Evolving a Single Scalable Controller for an Octopus Arm with a Variable Number of Segments

In: *Proceedings of the 11th International Conference on Parallel Problem Solving From Nature (PPSN 2010).* New York, NY: Springer

Brian G. Woolley and Kenneth O. Stanley

School of Electrical Engineering and Computer Science
University of Central Florida, Orlando, FL 32816
brian.woolley@ieee.org, kstanley@eecs.ucf.edu

Abstract. While traditional approaches to machine learning are sensitive to high-dimensional state and action spaces, this paper demonstrates how an indirectly encoded neurocontroller for a simulated octopus arm leverages regularities and domain geometry to capture underlying motion principles and sidestep the superficial trap of dimensionality. In particular, controllers are evolved for arms with 8, 10, 12, 14, and 16 segments in equivalent time. Furthermore, when transferred *without further training*, solutions evolved on smaller arms retain the fundamental motion model because they simply extend the general kinematic concepts discovered at the original size. Thus this work demonstrates that dimensionality can be a false measure of domain complexity and that indirect encoding makes it possible to shift the focus to the underlying conceptual challenge.

1 Introduction

Whether tackled through neuroevolution or temporal difference-based approaches, in reinforcement learning problems, the number of dimensions in the state and action space is often associated with problem difficulty [6, 11, 18, 19]. Yet the complexity of problems should *not* be determined by the dimensionality of such representations, which are a superficial proxy for the underlying *conceptual* problem. Instead, the problem complexity should correlate to the underlying complexity of the *principle* to be discovered. The argument in this paper is that indirect encoding, which means describing the solution as a *pattern* through a compressed representation [1, 9, 14, 16], is the essential ingredient that will allow reinforcement learning to transcend the superficial aspects of problem dimensionality.

To make this point, the problem domain in this paper is an octopus arm, which is approximated as a structure of interconnected muscles that must act together to create a coordinated behavior. Thus it induces a high-dimensional state space *and* action space (i.e. because each muscle in each segment can be articulated independently). In fact, the high dimensionality of the 10-segment arm provoked previous researchers to dramatically prune the action space by allowing only a small discrete set of pre-coordinated actions [5].

The octopus arm problem is thus an ideal departure for a study on the ability of indirect encoding to transcend such dimensionality. After all, the underlying kinematic control principle is similar regardless of the precise number of segments, muscles and

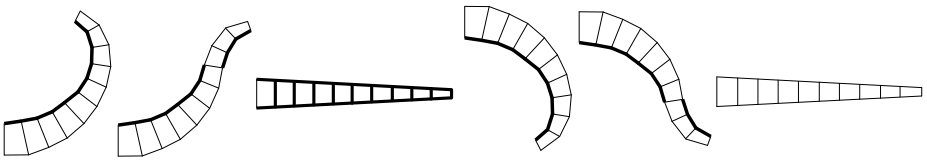


Fig. 1: **Octopus arm actions.** While there are theoretically many other combinations of muscle contractions possible, Engel et al. [5] limited their model to these six to make the domain tractable. Line thickness indicates the strength of the contractive force applied.

sensory inputs, suggesting that an approach that is sensitive to its particular dimensionality is missing something fundamental. Thus, to demonstrate this point in this paper, an indirect encoding called *hypercube-based neuroevolution of augmenting topologies* (HyperNEAT) evolves a *description* of how the weights of a neurocontroller relate to each other across the domain geometry irrespective of the arm's precise physical dimensionality [3, 8, 14]. This approach means that the HyperNEAT controller can actually learn to articulate all the muscles independently without the need to partition the action space up front. Furthermore, as should be the case in learning such problems, the indirect encoding learns equivalently across arms with a variable number of segments. Finally, neurocontrollers trained on arms with eight segments are scaled to a larger number of segments *without further training* and still work because they encode general control principles for the arm.

Thus the major contribution of this work is to demonstrate that indirect encoding is a potentially critical ingredient in reinforcement learning and control problems if they are to focus on the true problem complexity rather than the superficial dimensionality of the state or action space.

2 Background

This section reviews prior work in training multi-segment arms and in indirect encoding of neural networks.

