Evolving Policy Geometry for Scalable Multiagent Learning

In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010)

David B. D'Ambrosio, Joel Lehman, Sebastian Risi, and Kenneth O. Stanley
School of Electrical Engineering and Computer Science
University of Central Florida
4000 Central Florida Blvd.
Orlando, FL 32816-2362 USA
<ddambro,jlehman,risi,kstanley>@eecs.ucf.edu

ABSTRACT

A major challenge for traditional approaches to multiagent learning is to train teams that easily scale to include additional agents. The problem is that such approaches typically encode each agent's policy separately. Such separation means that computational complexity explodes as the number of agents in the team increases, and also leads to the *problem of reinvention*: Skills that should be shared among agents must be rediscovered separately for each agent. To address this problem, this paper presents an alternative evolutionary approach to multiagent learning called *multiagent HyperNEAT* that encodes the team as a *pattern* of related policies rather than as a set of individual agents. To capture this pattern, a *policy geometry* is introduced to describe the relationship between each agent's policy and its canonical geometric position within the team. Because policy geometry can encode variations of a shared skill across all of the policies it represents, the problem of reinvention is avoided. Furthermore, because the policy geometry of a particular team can be sampled at any resolution, it acts as a heuristic for generating policies for teams of *any* size, producing a powerful new capability for multiagent learning. In this paper, multiagent HyperNEAT is tested in predator-prey and room-clearing domains. In both domains the results are effective teams that can be successfully scaled to larger team sizes without any further training.

Categories and Subject Descriptors: I.2.6 [**Artificial Intelligence**]: Learning—*connectionism and neural nets, concept learning*; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent Systems*

General Terms: Algorithms

Keywords: CPPNs, HyperNEAT, Multiagent learning, NEAT, Neural Networks, Evolutionary Computation

1. INTRODUCTION

Two significant challenges in multiagent learning are training large teams and scaling the size of trained teams to productively employ an arbitrary number of agents. Current approaches to multiagent learning, such as cooperative coevolutionary algorithms (CCEAs [13, 21, 23, 24]) and multiagent reinforcement learning (MARL [4, 8, 20]), incur exponential computational cost as teams grow larger and also do not provide a principle for integrating new agents into an existing heterogeneous team. In contrast, this paper presents a novel heuristic for representing and training a team of agents based on discovering how agents' roles interrelate, which also allows teams to be scaled to arbitrary sizes without additional learning.

An important observation of human teams is that the behaviors of human agents tend to be related to their canonical position within a team (e.g. at the start of a match). Such teams effectively have a *policy geometry*, i.e. a functional relationship between an agent's location within the team and that agent's behavioral policy. Thus, rather than individually learning each agent's policy, an intriguing possibility for such teams is to instead learn a general relationship between team geometry and policies that can be *sampled* to *derive* the policies of individual agents.

While in traditional approaches the complexity of a multiagent learning problem increases exponentially with the size of the team [6, 21], by representing a multiagent team as a *pattern of policies* rather than as individual agents, this problem is avoided: Only a single unifying pattern must be learned regardless of the number of agents in the team. Furthermore, while most real-life teams' policies lie *between* extreme homogeneity or heterogeneity, many traditional multiagent learning methods do not exploit such partial homogeneity and thus face the *problem of reinvention*: Skills that should be shared across the team must be reinvented separately for each agent. In contrast, the policy geometry can represent such commonalities (at the same time as differences) without reinvention.

To implement this idea of exploiting policy geometry to create scalable teams, *hypercube-based neuroevolution of augmenting topologies* (HyperNEAT), an approach to evolving artificial neural networks (ANNs), is extended to encode *patterns of ANNs distributed across space*. The spatial distribution of ANNs matches with the initial locations of agents on a team, thereby allowing HyperNEAT to learn the policy geometry of a given task. In this way, HyperNEAT can reuse critical information to overcome both the problem of reinvention and the difficulty of scaling a team's size to an arbitrary number of agents. The approach to multiagent HyperNEAT in this paper substantially revises a prior such approach [11] to make it more amenable to scaling.

This idea is tested in both predator-prey and room-clearing tasks. In both tasks it evolves effective multiagent teams that scale to larger team sizes, in some cases up to 1,000 agents, that nonetheless coordinate seamlessly without any additional training.

Cite as: Evolving Policy Geometry for Scalable Multiagent Learning, D. B. D'Ambrosio, J. Lehman, S. Risi, and K. O. Stanley, *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. XXX-XXX.

2. BACKGROUND

This section reviews popular approaches to multiagent learning and the NEAT and HyperNEAT methods that form the backbone of multiagent HyperNEAT.

2.1 Cooperative Multiagent Learning

Multiagent reinforcement learning (MARL) encompasses several specific techniques based on off-policy and on-policy temporal difference learning [4, 8, 20]. The basic principle that unifies MARL techniques is to identify and reward promising cooperative states and actions among a team of agents [7,21]. The other major approach, cooperative coevolutionary algorithms (CCEAs), is an established evolutionary method for training teams of agents that must work together [13,21,23]. The main idea is to maintain one or more populations of candidate agents, evaluate them in groups, and guide the creation of new candidate solutions based on their joint performance.

