Crawling Before Walking: Improving Walking Soft Robot Evolution

Julian Segert

Abstract

Walking is a classic task for both evolving and engineered robots. The task is complex yet easy to score and holds much biological and practical relevance. Here, we look specifically at evolving voxel-based soft robots and how current evolutionary algorithms can be improved to better emulate biological systems and develop more sophisticated locomotor behavior. We assess and attempt to remedy the shortcomings of previous studies on evolving soft robots using compositional pattern-producing networks (CPPNs). We found that CPPNs are not necessary for the emergence of bilateral symmetry, although the relatively more complex CPPN networks performed better than traditional NEAT. We also propose and outline implementation of a biologically-inspired soft robot using a system of coevolving central pattern generators (CPGs).

1 Introduction

1.1 Background

The field of robotics has reached a dramatic turning point. It has become difficult for human engineers to program robots with starkly different topologies and sensors from themselves. Moreover, while robot controllers are adept at mathematical computations, it has proven very difficult to imbue them with many of the functions that come naturally to humans and other animals, such as vision or fluid and efficient movement. Instead, many computer scientists in the field have appealed to the forces of evolution with artificial genetic encodings of robot controllers and morphologies. Using evolutionary algorithms, random mutations are introduced into the robots in study, allowing for novel solutions engineers would have a hard time imagining. The best performing individuals can then be selected, and their genotypes recombined with other champions to introduce more genetic diversity. This process continues to run, optimizing solutions over many generations.

One of the most popular and effective evolutionary algorithms is known as NeuroEvolution of Augmenting Topologies (NEAT) [5]. As the name suggests, this algorithm is specifically for evolving artificial neural networks. One of the key strengths of NEAT is that it complexifies automatically, that is to say, neural networks start out simple, but new edges and hidden nodes are added as needed to increase the computational capacity of the network. NEAT is also effective because it allows and actively encourages speciation of "chromosomes," the encoding mechanism for its neural networks. This allows diverse solutions to emerge in each run without a slight edge early on causing one genotype to dominate.

One variant of the basic NEAT algorithm is the so-called compositional pattern-producing network (CPPN) [6]. This variant is designed to model biological development and induce complex patterned structures. It is typically implemented using a modified NEAT algorithm to allow for the same sophisticated evolutionary protocols, thus creating CPPN-NEAT.

CPPN-NEAT is a staple platform for evolving biologically-inspired walking robots. This algorithm was used in a previous study by Cheney et al. [3] that evolved walking behavior in the voxel-based softbots which form the basis of this study. An evolving CPPN-NEAT neural network is used to determine the morphology of these softbots, with each block being either empty, phase1 muscle, phase2 muscle, hard passive, or soft passive type. The muscle blocks expand and contract in relation to the temperature in the environment, and the two types are in opposite phase to each other (i.e. one expands while the other contracts). Softbots were evaluated based on the Euclidean distance of the center of mass over a fixed time interval when run in the soft body physics simulator voxelyze. Using this protocol, Cheney et al. successfully evolved robots that could traverse flat terrain. They further demonstrated the robustness of this evolutionary protocol by evolving these softbots under constraints, penalizing them for the number of total or actuated voxels. The robots managed to adapt to these constraints with only marginal drops in overall speed.

1.2 Limitations of Previous Research

While these results are interesting, previous studies lack much in the way of practical and biological consideration. First is the issue of what we will call "ghost" controllers. We use this phrase to describe robots whose central controller is not rooted in the same physical space as the robot itself. Take, for example, the iconic work by Karl Sims [4] which coevolved the morphology of locomoting robots along with controllers. Controllers are implemented as neural networks nested within each piece of the robots that can control the action of associated joints. This implementation ignores the physical presence of controllers within the robot and instead actuates them from disembodied ghosts.