2.1 Reinforcement Learning for Arm Controllers

An interesting study that provides inspiration for this paper demonstrated the ability of Gaussian Process Temporal Differencing (GPTD) [4] to learn value functions in a high-dimensional domain [5]. In it, a control policy is trained for an arm with many degrees-of-freedom (i.e. the octopus arm [20]). GPTD produced value functions for motion trajectories that touch targets at unknown locations within 20 trials.

The details of the original octopus arm experiment are interesting because they set a new standard for high-dimensional control that this paper pushes even further. The 10-segment arm had a state space with 88 dimensions (i.e. position and velocity for each vertex) that map to the six discrete actions shown in figure 1. Engel et al. [5] chose

these six actions to reduce the otherwise prohibitively large action space created by so many muscles.

The next section introduces the indirect encoding in HyperNEAT, which will make it possible to evolve such high-dimensional controllers without the need to shield the learner from the true dimensionality of the space.

2.2 Indirect Encoding and HyperNEAT

Neuroevolution, i.e. evolving ANNs, can produce solutions for a broad array of control tasks [6, 15, 17, 19]. Many such methods are based on *direct encodings*, which means each piece of structure in the phenotype is encoded by a single gene, making the discovery of repeating motifs expensive and improbable. Therefore, indirect encodings [1, 9, 14, 16] have become a growing area of interest in evolutionary computation.

One such indirect encoding designed explicitly for neural networks is the Hypercube-based NeuroEvolution of Augmenting Topologies (HyperNEAT) approach [14], which is an indirect extension of the directly-encoded NEAT approach [15, 17]. Rather than expressing link weights as distinct and independent parameters in the genome, HyperNEAT allows them to vary across the phenotype in a regular pattern through an encoding called a *compositional pattern producing network* (CPPN) [13].

The idea behind CPPNs is that geometric patterns can be encoded by a *composition of functions* that are chosen to represent common regularities. For example, the Gaussian function is symmetric, so when it is composed with any other function alone, the result is a symmetric pattern. The internal structure of a CPPN is a weighted network, similar to an ANN, that denotes which functions are composed and in what order, which means that instead of evolving ANNs as it normally does, NEAT [15, 17] can evolve *CPPNs* that generate connectivity patterns across an ANN.

Formally, CPPNs are *functions* of geometry (i.e. locations in space) that output connectivity patterns whose nodes are situated in n dimensions, where n is the number of dimensions in a Cartesian space. Consider a CPPN that takes four inputs labeled x_1, y_1, x_2 and y_2; this point in four-dimensional space also denotes the connection between the two-dimensional points (x_1, y_1) and (x_2, y_2). The output of the CPPN for that input thereby represents the weight of that connection (figure 2). By querying every pair of points in the space, the CPPN can produce an ANN, wherein each queried point is a neuron position. While CPPNs are themselves networks, the distinction in terminology between CPPN and ANN is important for explicative purposes because in HyperNEAT, CPPNs *encode* ANNs. Because the connection weights are produced as a function of their endpoints, the final structure is produced with *knowledge* of the domain geometry, which is literally depicted geometrically within the constellation of nodes.

As a rule of thumb, nodes are placed in a geometric space called the *substrate* to reflect the geometry of the domain (i.e. the state) [2, 8, 14]. For example, a visual field can be laid out in two dimensions such that nodes that receive input from adjacent locations in the image are literally adjacent in the network geometry. This way, knowledge of the domain geometry is preserved and exploited by HyperNEAT where regularities (e.g. adjacency, or symmetry, which the CPPN sees) are invisible to traditional encodings. This capability is exploited in the octopus arm substrate introduced next.

