Dynamic Resolution in the Co-Evolution of Morphology and Control

Joshua E. Auerbach[1] and Josh C. Bongard[1]

[1]Morphology, Evolution and Cognition Laboratory
Department of Computer Science
University of Vermont
Burlington, VT 05405
joshua.auerbach@uvm.edu

Abstract

Evolutionary robotics is a promising approach to overcoming the limitations and biases of human designers in producing control strategies for autonomous robots. However, most work in evolutionary robotics remains solely concerned with optimizing control strategies for existing morphologies. By contrast, natural evolution, the only process that has produced intelligent agents to date, may modify both the control (brain) and morphology (body) of organisms. Therefore, co-evolving morphology along with control may provide a better path towards realizing intelligent robots. This paper presents a novel method for co-evolving morphology and control using CPPN-NEAT. This method is capable of dynamically adjusting the resolution at which components of the robot are created: a large number of small sized components may be present in some body locations while a smaller number of larger sized components is present in other locations. Advantages of this capability are demonstrated on a simple task, and implications for using this methodology to create more complex robots are discussed.

Introduction

There are many reasons why it would be useful to have autonomous robots operating in our homes and offices. These range from freeing people from repetitive tasks to the ability to perform actions that humans are incapable of. However, with the exception of a few robots designed to accomplish simple tasks, the vast majority of autonomous robots currently in use operate only in factories and other highly structured environments. In order to make the migration out of the factories and into our everyday lives robots will need to be adaptive and exhibit intelligent behavior.

There has been much work in recent years in the area of embodied artificial intelligence (Brooks, 1999; Anderson, 2003; Pfeifer and Bongard, 2006; Beer, 2008) which has led to the conclusion that such intelligent behavior must arise out of the coupled dynamics between an agent's body, brain and environment. This means that the complexity of an agent's controller and morphology must increase commensurately with the task or tasks that it is required to perform. However, when designing complex autonomous robots it is often not clear how responsibility for different behaviors should be distributed across an agent's controller and morphology. A good example of this is that if a robot is solely tasked with moving over flat terrain while following a light source then wheels and a direct sensory motor mapping are an appropriate solution (Braitenberg, 1986), but if the robot must be able to navigate over varied terrains while performing more complicated tasks a more complex control strategy and/or morphology are required. This issue of scaling up morphological and control complexity has been a major obstacle in developing autonomous robots capable of operating in most real world situations.

Background

The only truly intelligent agents to have yet existed, as far as we are aware, are biological organisms. Therefore the only known pathway to creating intelligent agents is evolution by natural selection. Guided by this observation, the field of evolutionary robotics (Harvey et al., 1997; Nolfi and Floreano, 2000) attempts to realize intelligent agents by means of artificial evolution. Generally how this methodology works is that control policies for human designed or bio-mimicked robots are optimized to perform a desired task via evolutionary algorithms. This has allowed for the creation of robust, non-liner control strategies for autonomous agents that are not bound by the limits of human intuition. However, natural evolution does not operate on one part of an organism (brain) to the exclusion of others (body). In fact under evolution by natural selection any and all parts of an organism may be, and at some point in the past necessarily were, modified. This allows for the realization of organisms whose brains and bodies are co-optimized for specific ecological niches.

Luckily, artificial evolution is not necessarily limited to acting solely on a robot's brain or control strategy. Evolutionary frameworks in which the morphology and control of simulated machines are co-optimized in virtual environments are possible and indeed have been created, starting with Sims (1994) and followed by various other studies (Dellaert and Beer, 1994; Lund and Lee, 1997; Adamatzky et al., 2000; Mautner and Belew, 2000; Lipson and Pol-

lack, 2000; Hornby and Pollack, 2001a,b; Stanley and Miikkulainen, 2003; Eggenberger, 1997; Bongard and Pfeifer, 2001; Bongard, 2002; Bongard and Pfeifer, 2003). With this approach body plans and control policies uniquely suited for a machine's task environment may be found. This offers a substantial improvement over relying on body plans created by human designers who have inherent biases or copying animal body plans more suited to a given ecological niche.

