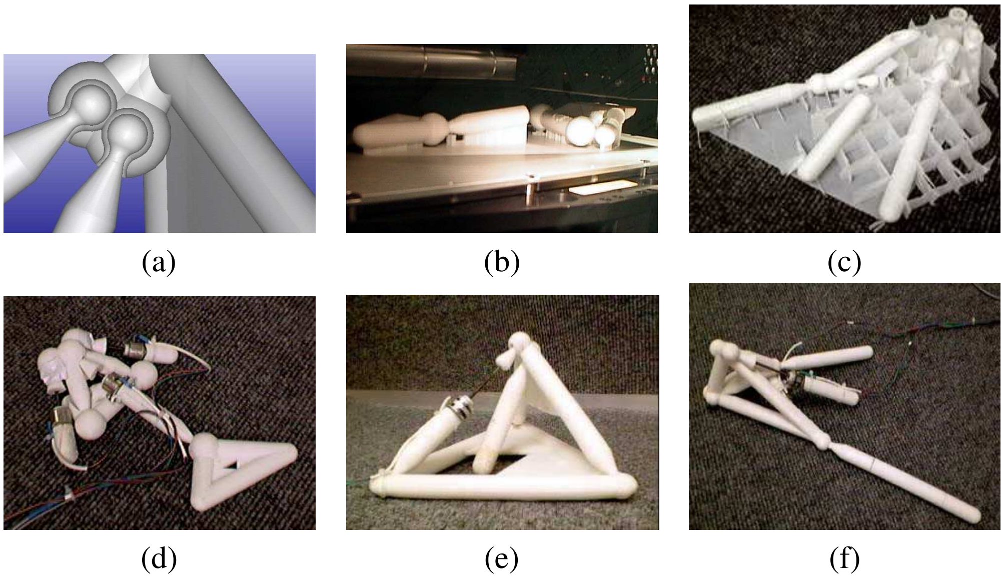
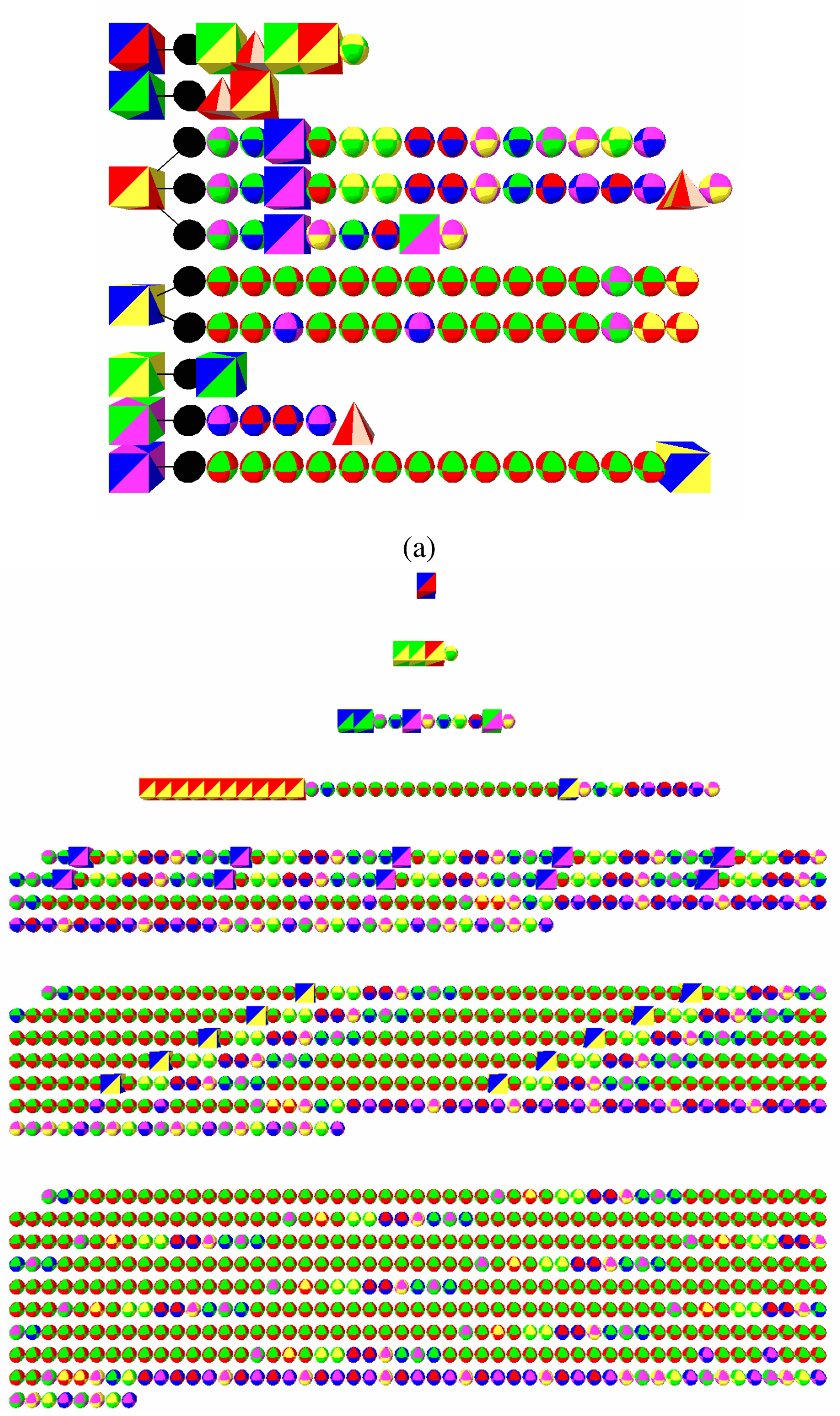