The softbots created by Cheney et al. fall victim to the ghost controller issue. These robots in fact lack controllers entirely, but one can think of the rhythmic activation of muscles as a simple and abstracted form of controller. Ghost controllers are suboptimal because they trivialize the idea of embodied intelligence which states that intelligence emerges as result of the complex interactions between a central controller, a physical body, and the environment. With the controller abstracted into a different physical plane as the body and the environment, the emergence of embodied intelligence is jeopardized. Moreover, this a poor approximation of biological systems, as neural controllers are inextricably linked to the physical body and evolve under the same mechanisms. Lastly, ghost controllers are impractical if one intends to port robots evolved in simulation into physical ones. In practice, robots need onboard processors, at the very least to execute remote commands and control their associated motors.

Another problem with previous research is the reliance on CPPN-NEAT to induce symmetric solutions, and the lack of recognition for selection's role in the emergence of bilateral symmetry. Here, we explore the differences between NEAT and CPPN-NEAT evolved walking softbots in both performance and symmetry. We also propose a model for evolving softbots with decentralized controller blocks, inspired by the central pattern generators used in vertebrate locomotion.

Auerbach and Bongard [1] used a similar CPPN-NEAT implementation to evolve bilaterally symmetric and intricately patterned robots for a movement task. The task was slightly different, because it focused on the morphological changes necessary to navigate surfaces with reduced friction ("ice"). This study showed that CPPNs generated more complex shapes to deal with the more complex task of traversing slippery terrain. However, it did not establish that CPPNs produce bilaterally symmetric objects on their own, since the objects generated were reflected over their short axis to ensure bilateral symmetry. Because Stanley's original CPPN paper [6] involved a

human experimenter selecting pictures, to the best of our knowledge, the evidence that CPPNs are more prone to establishing bilateral symmetry than are other algorithms is severely lacking.

The experiments described in this paper follow closely from the experiments described in the Cheney et al. paper [3], and much of the data are from modified versions of their provided source code.

2 Experiment 1: Traditional NEAT Control

2.1 Motivation and Procedure

In addition to the direct-encoded genetic algorithm control Cheney et al. used, we run the same evolutionary algorithm, except with what we will refer to as a "traditional" NEAT-evolved morphology. To clarify, the Cheney et al. used CPPN-NEAT evolved neural networks to determine morphology. The networks took the x, y, and z coordinates of each voxel, as well as the distance to the center. The CPPN networks used hidden nodes with one of several types of activation function. The network outputted values between 0 and 1 for each material type. The material that received the highest output for the given coordinate was placed into that position in the final phenotype. The solutions that evolved from this protocol tended towards bilateral symmetry with coordinated regions of the same material types. This result was compared to a run of a direct-encoded GA evaluated using the same criteria. Perhaps not surprisingly, the direct-encoded control performed poorly, and the evolved morphologies were largely uncoordinated. However, this does not establish that CPPNs are necessary for the emergence of patterned, symmetric structures. A more meaningful comparison is to traditional NEAT, which we define in this situation as a NEAT instance which takes only the x, y, z coordinates of each voxel and only uses nodes with sigmoid activation functions. This is a better control because it better isolates the difference between NEAT and CPPN-NEAT, whereas the direct-encoded GA control was confounded by the vast difference in evolutionary protocols and algorithmic complexity.

For our control runs, we ran 500 generations of the softbots experiment from a modified version of the available source code in the orignal experiment [3]. These results were compared to analagous runs of 500 generations of the unmodified CPPN-NEAT experiment. Experiments were also run with penalties as in the original paper. Details on experimental parameters can be found in the appendix. Unconstrained trials were performed in duplicate, averages are shown. We did not replicate the direct-encoded GA runs, but the results from Cheney et al. definitively demonstrated that this approach substantially underperforms both NEAT and CPPN-NEAT.

2.2 Results

Run without constraints, traditional NEAT slightly outperformed CPPN-NEAT in both average and generation champion (genChamp) fitness (fig. 1). Peak fitnesses were nearly identical (69.43cm for NEAT versus 69.04 for CPPN-NEAT), but NEAT was dramatically faster to optimize. Moreover, the CPPN champion was more of an outlier than the respective NEAT champion, which is apparent by the more dramatic difference best and average fitness for CPPN compared to NEAT runs.