The current work continues in this tradition while presenting several important advantages over previous approaches. First, the genomes of evolved agents are represented by compositional pattern producing networks (CPPNs) (Stanley, 2007), a form of indirect encoding that have been shown able to capture geometric symmetries appropriate to the system being evolved, are capable of reproducing outputs at multiple resolutions (Stanley et al., 2009), and have shown promise in producing neural network control policies for legged robots (Clune et al., 2009a,b). Second, through novel extensions of the CPPN outputs evolution can differentially optimize the resolution of the simulated robots such that a larger number of smaller sized components may be present in some body locations while a smaller number of larger sized components is present in other locations. To see why this is desirable consider evolving a creature capable of locomoting and grasping different objects. In this case evolution may choose to increase the resolution of the hands or grippers in order to achieve more fine grained control of the object to be grasped while at the same time using a lower resolution model of the trunk which will result in fewer components keeping the morphology from becoming unnecessarily complex and therefore providing faster simulations without sacrificing performance.

This paper extends the work presented in (Auerbach and Bongard, 2010) to allow for evolution of control as well as dynamic resolution as just discussed. The paper is organized as follows: the next section describers the CPPN encodings used, describes how they are evolved and presents how these encoding are used to grow actuated robots. Following that a description of two experiments is presented which compare this dynamic resolution method with a similar method lacking this ability. Some observations of how evolution makes use of the dynamic resolution capability are discussed, and finally some conclusions and directions for future work are presented.

Methods

This section presents a brief description of CPPNs and the evolutionary algorithm used to evolve them. This is followed by a description of the methods used for generating actuated robots from evolved genotypes. After this a description is presented of the fitness function used for evaluating these robots.

CPPNs

Compositional Pattern Producing Networks (CPPNs) are a form of artificial neural network (ANN). Unlike most ANNs where each internal node uses a form of sigmoid function, each internal node of a CPPN can have an activation function drawn from a diverse set of functions. This function set includes functions that are repetitive such as sine or cosine as well as symmetric functions such as gaussian. By composing these functions CPPNs can produce motifs seen in the majority of natural systems such as symmetry, repetition, and repetition with variation. It is important to note that these motifs come out of this encoding for free without the need for a human expert to explicitly enforce or select for them.

CPPN-NEAT

In this work the CPPNs are evolved via CPPN-NEAT (Stanley, 2007). CPPN-NEAT uses the NeuroEvolution of Augmenting Topologies (NEAT) method of neuro-evolution (Stanley and Miikkulainen, 2001) to evolve increasingly complex CPPNs. An extension of CPPN-NEAT —HyperNEAT— has been used (Stanley et al., 2009; Clune et al., 2009a,b) to evolve traditional ANNs, where each node of the ANN is embedded in a geometric space and whose coordinates are fed to an evolved CPPN to determine the presence and weights of connections. In effect these connections are "painted" on to the network from the output patterns produced by the CPPN. As shown by Stanley et al. (2009) this has the crucial benefit that a CPPN evolved to produce the connectivity patterns of small ANNs can be re-queried at a higher resolution to produce the connectivity patterns of larger ANNs without needing to re-evolve these large ANNs. Similarly as shown in (Auerbach and Bongard, 2010) it is possible to change the resolution at which CPPNs grow physical structures.

Growing Actuated Robots from CPPNs

In this work actuated robot morphologies and control strategies are grown from evolved CPPNs. Each robot is composed of many spherical cells which connect to each other either rigidly or via single degree of freedom rotational joints. For an example of robots produced in this way see Figure 1.

The growth procedure begins with a single cell, henceforth referred to as the root, with a predefined radius r_{init} located at a designated origin. A cloud composed of n points is cast around this cell with the n points being evenly distributed on the surface of the root sphere (all n points are at distance r from the center of the root). In the current work, n is restricted to 2, such that the points are directly opposite each other along the y-axis. In the coordinate system used here z is the vertical axis, and so the y-axis represents a horizontal axis that passes through the center of each cell. It is convenient to think of this as a cloud of points though, as is
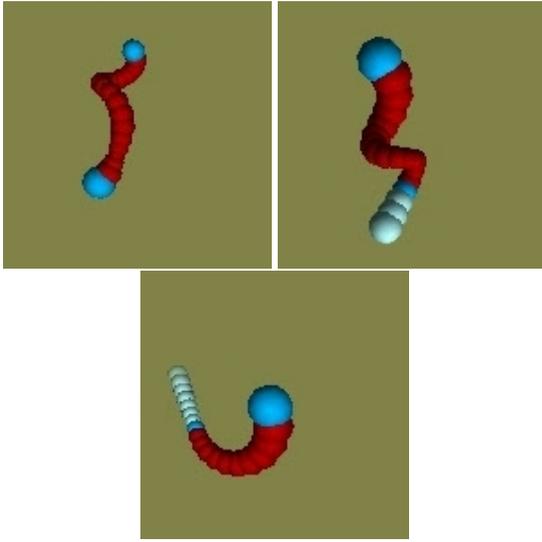
Figure 1: A few samples of robots evolved for directed locomotion.

the case in (Auerbach and Bongard, 2010), because in future work this restriction will once again be lifted allowing for a greater number of morphologies.

