Evolving Multimodal Controllers with HyperNEAT

In: *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2013).* New York, NY: ACM

Justin K. Pugh
Department of EECS
University of Central Florida
Orlando, FL 32816 USA
jpugh@eecs.ucf.edu

Kenneth O. Stanley
Department of EECS
University of Central Florida
Orlando, FL 32816 USA
kstanley@eecs.ucf.edu

ABSTRACT

Natural brains effectively integrate multiple sensory modalities and act upon the world through multiple effector types. As researchers strive to evolve more sophisticated neural controllers, confronting the challenge of multimodality is becoming increasingly important. As a solution, this paper presents a principled new approach to exploiting indirect encoding to incorporate multimodality based on the HyperNEAT generative neuroevolution algorithm called the *multi-spatial substrate* (MSS). The main idea is to place each input and output modality on its own independent plane. That way, the spatial separation of such groupings provides HyperNEAT an a priori hint on which neurons are associated with which that can be exploited from the start of evolution. To validate this approach, the MSS is compared with more conventional approaches to HyperNEAT substrate design in a multiagent domain featuring three input and two output modalities. The new approach both significantly outperforms conventional approaches and reduces the creative burden on the user to design the layout of the substrate, thereby opening formerly prohibitive multimodal problems to neuroevolution.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning—*connectionism and neural nets*

General Terms

Algorithms, Performance, Experimentation

Keywords

HyperNEAT, Multimodal Input, Multiagent Learning, Generative and Developmental Systems, Neuroevolution

1. INTRODUCTION

Among the long-term goals of the field of neuroevolution is to evolve artificial neural networks (ANNs) with capabilities similar to the human brain. Although current methods of evolving ANNs cannot approach the complexity of the brain, recent work focusing on indirect encoding has enabled the evolution of ANNs with thousands to millions of connections from genotypes with far fewer components (on the order of 20 to 30) [8, 24]. However, one aspect of the human brain that has not yet attracted significant attention in the neuroevolution literature is the diversity of senses. The human brain receives input from millions of receptors representing a wide array of sensory modalities (e.g. sight, touch, taste, smell, hearing, pain, temperature, balance, proprioception, etc.) [12]. This sensory richness provides the human brain with complementary representations of its environment that are critical to effective function.

As researchers continue to seek more advanced behaviors from evolved neural controllers, it will become increasingly important also to equip such controllers with multiple modalities that can potentially each support its own high resolution. However, successfully integrating different kinds of sensory information in an ANN is challenging because there is no explicit way to denote to the network which sensors are grouped with which. Even outside of the evolutionary community, there has been interest in recent years in multimodal integration. For example, Loyola and Coldewey-Egbers [13] developed a stacked neural network architecture that merges sensor data for long-term climate research. Interestingly, the field of *generative and developmental systems* (GDS) [1, 2, 10, 14, 18, 21] may be able to help with multimodal integration in evolved ANNs by grouping different types of sensors in different parts of the generated phenotype. Furthermore, the processing of different sensory modalities may share some principles, enabling information reuse. For example, in a foraging domain, an agent should act in a similar manner when seeing a food source as when smelling a food source. Inspired by this possibility, this paper presents an elaboration of the conventional HyperNEAT GDS approach for evolving large-scale ANNs [8, 24] called the *multi-spatial substrate* (MSS), which enables a principled integration of several sensory modalities.

While it has long been unclear how or where to place different types of sensors or outputs on the same HyperNEAT substrate, the key new idea is to place different sensory modalities on sheets encoded in HyperNEAT as entirely disconnected geometric spaces. That way there is no risk of confusing or conflating one modality with another. Thus a solution is offered to the longstanding dilemma about how to configure complex HyperNEAT substrates that at the same time yields a principled approach to the problem of

GECCO'13, July 6–10, 2013, Amsterdam, The Netherlands.

representing and learning from multiple modalities in general. This approach is demonstrated in a multiagent coordination domain that requires agents to perceive three modalities at once: walls, targets, and communications from other agents. The MSS approach significantly outperforms more conventional configurations in which all modalities are placed on the same plane and is also simpler to set up. This new principled ability to train in multimodal domains thereby opens up many such tasks to research that were perhaps prohibitive in the past.

2. BACKGROUND

This section reviews research foundational to the MSS approach.

2.1 Neuroevolution of Augmenting Topologies

The HyperNEAT approach extended in this paper is itself an extension of the original NEAT (Neuroevolution of Augmenting Topologies) algorithm that evolves increasingly large ANNs [20, 22]. NEAT starts with a population of simple networks that then *increase in complexity* over generations by adding new nodes and connections through mutations. By evolving ANNs in this way, the topology of the network does not need to be known a priori; NEAT searches through increasingly complex networks to find a suitable level of complexity. Because it starts simply and gradually adds complexity, it tends to find a solution network close to the minimal necessary size. However, as explained in the next section, the direct representation of nodes and connections in the NEAT genome cannot scale up to large brain-like networks. For a complete overview of NEAT, see Stanley and Miikkulainen [20] or Stanley and Miikkulainen [22].

2.2 HyperNEAT

Neuroevolution methods like NEAT are *directly encoded*, which means each component of the phenotype is encoded by a single gene, making the discovery of repeating motifs expensive and improbable. Therefore, indirect encodings [1, 2, 10, 14, 21], which are a key instrument of GDS, have become a growing area of interest in evolutionary computation.