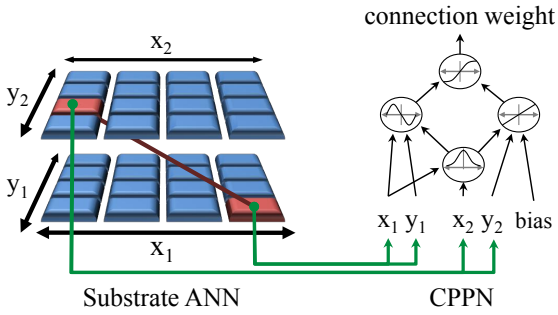


Fig. 2: **Encoding the connectivity pattern.** The four-dimensional CPPN encodes the connectivity pattern of the substrate through an evolved network of geometric functions. The substrate ANN is generated by querying the CPPN for the value of each potential connection from (x_1, y_1) to (x_2, y_2). In this way, CPPNs capture patterns and regularities in domain geometry.

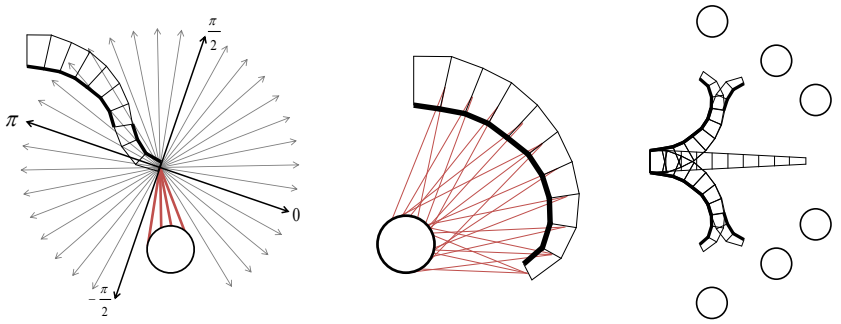
3 Scalable Neurocontroller for an Octopus Arm

The octopus arm domain formalized in this section is particularly challenging because its state space *and* action space are both high-dimensional. Although Engel et al. [5] reduced dimensionally by choosing only six canonical actions, the aim in this paper is to learn from the full unprocessed action space. Furthermore, unlike any system before, the learned controller will be asked to scale to even larger arms *without further learning*.

The simulation domain in this paper, based on Yekutieli et al. [20], models the kinematics and dynamics of a two-dimensional *muscular hydrostat* (which is the mechanism of the octopus arm [12]) as a chain of quadrilateral polygons with fixed area connected to a fixed base. The model constructs arms based on length (l), width (w), taper (t), mass (m), and number of segments (n). At the vertex of each quadrilateral is a point mass shared by adjacent segments. The *dorsal*, or upper, and *ventral*, or lower, edges of each segment represent longitudinal muscles while the vertical edges between sections represent transverse muscles. The muscles, modeled as spring-joints, are contracted by increasing the spring constant and relaxed by reducing the spring constant.

The fixed size and incompressible nature of the arm are the key features that enable the dynamic motion of the muscular hydrostat. These attributes are modeled by adjusting each segment's internal pressure: as external forces act to compress a segment, pressure increases; conversely, as forces stretch and expand the segment, internal pressure decreases. Thus segments change shape to restore the equilibrium between surface tension and internal pressure. Figure 1 shows the six basic actions utilized in Engel et al. [5] as examples of the model's motion effects. However, in this paper, the ANN will have independent control of all $3n$ muscles in the arm, creating a high-dimensional action space.

Because experiments in this paper involve moving towards a perceived object (unlike Engel et al. [5]), the arm state is defined by sensor inputs that allow the controller to infer the position of each segment relative to the target. Range sensors along the arm provide cues about target position. Each sensor at each segment produces 36 radial distance measurements across the range $[-\pi \ldots \pi]$ (figure 3a), allowing the target



(a) Target Sensor Operation (b) Target Perception (c) Target Placement

Fig. 3: **Target perception and placement.** The arm controller perceives the target through range sensors (a) placed at each segment along the arm; the combined effect is shown (b) with non-detecting beams removed for clarity. The training targets in this paper (c) are positioned beyond the reach of the simple actions in figure 1.

to be seen by multiple beams simultaneously, especially as the sensor approaches the target or as sensor resolution is increased. Thus the $36n$ total beams also create a high-dimensional input space. Figure 3b illustrates the arm's view of the target with the non-detecting beams removed for clarity.

3.1 Substrate Architecture

The octopus arm substrate (figure 4) closely couples sensing to acting. The input layer accepts sensor data directly and the output layer provides the contractive response for each muscle. Finally, a hidden layer is provided to support nonlinear operation required by the gravity and buoyancy effects acting on the arm.