When run with a penalty for the total number of voxels, CPPN-NEAT and NEAT were not substantially different (fig. 1). For both, the averages showed a characteristic bimodal pattern, with NEAT peaking slighly earlier but reaching similar maxima. In terms of genChamp fitness,

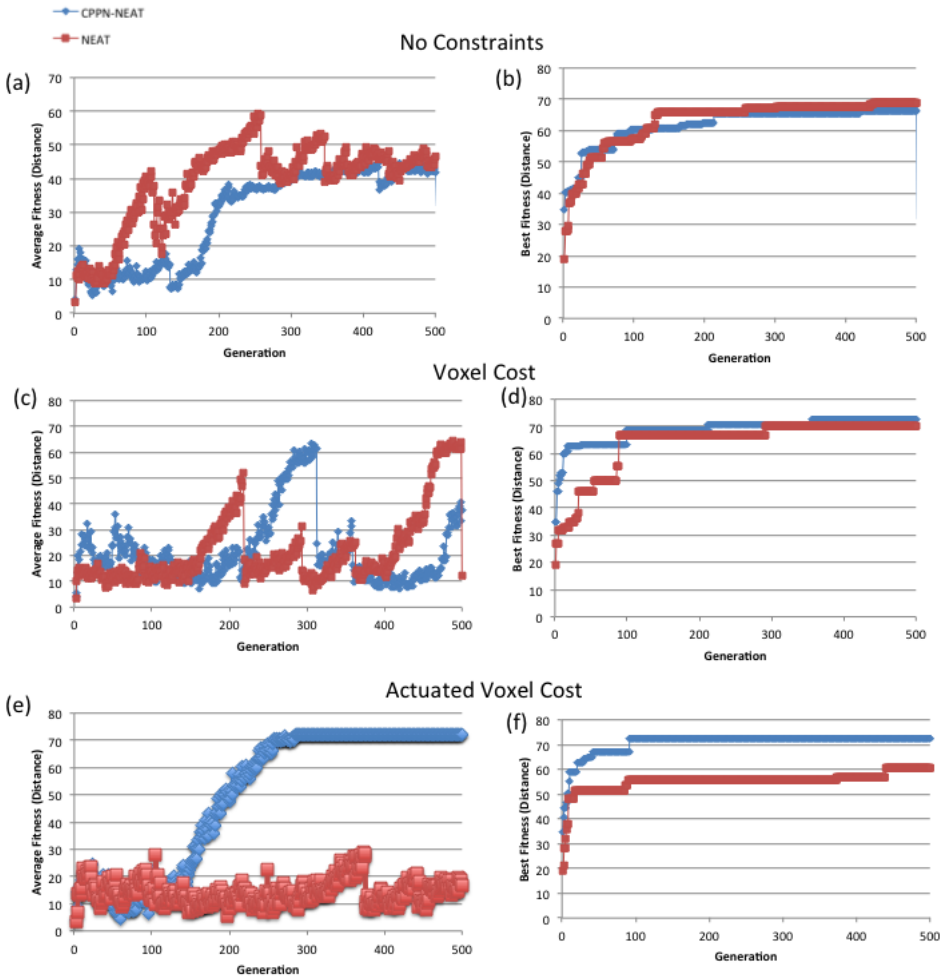

Figure 1: Average and best fitness scores for CPPN-NEAT and NEAT evolved voxel softbots. Left side is average fitness, right is the fitness of the genChamp for each generation.

CPPN-NEAT narrowly edged out traditional NEAT. CPPN-NEAT's relative fitness was more dramatic when solutions for the number of actuated voxels. Here, the CPPN genChamp consistently outperformed NEAT, and NEAT's average fitness did not increase appreciably over the entire 500 generations, while CPPN's average increased steadily before leveling at a fitness over 70cm.