One such indirect encoding designed explicitly for neural networks is in Hypercube-based NEAT (HyperNEAT) [8, 24], which is itself an indirect extension of the directly-encoded NEAT approach [20, 22] reviewed in the previous section. The geometric principles behind HyperNEAT lay the foundation for the multi-spatial approach introduced later. This section briefly reviews HyperNEAT; a complete introduction can be found in Stanley et al. [24] and Gauci and Stanley [8]. Rather than expressing connection weights as independent parameters in the genome, HyperNEAT allows them to vary across the phenotype in a regular pattern through an indirect encoding called a *compositional pattern producing network* (CPPN; [18]), which is like an ANN, but with specially-chosen activation functions.

CPPNs in HyperNEAT *encode* the connectivity patterns of ANNs as a *function of geometry*. That is, if an ANN's nodes are embedded in a geometry, i.e. assigned coordinates within a space, then it is possible to represent its connectivity as a single evolved function of such coordinates. In effect the CPPN paints a pattern of weights across the geometry of a neural network. Because the CPPN encoding is itself a network, it is evolved in HyperNEAT by the NEAT algorithm, which is designed to evolve networks of increasing complexity. To understand why this approach is promising, consider that a natural organism's brain is physically embedded within a three-dimensional geometric space, and that such embedding heavily constrains and influences the brain's connectivity. Topographic maps (i.e. ordered projections of sensory or effector systems such as the retina or musculature) in natural brains preserve geometric relationships between high-dimensional sensor and effector fields [11, 25]. In other words, there is important information *implicit* in geometry that can only be exploited by an encoding informed by such geometry.

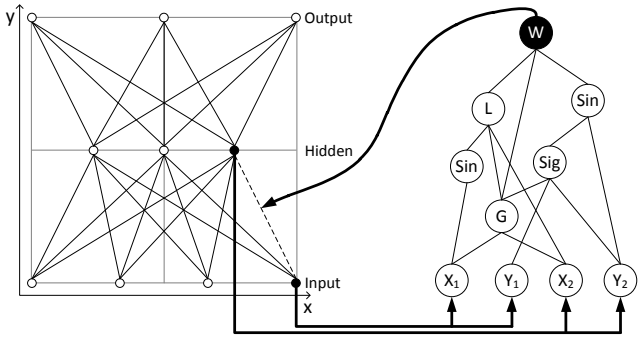


Figure 1: HyperNEAT example. An example substrate (left) for a simple ANN contains ten neurons that have been assigned (x, y) coordinates. The weight of every connection specified in the substrate is determined by the evolved CPPN (right): (1) The coordinates of the source (x_1, y_1) and target (x_2, y_2) neurons are input into the CPPN, (2) the CPPN is activated, and (3) the weight w of the connection being queried is set to the CPPN's output. CPPN activation functions in this paper can be *sigmoid (Sig), Gaussian (G), linear (L), or sine (Sin).*

In particular, geometric *regularities* such as symmetry or repetition are pervasive throughout the connectivity of natural brains. To similarly achieve such regularities, CPPNs exploit activation functions that induce regularities in HyperNEAT networks. The general idea is that a CPPN takes as input the geometric coordinates of two nodes embedded in the *substrate*, i.e. an ANN situated in a particular geometry, and outputs the weight of the connection between those two nodes (figure 1). In this way, a Gaussian activation function by virtue of its symmetry can induce symmetric connectivity and a sine function can induce networks with repeated elements. Note that because the size of the CPPN is decoupled from the size of the substrate, HyperNEAT can compactly encode the connectivity of an arbitrarily large substrate with a single CPPN.

Additionally, HyperNEAT can evolve controllers for *teams* of agents. This multiagent HyperNEAT algorithm is described by D'Ambrosio et al. [5]. It can be used to evolve both homogeneous and heterogeneous teams; however, the experiments in this paper only necessitate the homogeneous case (without loss of generality). Controllers for homogeneous teams are created by evolving a single controller that is duplicated for each agent on the team.

2.3 Sensory Input in Neuroevolved Agents

Researchers have evolved neural-controlled agents for diverse domains ranging in complexity from single-agent maze navigation [7] to multiagent video game warfare [23]. Early

work centered on behaviors that could be realized on available robot hardware, such as the Khepera. For example, Floreano and Mondada [7] evolve controllers for the Khepera robot that can navigate a simple maze without colliding with walls. These controllers feature only a single sensory modality: a set of infrared proximity sensors that surround the Khepera robot. In similar work, Nolfi [15] evolves controllers for a Khepera robot equipped with a "gripper" module capable of grabbing and releasing objects. Successful integration of the gripper module requires the addition of an additional sensory modality: a single input neuron that is activated when an object is present inside the gripper claw.

Some neural controllers are evolved in simulation without the intention of physical implementation. For example, video game non-player characters (NPCs) often must process a relatively diverse set of sensory inputs. Stanley et al. [23] present such a video game called NERO in which human players attempt to evolve a team of ANN-controlled agents for a combat simulation against other teams of such agents. Agents in NERO have sensory inputs encompassing at least four distinct modalities: enemies, walls, hostile line-of-fire, and a boolean sensor that detects whether the agent's gun is currently aimed at an enemy target. Schrum and Miikkulainen [17] demonstrate video game NPC controllers evolved for a variety of multitask combat-related domains that have thirteen distinct sensory modalities. While researchers such as Clune et al. [4] and Verbancsics and Stanley [26] have studied the potential for the learning algorithm to discover modular divisions on its own, the opportunity raised in this paper is to provide some kind of a priori *hint* to the algorithm about the ideal groupings of neurons (such as when they belong to different modalities). No such hint was possible to provide in the past because there is no existing technique for providing one.