Once this cloud is cast, every point in the cloud is used to query a CPPN. The CPPN is queried by providing as input the Cartesian coordinates (x, y, z) of the point in question, the radius r_{parent} of the sphere to which it will attach ($r_{\text{parent}} = r_{\text{init}}$ when considering points around the root), and a constant bias input. These values are propagated through the CPPN to produce multiple output values. The first of these outputs is m. This output value can be thought of as a concentration of matter at that point, such that when m is over a certain matter threshold, T_{matter}, a cell will be placed at that point. The more that m exceeds the matter threshold the denser the cell placed at that point will be. This creates a continuum from no cell existing at that location up to having a very dense cell at that location with all intermediate levels of density in between being possible. The second of these outputs is a radius scaling factor r_{scale} which will determine the size of the cell to be added at that location.

Once the m and r_{scale} values have been determined for all n points in the cloud the points are sorted in descending order of the matter output m. The sorted points are then looped through and the algorithm considers adding a cell centered at each point in turn. Specifically a cell, centered at point p is added to the structure if (a) the output value of point p is above the threshold T_{matter} and (b) no other cell, besides the one to which this new cell will be attached (its parent) has previously been added to the structure with center located at distance $< r$ away from p.

1. **GrowRobot**(`CPPN`)
2.   Initialize priority queue q, with priority based on cell density
3.   Create cell c at origin with full density and radius r_{init}, add to morphology M and flag its coordinates 'discovered'
4.   Enqueue c in q
5.   `WHILE` $\sim q$.isEmpty
6. $c \leftarrow q$.front
7.   Cast point cloud C centered at c
8.   Initialize vector V of neighboring cells
9.   `FOR EACH` point p in C
10.   Query `CPPN` at p to get output values m and r_{scale}
11.   Add p with values m and r_{scale} to vector V
12.   Sort V by descending value of m
13.   `FOR EACH` point p with value m in sorted vector V
14.   `IF` coordinates of p not yet 'discovered'
15.   Flag p 'discovered'
16.   `IF` CanAdd(p,m,c,r)
17.   Add cell centered at p with density $\propto m$ and radius $r = r_{\text{parent}} * r_{\text{scale}}$ to morphology M
18.   Re-query `CPPN` at $\frac{c+p}{2}$ to get output values j, θ and Δ.
19.   `IF` $j > T_{\text{joint}}$
20.   Determine joint normal $\vec{n}$ from θ
21.   Connect cell with 1-DOF rotational joint with normal $\vec{n}$, range $\propto j$ actuated by CPG with phase offset $\propto \Delta$
22.   `ELSE`
23.   Connect cell rigidly
24.   Enqueue (p, v) in q

25. **CanAdd**(p, m, c, r)
26.   `IF` $m > T_{\text{matter}}$ `AND` $\forall$ cells $d \in M, d \neq c$ dist(p,d) $\geq r$ `AND` p is within bounding cube
27.   `Return true`
28.   `ELSE`
29.   `Return false`

Figure 2: **Grow Robot pseudo code.** The growth procedure starts with a root cell at the origin (line 3). Then, as long as there are cells in the queue to consider it takes the cell at the front of the queue, casts a point cloud around it and considers adding a cell at each point in turn (lines 5-17). A cell is added at a given point if all of the following hold: it does not conflict with a previously added cell, the CPPN outputs a value above the threshold T_{matter} when queried at that point, and the point is within the bounding cube (lines 25-29). If a cell is to be added the CPPN is queried once again to determine connectivity and control parameters (lines 18-23).