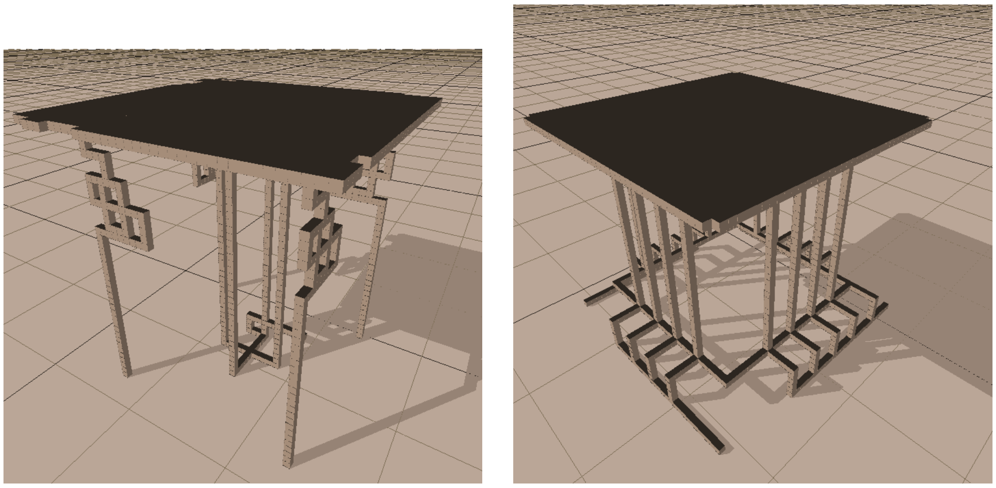
Figure 3: (a) Fleshed joints, (b) replication progress, (c) pre-assembled robot (figure 2f), (d,e,f) final robots with assembled motors. Images copyright Hod Lipson & Jordan Pollack, used by permission.



(a)

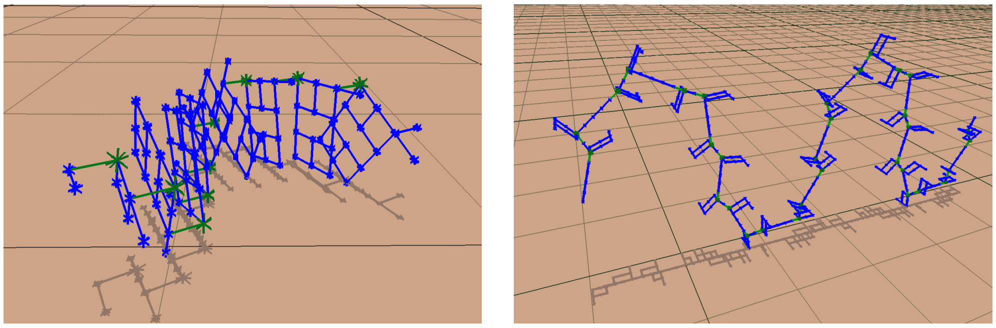
(b)

Figure 4: Example of an evolved L-system (a) along with the sequence of strings that it generates (b). Images copyright Gregory Hornby & Jordan Pollack, used by permission.



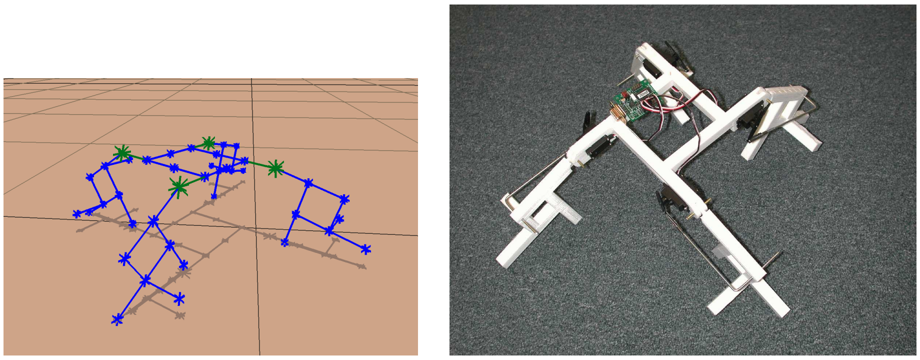
(a) (b)

Figure 5: Two tables evolved using a generative representation. Images copyright Gregory Hornby & Jordan Pollack, used by permission.



(a) (b)

Figure 6: Two examples of evolved genobots. Images copyright Gregory Hornby & Jordan Pollack, used by permission.



(a) (b)

Figure 7: A 4-legged, evolved genobot shown in simulation (a) and reality (b). Images copyright Gregory Hornby & Jordan Pollack, used by permission.

List of Figures