Neuroevolution methods based on an indirect encoding, such as HyperNEAT, offer a potential benefit to learning a large diversity of sensory input because geometric arrangements such as in HyperNEAT could in principle help to convey *belonging* to one modality or another. Furthermore, some modalities may share certain properties and thus benefit from information reuse. For example, in some cases modalities that consist of a single boolean value should be treated in a similar way that is different from the way that continuous rangefinder arrays should be treated. However, current applications of HyperNEAT feature relatively few sensory modalities [3, 5, 6, 8, 16, 24]. For example, Stanley et al. [24] evolve agents with HyperNEAT for a food gathering task. In this application, agents only have a single array of eight rangefinders (corresponding to a single sensory modality). Similarly, the agents featured in Risi and Stanley [16] have only two sensory modalities: wall sensors and target sensors. Although multiagent domains often necessitate the addition of extra sensory information to facilitate communication and the detection of friendly agent locations or status, multiagent HyperNEAT has so far only been used to evolve agents with one [5] or two [6] sensory modalities. The application of HyperNEAT that features the most diverse array of sensory information to date is Clune et al. [3]. In this application, HyperNEAT networks tasked with controlling the locomotion of a quadrupedal walker receive input from several modalities: the angles of the three joints in each of the legs, boolean values that are active when their corresponding leg is touching the ground, the rotation of the walker's body, and a sine input used to generate oscillations. With no known principle or precedent to follow, the arrangement of all these on a single substrate posed a difficult challenge to Clune et al. [3].

Perhaps the reason that HyperNEAT-evolved neural controllers often have few sensory modalities is that as more types of input are added, it becomes increasingly unclear how to organize the neurons on the substrate. The conventional approach to substrate design is to arrange the different modalities arbitrarily along one or two spatial dimensions, such as in Clune et al. [3]. The order and spacing of modalities along these dimensions can significantly impact the results. By convention, the best ordering is generally determined experimentally. However, this approach is computationally expensive and impractical. This paper instead offers a principled solution that places each sensory modality in its own space, similarly to (though much expanded from) how the input, hidden, and output layers are divided in past work such as Gauci and Stanley [8]. This approach is explained in detail in the next section.

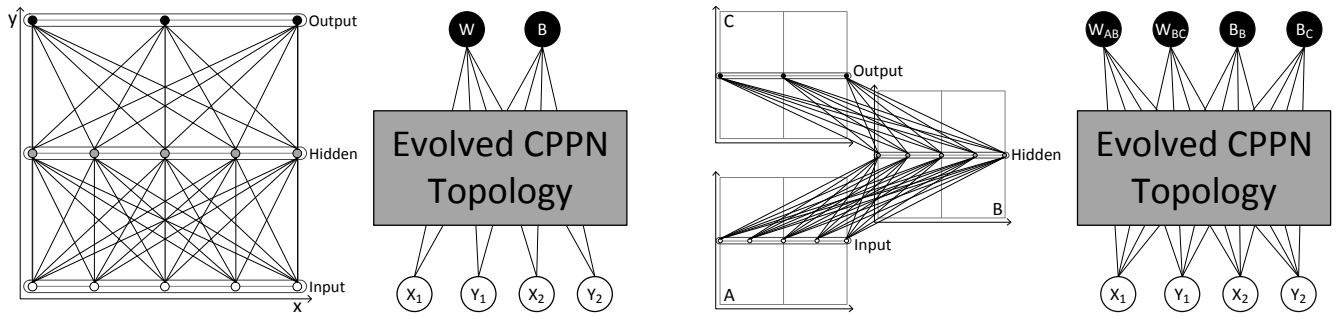
3. APPROACH: MULTI-SPATIAL SUBSTRATE

The primary challenge in designing a HyperNEAT substrate is deciding how to spatially arrange several sets of geometrically unrelated neurons. This section begins by explaining the main idea for addressing this problem with simple substrates, and then turns to more complex multimodal arrangements.

3.1 Simple Multi-Spatial Substrates

The most common example that appears in nearly every substrate design problem is deciding how the input, hidden, and output sets of neurons should appear in the substrate space relative to each other. Imagine a simple agent that has a set of five rangefinder sensors equally spaced along the front 180 degrees of the agent and three effectors to perform the actions of turning left, moving forward, and turning right. Each sensor corresponds to a single input neuron and each effector corresponds to a single output neuron. This neural controller also has a set of five hidden neurons to which the input neurons are fully connected and that are fully connected to the output neurons. A common design for such a substrate is shown in figure 2a. Because there exists a clear geometric relationship among the inputs, a good practice is to arrange them in a line along one dimension of the substrate (such as parallel to the x-axis in figure 2a) in the same order that the corresponding sensors appear on the agent. A similar geometric relationship exists among the outputs, which are therefore often arranged in a line as well. Arranging the inputs and outputs in this way allows HyperNEAT to discover the concept of left-right symmetry along the inputs and outputs in a single step. The hidden nodes are often arranged in a line in the same way as the inputs and outputs to convey an identical left-right symmetry along the hidden nodes.