The radius r of a cell is determined from the radius of its parent r_{parent} and the output value r_{scale}. Specifically

$$r = \begin{cases} r_{\text{parent}} * r_{\text{scale}} & r_{\text{min}} \leq r_{\text{parent}} * r_{\text{scale}} \leq r_{\text{max}} \\ r_{\text{min}} & r_{\text{parent}} * r_{\text{scale}} < r_{\text{min}} \\ r_{\text{max}} & r_{\text{parent}} * r_{\text{scale}} > r_{\text{max}} \end{cases}$$

That is, the cell to be added will have radius equal to that of its parent scaled by a factor determined by the CPPN output capped by a minimum and maximum possible radius.

If a cell has been selected for addition to the robot the CPPN will be queried once more to determine connectivity and control parameters. In particular the CPPN will be fed the coordinates where a joint may be added: a cell centered at point p connecting to a parent cell centered at point p_{parent} may be connected by a single degree of freedom (DOF) rotational joint located halfway between p and p_{parent} ($\frac{p + p_{\text{parent}}}{2}$). These coordinates are input to the CPPN along with r_{parent} to retrieve additional outputs: a joint "concentration" j, an angle θ and a phase offset Δ.

If the output j exceeds a joint threshold T_{joint} the cell will attach to its parent with a 1-DOF rotational joint. The more j exceeds this threshold the greater the range of motion of the connecting joint will be. Similar to the matter case this creates a continuum from connecting rigidly when $j \leq T_{\text{joint}}$ to connecting via a joint with a very narrow range to connecting via a joint with a large range of motion.

If indeed a given cell will connect to its parent via a joint there are two more important properties of this connection to be determined. First, the direction of motion of this joint is defined by a normal vector $\vec{n}$. This vector will be normal to the axis $\vec{a}$ defined by the center of the cell and the center of its parent. To choose one vector out of the infinitely many such vectors the cross product of $\vec{a}$ and a default vector $\vec{d}$ is taken. This results in a single vector normal to $\vec{a}$ which is then rotated around $\vec{a}$ by angle θ. In this way all possible vectors normal to $\vec{a}$ may be used in constructing the joint and it is left up to the CPPN to output a single angle to choose a specific normal vector.

The second property to be determined in the case where a cell connects via a joint is what control signal drives the motor actuating this joint. In this work all motors are controlled by time dependent harmonic oscillators. A central sinusoidal oscillation is used, but each individual motor is allowed to be out of phase with this central control signal. The phase offset of each motor is determined by the final CPPN output Δ when queried at the joint's location. In this way the CPPN also determines the control policy of the robot being grown in addition to its morphology.

Once a cell is added to the structure and its connectivity and control have been determined it gets placed into a priority queue whose priority is based on its matter concentration m. When all points from the current cloud have been considered the algorithm takes the cell at the top of the priority queue and casts a point cloud around it, and this process continues until there are no valid possible points at which to place cells. Points are valid if they are within a bounding cube with side lengths l. This bounding cube constraint was imposed so that in the future it will be possible to physically fabricate the entire evolved robots within the confines of a 3D-printer. Figure 2 gives pseudo code for this growth procedure.

There are several reasons why it is desirable to have a growth procedure such as this. Merely querying CPPNs over a sampling of three-dimensional space may lead to disconnected objects. Even if all but one of these objects are thrown out much computational resources will have been wasted querying these regions of space. Additionally, imposing a grid over space to determine which points to query imposes a specific resolution on the morphology and thus removes much of the benefit of the dynamic resolution (radius) method used in this work because the spacing of the cells will have been predetermined by the grid.

Selecting for robots with desirable properties

This paper aims to demonstrate that CPPN-NEAT coupled with the growth procedure just presented is capable of evolving actuated robot morphologies and control policies for a given task. In particular the property selected for in this work is maximum directed displacement of the robot in a fixed amount of time.

To select for this property, an evolved virtual robot is placed in a physical simulator[1] for that set amount of time. The fitness of this robot (and hence its encoding CPPN) that CPPN-NEAT attempts to maximize is simply the y-coordinate of the robot's center of mass after the simulation completes subject to a few conditions. The first of these conditions is to prevent robots from exploiting simulation faults. There are a number of ways these faults could be avoided such as reducing the step size used in running the simulation, but this would lead to increased simulation runtimes. The technique used here is to throw out any solution where the robot's linear or angular acceleration exceed predefined thresholds by giving 0 fitness. The second condition is to prevent solutions where the robot moves by rolling on a subset of its cells. These solutions tend to be common but are less interesting than other solutions that may be found, therefore any robot that has a subset of its cells remain in contact with the ground for over 95% of the time is discarded and given a fitness of 0 once again.