Phenotypically, CPPN-NEAT evolved along similar lines. In all variations, the highest-performing CPPN-NEAT solutions took the form of an approximate sphere divided into two sections, the larger of which is a solid mass of contracting muscle, and the smaller is hard passive type at the back of the robot (fig. 2). Traditional NEAT solutions were more diverse; some strongly resembled the characteristic sphere softbot from CPPN-NEAT, although the form was often slightly less symmetric or the edges of material types were less well defined. Another common solution among NEAT networks was a staircase-like shape with a section of muscle on the bottom and section of soft passive material on top (fig. 3).

As far as complexity, CPPN-NEAT consistently produced more complex solutions than did

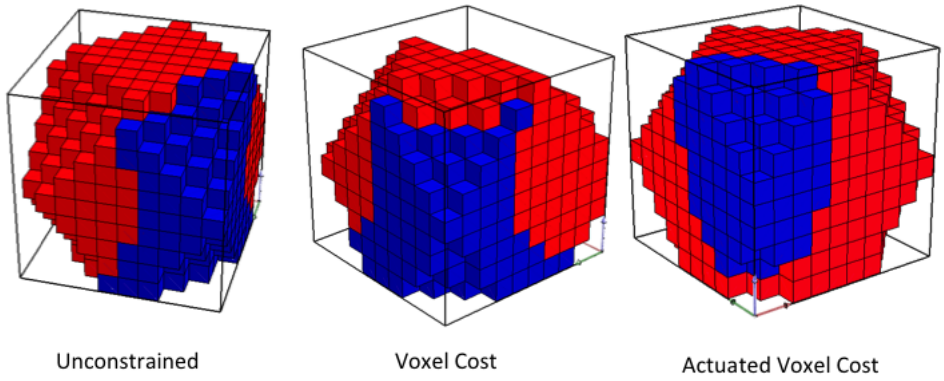

Figure 2: Characteristic phenotypes for CPPN-NEAT evolved softbots. Red is phase1 muscle, blue is hard passive.

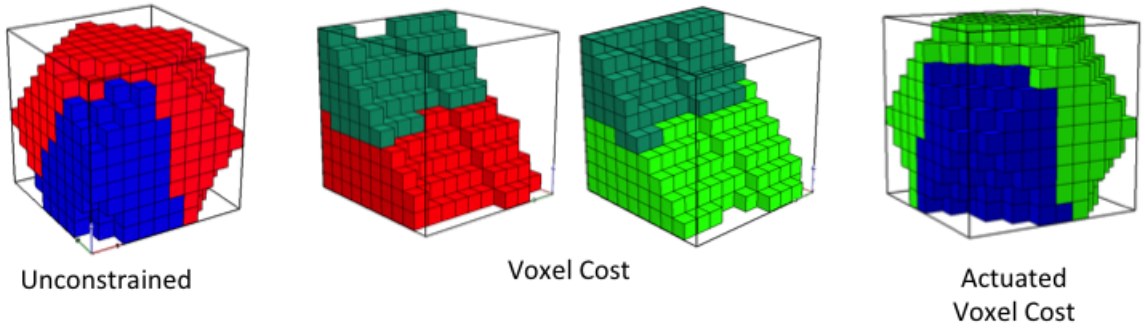

Figure 3: Characteristic phenotypes for traditional NEAT evolved softbots. Red is phase1 muscle, light green is phase2 muscle, blue is hard passive, and dark green is soft passive.

standard NEAT in all three conditions (fig. 4). The two networks were configured with the same probabilities of adding nodes, although CPPN-NEAT does have an additional input node. For CPPN-NEAT, the most complex networks arose from the voxel cost constraint, followed by the unconstrained condition and actuated voxel cost constraint. Traditional NEAT developed the most complex networks when unconstrained, followed by voxel cost constraint and then actuated voxel cost constraint, although this condition had a rapid spike in complexity near the end of the run (fig. 4).