Overall, the substrate designer is left to situate three distinct sets of neurons, each arranged in a line parallel to the x-axis. However, it is unclear how to place these sets in the final substrate *relative to each other* because there is no clear geometric relationship among them. A common practice is to arrange them arbitrarily along another, orthogonal axis



(a) Crowded substrate (CS) **(b)** CPPN (CS) **(c)** Multi-spatial substrate (MSS) **(d)** CPPN (MSS)

Figure 2: Example: Converting a crowded substrate to a multi-spatial substrate. Two substrate designs are depicted for a simple neural controller with one set of inputs, one hidden layer, and one set of outputs. The first substrate (a) follows the crowded substrate approach, which is typical for designing substrates: Neuron groups that are not geometrically related are arranged arbitrarily along an "extra" spatial dimension (in this case, the y-axis) that is otherwise not used to convey geometric information. The second substrate (c) follows the multi-spatial substrate approach, wherein geometrically independent neuron groups are placed in separate spaces (i.e. planes). Also shown are the corresponding CPPNs for both substrates (b, d). Note that the addition of extra planes in the multi-spatial substrate necessitates extra CPPN outputs (d). W_{AB} is queried when determining the weight of connections from plane A to plane B and W_{BC} is queried in the same way for connections from plane B to plane C. B_B and B_C are queried when determining the implicit bias for neurons on plane B and plane C, respectively (there is no B_A because input neurons do not have an implicit bias). In this example, the conversion to a multi-spatial substrate removes the need for the y-axis entirely; thus, the Y_1 and Y_2 inputs in (d) could be removed (not shown). However, the y-axis may also be repurposed (for example, to expand the hidden layer to a grid of neurons).

(such as the y-axis in figure 2a). This arbitrary placement of geometrically unrelated neuron groups is referred to in this paper as the *crowded substrate* because it involves squeezing several different types of neurons into the same space. To understand why it is a single shared space, notice that the CPPN in figure 2b that encodes the crowded substrate has a single weight output W (and similarly a single bias output B). In effect, that means mathematically that all weights and biases are projected onto the same plane.

The main contribution of this paper is to introduce an approach to designing HyperNEAT substrates that avoids the arbitrary placement of geometrically unrelated neurons by first grouping neurons that are logically related. That is, each input and output modality forms a separate group, and hidden neurons can be grouped according to their connectivity to other neuron groups in the network. Then each group is placed in a *separate* space according to the principle: *Two groups of neurons with no obvious geometric relationship between them should not be placed in the same space.*

In the case of the example in figure 2, each "space" can be conceived as a separate plane and there are three logical groupings of neurons. Thus the resulting substrate contains three separate planes (figure 2c). Substrates organized in this way are called *multi-spatial substrates* (MSS) because geometrically unrelated sets of neurons are placed in different spaces. In practice, adding spaces to the substrate accordingly requires extra CPPN outputs to differentiate among the spaces being connected. While the crowded (single-spatial) substrate requires two CPPN outputs (one that is read when querying connection weights and one that is read when querying the implicit bias[1] on every non-input neuron), the multi-spatial substrate requires $N + M$ CPPN outputs where N is the number of unique space-space pairings (i.e. where there exist neurons from one space that are connected to neurons on the other space) and M is the number of planes that contain non-input neurons. It should be noted that a space with internal connections counts as a separate pairing (i.e. a space paired with itself) for the purposes of calculating N. Figure 2d shows the MSS encoding that is an alternative to the crowded encoding in figure 2b.

In addition to being more challenging to design, crowded substrates impose an evolutionary burden on HyperNEAT. If geometrically unrelated neuron groups are arranged along the y-axis, the HyperNEAT CPPN must learn a function of y that expresses the differences among these logical groups of neurons. This function becomes more complicated and thus more difficult to discover during evolution as more groups of neurons are added to the substrate (e.g. when the number of input or output modalities increases, or when more layers of hidden neurons are added to the network). By providing a separate CPPN output for each logically separate set of connections, multi-spatial substrates do not need to learn a potentially complex function to differentiate among these sets. Rather, the separation of each set into its own CPPN output conveys this information a priori, thereby providing the hint that has heretofore been missing. While CPPNs such as the one in figure 2d have appeared before with little commentary, such as in Gauci and Stanley [8], the novel insight here is that the separation provided by the multiple outputs is actually a *solution* to the problem of configuring and training networks of many modalities. In particular, the hypothesis of this paper is that this additional information provided to the HyperNEAT CPPN will improve evolutionary performance,

[1]The implicit bias is a convention in HyperNEAT that simulates a single bias neuron that is fully connected to all other neurons except the inputs [19]. The CPPN is queried for the bias of a neuron by setting the X_1 and Y_1 inputs to the neuron's coordinates and setting X_2 and Y_2 to zero. Implementing the bias in this way effectively places the bias at the center of a separate plane.

particularly in cases where there are many logically separate neuron groups, as described next.

