A Novel Generative Encoding for Exploiting Neural Network Sensor and Output Geometry

David B. D'Ambrosio
Evolutionary Complexity Research Group
School of EECS
University of Central Florida
Orlando, FL 32836
ddambro@cs.ucf.edu

Kenneth O. Stanley
Evolutionary Complexity Research Group
School of EECS
University of Central Florida
Orlando, FL 32836
kstanley@cs.ucf.edu

ABSTRACT

A significant problem for evolving artificial neural networks is that the physical arrangement of sensors and effectors is invisible to the evolutionary algorithm. For example, in this paper, directional sensors and effectors are placed around the circumference of a robot in analogous arrangements. This configuration ensures that there is a useful geometric correspondence between sensors and effectors. However, if sensors are mapped to a single input layer and the effectors to a single output layer (as is typical), evolution has no means to exploit this fortuitous arrangement. To address this problem, this paper presents a novel generative encoding called *connective Compositional Pattern Producing Networks (connective CPPNs)* that can effectively detect and capitalize on geometric relationships among sensors and effectors. The key insight is that sensors and effectors with consistent geometric relationships can be exploited by a *repeating motif* in the neural architecture. Thus, by employing an encoding that can discover such motifs as a function of network geometry, it becomes possible to exploit it. In this paper, a method for evolving connective CPPNs called *Hypercube-based Neuroevolution of Augmenting Topologies (HyperNEAT)* discovers sensible repeating motifs that take advantage of two different placement schemes, demonstrating the utility of such an approach.

Categories and Subject Descriptors: I.2.6 [Artificial Intelligence]: Learning–Connectionism and neural nets, concept learning

General Terms: Experimentation, Algorithms

Keywords: Compositional Pattern Producing Networks, NEAT, HyperNEAT, large-scale artificial neural networks

1. INTRODUCTION

The geometry of physical space can provide a potentially useful constraint on the evolution of artificial neural networks (ANNs). In particular, physical space by virtue of its Cartesian organization produces inherent regularities within physical tasks. For example, many tasks on land are inherently symmetric. Thus, genetic encoding can exploit such symmetry advantageously by simply repeating the same connectivity motifs throughout the brain [13].

In biological organisms, the placement and configuration of sensors and effectors bridges between the geometry of the physical world and the geometry of the brain. By preserving the most salient physical relationships of the outside world, such as symmetry (e.g. left and right eyes and ears) and locality (e.g. the retinotopic map of visual experience), sensory configuration allows neural organization to largely mirror the same relationships. For example, the visual cortex preserves the retinotopic layout of the eye [4]. In this way, nearby events in physical space are easily represented by similarly proximal neurons.

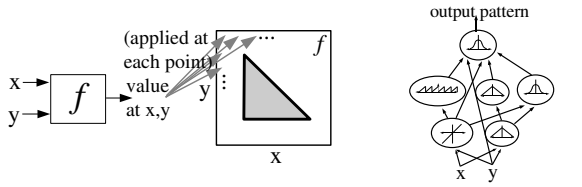
Interestingly, existing approaches to evolving ANNs generally employ a single layer of inputs and single layer of outputs with no explicit or implicit geometry in the representation [18, 22]. Therefore, evolution cannot exploit the geometry of such neurons even if it theoretically could provide a powerful bias. In fact, exploiting sensor geometry requires a generative encoding because it is necessary to correlate repeated connectivity motifs to regularities in the physical placement of sensors and effectors. This paper introduces such an encoding, called *connective Compositional Pattern Producing Networks (connective CPPNs)* and a method to evolve them called *Hypercube-based Neuroevolution of Augmenting Topologies (HyperNEAT)*. Connective CPPNs abstract the properties of developmental encoding at a high level and apply this abstraction to encoding connectivity patterns. In this way, HyperNEAT is able to find solutions that truly exploit task geometry.

This capability is demonstrated through a food gathering experiment designed to show how different layouts of sensors and effectors lead to contrasting solutions with varying efficiency and ability to scale. The main conclusion is that with HyperNEAT an experimenter can now significantly enhance evolution's chance of solving a task by simply arranging sensors in a way that reflects the task geometry. Thus the geometry of inputs and outputs now becomes a crucial constraint for the learning algorithm.

2. BACKGROUND

This section provides an overview of CPPNs and NEAT, which are foundational to the novel approach introduced in

GECCO'07, July 7–11, 2007, London, England, United Kingdom.



(a) Mapping (b) Composition

Figure 1: CPPN Encoding. (a) The function f takes arguments x and y, which are coordinates in a two-dimensional space. When all the coordinates are drawn with an intensity corresponding to the output of f, the result is a spatial pattern, which can be viewed as a phenotype whose genotype is f. (b) The CPPN is a graph that determines which functions are connected. The connections are weighted such that the output of a function is multiplied by the weight of its outgoing connection.

this paper, in which NEAT evolves CPPNs that are interpreted as ANN connectivities.