While reinforcement learning and evolution are mainly the focus of separate communities, Panait, Tuyls, and Luke [22] showed recently that they share a significant common theoretical foundation. One key commonality is that they break the learning problem into separate roles that are semi-independent and thereby learned separately through interaction with each other. Although this idea of separating multiagent problems into parts is appealing, it does create challenges for achieving certain desirable capabilities such as scaling to larger team sizes, which is a focus in this paper. The problem is that when individual roles are learned separately, there is no representation of how roles relate to the team structure and therefore no principle for assigning new roles automatically to new individuals.

Both CCEAs and MARL face the *problem of reinvention*. That is, because agents are treated as separate subproblems they must usually separately discover and represent all aspects of the solution, even though optimally there may be a high degree of overlapping information among the policies of each agent. CCEAs commonly separate agents into different populations, creating strict divisions among agents, and in MARL methods, each agent learns a separate reward function based upon individual or joint experiences. There have been attempts to address the problem of reinvention such as introducing existing agents that "train" new agents [25, 32] or implementing specially-designed genetic operators [18]. However, an intriguing alternative is to exploit the *continuum of heterogeneity*, which means distributing shared skills optimally among the agents and only representing such skills once. At the same time, unique abilities could be isolated and assigned appropriately. The method in this paper addresses the problem of reinvention by finding the right point on the continuum of heterogeneity for a given team.

The next section reviews the Neuroevolution of Augmenting Topologies (NEAT) method, the foundation for the multiagent learning approach introduced in this paper.

2.2 Neuroevolution of Augmenting Topologies

The HyperNEAT method that enables learning from geometry in this paper is an extension of the NEAT algorithm for evolving ANNs. NEAT performs well in a variety of control and decision-making problems [1, 29, 31]. It starts with a population of small, simple neural networks and then *complexifies* them over generations by adding new nodes and connections through mutation. By evolving networks 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. Furthermore, it allows NEAT to establish high-level features early in evolution and then later elaborate on them.

The important property of NEAT for this paper is that it evolves *both* the topology and weights of a neural network. Because it starts simply and gradually adds complexity, NEAT tends to find a solution network close to the minimal necessary size. In principle, another method for learning the topology and weights of networks could also fill the role of NEAT in this paper. Nevertheless, what is important is to begin with a principled approach to learning both such features, which NEAT provides. Stanley and Miikkulainen [29,31] provide a complete overview of NEAT.

The next section reviews the HyperNEAT extension to NEAT that is itself extended in this paper to generate multiagent teams.

2.3 HyperNEAT

A key similarity among many neuroevolution methods, including NEAT, is that they employ a *direct* encoding, that is, each part of the solution's representation maps to a single piece of structure in the final solution. Yet direct encodings impose the significant disadvantage that even when different parts of the solution are similar, they must be encoded and therefore discovered separately. This challenge is related to the problem of reinvention in multiagent systems: After all, if individual team members are encoded by separate genetic code, even if a component of their capabilities is shared, the learner has no way to exploit such a regularity. Thus this paper employs an *indirect* encoding instead, which means that the description of the solution is compressed such that information can be reused, allowing the final solution to contain more components than the description itself. Indirect encodings are powerful because they allow solutions to be represented as a pattern of policy parameters, rather than requiring each parameter to be represented individually [2, 3, 19, 27, 30]. HyperNEAT, reviewed in this section, is an indirect encoding extension of NEAT that is proven in a number of challenging domains that require discovering regularities [9,10,14,15,28]. For a full description of HyperNEAT see Stanley et al. [28] and Gauci and Stanley [15].

In HyperNEAT, NEAT is altered to evolve an indirect encoding called *compositional pattern producing networks* (CPPNs [27]) *instead* of ANNs. CPPNs, which are also networks, are designed to encode *compositions of functions*, wherein each function in the composition (which exists in the network as an activation function for a node) loosely corresponds to a useful regularity. For example, a Gaussian function induces symmetry. Each such component function also creates a novel geometric *coordinate frame* within which other functions can reside. For example, any function *of* the output of a Gaussian will output a symmetric pattern because the Gaussian is symmetric.

The appeal of this encoding is that it allows spatial patterns to be represented as networks of simple functions (i.e. CPPNs). Therefore NEAT can evolve CPPNs just like ANNs; CPPNs are similar to ANNs, but they rely on more than one activation function (each representing a common regularity) and act as an *encoding* rather than a network.

The indirect CPPN encoding can compactly encode patterns with regularities such as symmetry, repetition, and

repetition with variation [26, 27]. For example, while including a Gaussian function, which is symmetric, causes the output to be symmetric, a periodic function such as sine creates segmentation through repetition. Most importantly, *repetition with variation* (e.g. such as the fingers of the human hand) is easily discovered by combining regular coordinate frames (e.g. sine and Gaussian) with irregular ones (e.g. the asymmetric x-axis). For example, a function that takes as input the sum of a symmetric function and an asymmetric function outputs a pattern with imperfect symmetry. In this way, CPPNs produce regular patterns with subtle variations. The potential for CPPNs to represent patterns with motifs reminiscent of patterns in natural organisms has been demonstrated in several studies [27] including an online service on which users collaboratively breed patterns represented by CPPNs [26].

The main idea in HyperNEAT is that CPPNs can naturally encode *connectivity patterns* [14, 15, 28]. That way, NEAT can evolve CPPNs that represent large-scale ANNs with their own symmetries and regularities. This capability will prove essential to encoding multiagent policy geometries in this paper because it will ultimately allow connectivity patterns to be expressed as a function of team geometry, which means that a smooth gradient of policies can be produced across possible agent locations.