3.2 Multi-Spatial Substrates for Multimodal Controllers

Substrates for multimodal controllers tend to have more logically separate neuron groups than substrates for unimodal controllers such as the example in figure 2. A class of domains that naturally requires multimodality is multiagent learning. Consider a team of robots with three input modalities, (1) target sensing, (2) communication sensing, and (3) wall sensing, as well as two output modalities: (1) steering and (2) a "voice box" for sending communication signals. Target sensors consist of five pie slice sensors across the front 180 degrees of the robot. Wall sensors are similarly arranged except rangefinders are used instead of pie slices. Communication sensors consist of ten pie slice sensors that surround the robot. Steering is determined by three outputs: turn left, move straight, and turn right. The "voice box" output activates other robots' communication sensors that point in the direction of the vocalizing robot. A controller for such a robot may require a more complex system of hidden layers than a single layer of five neurons to successfully process its variety of sensory modalities. Due to the additional modalities and hidden layers, the resulting network architecture would have significantly more logically separate neuron groups than the simple controller discussed earlier. Figure 3 depicts a possible logical architecture for the connectivity of such a controller. From a design perspective, actually realizing this logical configuration is difficult with a single-plane crowded substrate and there are many possibilities; several such alternative crowded substrate designs are shown in figure 4. However, the multi-spatial substrate (figure 5) is simple to configure because different modalities are simply placed on different planes.

The next section presents an experiment comparing the performance of a multi-spatial substrate to that of substrates designed with the crowded substrate approach.

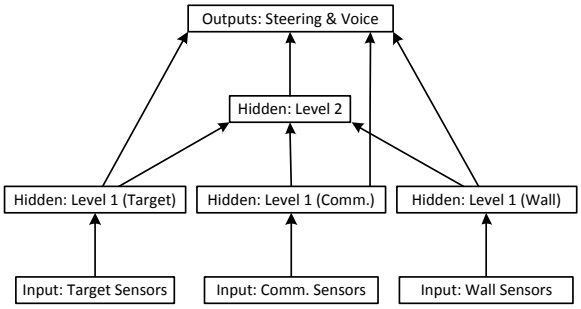


Figure 3: Substrate logical connectivity. The logical connectivity for all the substrates compared experimentally in this paper is shown. Neuron groups shown as connected are potentially fully connected (i.e. all neurons in the first group are queried for connections to all in the second group). Each input modality is connected in turn to a dedicated "level 1" hidden layer. All level 1 hidden layers are connected to a "level 2" hidden layer. Finally, all hidden layers are connected to the two output layers. Steering outputs and voice outputs are logically separated in this paper, although they are merged in this diagram to reduce clutter.

4. EXPERIMENT

The multiagent example domain just described in section 3.2 is well-suited to illustrating the evolutionary advantage of the MSS approach because it relies on multimodal input. For this purpose, five agents matching the description in Section 3.2 are evolved with multiagent HyperNEAT[2] for a maze exploration and group coordination task.

The agents are placed in a maze containing one target point that is initially out of vision range of all agents. The goal of the task is that the agents navigate their way to the target point and remain there. The best performance requires one agent to call the others over after discovering the target. Fitness is accumulated as follows: Each agent earns one point per tick of the clock that it is within a small distance (slightly larger than collision range) of the target point. The duration of the simulation is 1,000 ticks, which corresponds to a maximum fitness of 5,000 for the five agents. However, due to the time taken for the initial discovery of the target point and for all agents to move to it[3], the maximum achievable fitness is lower. Nearly perfect solutions wherein agents explore efficiently and move directly to the target point once it is found achieve a fitness of about 4,200. Agents' sensors for seeing walls and the target points have limited range[4] and are occluded by walls. However, agent communication sensors have unlimited range and are not occluded. The primary purpose of communication therefore is to share the location of the target point with other agents when it is too far away or occluded by a wall.

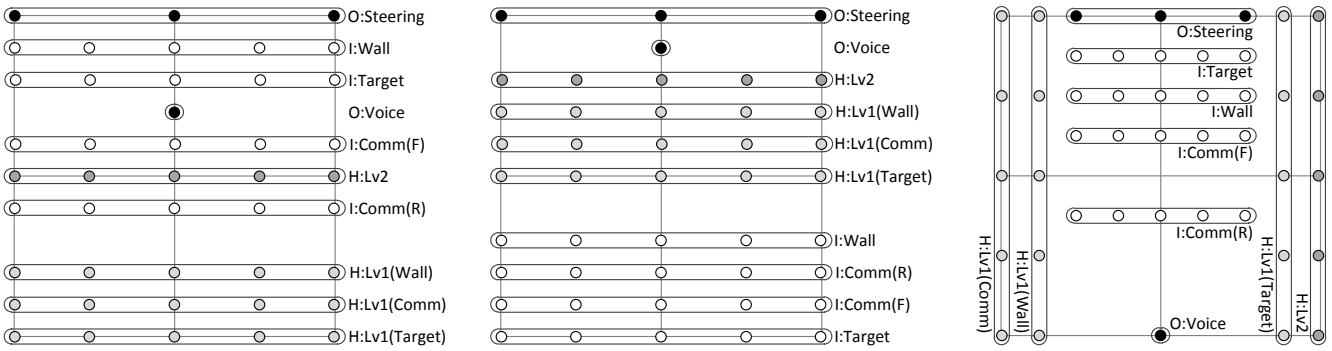
Teams are trained on a total of seven environments, shown in figure 6. Environments are designed to encompass a diversity of situations, including initial isolation from team members. During evolution, fitness is averaged over all seven.