2.1 Compositional Pattern Producing Networks

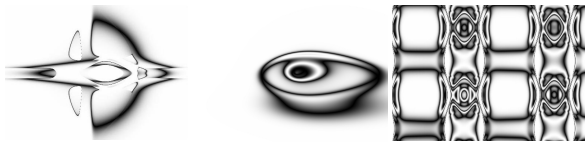
In biological genetic encoding the mapping between genotype and phenotype is *indirect*. The phenotype typically contains orders of magnitude more structural components than the genotype contains genes. Thus, the only way to discover such high complexity may be through a mapping between genotype and phenotype that translates few dimensions into many, i.e. through an *indirect encoding*. A promising research area in indirect encoding is *generative and developmental encoding*, which is motivated from biology [1, 2, 8]. In biological development, DNA maps to the phenotype through a process of growth that builds the phenotype over time. Development facilitates the reuse of genes because the same gene can be activated at any location and any time during the development process.

This observation has inspired significant research in generative and developmental systems [2, 3, 5, 8, 9, 11, 12, 19, 21]. The aim is to find the right abstraction of natural development for a computer running an evolutionary algorithm, so that EC can begin to discover complexity on a natural scale [19].

Compositional Pattern Producing Networks (CPPNs) are a novel abstraction of development that can represent sophisticated repeating patterns in Cartesian space [15, 16]. Unlike most generative and developmental encodings, CPPNs do not require an explicit simulation of growth or local interaction, yet still exhibit their essential features. The remainder of this section briefly reviews the theory behind CPPNs, which will be augmented in this paper to represent connectivity patterns and ANNs.

Consider the phenotype as a function of n dimensions, where n is the number of dimensions in physical space. For each coordinate in that space, its level of expression is an output of the function that encodes the phenotype. Figure 1a shows how a two-dimensional phenotype can be generated by a function of two parameters.

Stanley [15] showed how simple canonical functions can be composed to create networks that produce complex regularities and symmetries. Each component function creates a novel geometric *coordinate frame* within which other functions can reside. The main idea is that these simple canonical functions are abstractions of specific events in development such as establishing bilateral symmetry (e.g. with a symmetric function such as Gaussian) or the division of the body into discrete segments (e.g. with a periodic function such as sine). Figure 1b shows how such a composition is represented as a network.



(a) Symmetry (b) Imperf. Sym. (c) Rep. with var.

Figure 2: CPPN-generated Regularities. Spatial patterns exhibiting (a) bilateral symmetry, (b) imperfect symmetry, and (c) repetition with variation are depicted. These patterns demonstrate that CPPNs effectively encode fundamental regularities of several different types.

Such networks are called *Compositional Pattern Producing Networks* because they produce spatial patterns by composing basic functions. While CPPNs are similar to ANNs, they differ in their set of activation functions and how they are applied. Furthermore, they are an abstraction of development rather than of biological brains.

Through interactive evolution, Stanley [15] showed that CPPNs can produce spatial patterns with important geometric motifs that are expected from generative and developmental encodings and seen in nature. Among the most important such motifs are symmetry (e.g. left-right symmetries in vertebrates), imperfect symmetry (e.g. right-handedness), repetition (e.g. receptive fields in the cortex [23]), and repetition with variation (e.g. cortical columns [6]). Figure 2 shows examples of several such important motifs produced through interactive evolution of CPPNs.

These patterns are generated by applying the right activation functions (e.g. symmetric functions for symmetry; periodic functions for repetition) in the right order in the network. The order of activations is an abstraction of the unfolding process of development.

2.2 CPPN-NEAT

Because NEAT was originally designed to evolve increasingly complex ANNs, it is naturally suited to doing the same with CPPNs, which are also represented as graphs. The NEAT method begins evolution with a population of small, simple networks and *complexifies* them over generations, leading to increasingly sophisticated solutions. While the NEAT method was originally developed to solve difficult control and sequential decision tasks, [17, 18, 20], in this paper it is used to evolve CPPNs. This section briefly reviews the NEAT method. See also Stanley and Miikkulainen [18, 20] for detailed descriptions of original NEAT.

NEAT is based on three key ideas. First, in order to allow network structures to increase in complexity over generations, a method is needed to keep track of which gene is which. Otherwise, it is not clear in later generations which individual is compatible with which, or how their genes should be combined to produce offspring. NEAT solves this problem by assigning a unique *historical marking* to every new piece of network structure that appears through a structural mutation. The historical marking is a number assigned to each gene corresponding to its order of appearance over the course of evolution. The numbers are inherited during

crossover unchanged, and allow NEAT to perform crossover without the need for expensive topological analysis. That way, genomes of different organizations and sizes stay compatible throughout evolution.

Second, NEAT speciates the population, so that individuals compete primarily within their own niches instead of with the population at large. This way, topological innovations are protected and have time to optimize their structure before competing with other niches in the population. NEAT uses the historical markings on genes to determine to which species different individuals belong.

Third, NEAT begins with a uniform population of simple networks with no hidden nodes, differing only in their initial random weights. Speciation protects new innovations, allowing diverse topologies to gradually complexify over evolution. Thus, NEAT can start minimally, and grow the necessary structure over generations. A similar process of gradually adding new genes has been confirmed in natural evolution [10]. Through complexification, high-level features can be established early in evolution and then elaborated and refined as new genes are added.

For these reasons, in this paper the NEAT method is used to evolve increasingly complex CPPNs. CPPN-generated patterns evolved with NEAT exhibit several essential motifs and properties of natural phenotypes [14, 16]. If such properties could be transferred to evolved *connectivity patterns*, the representational power of CPPNs could potentially evolve large-scale ANNs and other graph structures, as explained in the next section.