To represent the sensor array described above, the controller must interpret 36 rangefinder inputs per segment. The arm model is composed of segments that have a necessary order and relationship to the other segments in the arm, i.e. segment 1 connects to segment 2, segment 2 connects to segment 3, etc. Thus, the perception layer is constructed as a two-dimensional sheet with θ as one axis and the arm's proximal-distal (PD) geometry as the other (figure 4, layer A).

To represent the action space, the substrate provides an output for each of the $3n$ muscles in the arm. To take advantage of HyperNEAT's ability to leverage domain geometry, the proximal-distal axis of the sensor layer is mirrored by the output (contractive) layer. Furthermore, note in figure 1 how the dorsal and ventral muscles act together to form coordinated reaching behaviors. This configuration suggests aligning the dorsal, transverse, and ventral muscles along the proximal-distal axis (figure 4, layer C).

By viewing the substrate architecture as a feedforward network spanning from the sensor input layer (A), to the hidden layer (B), to the contractive output layer (C), a CPPN with inputs $(x_1,\ y_1,\ x_2,\ y_2)$ and outputs $(AB,\ BC)$ provides a complete

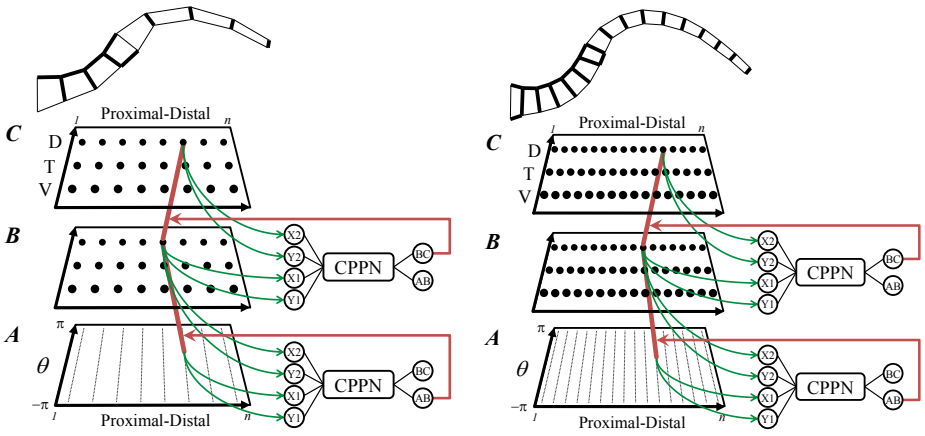


Fig. 4: **Scalable neurocontroller architecture.** The eight-segment substrate (a) can be scaled to a 16-segment substrate (b) without further training. The substrate at any resolution contains a two-dimensional input layer A that corresponds to the arm's sensory input, a two-dimensional hidden layer B, and a two-dimensional output layer C that controls the musculature. To query connections between A and B, the proximal-distal (PD) axis is the x_1 input and θ is the y_1 input. To query connections between B and C, the x_1 input is also PD, and DTV (Dorsal-Transverse-Ventral) is the y_2 input. Because the CPPN encodes kinematic principles, resampling with the node positions in (b) can produce a similar contractive pattern and arm pose (shown above).

encoding of the phenotype. Figure 4a illustrates how each of the 7,488 link weights in an eight-segment controller (left) are set by a single CPPN.

The hope is that the principle that underlies moving a hydrostat is regular across the segments of the arm and therefore can be captured by the CPPN.

3.2 Scaling

By constructing the substrate to reflect the domain geometry (figure 4a), larger arm controllers are generated without further evolution by *requerying the same CPPN* at higher-resolutions (figure 4b). This approach works because adding segments to the arm is analogous to increasing the resolution of the hydrostat model. The CPPN provides a nonlinear interpolation of the behavior policy for each of the $3n$ muscles.