Formally, CPPNs are *functions* of geometry (i.e. locations in space) that output connectivity patterns whose nodes are situated in n dimensions, where n is the number of dimensions in a Cartesian space. Consider a CPPN that takes four inputs labeled x_1, y_1, x_2, and y_2; this point in four-dimensional space *also* denotes the connection between the two-dimensional points (x_1, y_1) and (x_2, y_2), and the output of the CPPN for that input thereby represents the weight of that connection (Figure 1). By querying every possible connection among a set of points in this manner, a CPPN can produce an ANN, wherein each point is a neuron position. Because the connection weights are produced by a function of their endpoints, the final structure is produced with *knowledge* of its geometry. In effect, the CPPN is painting a pattern on the inside of a four-dimensional hypercube that is interpreted as the isomorphic connectivity pattern, which explains the origin of the name *hypercube-based NEAT* (HyperNEAT). Connectivity patterns produced by a CPPN are called *substrates* in order to verbally distinguish them from the CPPN itself, which has its own internal topology.

Each queried point in the substrate is a node in a neural network. The experimenter defines both the location and role (i.e. hidden, input, or output) of each such node. Nodes should be placed on the substrate to reflect the geometry of the task [9, 10, 14, 15, 28]. That way, the connectivity of the substrate is a function of the the task structure.

For example, the sensors of an autonomous robot can be placed from left to right on the substrate in the same order that they exist on the robot. Outputs for moving left or right can also be placed in the same order, allowing HyperNEAT to understand from the outset the correlation of sensors to effectors. In this way, knowledge about the problem geometry can be injected into the search and HyperNEAT can exploit the regularities (e.g. adjacency, or symmetry) of a problem that are invisible to traditional encodings.

In summary, the capabilities of HyperNEAT are important for multiagent learning because they provide a formalism for producing policies (i.e. the output of the CPPN) as a function of geometry (i.e. the inputs to the CPPN). As explained next, not only can such an approach produce a single network but it can also produce a set of networks that are each generated as a function of their location in space.

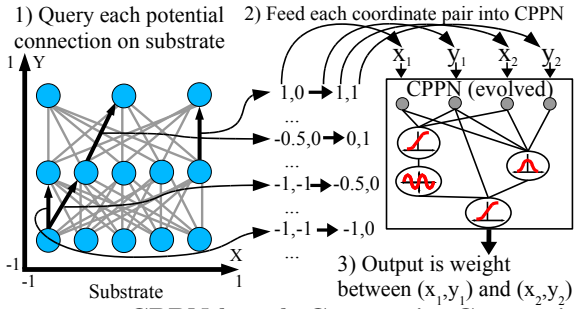


Figure 1: CPPN-based Geometric Connectivity Pattern Encoding. A collection nodes, called the *substrate*, is assigned coordinates that range from -1 to 1 in all dimensions. (1) Every potential connection in the substrate is queried to determine its presence and weight; the dark directed lines in the substrate depicted in the figure represent a sample of connections that are queried. (2) Internally, the CPPN (which is evolved by NEAT) is a graph that determines which activation functions are connected. As in an ANN, the connections are weighted such that the output of a function is multiplied by the weight of its outgoing connection. 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, CPPNs can produce regular patterns of connection weights in space.

3. APPROACH: Multiagent HyperNEAT

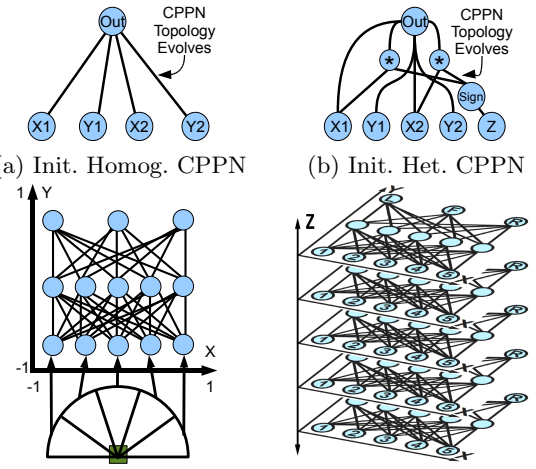
Recall that the *policy geometry* of a team is the relationship between the canonical starting positions of agents on the field and their behavioral policies. Multiagent HyperNEAT exploits the fact that policy geometry can be encoded naturally as a pattern. To understand how the policy geometry of a team can be encoded, it helps to begin by considering *homogeneous teams*, which in effect express a trivial policy geometry in which the same policy is uniformly distributed throughout the team at all positions. Thus this section begins by exploring how teams of purely homogeneous agents can be evolved with an indirect encoding, and then transitions to a method for evolving heterogeneous teams that are represented by a single genome in HyperNEAT.

3.1 Pure Homogeneous Teams

A homogeneous team is composed of a *single* controller that is copied for each agent on the team. To generate such a controller, a CPPN with inputs x_1, y_1, x_2, and y_2 (Figure 2a) queries the substrate shown in Figure 2c, which has five inputs, five hidden nodes, and three output nodes, to determine its connection weights. This substrate is designed to geometrically correlate sensors to corresponding outputs (e.g. seeing something on the left and turning left). Thus the CPPN can exploit the geometry of the agent [28]. However, the agents themselves have exactly the same policy no matter where they are positioned; while each agent is informed by geometry, their policies cannot differentiate genetically.