3. HyperNEAT

The spatial patterns in Section 2.1 present a challenge: How can such spatial patterns describe connectivity? This section explains how CPPN output is effectively interpreted as a connectivity pattern rather than a spatial pattern. Furthermore, this novel interpretation allows neurons, sensors, and effectors to exploit meaningful geometric relationships. The next section introduces the key insight, which is to assign connectivity a geometric interpretation.

3.1 Geometric Connectivity Patterns

The main idea is to input into the CPPN the coordinates of the *two points* that define a connection rather than inputting only the position of a single point as in Section 2.1. The output is then interpreted as the *weight* of the connection rather than the intensity of a point. This way, connections are defined in terms of the locations that they connect, thereby taking into account the network's geometry.

For example, consider a 5×5 grid of nodes. The nodes are assigned coordinates corresponding to their positions within the grid (labeled *substrate* in figure 3), where $(0, 0)$ is the center of the grid. Assuming that these nodes and their positions are given *a priori*, a *geometric connectivity pattern* is produced by a CPPN that takes any two coordinates (source and target) as input, and outputs the weight of their connection. The CPPN is queried in this way for every potential connection on the grid. Because the connection weights are thereby a function of the *positions* of their source and target nodes, the distribution of weights on connections throughout the grid will exhibit a pattern that is a function of the geometry of the coordinate system.

Such a CPPN in effect computes a four-dimensional function $CPPN(x_1, y_1, x_2, y_2) = w$, where the first node is at (x_1, y_1) and the second node is at (x_2, y_2). This formalism

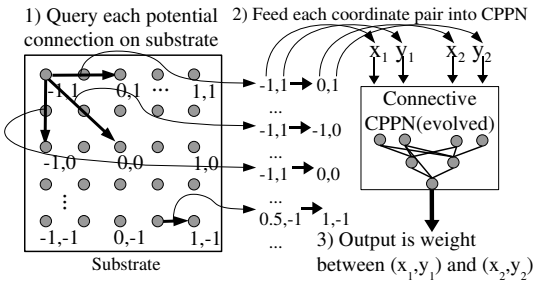


Figure 3: Hypercube-based Geometric Connectivity Pattern Interpretation. A grid of nodes, called the *substrate*, is assigned coordinates such that the center node is at the origin. 1. Every potential connection in the substrate is queried to determine its presence and weight; the dark directed lines in the substrate in the figure represent a sample of connections that are queried. 2. For each query, the CPPN takes as input the positions of the two endpoints and (3) outputs the weight of the connection between them. Thus, *connective CPPNs* can produce regular patterns of connections in space.

returns a weight for every connection between every node in the grid, including recurrent connections. By convention, a connection is not expressed if the magnitude of its weight, which may be positive or negative, is below a minimal threshold w_{min}. Magnitudes above this threshold are scaled to be between zero and a maximum magnitude in the substrate. That way, the pattern produced by the CPPN can represent any network topology (figure 3).

The connectivity pattern produced by a CPPN in this way is called the *substrate* so that it can be verbally distinguished from the CPPN itself, which has its own internal topology. Furthermore, in the remainder of this paper, CPPNs that are interpreted to produce connectivity patterns are called *connective CPPNs* while CPPNs that generate spatial patterns are called *spatial CPPNs*. This paper focuses on neural substrates produced by connective CPPNs.

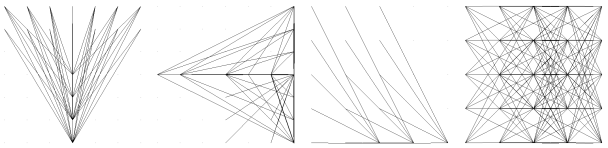
Because the CPPN is a function of four dimensions, the two-dimensional connectivity pattern expressed by the CPPN is isomorphic to a spatial pattern embedded in a four-dimensional hypercube. Thus, because CPPNs generate regular spatial patterns (Section 2.1), by extension they can be expected to produce geometric connectivity patterns with corresponding regularities. The next section demonstrates this capability.

3.2 Producing Regular Connectivity Patterns

Simple, easily-discovered substructures in the connective CPPN produce connective motifs in the substrate. The key difference between connectivity patterns and spatial patterns is that each discrete unit in a connectivity pattern has *two* x values and *two* y values. Thus, for example, symmetry along x can be discovered simply by applying a symmetric function (e.g. Gaussian) to either x_1 or x_2 (figure 4a).

The human brain is roughly symmetric at a gross resolution, but its symmetry is imperfect. Thus, imperfect symmetry is an important structural motif in ANNs. Connective CPPNs can produce imperfect symmetry by composing symmetric functions of one axis along with asymmetric coordinate frames such as the axis itself. In this way, the CPPN produces varying degrees of imperfect symmetry (figure 4b).

Another important motif in biological brains is repetition, particularly repetition with variation. Just as sym-



(a) Sym. (b) Imperf. (c) Repet. (d) Var.

Figure 4: Connectivity Patterns Produced by Connective CPPNs. These patterns, produced through interactive evolution, exhibit important connectivity motifs: (a) bilateral symmetry, (b) imperfect symmetry, (c) repetition, and (d) repetition with variation. That these fundamental motifs are compactly represented and easily produced suggests the power of this type of encoding.