Results

This section presents experiments comparing how the dynamic resolution method presented above performs is comparison to a similar method restricted to using cells with a

---

[1]Simulations are conducted in the Open Dynamics Engine (http://www.ode.org), a widely used open source, physically realistic, simulation environment
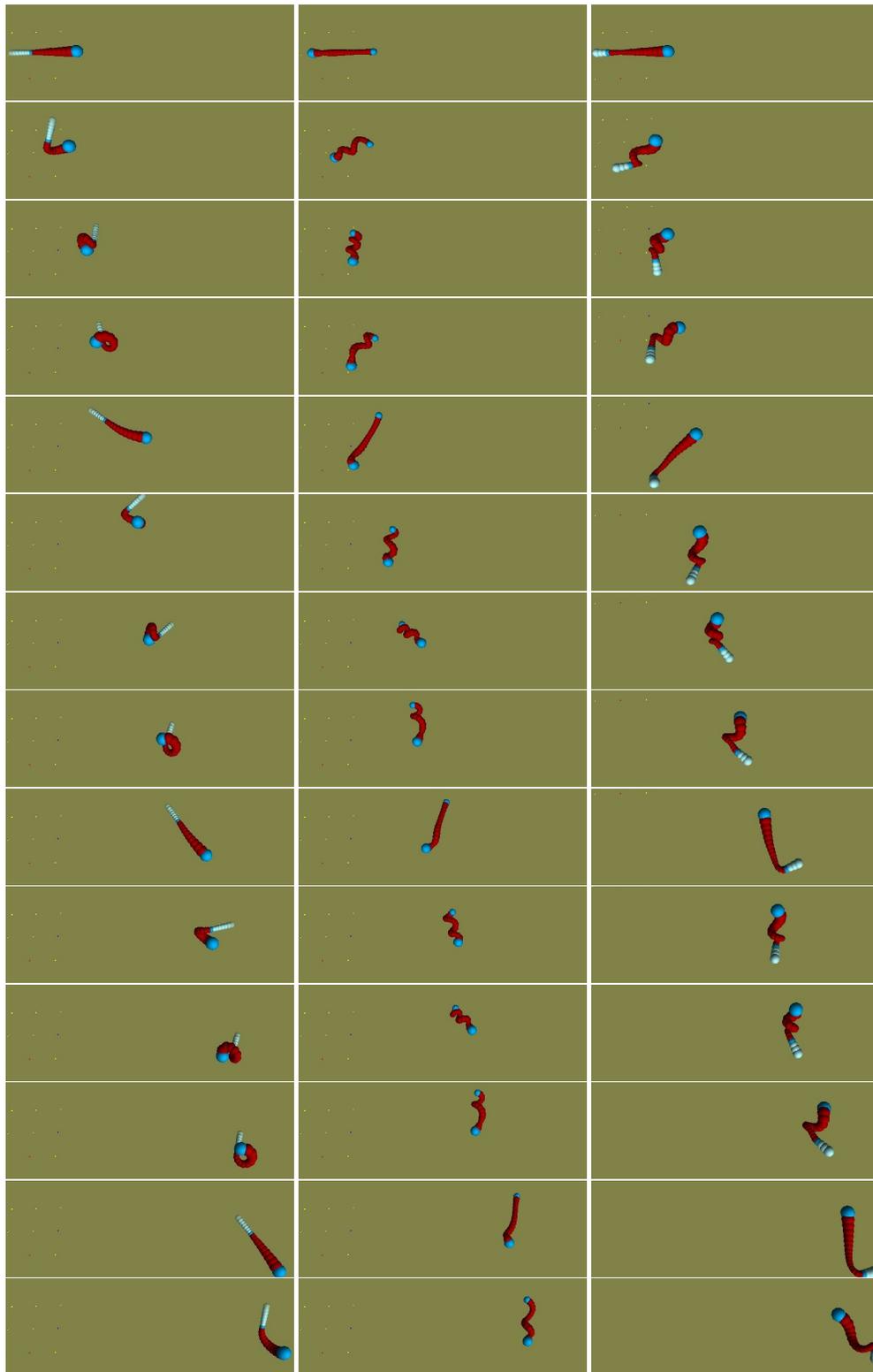
Figure 3: Each column shows the behavior of a different dynamic resolution robot evolved for directed locomotion (with time going from top to bottom). Three different robots are shown. Red cells are attached to two joints while the darker blue cells attach to a single joint. The lighter blue cells all connect rigidly. Enlarged pictures of each of these robots are shown in Fig. 1.

fixed radius. It should be noted that using a fixed radius in this case would be equivalent to omitting the growth procedure and merely querying the evolved CPPN over a gridded region of space and then taking those cells which connect to the cell at the origin as the resulting morphology, however as mentioned above this procedure would require more computational resources than using the growth procedure to accomplish the same result.