4 Experiment

The first aim of the experiment is to investigate the ability of indirect encoding to facilitate learning to control a hydrostat with dozens of degrees of freedom that are not a priori restricted or pruned in any way. The second aim is to test the ability of an evolved CPPN to generate controllers for higher resolution arms without further training. Both

tests can validate that HyperNEAT learns *general principles* of hydrostat control rather than a single solution at a particular dimensionality.

The fitness function is designed to select controller behaviors that approach targets quickly. The simulator records the distance between the tip of the arm and the target at each timestep and calculates the average distance over a trial with a single target as:

$$d_{avg} = \sum_{t=0}^{t_{max}} \frac{d_t}{t_{max}}, \tag{1}$$

where d_t is the distance from the tip of the arm to the target center at time t and t_{max} is the maximum number of timesteps in the trial. Individuals in the population are evaluated in six trials against six training targets (figure 3c) that are beyond the reach of the simple movements shown in figure 1. Because the goal is to reduce the average distance, fitness for a single trial can be expressed as $f_{trial} = d_0 - d_{avg}^2$, where d_0 is the initial distance and squaring d_{avg} emphasizes early innovations that move towards the target by providing larger rewards for small improvements. Negative fitness values are set to zero and arms that succeed in touching the target with the tip earn a 25% bonus.

Because HyperNEAT differs from original NEAT only in its set of activation functions, it uses the same parameters [15]. All experiments were run with a version of the public domain ANJI package [10] augmented to implement HyperNEAT. The population size was 100 and each run lasted 500 generations. The speciation threshold, δ_t, was 0.2 and the compatibility modifier was 0.3. Available CPPN activation functions were sigmoid, Gaussian, sine, and linear functions. Recurrent connections within the CPPN were not enabled. Signed activation was enforced in the CPPN, but the substrate was unsigned, resulting in a node output range of $[-1, 1]$. By convention, a connection was not expressed if the magnitude of its weight is below a minimal threshold of 0.2 [7]. These parameters were found to be robust to moderate variation.

To validate that eight-segment solutions can scale, their evolved CPPNs are requeried to generate controllers for arms with 10, 12, 14, 16, 18, and 20 segments with no further training. It is important to note that these dimensionalities are indeed high because they impact the necessary dimensionality of the corresponding *neurocontroller*, (i.e. an eight-segment controller must set 7,488 connection weights while a 20-segment controller must set 46,800). Also, results cannot be compared directly to controllers trained by Engel et al. [5] because they seek a single target blindly while those in this paper can actively touch targets at multiple locations based on sensory inputs.

5 Results

Figure 5 shows training performance over generations when controllers are separately evolved (i.e. not scaled) for arms with 8, 10, 12, 14, and 16 segments. 20 runs were completed at each resolution. Remarkably, the number of degrees-of-freedom has *no significant effect* on the training curve, suggesting that indirect encoding really is making it possible to focus on learning the underlying *principle* independently of dimensionality.

Across all variants, CPPNs with an average of only 10.1 connections (stdev $= 2.3$) encode substrates with between 7,488 (8 segments) and 29,952 connections (16 segments), demonstrating the considerable compression of the indirect encoding.

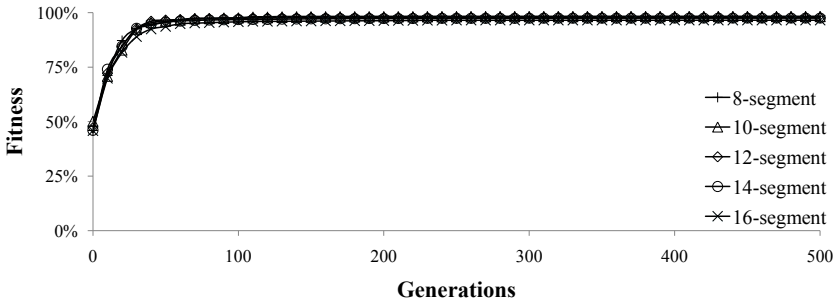


Fig. 5: **Training at different arm resolutions.** HyperNEAT evolves neurocontrollers for arms with 8, 10, 12, 14 and 16 segments in equivalent time because the CPPN discovers the underlying kinematic patterns. Measurements are averaged over 20 runs.