(a) Grid (b) Circular

Figure 5: Alternative Substrate Configurations. There exist many potential substrate configurations. This figure shows (a) a grid and (b) a concentric configuration. Different configurations are likely suited to problems with different geometric properties. The generality of connective CPPNs is supported by their ability to generate connectivity patterns across any such configuration.

metric functions produce symmetry, periodic functions such as sine produce repetition (figure 4c). Patterns with variation are then produced by composing a periodic function with a coordinate frame that does not repeat, such as the axis itself (figure 4d). Repetitive patterns can also be produced in connectivity as functions of invariant properties between the two nodes, such as connection length. Thus, symmetry, imperfect symmetry, repetition, and repetition with variation, key structural motifs in biological brains, are compactly represented and therefore easily discovered by connective CPPNs.

3.3 Substrate Configuration

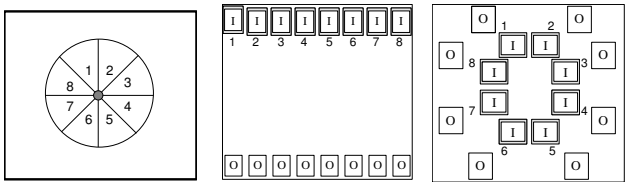
CPPNs produce connectivity patterns among nodes on the substrate by querying the CPPN for each pair of points in the substrate to determine the weight of the connection between them. The layout of these nodes can take forms other than the planar grid (figure 5a) discussed thus far. Different such *substrate configurations* are likely suited to different kinds of problems.

For example, the nodes need not be distributed in a grid. The nodes within a substrate that controls a radial entity such as a starfish might be best laid out with radial geometry (figure 5b) so that the connectivity pattern can be situated with perfect polar coordinates.

3.4 Input and Output Placement

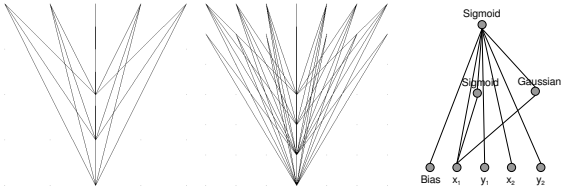
Part of substrate configuration is determining which nodes are inputs, which are outputs, and which are hidden. The flexibility to assign inputs and outputs to specific coordinates in the substrate creates a powerful opportunity to exploit geometric relationships advantageously.

In many ANN applications, the inputs are drawn from a set of sensors that exist in a geometric arrangement in space. Because connective CPPN substrates are aware of their inputs' and outputs' geometry, they can use this information to their advantage.



(a) Robot (b) Parallel (c) Concentric

Figure 6: Placing Inputs and Outputs. Unlike tradition ANN representations, neurons on the substrate exist at defined locations in space that are accessible to the learning algorithm. The figure depicts a robot (a) with eight radar sensors along its circumference and eight motion effectors set at the same angle. In (b), the inputs (labeled *I*) and outputs (labeled *O*) are placed such at their location along x determines whether they represent a corresponding direction. In (c), the inputs and outputs are laid out literally according to the eight directions in space. Both arrangements create a geometric relationship between each input and its corresponding output.



(a) 5×5 Concept (b) 7×7 Concept (c) CPPN

Figure 7: Equivalent Connectivity Concept at Different Substrate Resolutions. A connectivity concepts is depicted that was evolved through interactive evolution. The CPPN that generates the concept at (a) 5×5 *and* (b) 7×7 is shown in (c). This illustration shows that CPPNs represent a mathematical concept rather than a single structure. Thus, the same connective CPPN can produce patterns with the same underlying concept at different substrate resolutions (i.e. node densities).

There is room to be creative and try different configurations with different geometric advantages. For example, figure 6 depicts two methods in which the inputs and outputs of a circular robot can be configured, both of which create an opportunity to exploit a different kind of geometric relationship.

3.5 Substrate Resolution

As opposed to encoding a specific pattern of connections among a specific set of nodes, connective CPPNs in effect encode a general *connectivity concept*, i.e. the underlying mathematical relationships that produce a particular pattern. The consequence is that *same connective CPPN* can represent an equivalent concept at different resolutions (i.e. node densities). Figure 7 shows a connectivity concept at different resolutions.

For neural substrates, a significant implication is that the same ANN can be generated at different resolutions. *Without further evolution*, previously-evolved connective CPPNs can be re-queried to specify the connectivity of the substrate at a new, higher resolution, thereby producing a working solution to the same problem at a higher resolution! This

operation, i.e. increasing substrate resolution, introduces a powerful new kind of complexification to ANN evolution. It is an interesting question whether, at a high level of abstraction, the evolution of brains in biology in effect included several such increases in density on the same connectivity concept. Not only can such an increase improve the immediate resolution of sensation and action, but it can provide additional substrate for increasingly intricate local relationships to be discovered through further evolution.

3.6 Computational Complexity

Querying every potential connection in the substrate is realistic for modern computers. For example, a CPPN generating a substrate with 250,000 potential connections is queried 250,000 times, which can be computed e.g. in 4.64 seconds on a 3.19 Ghz Pentium 4. Note that this substrate is an enormous ANN with up to a quarter-million connections. Connective CPPNs present an opportunity to evolve structures of a complexity and functional sophistication genuinely commensurate with available processing power.