Specifically, two experiments are conducted each consisting of a set of 30 evolutionary trials. All experiments attempt to evolve simulated robots with CPPN-NEAT capable of directed locomotion using the fitness criteria presented above. Moreover, all experiments are configured to use a population size of 150, and run for 500 generations with each fitness evaluation given 2500 time steps. Additionally in all experiments the values T_{matter} and T_{joint} are both fixed at 0.7, and each cell of the structure is restricted to having its center initially located in interval $(0, [-2, 2], 0)$ (coordinates all in meters). Before being placed in the simulator the morphologies are translated vertically such that the largest component is resting on the ground. The CPPN internal nodes are allowed to use the signed cosine, gaussian, and sigmoid activation functions. All other parameters of the evolutionary algorithm are kept at the default values provided with the C++ implementation of HyperNEAT[2].

The trials in the first experiment grow structures using the dynamic resolution method introduced in this paper. In this case r_{init} was set to 0.1 meters, r_{min} set to 0.01 meters, and r_{max} set to 0.5 meters. Additionally the output value r_{scale} is normalized to the range $[0.5, 1.5]$; that is, a newly added cell can have radius at the most 50% larger and at the least 50% smaller than its parent. Figure 3 demonstrates the behavior of a few of the more successful robots to evolve in evolutionary trials in this experiment.

The second experiment is exactly the same as the first one, but it is restricted to growing robots composed of cells with a fixed radius. CPPN-NEAT is still used to evolve CPPNs which are used to grow the morphologies and control strategies under the procedure outlined above, but the r_{scale} output is not included in the CPPNs. In lieu of determining cell size from this output this experiment builds robots from cells all having radius $r_{\mathrm{fixed}} = 0.1$ meters.

Discussion

One advantage of using the dynamic resolution method over keeping resolution fixed is that it allows evolution to explore a greater variety of possible solutions. The first evidence of this is observational. Looking at the behavior of the three robots shown in Figure 3 a variety of dynamics can be observed. The left most robot resembles a whip in that it has one thicker end and tapers off to a thinner end. Additionally

we see that the thin end is rigid. This can be inferred from the light blue coloring of the cells at that end which represent cells that are not connected to any joint (while red cells connect to two joints and dark blue cells to a single joint). Scanning down the panels one can see that this rigid end is utilized as a paddle to propel the robot forward while curling over at the other end.

The middle robot on the other hand has no rigid connections. This robot moves by coiling and uncoiling to move itself in the desired direction. The right most robot has yet a different morphology and movement pattern than the other two. While it has one rigid end like the left most robot this end is composed of fewer spheres and actually includes cells that are larger than those in the middle of its body, flaring back out like a baseball bat. This configuration is actually the most successful one discovered and its movement pattern is different from the other two robots.
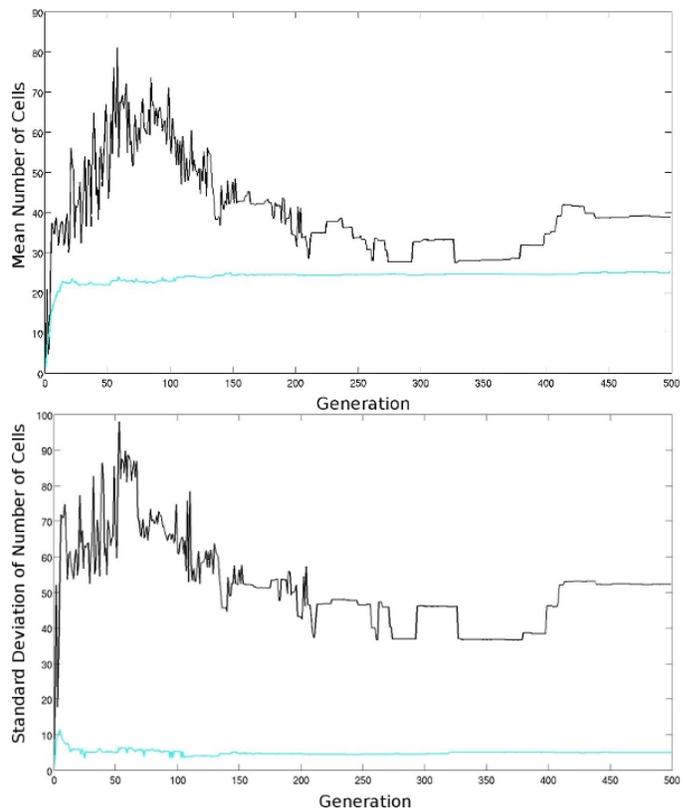