3.2 The Continuum of Heterogeneity

Heterogeneous teams are a greater challenge; how can a single CPPN encode a *set* of networks in a pattern, all with



(a) Init. Homog. CPPN (b) Init. Het. CPPN

(c) Homog. Substrate (d) Het. Substrate

Figure 2: Multiagent HyperNEAT. This figure depicts the CPPNs and substrates for homogeneous and heterogeneous learning in HyperNEAT. The CPPN in (a) generates a single controller for a single agent or a homogeneous team of agents. The single controller substrate that is queried by this CPPN is shown in (c). In contrast, the CPPN in (b) encodes heterogeneous teams by sampling the substrate in (d), which is made up of the single substrate (c) repeated across the substrate five times. This repetition of coordinate frames is accomplished through the z-axis, because each value of z corresponds to a new set of x and y coordinates. Note that CPPNs depicted in (a) and (b) increase in complexity over evolution through the NEAT algorithm and the CPPN in (b) begins with additional structure that provides knowledge of mirrored symmetry along z through special nodes that multiply the sum of one set of their inputs by the sum of a second set of inputs.

related yet varying roles? Indirect encodings such as HyperNEAT are naturally suited to capturing such patterns by encoding the policy geometry of the team as a pattern. The remainder of this section discusses the method by which HyperNEAT can encode such teams.

The main idea is that the CPPN is able to create a pattern based on *both* the agent's internal geometry (x and y) *and* its position on the team (z) (Figure 2b,d). The CPPN can thus emphasize connections from z for increasing heterogeneity or minimize them to produce greater homogeneity. Furthermore, because z is a spatial dimension, the CPPN can literally generate policies based on their positions on the team. Note that because z is a single dimension, the policy geometry of this team (and those in this paper) is on a one-dimensional line. However, in principle, more inputs could be added, allowing two- or more dimensional policy geometry to be learned as well. The starting structure of the initial heterogeneous CPPN (Figure 2b) includes some additional structure to start evolution with knowledge of mirrored team symmetry; in particular, the sign function of z connected to multiplication operators on the x-axis means that the CPPN knows which side of the team is which and can opt to reflect one side across the midline.

The heterogeneous substrate (Figure 2d) formalizes the idea of encoding a team as a pattern of policies. This capability is powerful because generating each agent with the same CPPN means they can share tactics and policies while still exhibiting variation across the policy geometry. In other words, policies are spread across the substrate in a pattern just as role assignment in a human team forms a pattern across a field. However, even as roles vary, many skills are shared, an idea elegantly captured by indirect encoding. The complete multiagent HyperNEAT algorithm is enumerated in Algorithm 1.

1. Set the substrate to contain the necessary number of agents (Figure 2d).
2. Initialize a population of CPPNs with random weights.
3. Repeat until a solution is found or the maximum number of generations are reached:
   (a) For each CPPN in the population:
      i. Query the CPPN for the weight of each connection in the substrate within each agent's ANN. If the absolute value of the output exceeds a threshold magnitude, create the connection with a weight scaled proportionally to the output value (Figure 1).
      ii. Assign the generated ANNs to the appropriate agents and run the team in the task domain to ascertain fitness.
   (b) Reproduce the CPPNs according to the NEAT method to create the next generation.

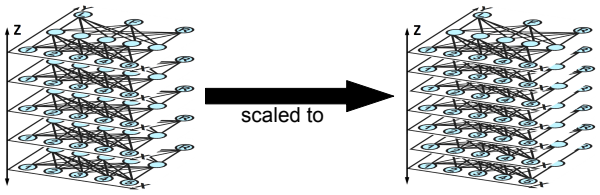
Algorithm 1: Multiagent HyperNEAT

A key property of the heterogeneous substrate is that if a new network is added to the substrate at an intermediate location, its policy can theoretically be interpolated from the policy geometry embodied in the CPPN. Thus, as the next section describes, it becomes possible to scale teams without further training by interpolating new roles.

3.3 Scaling

As discussed in Section 2.1, traditional multiagent learning techniques struggle to represent cooperative teams of more than a few heterogeneous agents and there are typically few rules or principles from which to determine how *additional* agents could be added after learning has taken place. However, in real world tasks, it would be most convenient if the number of possible agents is unbounded and independent of the number initially trained. Furthermore, because agents may break down or additional ones may become available, ideally the size of a learned team should be dynamically adjustable *after* deployment.

Consider a soccer team, which includes eleven agents with assigned roles. How might additional agents be added to such a team, e.g. between the midfielders and the forwards? Intuitively, the implicit policy geometry suggests that these new agents should interpolate between the policies of the surrounding agents. That is, they should be relatively offensive, but not as offensive as the players in front of them. Traditional techniques have no way to exploit this policy geometry because they treat each agent independently and would thus require retraining to assign roles to the new agents. However, because teams in multiagent HyperNEAT are represented by the CPPN as a *pattern of policies* rather than as individual agents, the CPPN effectively encodes an infinite number of heterogeneous agents that can be sampled as needed without the need for additional learning. Thus if more agents are required, the substrate can be updated to encompass the new agents and resampled by the CPPN to



(a) 5 Agent Team (b) 7 Agent Team

Figure 3: Heterogeneous Scaling. Because multiagent HyperNEAT represents teams as a pattern of policies, it is possible to interpolate new polices in the policy geometry for additional agents by sampling new points on the substrate. The original substrate (a) is scaled by inserting new two-dimensional substrate slices along the z-axis (b).

assign policies to them without further evolving the CPPN.