The experiment consists of a comparison between the evolutionary performance of HyperNEAT with each of the three crowded substrates in figure 4 as well as the two multi-spatial substrates described in figure 5. The crowded substrates test several possible ways of arranging neuron groups that have no obvious geometric relationship. In the first crowded substrate, CS1 (figure 4a), neuron groups are arranged such that front sensors appear in the positive y-axis and rear sensors appear in the negative y-axis. Additionally, the outputs are spaced reasonably far apart to prevent a correlation between the activations of the forward steering and voice box outputs from emerging early in evolution. CS2 (figure 4b) is designed such that all connections flow from neuron groups located at lower values of y to neuron groups located at higher values of y. In CS3 (figure 4c), horizontal neuron groups are compressed to make room for the hidden layers to be positioned vertically along the left and right edges of the substrate. The first multi-spatial substrate, MSS1, is shown in figure 5. Here, the level 2 hidden layer (shown at upper-left) is converted from a horizontal line of neurons to a grid of neurons (made possible because of the MSS), which

[2]Teams are composed of homogeneous agents in this paper; all agents on a team have identical ANNs.

[3]Agents move at a maximum rate of 5 units per tick and have a maximum turn rate of 36 degrees per tick. The width of each environment's bounding walls is 900 units. Thus it takes 180 ticks for an agent to move across the world; extra time is required to navigate around obstacles.

[4]Target and wall sensor range is 150 units.



(a) CS1 **(b)** CS2 **(c)** CS3

Figure 4: Alternate crowded substrates. In each crowded substrate, the input, hidden, and output neurons are all placed together on a single plane, organized geometrically into groups of related neurons. Input neurons (denoted by I) are colored white, hidden neurons (denoted by H) are colored light gray (level 1) and dark gray (level 2), and output neurons (denoted by the letter O) are colored black. The communication ($Comm$) inputs are divided into two groups, corresponding to the front sensors (F) and the rear sensors (R). Connections are omitted for clarity (see figure 3 for connectivity).

makes more complex functionality possible to express. MSS2 (not pictured) is identical to MSS1 except that its level 2 hidden layer consists of a single horizontal line of 5 neurons. This variation helps to validate that any advantage for the MSS is not only from having a grid of neurons at level 2.

Evolution was allowed 500 generations, after which point fitness improvements stagnate, even for runs that do not find a good solution. Because HyperNEAT differs from original NEAT only in its set of activation functions, it uses the same parameters [20]. The experiment was run with a modified version of the public domain SharpNEAT package [9]. The size of the population was 500 with 20% elitism. Sexual offspring (50%) did not undergo mutation. Asexual offspring (50%) had 0.96 probability of link weight mutation, 0.03 chance of link addition, and 0.01 chance of node addition. The coefficients for determining species similarity were 1.0 for nodes and connections and 0.1 for weights. The available CPPN activation functions were sigmoid, Gaussian, linear, and sine, all with equal probability of being added to the CPPN. Parameter settings are based on standard SharpNEAT defaults and prior reported settings for NEAT [20, 22]. They were found to be robust to moderate variation through preliminary experimentation.

5. RESULTS

Evolutionary performance of each substrate design, determined by the best fitness achieved at each generation, is averaged across 30 runs and presented in figure 7. The differences between MSS1 and MSS2 are not statistically significant.[5] Both MSS1 and MSS2 significantly outperform all three crowded substrates ($p < 0.001$), as determined by the final fitness achieved at generation 500. CS2 outperforms the other two crowded substrate designs ($p < 0.05$).

To provide further perspective on the disparity between CS and MSS, a good solution is defined as a fitness greater than 3,333, corresponding to all agents spending two-thirds of the total available time at the target point. This threshold allows agents to cross the world twice before finding the target point and excludes nearly all cases in which one or more agents never find it. The proportion of runs that find a good solution for each approach is presented in figure 8. Both MSS designs find a good solution almost every time, whereas even the best crowded substrate designs find a good solution only half the time. The worst crowded substrate, CS3, finds a good solution in only two out of 30 runs.

The agents on the best solution teams for MSS1 and MSS2 explore the map individually and only activate their voice box output after finding the target point. The other agents move towards the signal, navigating around walls, until the entire team has reached the target. Thus agents effectively share information about the location of the target point with team members, expediting the exploratory process. Many of the best performing CS solution teams behave in the same way. However, some of the CS2 teams solve the problem without communication because the agents keep their voice box output activated at all times. These teams achieve slightly lower scores than communicating teams, but nonetheless exceed the success threshold.

6. DISCUSSION

The crowded substrate faces the problem that the CPPN tends to assign similar weights (especially early in evolution) to connections that are near each other, such as those connecting to the voice box and forward movement neurons on the CS2 substrate (figure 4b). Such a correlation can become entrenched during the course of evolution if the candidate solution learns to rely on it, which explains why several of the best CS2 teams exhibit a correlation between forward movement and communication at generation 500 even though a higher score is possible without such correlation.

As more neurons are added to a crowded substrate to solve more complex problems, it is ultimately inevitable that some unrelated neurons will have to be placed close enough together to cause unintended correlations in early generations. In contrast, neurons in a multi-spatial substrate easily avoid unintended correlations because each modality exists in an entirely separate space. In a MSS, logical separation is not conveyed by spatial distance on the substrate but by separate CPPN outputs, which is a cleaner distinction that is easily followed by HyperNEAT.