Figure 4: **Top**: Mean number of cells of best individual in each generation across the 30 evolutionary trials for the dynamic resolution set (black) and the fixed resolution set (light blue). **Bottom**: Standard deviation from the mean number of cells by generation.

Additional evidence of the dynamic resolution runs exploring a greater variety of morphologies is shown in Figure 4. The top part of this figure shows the mean number of cells used by the best individual from each generation across the

30 evolutionary trials from both the dynamic resolution set and the fixed resolution set. The bottom portion of this figure shows the standard deviation from the means shown in the top. One can see here that the trials in the dynamic resolution set tend to explore morphologies with a large number of small cells early on, followed by exploring a fewer number of cells on average later on in the trials. However, while the fixed resolution robots tend to converge to a narrow range of cell numbers as exemplified by the constant mean and small standard deviation, the dynamic resolution robots continue to explore a wide array of different number of cells and cell sizes which can be inferred by observing that their standard deviation never comes back down.
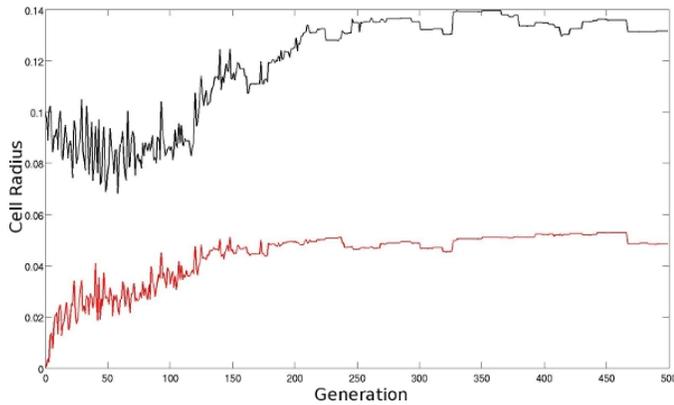

Figure 5: Mean (black) and standard deviation from the mean (red) of cell radii *within* each best of generation individual from the dynamic resolution set averaged across the 30 evolutionary trials.

This evidence is corroborated by Figure 5 which plots the mean and standard deviation of cell radii *within* each best of generation individual averaged across the 30 evolutionary trials. Here it is shown in a different way how the dynamic runs tend to explore smaller cell sizes early on in the evolutionary trials followed by larger cell sizes later. While this is the case on average, by looking at the standard deviations we see that as evolution progresses morphologies with a wide variety of cell sizes come into being (the standard deviation trends upwards). This means that the dynamic resolution runs are exploring the space of solutions with variable cell sizes which is not possible in the fixed resolution case.

Conclusion

This paper has demonstrated how one can implement a growth mechanism that can generate robots composed of variable sized components. This ability was then shown to be actually utilized by demonstrating how evolutionary trials that incorporate this dynamic resolution mechanism explore a greater variety of possible solutions than evolutionary trials that are restricted to constructing robots out of fixed sized components.

While it is not directly evident what performance advantage using dynamic resolution offers on a task as simple as the one utilized in this work, intuitively one can see the benefit of such a mechanism when generating more complex robots for more complex tasks. Specifically in any task that requires object manipulation it will be useful to adapt the component sizes of the parts of the morphology that will be in contact with external objects while not creating overly complex morphologies as would be the case if such a high resolution were employed for the entire robot. Additionally, it may not be possible to know the ideal component size *a priori*, and so using a dynamic resolution method such as this can help steer evolution towards constructing robot morphologies with the proper component sizes.