In fact, the heterogeneous substrate accommodates additional agents by simply redistributing their controllers on the z-axis so that they are uniformly spaced in accordance with their new number (Figure 3).

Note that these steps can be taken *after* learning is completed and the new agent policies will be automatically interpolated based on the policy geometry by simply requerying the CPPN. There is no limit to the size to which such a substrate can be scaled in this way. Thus, through this approach, a new form of post-deployment scaling is introduced.

4. EXPERIMENTS

This section discusses the two experiments in this paper, a predator-prey task and a room-clearing task, which are designed to demonstrate multiagent HyperNEAT's novel capability to flexibly scale heterogeneous teams.

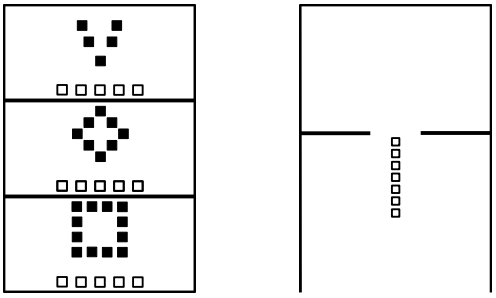
4.1 Predator-Prey

In the cooperative multiagent predator-prey domain, a group of five predators cooperate to capture a group of prey. Because a predator may easily undermine a teammate's pursuit of a prey, predators must learn consistent, complementary roles. All of the predators also need basic skills to interpret and react to their sensors. Because multiagent HyperNEAT creates all the agent ANNs from the *same* CPPN, it has the potential to balance these contrasting ingredients over teams with any number of agents.

Each predator agent is controlled by the ANN in Figure 2c (which is a single slice of the heterogeneous substrate in Figure 2d). Predators are equipped with five rangefinder sensors that detect nearby prey, but *cannot* sense other predators, making the task highly challenging by requiring careful coordination. At each discrete moment, a predator can turn and/or move in small increments. The predators' goal is to capture the prey agents by running into them.

Prey agents have a fixed policy; they maintain their current location until threatened by nearby predators, and then move in the opposite direction of the nearest predator until they are once again at a safe distance. Because prey move at the maximum speed of predator agents, it is impossible for a solitary predator to catch a prey, so they must coordinate. Each team of predators is trained on two variations of one of three prey formations (triangle, diamond, and square; Figure 4a) to balance generalization and specialization.

The predator team starts each trial in an evenly-spaced line, facing the prey (Figure 4a). The environment is physically unbounded, but each trial has finite length (at most



(a) Formations (b) Room

Figure 4: Domains. In predator-prey (a), the prey (upper, shaded agents) are arranged in one of three formations. The predators (lower, hollow agents) are always placed in an evenly-spaced line (which mirrors their policy geometry) below the prey. In room clearing (b), agents lined up in front of the entrance to a room must cooperate to enter and observe as much of the room as possible at once.

1,000 time steps). During a trial the predators attempt to capture the prey, and at the end of each trial the team receives a score of $10{,}000p + (1{,}000 - t)$, where p is the number of prey captured and t is the time it takes to catch all the prey. If all prey are not captured, t is set to 1,000. Team fitness is the sum of the scores from the two trials on which the team is evaluated, which encourages the predators to capture all the prey as quickly as possible.

The challenge for the predators is to coordinate despite their inability to see one another. This restriction encourages establishing *a priori* policies for cooperation because agents thus have little information to infer each others' current states. Such scenarios are not uncommon; military units often form plans, split up, and execute complicated maneuvers with little to no contact with each other [12].

4.2 Room Clearing

Room clearing is a good domain to further demonstrate the abilities of multiagent HyperNEAT because in this domain, sensory information alone is inadequate to robustly solve the task and physical collisions between agents are enforced even though they cannot see each other. It is also an important military task that may one day be performed by unmanned ground vehicles. In the room-clearing domain, a group of agents enter a room and position themselves to visually cover as much of the room as possible (assuming 360 degree rotating cameras). The optimal solution is for each agent to be as far as possible from other agents. However, the agents' rangefinder sensors do not directly specify whether a particular area is already covered by another agent; thus, a priori role assignment is critical to obtaining good performance. Coordination among varied policies is also necessary because colliding with other unseen agents may prevent an agent from reaching its desired destination. The ability to avoid walls and navigate the room are also important for all the agents. As in the predator-prey domain, multiagent HyperNEAT has the ability to balance the need for these heterogeneous and homogeneous elements of policy across teams with any number of agents.

Agents are controlled by ANNs similar in architecture to those in the predator-prey domain, but augmented with recurrent connections so that agents can potentially retain a basic memory of past states. Each agent has 11 rangefinder sensors that indicate the distance to nearby walls but not

nearby agents, which reflects a possible configuration of sensors on real robots and also makes the task more difficult. On each discrete time step, an agent's ANN outputs dictate whether it turns and/or moves in small increments. If an agent requests to move at a velocity that is below a certain threshold, it will be interpreted as a request to stop, and the agent's motor will be disabled for the remainder of the evaluation. The room-clearing teams are trained at a size of seven agents because preliminary experiments showed that seven is the smallest team size for which avoiding overlap is non-trivial. Thus, because the principle of spreading is the key to scaling effectively, seven is a good size on which to start to produce a team optimized to spread.