3 Experiment 2: Coevolving Softbots with Controllers

3.1 Motivation

In the animal kingdom, motor control is shockingly decentralized. The brain is typically responsible only for high-level commands that specify a type of movement, but encodes nothing about the mechanism by which to execute these commands. More localized structures are necessary for this.

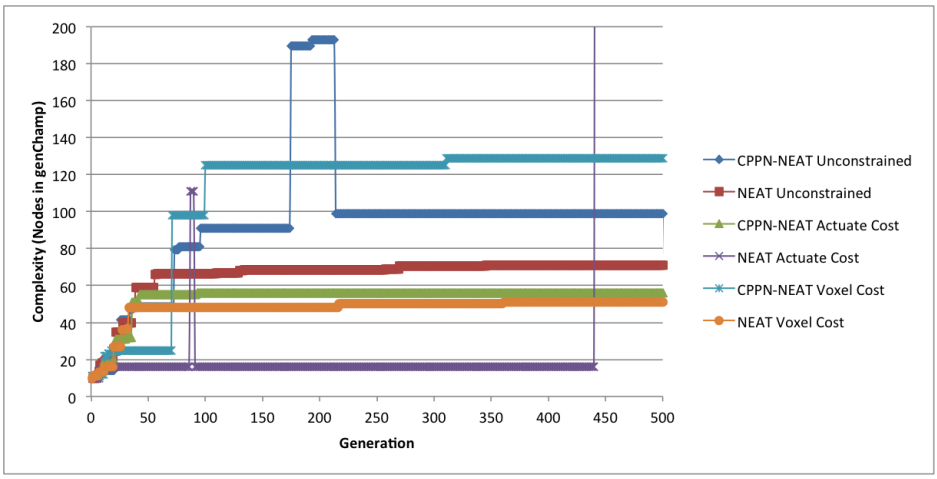

Figure 4: Complexity of evolved networks, defined as the number of nodes in the genChamp for each generation. NEAT actuate cost trace extends above the top of the axis to 500.

For vertebrates, these structures are the central pattern generators (CPGs), which coordinate the rhythmic contractions of muscles needed for locomotion. A network of decentralized controllers turns out to be an extremely efficient control scheme. Vertebrates have evolved central pattern generators, and cephalopods evolved the similar local pattern generators independently. Neurons are energetically expensive, so there is selective pressure against having more complex nervous systems than are necessary. By extension, we can conjecture that decentralized motor control mechanisms are efficient because they have been favored by evolution.

Biological CPGs achieve rhythmic output by systems of mutually inhibitory neurons that fatigue so that the two are activated in an alternating manner [2]. In order to emulate this in a NEAT evolved neural networks, NEAT would be parameterized so that it is not strictly feed-forward, and rhythmic output would hopefully evolve on its own.

Adding CPG blocks to the morphology of the robot also helps to address the ghost controller problem. The controllers exist alongside the rest of the body in physical space. Moreover, the decentralized architecture of this approach also allows muscle blocks to integrate inputs from multiple sources, making the entire body part of the computational process, a prime example of embodied intelligence that would not be possible with a disembodied ghost controller.

3.2 The CPG Model Applied to Voxel Softbots

To see this hypothetical CPG model in action, several attempts were made to design an evolutionary algorithm along these lines. The framework was based around the Cheney et al. [3] protocol, but with some important differences. Because of time and computational constraints, most of the following protocol is hypothetical.

The morphology of the softbots is still determined by a CPPN-NEAT network in much the same way as before. However, in this iteration, voxels are either empty, muscle, or CPG. Muscles do not contract in response to temperature change, but instead in response to activation signals from neighboring CPGs. When a CPG voxel is instantiated, it goes through an initializtion step in which

it is linked to nearby CPGs and muscles. Ideally, each CPG could link to a variable number of other voxels, with the probability of linking to any particular voxel inversely proportional to Euclidean distance. However, this is very difficult to implement, and CPGs currently link to a fixed number of other voxels, 3 CPGs and 6 muscles. Linking to other CPGs allows control networks to form, similar to what is seen in biological control networks. Each CPG has nested within it an evolving NEAT neural network, which takes as input the signals from associated CPGs and outputs signals to its associated CPGs and muscles.