Much work remains to be done in exploring the possibilities of this methodology. The logical next step will be to relax some of the restrictions imposed in this work such as allowing robots to grow in arbitrary trajectories as opposed to along only a single axis. The authors additionally plan to tackle more complex tasks including object manipulation to test whether using dynamic resolution will result in the additional predicted advantages discussed here. This will require the use of more complex control strategies such as neural networks, and the inclusion of a mechanism for endowing the robots with sensors in order to close the control loop. The methods used here for generating joint and motor parameters via additional CPPN outputs seem promising and the authors plan to further leverage this technique for determining sensor and neuron positions and parameters.

References

Adamatzky, A., Komosinski, M., and Ulatowski, S. (2000). Software review: Framsticks. *Kybernetes: The International Journal of Systems & Cybernetics*, 29(9/10):1344–1351.

Anderson, M. (2003). Embodied Cognition: A field guide. *Artificial Intelligence*, 149(1):91–130.

Auerbach, J. E. and Bongard, J. C. (2010). Evolving CPPNs to Grow Three-Dimensional Physical Structures. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*. To Appear.

Beer, R. D. (2008). The dynamics of brain-body-environment systems: A status report. In Calvo, P. and Gomila, A., editors, *Handbook of Cognitive Science: An Embodied Approach*, pages 99–120. Elsevier.

Bongard, J. and Pfeifer, R. (2001). Repeated structure and dissociation of genotypic and phenotypic complexity in Artificial Ontogeny. *Proceedings of The Genetic and Evolutionary Computation Conference (GECCO 2001)*, pages 829–836.

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

Bongard, J. C. (2002). Evolving modular genetic regulatory networks. In *Proceedings of The IEEE 2002 Congress on Evolutionary Computation (CEC2002)*, pages 1872–1877.

Braitenberg, V. (1986). *Vehicles: Experiments in Synthetic Psychology*. MIT Press.

Brooks, R. (1999). *Cambrian intelligence*. MIT Press Cambridge, Mass.

Clune, J., Beckmann, B., Ofria, C., and Pennock, R. (2009a). Evolving Coordinated Quadruped Gaits with the HyperNEAT Generative Encoding. In *Proceedings of the IEEE Congress on Evolutionary Computing*, pages 2764–2771.

Clune, J., Pennock, R. T., and Ofria, C. (2009b). The sensitivity of hyperneat to different geometric representations of a problem. In *Proceedings of the Genetic and Evolutionary Computation Conference*.

Dellaert, F. and Beer, R. (1994). Toward an evolvable model of development for autonomous agent synthesis. *Artificial Life IV, Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*.

Eggenberger, P. (1997). Evolving morphologies of simulated 3D organisms based on differential gene expression. *Procs. of the Fourth European Conf. on Artificial Life*, pages 205–213.

Harvey, I., Husbands, P., Cliff, D., Thompson, A., and Jakobi, N. (1997). Evolutionary robotics: the sussex approach. *Robotics and Autonomous Systems*, 20:205–224.

Hornby, G. and Pollack, J. (2001a). Body-brain co-evolution using l-systems as a generative encoding. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 868–875.

Hornby, G. and Pollack, J. (2001b). Evolving L-systems to generate virtual creatures. *Computers & Graphics*, 25(6):1041–1048.

Lipson, H. and Pollack, J. B. (2000). Automatic design and manufacture of artificial lifeforms. *Nature*, 406:974–978.

Lund, H. H. and Lee, J. W. P. (1997). Evolving robot morphology. *IEEE International Conference on Evolutionary Computation*, pages 197–202.

Mautner, C. and Belew, R. (2000). Evolving robot morphology and control. *Artificial Life and Robotics*, 4(3):130–136.

Nolfi, S. and Floreano, D. (2000). *Evolutionary Robotics: The Biology,Intelligence,and Technology*. MIT Press, Cambridge, MA, USA.

Pfeifer, R. and Bongard, J. (2006). *How the Body Shapes the Way We Think: A New View of Intelligence*. MIT Press.

Sims, K. (1994). Evolving 3D morphology and behaviour by competition. *Artificial Life IV*, pages 28–39.

Stanley, K., D'Ambrosio, D., and Gauci, J. (2009). A Hypercube-Based encoding for evolving Large-Scale neural networks. *Artificial Life*, 15(2):185–212.

Stanley, K. and Miikkulainen, R. (2003). A taxonomy for artificial embryogeny. *Artificial Life*, 9(2):93–130.

Stanley, K. O. (2007). Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines*, 8(2):131–162.

Stanley, K. O. and Miikkulainen, R. (2001). Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10:2002.