Agents are evaluated in a single rectangular room for 35 simulated seconds comprised of 0.2 second discrete time steps. Agents begin outside the room in an evenly spaced vertical line; a fixed policy of moving directly forward is maintained for each agent until they are inside the room, at which point each agent is under control of its evolved ANN for the remainder of the evaluation. To measure how much of the room is covered by the team of agents, a grid is superimposed upon the room. A grid square is deemed covered if an agent is in close proximity to it. For each of the last 15 seconds of the evaluation, the number of grid squares inside the room covered by the agents is counted to determine the team's fitness score. This measure of fitness encourages agents to spread quickly throughout a room just as a real team of humans would, to rapidly assess risk.

4.3 Homogeneous vs. Heterogeneous Policies

To demonstrate that HyperNEAT gains from heterogeneous role-differentiation, both purely homogeneous and heterogeneous teams are trained and tested. Homogeneous teams must rely on their current perceived state to differentiate themselves, which is effective in some tasks [5]. In contrast, heterogeneous teams have more tactical options available because the search does not need to find one global policy that works for all agents in all cases, that is, it can separate the problem among agents. Such separation can be distributed logically across the team (e.g. agents on the left attack prey on the left). Additionally, while the policies may be heterogeneous, they likely should overlap by a significant amount (e.g. all predators know how to turn to face prey).

The aim is to establish in both domains that the continuum of heterogeneity is indeed being exploited.

4.4 Scaling

Adding agents to a team that was trained at a different size is challenging because the new agents must be assigned policies automatically. Yet such a capability could benefit real-world multiagent applications such as the room-clearing task; the ability to immediately integrate these new agents could be critical. The *interpolated scaling* (Figure 3) made possible by the heterogeneous substrate is a unique capability of HyperNEAT that addresses this problem.

To test this capability, the best CPPN of each generation of training is tested on several different team sizes *without further learning*. While the methodology for scaling is the same, the details of adding new agents to each domain varies slightly. In the predator-prey domain the teams are tested on sizes 7, 9, 25, 100, and $1,000$, which are chosen to test the teams' abilities to scale on many levels. However, the room in the room-clearing domain is too small to accommodate scaling to very large team sizes. Thus these teams are tested on all odd-numbered team sizes from 9 to 23. The scaled teams are tested on the same tasks as the original teams, isolating the effects of scaling.

4.5 Experimental Parameters

Because HyperNEAT differs from original NEAT only in its set of activation functions, it uses the same parameters [29]. Both experiments were run with a modified version of the public domain SharpNEAT package [16]. The size of each population was 150 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 available CPPN activation functions were sigmoid, Gaussian, absolute value, 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 [29, 31]. They were found to be robust to moderate variation through preliminary experimentation.

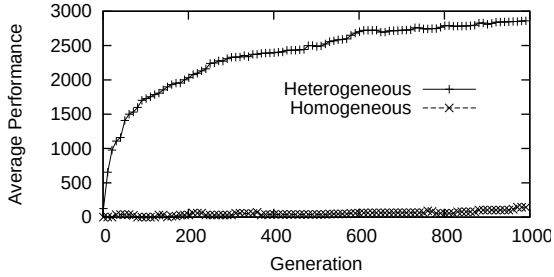
5. RESULTS

Figure 5 shows the training performance over generations of the teams in both domains averaged over 20 runs in each configuration of predator-prey (60 total) and 20 runs of room clearing. In both cases, heterogeneous teams find significantly more efficient policies and do so more quickly than the homogeneous teams ($p < 0.001$ in both domains), confirming the utility of a team-wide policy geometry. In fact, the homogeneous teams rarely solve the task at all in the predator-prey domain. Thus the heterogeneity of policies discovered by HyperNEAT is helpful in both domains.

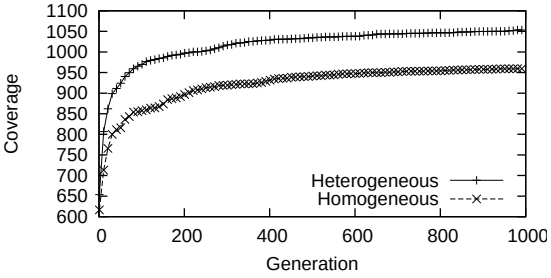
5.1 Scaling Performance

Multiagent HyperNEAT encodes a team as a pattern of policies distributed across the geometry of the team rather than as a group of individual agents. Thus each CPPN can encode the policies of any number of agents, allowing the team size to change *without further training*. To confirm this capability the teams in both domains were scaled to include more agents without any additional learning. The policies of new agents in heterogeneous configurations are determined by inserting new networks into the substrate as in Figure 3. The main question is whether performance can be maintained (and even improved) after scaling. Figure 6 shows the scaling performance of the the teams in the two domains. For the predator-prey domain, scaling results are shown as speed (defined as the time taken to capture all prey, where a full 2,000 time steps per formation is given if the team is unable to capture all of them) at which the task was completed for the new team sizes (lower is better). In the room-clearing domain, results are measured as room coverage over the last 15 seconds (higher is better). The data was collected by testing every generation champion from each of the 20 runs on all scaling sizes. The team that scores the highest in the most new scenarios is chosen to represent each run. The scores of these best scalers from each of the 20 runs are then averaged. This method of testing follows from Gruau et al. [17] and is designed to compare the best overall teams that each method can create.