Muscles then integrate the activation signals from multiple CPGs. CPG outputs are scaled from -1 to 1, allowing for excitatory or inhibitory signals. Muscle blocks have linear activation functions, mimicking biological muscle bundles which contract roughly linearly with neural activation. The one aspect of this model that is glaringly out of line with analogous biological systems is the way in which muscle blocks can expand and contract, whereas biological muscles can only contract. One can, however, abstract this as saying every region of muscle voxels is analagous to a pair of opposing muscles which are usually found in close proximity in biological systems.

What evolves in this experiment is a meta-genotype consisting of the morphology-determining CPPN-NEAT network as well as the associated NEAT networks within each CPG. Because of the use of nested neural networks and the complexity of the task these robots must learn, the computational time of running this experiment would be staggering. Because the algorithm was never completed, and insufficient time and computational resources were afforded, there are no data to report for this experiment, although this remains an open field of investigation.

4 Discussion

4.1 NEAT Control Trials

We have seen that, without additional constraints, traditional NEAT can in fact outperform CPPN-NEAT at a walking task. This is a slightly surprising result since one would expect an algorithm designed to mimic the patterning of biological organisms to perform better at a walking, a task mastered by many animals but still challenging for robots. To that effect, NEAT softbots showed a surprising degree of bilateral symmetry, but this may be because of selective pressure from the fitness function. Fitness in this experiment was measured by the Euclidean distance from the center of mass at the beginning of the simulation to the end, and not by the total distance traveled by the softbot. One can see that a robot that moves in a snaking or curved path will get a lower fitness score than one that moves the same total distance but in a straight line. One can also see that bilateral symmetry would lead to a robot most likely to locomote in a straight line. Therefore, this task may not be an effective test for the ability of CPPNs to induce bilateral symmetry because it is an implicit part of the fitness function. An interesting comparison would be to run the same experiment but evaluation the robots based on total distance. This could be computed easily by summing the Euclidean distances traveled between each timestep.

It is interesting that CPPN-NEAT consistently generated more complex networks than traditional NEAT, although this could be partially explained by the additional input. It's also likely that a network with a wide range of possible activation functions is more difficult to optimize, so more nodes are required. It's strange that the CPPN networks in the voxel cost condition were comparatively simpler since one would imagine optimizing under harsher constraints would lead to more complex solutions. Either this is not a sound conclusion, or it is more difficult for CPPN

networks to optimize under a total voxel constraint. It's also strange that network complexity varied as much as it did considering how similar the final phenotypes turned out to be, but this could again be an artifact from having so few total lineages in the limited number of experimental runs. Auerbach [1] showed that robot topologies complexified in response to a more complex environment, but that may have been specific to that particular environment and does not apply to more complex tasks in general.

It was also unexpected that traditional NEAT developed more varied solutions than did CPPN. One would expect that the more complex networks would create more complex and varied robots, but NEAT robots were much more diverse. Maybe it was the case that CPPN was just faster to find the optimal solution to this task, while NEAT explored many dead-end paths. As seen in figure 3, NEAT demonstrated convergent evolution that was not seen in CPPN. Robots often evolved into similar morphologies from different lineages, as evidenced by the use of the two muscle types to fulfill the same role.

Computational constraints proved to be a problem. In a preliminary run, CPPN-NEAT outperformed NEAT on average which opposes the finding presented here in which this run was aggregating with a duplicate run. This highlights the importance of replication, and, ideally, we would have performed several more runs of each condition had we not been limited by time. Auerbach [1] mentioned that their evolutionary algorithm ran on a 7.1 teraflop supercomputing cluster, and that the process would have been infeasibly slow on a standard computer. The limiting factor is evaluation time in the voxelyze soft body physics simulator. Because of this, the added complexity of nested neural networks in CPG blocks may not have had a huge effect on computational time, although that does involve modifying the physics simulator which would presumably introduce a marginal runtime hit.