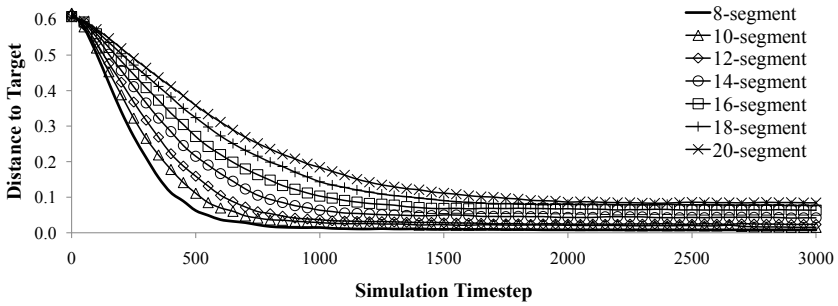


Fig. 6: **Scaling solutions to larger arm controllers.** The distance from the target surface at each time-step is shown, demonstrating that the ability to move towards the target is preserved as controllers are scaled to support arms with additional segments. Measurements are averaged over 20 runs.

The main scaling result (figure 6) is that the evolved contractive patterns transfer well from controllers trained on eight segments to arms with an increasing number of segments with no additional training. In the figure, the distance from the arm tip to the target surface is graphed over timesteps, demonstrating that controllers maintain the ability to approach targets as the physical structure scales; on average, even the 20-segment (worst) case approaches within 0.084 (± 0.05 at 95% confidence) units of the target surface. It is important to note that the qualitative behavior of the arm at all scales in figure 6 is the same (i.e. they all still approach the target) although the speed of movement slows gradually and emergent physical characteristics begin to render the original solution less effective.

The sequence shown in figure 7 demonstrates a typical scaled reaching behavior. The contractive pattern shown was evolved as an eight-segment arm and applied to a 16-segment arm with no further training. Videos of evolved arms and scaling are available at `http://eplex.cs.ucf.edu/octopusArm`.

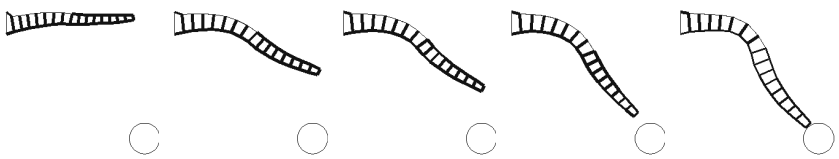


Fig. 7: **Typical reaching motion of a scaled hydrostat.** This 16-segment controller is scaled directly from an eight-segment arm solution and illustrates how contracting the transverse muscles allows the arm to extend beyond its relaxed length.

6 Discussion and Future Work

Finding solutions to problems in control should be about discovering an underlying principle and not about the number of dimensions in the action or state representation. Traditional approaches [15, 17, 18] map state information to effectors as if each is an independent dimension when in fact they are related. This traditional view of the problem domain ties complexity to the dimensionality of the physical domain and thus obfuscates the underlying concept.

The ability to evolve controllers for arms with 8, 10, 12, 14, and 16 segments (which contain 7,488, 11,700, 16,848, 22,932, and 29,952 connections, respectively) in equivalent time demonstrates that this physical structure's dimensionality is a false measure of the domain complexity. By exploring the space of kinematic principles, the indirect encoding approach bypasses the increasing dimensionality of the physical structure. Similarly, the scaling results demonstrate that solutions evolved specifically for the eight-segment arm model encompass fundamental kinematic strategies that apply directly to arms with additional segments.

Thus indirect encoding becomes an important consideration for any problem in which state or action dimensionality may be misleading, or for learning scalable control policies. Whether it is a multi-segment arm, a robot hand that can add more fingers, or a centipede with a variable number of legs, indirect encoding shifts the problem away from the precise configuration towards the underlying principle, thereby opening up such problems to machine learning.