3.7 Evolving Connective CPPNs

The approach in this paper is to evolve connective CPPNs with NEAT. This approach is called *HyperNEAT* because NEAT evolves CPPNs that represent spatial patterns in hyperspace. Each point in the pattern, within a hypercube, is interpreted as a connection in a lower-dimensional connected graph. NEAT is the natural choice for evolving CPPNs because it evolves increasingly complex network topologies. Therefore, as CPPNs complexify, so do the regularities and elaborations (i.e. the *global dimensions of variation*) that they represent in their corresponding connectivity pattern. Thus, HyperNEAT is a powerful new approach to evolving large-scale connectivity patterns and ANNs. The next section describes initial experiments that demonstrate how HyperNEAT can exploit input and output geometry.

4. FOOD GATHERING EXPERIMENT

If sensors and outputs are placed such that they respect regularities in the outside world, HyperNEAT can discover those regularities through connective CPPNs and exploit them to solve the problem. For example, HyperNEAT can discover that the way one reacts to stimulus on the left is related to the way one react to similar stimulus on the right. In this way, HyperNEAT can solve problems with high-dimensional input because it does not need to learn the meaning of each sensor independently.

The experiments in this paper demonstrate this capability and its implications through a food gathering task. This task was chosen for its simplicity as a proof-of-concept; it effectively isolates the issue of sensor and output placement, which is the primary topic of this paper. Furthermore, although the task is simple, at high resolutions (i.e. with large numbers of sensors and effectors) it becomes a difficult optimization problem because of its increasing dimensionality. In the experiments, two different sensor placement arrangements are compared that both present a chance to exploit regularity in different ways.

The food gathering domain works as follows. A single piece of food is placed within a square room with a robot at the center (figure 6a). A set of n rangefinder sensors, placed at regular angular intervals, encircle the robot's perimeter. The robot has a compass that allows it to maintain the same orientation at all times, that is, its north-facing side *always* faces north and never rotates. Internally, the robot also contains a set of n effectors. Each effector, when activated, causes the robot to move in one of n directions. Thus, there is one effector for each sensor that points in the same direction. The robot's objective is to go to the food.

The interpretation of effector outputs constrains the problem and the potential solutions. For the experiments in this paper, the motion vector resulting from effector output is interpreted to incentivize HyperNEAT to find holistic solutions, i.e. solutions that do not only require a single connection. The robot moves in the direction corresponding to the largest effector output. In the case of a tie, the robot moves in the direction of the first tied output in sampling order. The robot's speed s is determined by

$$s = (s_{max} o_{max})(\frac{o_{max}}{o_{tot}}), \quad (1)$$

where s_{max} is the maximum possible speed, o_{max} is the maximum output, and o_{tot} is the sum of all outputs. The first term correlates speed with output, such that to go the maximum speed, the robot must maximize the output corresponding to the direction of the food. The second term encourages the robot to excite a single output by penalizing it for activating more than one at a time. Furthermore, outputs have *sigmoidal activation*, which means that if their input is zero, they will output 0.5. Thus, the robot also needs to inhibit effectors that point in the wrong direction because they will otherwise slow down motion in the chosen direction. Thus, while diverse solutions still work in this domain, many are not optimal in terms of speed. The best solutions require a correct pattern connecting to all the outputs from all the sensors.

Each robot attempts r trials, where r is twice the resolution; thus higher resolutions are evaluated on more trials. For each trial a single piece of food is placed 100 units away from the robot at either the center of a sensor or the border between two sensors. Each trial tests a different such location. If a robot is not able to get food for a particular trial after 1000 ticks, its trial ends. Individuals are evaluated based on their amount of food collected and the average speed at which they obtain each item:

$$fitness = (10000\frac{f_c}{r}) + (\frac{t_{tot}}{1000r}), \quad (2)$$

where f_c is the total number of food collected and t_{tot} is the total time spent on all trials.

This task is a good proof of concept because it requires discovering the underlying regularity of the domain: Two nodes at the same angle (i.e. the sensor and effector) should be connected and the others inhibited. If this concept is discovered, the task is effectively trivial. However, direct encodings would need to discover the connection between each pair *independently*. That is, they cannot discover the underlying *concept* that describes the solution. In contrast, HyperNEAT can discover the general concept. Demonstrating this fact helps to establish the promise of HyperNEAT in more complex domains.

4.1 Sensor Placement

HyperNEAT makes it possible to decide how the sensors and effectors should be placed in the substrate to allow the concept to be discovered. In general, the geometry of the placement scheme should reflect the correlation between sensors and effectors. Two different configurations are tested:

(1)**Two parallel lines of nodes (figure 6b).** The top row of sensors are placed in clockwise order starting from the sensor closest to 45° above west. In the bottom row, effectors are placed in the same order. In this way, the key regularity is geometrically expressed as two nodes being in the same column.

(2)**Two concentric circles of nodes (figure 6c).** The inner circle (radius .5) is the sensors and the outer circle (radius 1) is the effectors. The nodes are placed at the same angle as they exist in the robot. In this layout, the key regularity is captured by shared angle. It is also interesting because the sensors and effectors are placed exactly in the shape of the robot, an intuitive scheme that would not be meaningful to traditional methods.