4.2 Implications and Future Directions

While the preceding experiments did not manage to alleviate the endemic ghost controller issue, they did provide valuable insights into the role of CPPNs in bilateral symmetry and patterning. CPPN softbots were not appreciably more complex than were NEAT softbots. If anything, they were simpler and solutions were certainly less diverse. It's hard to make claims about bilateral symmetry because all softbots were relatively simple shapes, so a high degree of symmetry should not be surprising.

The next steps would be to run more trials to reduce noise and establish statistical significance. It would also be worthwhile to run the experiment using a fitness function for total distance traveled and observe the effect on bilateral symmetry. To better investigate this, it would be useful to develop a quantifiable measure of bilateral symmetry. This would be relatively simple to do by running a χ^2 test between the sides of the line of symmetry on the abundance of voxel types, but the test should also account for simplicity of the overall structures, which would influence the likelihood of strong symmetry.

Ideally also, more of the control mechanisms would also be left to evolution instead of being hard-coded into the simulator. Adding central pattern generators is a good start towards a more intelligent softbot, and onto that could be added sensory structures and a central controller. Softbots that evolve to walk only on flat ground show little in the way of adaptation or intelligence because the environment and task are the same every time. It would be more interesting and meaningful if the environment contained obstacles that were randomized in every iteration and the robots learned to sense and respond to these using tactile sensory voxels and a network of control voxels.

Moreover, an objective-driven central controller would fulfill all the components of an embodied intelligence.

Acknowledgements

I would like to thank Professor Lisa Meeden for her instruction and support, Jeff Knerr for his technical assistance, and Nick Cheney for his valuable advice and source code.

References

[1] Joshua E. Auerbach and Josh C. Bongard. On the relationship between environment and morphological complexity in evolved robots. *GECCO*, 2012.

[2] Auke Jan Ijspeert. Central pattern generators for locomotion control in animals and robots: A review. *Neural Networks*, 21:642–653, 2008.

[3] Jeff Clune Nick Cheney, Robert MacCurdy and Hod Lipson. Unshackling evolution: Evolving soft robots with multiple materials and a powerful generative encoding. *GECCO*, 2013.

[4] Karl Sims. Evolving virtual creatures. *ACM*, 1994.

[5] Kenneth Stanley. Competitive coevolution through evolutionary complexification. *Journal of Artificial Intelligence Research*, 21, 2004.

[6] Kenneth Stanley. Compositional pattern producing networks: A novel abstraction of development. *Genet Program Evolvable Mach*, 8:131–162, 2007.

Appendix: NEAT Parameters

Parameter	Setting
Population size	30
Max generations	500
Tournament size	2
Random seed	-1
Species size target	1.0
Survival threshold	1.0
Compatability threshold	60.0
Compatibility modifier	30
Disjoint coefficient	2.0
Excess coefficient	20
Weight difference coefficient	1.0
Age significance	1.0
Adult link age	1.0
Dropoff age	15000
Mutate add node probability	0.1
Mutate add link probability	0.1
Mutate demolish link probability	0.03
Mutate link weights probability	0.95
Mutate link probability	0.8
Mutation power	0.2
Link gene minimum wight for phenotype	0.0
Mutate only probability	0.25
Smallest species size with elitism	1.0
Force copy generation champion	1.0
Allow recurrent connections	0.0
Add bias to hidden nodes	0.0
Signed activation	1.0
Bounding box x, y and z	10
Actuations per second	20
Num actuation cycles	50
Min percent voxels filled	0.05

Table 1: CPPN-NEAT Parameters used in experiments. NEAT parameters were the same except it only used sigmoid activation functions and did not incorporate the distance from center input.