Note that in general as team size increases, performance should improve because either there are more predators available against the same number of prey or there are more



(a) Predator-Prey Training



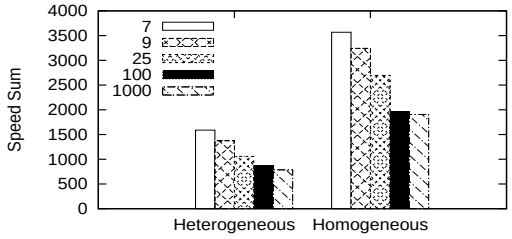
(b) Room Clearing Training

Figure 5: Training Performance. (a) For the predator-prey domain the training performance (defined as the average time remaining after catching all the prey) of the team is averaged over the three formation shapes and over twenty runs on each shape. (b) In the room-clearing domain performance (defined as sum of the number of grid squares covered by the team during the evaluation period) is averaged over twenty runs. In both cases the heterogeneous teams learned the best solutions and learned most quickly.

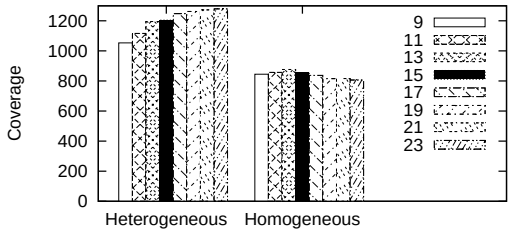
agents to cover the same size room. However, such improvement is not guaranteed because the agents cannot sense each other so it is possible for new agents to unknowingly interfere or become redundant with the previously existing agents. Thus the question is whether coordination is maintained so that performance improves as more agents are added.

The main result is that multiagent HyperNEAT makes it possible for heterogeneous teams to increase in size without additional learning (Figure 6). Scaling performance generally mirrors training performance in the sense that the heterogeneous teams significantly outperform ($p < 0.001$) the homogeneous teams in both domains. Interestingly, while homogeneous teams also benefit from more agents in the predator-prey task, their performance begins to *decline* as more agents are added in room clearing.

In both domains the difference between both teams is quantitative *and* qualitative; the heterogeneous teams typically employ complex and scalable strategies while the homogeneous teams apply unsophisticated approaches. For example, in the predator-prey domain one strategy used by the heterogeneous teams is corralling, wherein several predators encircle the prey and push them inward. In contrast, homogeneous teams typically just chase prey in circles until they encounter another predator. Similarly, in the room-clearing domain the heterogeneous agents tend to split in half and stop moving when they are in advantageous positions, while the homogeneous agents bunch into groups and then split apart. Such a strategy is problematic because additional agents cause more collisions, making it hard to split apart, which explains the declining homogeneous performance in room clearing as team size increases.



(a) Predator-Prey Scaling (lower is better)



(b) Room Clearing Scaling (higher is better)

Figure 6: Scaling Performance. Scaling performance (time remaining for predator-prey and grid squares observed for room clearing) is shown for teams in both domains. It is determined by averaging the performance of the team from each run with the best performance on all team sizes over all twenty runs. Heterogeneous teams significantly outperform homogeneous teams at scaling in each domain.

Video demonstrations of these behaviors are available at `http://eplex.cs.ucf.edu/mahnaamas2010.html`.

Thus the main conclusion is that heterogeneous teams, which can interpolate new roles through the policy geometry, scale best (indeed it is significant that *heterogeneous* teams, which perform better, can scale at all). This result suggests that multiagent HyperNEAT is an effective new approach to training *scalable* heterogeneous multiagent teams.

6. DISCUSSION

In their recent survey of cooperative multiagent learning, Panait and Luke [21] cite scalability to be a "major open topic" in the field. In a first step in this direction, multiagent HyperNEAT was able to create teams of heterogeneous agents that could scale several orders of magnitude in size without further training. Such a task is prohibitive for traditional heterogeneous multiagent learning techniques, which do not encode the team as a pattern, yet multiagent HyperNEAT accomplished it by representing the teams as patterns rather than as individual agents and exploiting the policy geometry of the teams.

In this sense, the major contribution of this work is conceptual because it offers a novel perspective on multiagent learning. Nevertheless, as a practical matter, undoubtedly there will be interest in applying more traditional techniques such as CCEAs and MARL to scaling in similar domains. Yet the lack of a principled approach to interpolating policies in such methods is a significant obstacle that will make direct comparison difficult, at least until an alternative such capability is introduced.

An important difference between the results in predator-prey and in room clearing is that while the heterogeneous policies improve on average in both domains with more agents, the homogeneous policies *degrade* on average as the team gets larger in room clearing. The technical reason for this

disparity is that the collisions in the small room mean that adding more agents that all do the same thing will eventually cause a pile-up. However, more generally, it shows that adding more agents is not always automatically helpful; thus the ability to intelligently interpolate intermediate policies along a cooperative spectrum will sometimes be critical to enabling scaling, highlighting the utility of policy geometry. Furthermore, even though the homogeneous team does improve with more agents in predator-prey, its performance is still significantly worse than every heterogeneous team size, which means that simply improving through scaling is not enough; the idea is to improve a policy that is already good, which is possible through heterogeneous policy geometry.