4.2 Additional Geometric Bias

Connective CPPNs already provide a strong geometric bias, but they also make possible injecting additional *a priori* knowledge about the problem into the search. In biology, distance is a significant bias simply because in the physical world it is easier to connect to something closer than farther away. Thus, if a CPPN is *given* connection length as an input, related concepts placed close together on the substrate can be treated specially by the CPPN.

In *both* placement schemes, sensors should excite their nearest neighbor effector and inhibit those farther away. While CPPNs have the capability to discover the concept of distance themselves, they may solve tasks more efficiently if it is given. To explore this capability, a separate experiment is performed on both substrate configurations in which the CPPN receives an extra input representing the Euclidean distance between the two points being queried.

4.3 Scaling

An important question is whether HyperNEAT will discover the key regularity and whether one configuration has an advantage over another. Furthermore, *if* HyperNEAT does discover the underlying concept, then solutions should scale to more sensors and effectors without further evolution simply by increasing the resolution of the substrate.

Therefore, the number of inputs and outputs of each generation champion from all runs of both configurations is doubled several times starting from the evolved 8×8 resolution. Each double-resolution substrate samples twice as many points as its predecessor by decreasing the angular sampling interval around the robot by half. Doubling for each champion proceeds from the initial 8×8 resolution to a maximum size of 128×128, a 16-fold increase in resolution.

4.4 Experimental Parameters

Because HyperNEAT differs from original NEAT only in its set of activation functions, it uses the same parameters [18]. Experiments were run using a modified version of the public domain SharpNEAT package [7]. The size of each population was 150 with 20% elitism. Reproduction had an equal chance of being sexual or asexual. Asexual offspring had .96 probability of link weight mutation, .03 chance of link addition, and .01 chance of node addition. The coefficients for species similarity were 1.0 for nodes and connections, and 0.1 for weights. CPPN connection weights ranged between -3.0 and 3.0. Sigmoid, Gaussian, absolute value, and sine nodes were created with equal probability. Parameter settings are based on standard SharpNEAT defaults and prior reported settings for NEAT [18, 20, 17]. They were found to be robust to moderate variation through preliminary experimentation.

5. RESULTS

All sensor configurations were able to collect food at all positions within the first few generations except unbiased concentric, which took on average 33 generations to learn how to get food at every position. Thus, for most configurations, the main challenge was to learn to get food *efficiently*. The performance measure in this section is thus the average time (i.e. number of ticks) it takes the robot to get a piece of food over all its trials. Robots that cannot get the food in a trial are given the maximum time 1,000 for that trial. Results are averaged over 20 runs.

Figure 8a shows how performance improved over 500 generations for both placement schemes, with and without the length-input geometric bias. Parallel placement on average evolved significantly faster strategies ($p < .05$) than concentric. This disparity is explained by a more complex relationship, in Cartesian coordinates, between corresponding nodes in the concentric case. However, the geometric bias significantly increased performance of both methods after the fourth generation ($p < .01$). Furthermore, the geometric bias is so useful that it erases the difference between the two schemes, causing both to perform similarly when present. Thus, parallel placement is easier to exploit for HyperNEAT except when the CPPN is provided connection length as input.

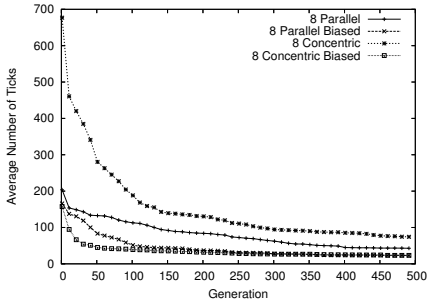
5.1 Scaling Performance

As described in Section 4.3, after evolution at resolution eight is completed, generation champions are reevaluated by doubling their resolution repeatedly without further evolution. Individuals generally *did* retain the ability to collect food although there is some degradation in performance at each increment in resolution. To illustrate this degradation for different configurations, figure 8b shows the average *difference* in efficiency between resolution eight and resolution 32; lower numbers imply better ability to scale.

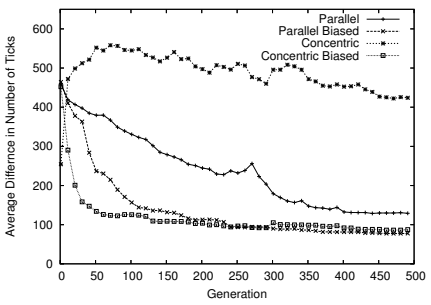
Parallel placement scaled significantly more effectively than concentric except in the earliest generations ($p < .01$). However, as with efficiency, when concentric placement's CPPN was provided length as input, it scaled as well as parallel placement did with length input. In both biased cases, scaling significantly improved over runs using concentric placement without the geometric bias ($p < .01$), but not significantly over unbiased parallel placement.

As figure 8b shows, unbiased concentric placement degraded significantly between resolution eight and 32; in fact individuals could no longer collect food in every position. However, it turns out that information about the task was still retained implicitly at the higher resolution: When allowed to *continue* evolving at the higher resolution, solutions that collect all the food were always found within five generations (2.5 on average). On the other hand, when concentric evolution is started from scratch at a *lower* resolution, it takes on average 33 generations to learn to get food on every trial. Thus, even when performance degrades significantly after scaling, the resultant individual still retains important geometric information that can be quickly tweaked to work at the higher resolution.