[5]Statistical significance is determined by an unpaired two-tailed Student's t-test in all reported cases.

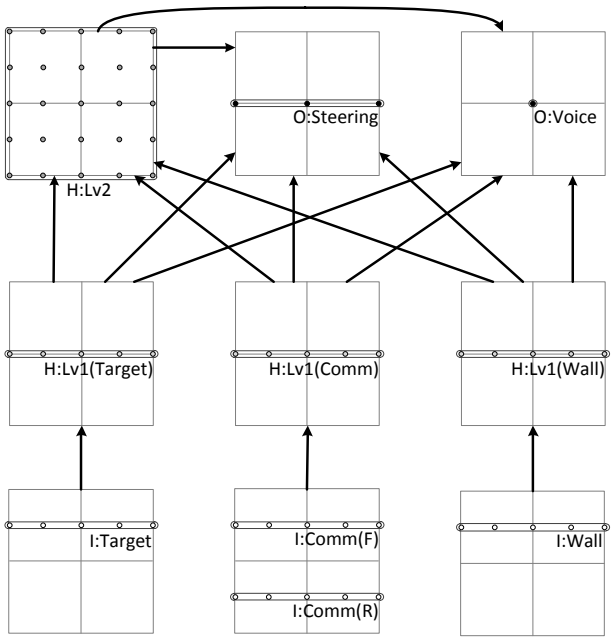


Figure 5: MSS1 and MSS2 (not shown). Each input and output modality and each hidden layer is placed on its own plane. The level 2 hidden layer (upper left) is expanded to a grid of neurons to allow the expression of more complex functionality. Front sensors are located at $y = 0.5$ and rear sensors are located at $y = -0.5$ in their input plane. Neural connections are omitted for clarity. Instead, arrows indicate the existence of neural connections between two planes. Planes shown as connected are potentially fully connected (all neurons on the first plane are queried for connections to all neurons on the second plane). The CPPN for this substrate has a total of 20 outputs (connections exist between 14 unique plane-plane pairs and there are 6 planes that have neurons with an implicit bias). MSS2 (not shown) is identical to MSS1 except its level 2 hidden layer (H:Lv2) only has a single line of five neurons at $y = 0$.

Figure 7 shows that the way crowded substrates are arranged can significantly impact their performance. Indeed, some CS substrate designs even prevent solving the problem (e.g. CS3, figure 8). More complicated domains that necessitate additional modalities and hidden layer structure present an even greater challenge to the substrate designer because the degree of neuron crowding is greatly increased. In effect, as the number of modalities and associated modality-specific hidden layers increases, the burden on the user to design an effective crowded substrate increases until eventually the problem becomes intractable. While it is conceivable that the dimensionality of the crowded substrate could be expanded to reduce crowding (e.g. more CPPN inputs), this would only perpetuate the problem of deciding where to place unrelated modalities within the substrate.

On the other hand, the MSS approach eliminates the need to make such difficult design decisions even while potentially performing better. This result opens the door for future research evolving ANN controllers capable of processing a rich diversity of sensory information that was not previously possible, which can impact domains ranging from biped walking to multiagent control. Furthermore, the MSS technique may

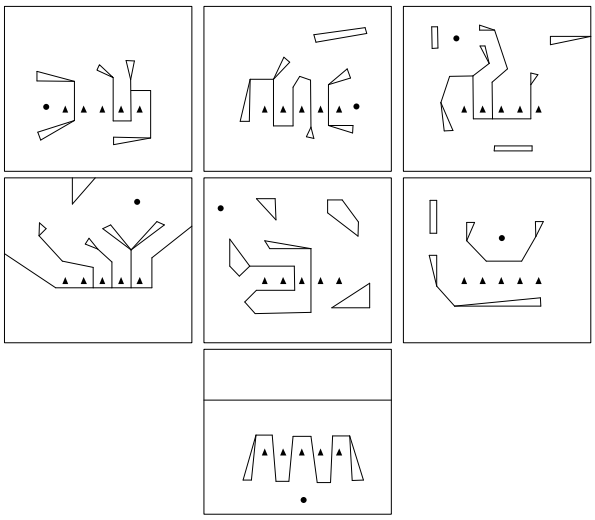


Figure 6: Training environments. Team performance is averaged over seven environments, depicted here. Agent starting positions are represented by triangles and target points are represented by circles. The ends of walls are thick to ensure agents can see them.

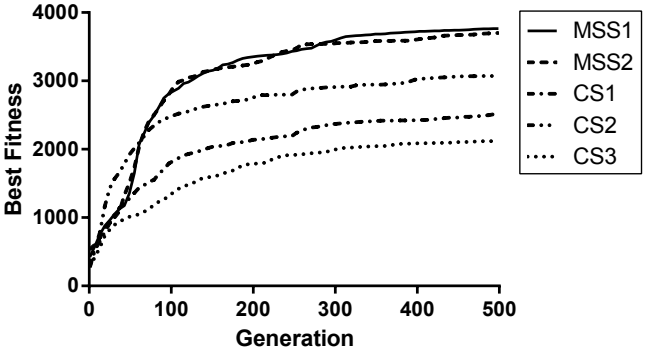


Figure 7: Evolutionary performance by generation. The best fitness achieved at each generation by each of the five substrate designs is shown, averaged over 30 runs each. The MSS variants reach significantly higher fitnesses by generation 500 ($p < 0.001$).

be useful whenever there are several a priori logical groupings of neurons within a substrate, which is not restricted to multimodal controllers. In the future it will be important to study the scalability of this approach, as well as how well it may complement more organic approaches to modularity such as the Link Expression Output [26] or optimizing for connection costs [4].