7. CONCLUSION

This paper presented a new, scalable method for training multiagent teams called multiagent HyperNEAT. By representing teams as patterns of policies, rather than as several distinct agents, the teams are able to dynamically change their size, up to several orders of magnitude, without any further training. Additionally, this approach to learning is able to overcome the *problem of reinvention* faced in traditional multiagent learning by exploiting team geometry and the continuum of heterogeneity. These novel capabilities were demonstrated in both predator-prey and room-clearing domains. More fundamentally, multiagent HyperNEAT offers a new perspective on multiagent learning that focuses on how agents on a team relate to one another and how those relationships can be exploited to foster cooperation and principled scaling.

8. ACKNOWLEDGMENTS

This research was supported by DARPA under grants HR0011-08-1-0020 and HR0011-09-1-0045 (Computer Science Study Group Phases I and II).

9. REFERENCES

[1] T. Aaltonen. *et al.* Measurement of the top quark mass with dilepton events selected using neuroevolution at CDF. *Physical Review Letters*, 2009.

[2] P. J. Bentley and S. Kumar. The 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.

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

[4] M. Bowling and M. Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136(2):215–250, 2002.

[5] B. D. Bryant and R. Miikkulainen. Neuroevolution for adaptive teams. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC 2003)*, volume 3, pages 2194–2201, Piscataway, NJ, 2003. IEEE.

[6] L. Busoniu, R. Babuška, and B. De Schutter. A comprehensive survey of multi-agent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 38(2):156–172, Mar. 2008.

[7] L. Busoniu, B. D. Schutter, and R. Babuska. Learning and coordination in dynamic multiagent systems. Technical Report 05-019, Delft University of Technology, 2005.

[8] C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the National Conference on Artificial Intelligence*, pages 746–752. John Wiley & Sons LTD, 1998.

[9] J. Clune, B. E. Beckmann, C. Ofria, and R. T. Pennock. Evolving coordinated quadruped gaits with the hyperneat generative encoding. In *Proc. of the Congress on Evolutionary Computation (CEC-2009) Spec. Sect. on Evolutionary Robotics*, Piscataway, NJ, USA, 2009. IEEE Press.

[10] J. Clune, R. T. Pennock, and C. Ofria. The sensitivity of hyperneat to different geometric representations of a problem. In *Proc. of the Genetic and Evolutionary Computation Conf. (GECCO-2009)*, New York, NY, USA, 2009. ACM Press.

[11] D. B. D'Ambrosio and K. O. Stanley. Generative encoding for multiagent learning. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2008)*, New York, NY, 2008. ACM Press.

[12] T. N. Dupuy. *The Evolution of Weapons and Warfare*. Da Capo, New York, NY, USA, 1990.

[13] S. Ficici and J. Pollack. A game-theoretic approach to the simple coevolutionary algorithm. *Lecture notes in computer science*, pages 467–476, 2000.

[14] J. Gauci and K. O. Stanley. A case study on the critical role of geometric regularity in machine learning. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI-2008)*, Menlo Park, CA, 2008. AAAI Press.

[15] J. Gauci and K. O. Stanley. Autonomous Evolution of Topographic Regularities in Artificial Neural Networks. *Neural Computation Journal*, 2010. To appear.

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

[17] F. Gruau, D. Whitley, and L. Pyeatt. A comparison between cellular encoding and direct encoding for genetic neural networks. In J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, editors, *Genetic Programming 1996*, pages 81–89, Cambridge, MA, 1996. MIT Press.

[18] T. Haynes and S. Sen. Co-adaptation in a team. *International Journal of Computational Intelligence and Organizations*, 1(4):1–20, 1996.

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

[20] J. Hu and M. P. Wellman. Multiagent reinforcement learning: theoretical framework and an algorithm. In *Proc. 15th International Conf. on Machine Learning*, pages 242–250. Morgan Kaufmann, San Francisco, CA, 1998.

[21] L. Panait and S. Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 3(11):383–434, November 2005.

[22] L. Panait, K. Tuyls, and S. Luke. Theoretical Advantages of Lenient Learners: An Evolutionary Game Theoretic Perspective. *The Journal of Machine Learning Research*, 9:423–457, 2008.

[23] L. Panait, R. Wiegand, and S. Luke. Improving coevolutionary search for optimal multiagent behaviors. *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 653–658, 2003.

[24] M. A. Potter, K. A. De Jong, and J. J. Grefenstette. A coevolutionary approach to learning sequential decision rules. In L. J. Eshelman, editor, *Proc. of the Sixth Intl. Conference on Genetic Algs.* . San Francisco: Kaufmann, 1995.

[25] B. Price and C. Boutilier. Implicit imitation in multiagent reinforcement learning. In *Machine Learning*, pages 325–334. Morgan Kaufmann Publishers, INC., 1999.

[26] J. Secretan, N. Beato, D. B. D'Ambrosio, A. Rodriguez, A. Campbell, and K. O. Stanley. Picbreeder: Evolving pictures collaboratively online. In *CHI '08: Proc. of the twenty-sixth annual SIGCHI conf. on Human factors in computing systems*, pages 1759–1768, New York, NY, USA, 2008. ACM.

[27] 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.

[28] 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.

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

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

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

[32] M. Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. *Readings in agents*, pages 487–494, 1997.