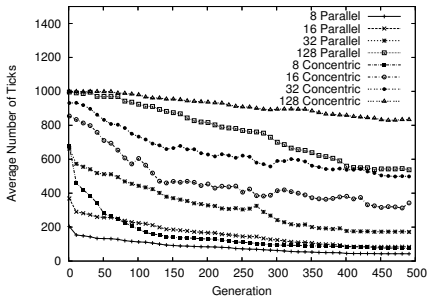
Figure 8c shows the average absolute performance at dif-



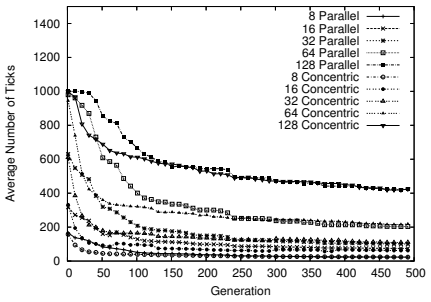
(a) Initial Performance



(b) Scalability



(c) Unbiased Scaling

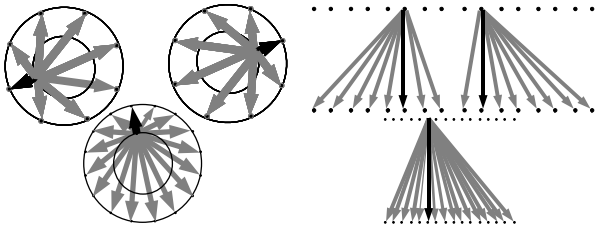


(d) Biased Scaling

Figure 8: HyperNEAT Performance. The performance of both sensor layouts at resolution eight with and without biasing is shown in (a). The difference in speeds when different methods are scaled from 8 to 32 is shown in (b). Graphs (c) and (d) show the speeds of the two sensor placement schemes at all resolutions with and without biasing, respectively. The conclusion is that HyperNEAT learns to exploit the placement geometry.

ferent resolutions for CPPNs without the geometric bias, and 8d shows the same comparison for those with the bias. Parallel placement consistently outperformed concentric on the same resolution (figure 8c). However, again, when provided the length input (figure 8d), the performance of the two placement schemes no longer significantly differed. Although each increment in resolution leads to a graceful degradation in performance, scaled individuals at all resolutions and in all configurations significantly outperformed a set of random individuals, showing further that scaling indeed retains abilities present in the lower-resolution network ($p < 0.01$ vs. random) Furthermore, remarkably, although *average time* degrades, most scaled networks could still collect all the food if their unscaled predecessor could.

In a more dramatic demonstration of scaling, one high fitness individual of each sensor placement scheme was scaled to a resolution of 1024. The resulting ANNs each had over *one million* connections and could still gather all the food.



(a) Concentric (b) Parallel

Figure 9: Repeated patterns in solutions. The figure depicts repeated motifs found by HyperNEAT at 8×8 resolution (top rows, two examples) and 16×16 resolution (bottom rows, same CPPNs). (a) Concentric and (b) parallel solutions are both shown. These patterns are encoded compactly by CPPNs with (a) 2 hidden nodes and 11 connections and (b) 2 hidden nodes and 9 connections. Thus an appropriate motif is discovered for both configurations for different resolutions.

5.2 Typical Solutions

In the optimal solution for both placement schemes, the sensor excites its corresponding effector while inhibiting all other connections (figure 9). There are also a surprising diversity of alternative solutions that are not optimal but can still collect all the food. Within a run, early solutions generally employ excitatory connections to several adjacent effectors or inhibitory connection to farther ones, while later solutions gradually pruned excess excitatory connections. Especially with concentric placement, suboptimal solutions sometimes included a few "good" sensor-effector connections that could quickly gather food, while using their other effectors only to drive the robot until the food was within range of a good sensor.

No matter the form of the solution, symmetry and repetition were ubiquitous. Figure 9 shows actual outgoing connections from sensors in evolved optimal solutions for each placement scheme. The same motif is duplicated at each sensor and continues to appear as the resolution is scaled. Although concentric layouts typically had more difficulty discovering fine-grained repetition, they frequently

created bilaterally-symmetric solutions that exploit the inherent symmetry of moving in two dimensions.

6. DISCUSSION AND FUTURE WORK

The main result is that different placement schemes create geometric relationships that are exploitable in different ways, some more easily than others. Connective CPPNs exploit regularities in parallel placement more efficiently than in concentric. The reason is that angular differences necessary to exploit concentric regularity take more structure to compute. Thus the activation functions in the CPPN are like a *language* that describes geometric structure. With the right words, it is easier to describe a particular relationship.

However, the results also show that the coordinate frames provided as input to the connective CPPN can significantly simplify the structure necessary to describe key motifs. When the geometric bias *connection length* is provided, both placement schemes produce equivalent results.

Furthermore, solutions are able to scale to higher resolutions while still retaining the general connectivity concept. This ability shows that even with their differences, both placement schemes can leverage the regularities in the physical task to encode a solution. The solutions thus cannot be simple explicit descriptions of which node connects to which, and thus must be generative. Slight degradation between successive resolutions occurs because the newly sampled points may contain elements of the pattern that were not evaluated by the fitness function. Nevertheless, that such solutions can recover the same performance as their lower-resolution predecessor after no more than five generations shows that the general concept is still present and only needs slight refinement to recover completely.