7 Conclusions

For many problems, complexity is independent of the number of dimensions. The challenge is to transcend the distraction of superficial dimensionality by preserving meaningful relationships, e.g. geometric principles like order, orientation, and proximity. The octopus arm model is a good platform to test this idea because it can include an increasing number of segments. By discovering an underlying kinematic pattern, the HyperNEAT approach is able to sidestep the increasing dimensionality of the physical structure. Experimental results demonstrate that this approach yields controllers for arms with 8, 10, 12, 14, and 16 segments in equivalent time and that evolved solutions can provide controllers for arms with up to twice as many segments without further training. Thus this paper provides a lesson on the important role of indirect encoding in reinforcement learning.

References

1. Bongard, J.C., Pfeifer, R.: Evolving complete agents using artificial ontogeny, pp. 237–258. Morpho-functional Machines: The New Species (Designing Embodied Intelligence), Springer-Verlag (2003)
2. Clune, J., Beckmann, B.E., Ofria, C., Pennock, R.T.: Evolving coordinated quadruped gaits with the hyperneat generative encoding. In: Proceedings of the IEEE Congress on Evolutionary Computing Special Section on Evolutionary Robotics. IEEE Press (2009)
3. D'Ambrosio, D.B., Stanley, K.O.: A novel generative encoding for exploiting neural network sensor and output geometry. In: Proceedings of the 9th annual conference on Genetic and evolutionary computation. pp. 974–981. ACM, New York, NY (2007)
4. Engel, Y., Mannor, S., Meir, R.: Bayes meets Bellman: The Gaussian process approach to temporal difference learning. In: Proceedings of the 20th international conference on machine learning. vol. 20, pp. 154–161. AAAI Press (2003)
5. Engel, Y., Szabo, P., Volkinshtein, D.: Learning to control an octopus arm with gaussian process temporal difference methods. Advances in Neural Information Processing Systems 18, 347–354 (2006)
6. Floreano, D., Dürr, P., Mattiussi, C.: Neuroevolution: from architectures to learning. Evolutionary Intelligence 1(1), 47–62 (2008)
7. Gauci, J., Stanley, K.O.: Generating large-scale neural networks through discovering geometric regularities. In: Proceedings of the 9th annual conference on Genetic and evolutionary computation. pp. 997–1004. ACM, New York, NY (2007)
8. Gauci, J., Stanley, K.O.: Autonomous evolution of topographic regularities in artificial neural networks. Neural Computation p. 38 (2010), to appear
9. Hornby, G.S., Pollack, J.B.: Creating high-level components with a generative representation for body-brain evolution. Artificial Life 8(3), 223–246 (2002)
10. James, D., Tucker, P.: ANJI homepage. `http://anji.sourceforge.net/` (2004)
11. Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement learning: A survey. Journal of artificial intelligence research 4(1), 102–138 (1996)
12. Kier, W.M., Smith, K.K.: Tongues, tentacles and trunks: the biomechanics of movement in muscular-hydrostats. Zoological Journal of the Linnean Society 83, 307–324 (1985)
13. Stanley, K.O.: Compositional pattern producing networks: A novel abstraction of development. Genetic programming and evolvable machines 8(2), 131–162 (2007)
14. Stanley, K.O., D'Ambrosio, D.B., Gauci, J.: A hypercube-based indirect encoding for evolving large-scale neural networks. Artificial Life 15(2) (2009)
15. Stanley, K.O., Miikkulainen, R.: Evolving neural networks through augmenting topologies. Evolutionary computation 10(2), 99–127 (2002)
16. Stanley, K.O., Miikkulainen, R.: A taxonomy for artificial embryogeny. Artificial Life 9(2), 93–130 (2003)
17. Stanley, K.O., Miikkulainen, R.: Competitive coevolution through evolutionary complexification. Journal of Artificial Intelligence Research 21(1), 63–100 (2004)
18. Sutton, R.S., Barto, A.G.: Reinforcement Learning: an introduction. MIT Press, Cambridge, MA (1998)
19. Yao, X.: Evolving artificial neural networks. Proc. of the IEEE 87(9), 1423–1447 (1999)
20. Yekutieli, Y., Sagiv-Zohar, R., Aharonov, R., Engel, Y., Hochner, B., Flash, T.: Dynamic model of the octopus arm. I. Biomechanics of the octopus reaching movement. Journal of neurophysiology 94(2), 1443–1506 (2005)