7. CONCLUSIONS

This paper introduced a principled new approach to substrate design for the HyperNEAT algorithm called the multi-spatial substrate, which provides a priori information to HyperNEAT about the distinctions among neuron groups, both improving evolutionary performance and reducing the creative burden on the user. The approach was validated through a comparison with conventional methods of substrate design on a multiagent domain wherein agents have three input and two output modalities; the multi-spatial sub-

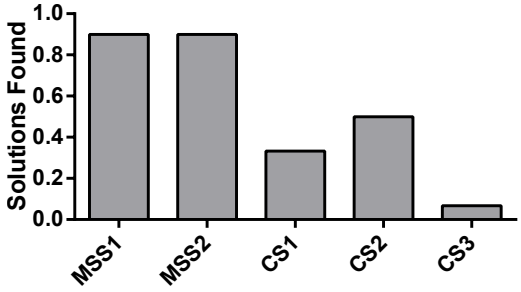


Figure 8: Proportion of runs that find a good solution. Proportions are taken out of 30 runs. A good solution is defined by a fitness threshold of 3,333. MSS finds such solutions 80% more often than the best CS.

strate found a solution 80% more often than even the best conventional substrate design in the tested domain. This new approach brings within the reach of neuroevolution a new tier of domains by providing a solution to the problem of multimodality that was not previously available.

Acknowledgments

This work was supported through a grant from the US Army Research Office (Award No. W911NF-11-1-0489). This paper does not necessarily reflect the position or policy of the government, and no official endorsement should be inferred.

References

[1] P. J. Bentley and S. Kumar. Three ways to grow designs: A comparison of embryogenies for an evolutionary design problem. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 1999)*, pages 35–43, San Francisco, 1999. Kaufmann.

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

[3] J. Clune, B. E. Beckmann, C. Ofria, and R. T. Pennock. Evolving coordinated quadruped gaits with the HyperNEAT generative encoding. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2009)*, Piscataway, NJ, USA, 2009. IEEE Press.

[4] J. Clune, J.-B. Mouret, and H. Lipson. The evolutionary origins of modularity. *Proceedings of the Royal Society B: Biological Sciences*, 280(1755), 2013.

[5] D. B. D'Ambrosio, J. Lehman, S. Risi, and K. O. Stanley. Evolving policy geometry for scalable multiagent learning. In *Proceedings of the Ninth International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, pages 731–738. 2010.

[6] D. B. D'Ambrosio, J. Lehman, S. Risi, and K. O. Stanley. Task switching in multirobot learning in multiagent learning through indirect encoding. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS 2011)*, Piscataway, NJ, 2011. IEEE.

[7] D. Floreano and F. Mondada. Automatic creation of an autonomous agent: Genetic evolution of a neural-network driven robot. In *From Animals to Animats III: Proceedings of the Third International Conference on Simulation of Adaptive Behavior*. MIT Press, 1994.

[8] J. Gauci and K. O. Stanley. Autonomous evolution of topographic regularities in artificial neural networks. *Neural Computation*, 22(7):1860–1898, 2010.

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

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

[11] D. Hubel and T. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, 160(1):106, 1962.

[12] E. R. Kandel, J. H. Schwartz, and T. M. Jessell. *Principles of Neural Science*. McGraw-Hill, New York, fourth edition, 2000.

[13] D. G. Loyola and M. Coldewey-Egbers. Multi-sensor data merging with stacked neural networks for the creation of satellite long-term climate data records. *EURASIP Journal on Advances in Signal Processing*, 2012(1):1–10, 2012.

[14] J. Miller. Evolving a self-repairing, self-regulating, french flag organism. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2004)*, pages 129–139. Springer, 2004.

[15] S. Nolfi. Evolving non-trivial behavior on real robots: A garbage collecting robot. *Robotics and Autonomous Systems*, 22:187–198, 1997.

[16] S. Risi and K. O. Stanley. An enhanced hypercube-based encoding for evolving the placement, density and connectivity of neurons. *Artificial Life*, 18(4):331–363, 2012.

[17] J. Schrum and R. Miikkulainen. Evolving multimodal networks for multitask games. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(2):94–111, 2012.

[18] K. O. Stanley. Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines Special Issue on Developmental Systems*, 8(2):131–162, 2007.

[19] K. O. Stanley. HyperNEAT user's page. `http://eplex.cs.ucf.edu/hyperNEATpage/`, 2009–2013.

[20] K. O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127, 2002.

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

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

[23] 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.

[24] K. O. Stanley, D. B. D'Ambrosio, and J. Gauci. A hypercube-based indirect encoding for evolving large-scale neural networks. *Artificial Life*, 15(2):185–212, 2009.

[25] S. Udin and J. Fawcett. Formation of topographic maps. *Annual Review of Neuroscience*, 11(1):289–327, 1988.

[26] P. Verbancsics and K. O. Stanley. Constraining connectivity to encourage modularity in hyperneat. In *Proc. of the Genetic and Evolutionary Computation Conf. (GECCO 2011)*, pages 1483–1490. ACM, 2011.