That different placement schemes and coordinate frames *do* produce different results creates a new category of research in ANN evolution. HyperNEAT allows the experimenter to inject knowledge into the search through such configurations, e.g. by grouping correlated objects or arranging sensors analogously to their real world geometry.

It is now possible to exploit the geometry of many domains and tasks for the first time. Thus, by applying HyperNEAT to tasks in control, vision, and decision-making, new kinds of solutions may be discovered that genuinely exploit regularities in such tasks. Finally, in the future, HyperNEAT will also choose the connectivity of hidden nodes.

7. CONCLUSIONS

A novel encoding called connective CPPNs was introduced that generates a connectivity pattern as a function of substrate geometry. This capability allows the evolutionary algorithm HyperNEAT to discover ANNs that exploit geometric regularities among sensors and effectors by discovering their underlying *connectivity concept*. Results from two different sensor placement schemes in a food gathering domain showed that some placement schemes are easier to exploit than others, although HyperNEAT ultimately could discover perfect regular solutions for both. Because they are encoded as concepts, these solutions could scale to up to over one million connections and still retain the ability to collect all the food. The long term implication of this work is that sensor and effector spatial geometry is now a crucial and useful factor in the evolution of ANNs.

8. ACKNOWLEDGMENTS

Special thanks to Colin Green for creating SharpNEAT, which is available through `http://www.cs.ucf.edu/~kstanley`.

9. REFERENCES

[1] P. J. Angeline. Morphogenic evolutionary computations: Introduction, issues and examples. In J. R. McDonnell, R. G. Reynolds, and D. B. Fogel, editors, *Evolutionary Programming IV: The Fourth Annual Conference on Evolutionary Programming*, pages 387–401. MIT Press, 1995.

[2] P. J. Bentley and S. Kumar. The ways to grow designs: A comparison of embryogenies for an evolutionary design problem. In *Proc. of the Genetic and Evol. Comp. Conf. (GECCO-1999)*, pages 35–43, San Francisco, 1999. Kaufmann.

[3] J. C. Bongard. Evolving modular genetic regulatory networks. In *Proceedings of the 2002 Congress on Evol. Comp.*, 2002.

[4] D. B. Chklovskii and A. A. Koulakov. MAPS IN THE BRAIN: What can we learn from them? *Annual Review of Neuroscience*, 27:369–392, 2004.

[5] D. Federici. Using embryonic stages to increase the evolvability of development. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2004) Workshop Program*, Berlin, 2004. Springer Verlag.

[6] G. J. Goodhill and M. A. Carreira-Perpinn. Cortical columns. In L. Nadel, editor, *Encyc. of Cognitive Science*, volume 1, pages 845–851. MacMillan Publishers Ltd., London, 2002.

[7] C. Green. SharpNEAT homepage. http://sharpneat.sourceforge.net/, 2003–2006.

[8] G. S. Hornby and J. B. Pollack. Creating high-level components with a generative representation for body-brain evolution. *Artificial Life*, 8(3), 2002.

[9] A. Lindenmayer. Adding continuous components to L-systems. In G. Rozenberg and A. Salomaa, editors, *L Systems, Lecture Notes in Computer Science 15*, pages 53–68. Springer-Verlag, Heidelberg, Germany, 1974.

[10] A. P. Martin. Increasing genomic complexity by gene duplication and the origin of vertebrates. *The American Naturalist*, 154(2):111–128, 1999.

[11] J. F. Miller. Evolving a self-repairing, self-regulating, French flag organism. In *Proc. of the Genetic and Evol. Comp. Conf. (GECCO-2004)*, Berlin, 2004. Springer Verlag.

[12] E. Mjolsness, D. H. Sharp, and J. Reinitz. A connectionist model of development. *Journal of Theoretical Biology*, 152:429–453, 1991.

[13] O. Sporns. Network analysis, complexity, and brain function. *Complexity*, 8(1):56–60, 2002.

[14] K. O. Stanley. Comparing artificial phenotypes with natural biological patterns. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) Workshop Program*, New York, NY, 2006. ACM Press.

[15] K. O. Stanley. Exploiting regularity without development. In *Proceedings of the AAAI Fall Symposium on Developmental Systems*, Menlo Park, CA, 2006. AAAI Press.

[16] K. O. Stanley. Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines Special Issue on Developmental Systems*, 2007. To appear.

[17] K. O. Stanley, B. D. Bryant, and R. Miikkulainen. Real-time neuroevolution in the NERO video game. *IEEE Transactions on Evolutionary Computation Special Issue on Evolutionary Computation and Games*, 9(6):653–668, 2005.

[18] K. O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evol. Comp.*, 10:99–127, 2002.

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

[20] K. O. Stanley and R. Miikkulainen. Competitive coevolution through evolutionary complexification. *Journal of Artificial Intelligence Research*, 21:63–100, 2004.

[21] A. Turing. The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society B*, 237:37–72, 1952.

[22] X. Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, 1999.

[23] M. J. Zigmond, F. E. Bloom, S. C. Landis, J. L. Roberts, and L. R. Squire, editors. *Fundamental Neuroscience*. Academic Press, London, 1999.