Evolving Artificial Neural Networks with Communicating Heterogeneous Agents

Independent Work 2013

J.T. Glaze

Advisor: Robert Stengel

Princeton University

May 7, 2013

Abstract

This document examines the applicability of Neuroevolution to the field of multi-agent robotics. Specifically, it seeks to discover if the performance of robotic teams may be improved by combining the HyperNEAT qualities of direct communication and heterogeneity between agents. This is explored through several experiments which require sophisticated coordination and specialization among agents in a robotic team.

1. Introduction

The rapidly increasing power of computational hardware has created a boom in robotics research over the last two decades, leading to a proportional increase in the demand for more and more sophisticated software algorithms to control these robotic agents. Early robotics research featured a large focus on hardware development, with robots' motions often controlled directly by human input. However, as robotics research matures, a need has developed for software that can allow a robotic agent to analyze sensor inputs, develop a plan of action, and carry out that plan in the environment – all with little or no input from a human controller.

Additionally, the need for autonomous control schemes for multi-agent teams of robotic agents has become very important. Replacing a single agent with teams of coordinating robotic agents can allow for greater efficiency in many different applications, especially search tasks[1]. Multi-agent teams also are more robust than a single robot, and can adapt to the failure of several robots in the team. These teams often feature

[1] Russell, S. and Norvig, P. (2010) *Artificial Intelligence*, *3rd Edition* (New Jersey: Prentice Hall), p. 425

decentralized control, where each agent makes decisions based on its own set of rules; no one agent is in charge, yet the teams are often able to exhibit complex coordination behaviors[2].

One particularly useful scheme for developing autonomous control in robotic agents is an Artificial Neural Network (ANN). In this system, the robot's sensors send information as inputs into a specialized neural network, which then outputs control signals to the robot's motors. This allows the robot to make navigation decisions without storing large sets of probabilistic data or requiring extensive computation time, as is necessary for the Particle Filtering[3] and Markov Localization[4] methods. This is particularly useful in a multi-agent team, which emphasizes simple hardware and on-the-fly decision making.

The ANNs may be optimized through the process of neuroevolution, which uses genetic algorithms to "[search] through the space of behaviors for a network that performs well at a given task."[5] A population of ANNs are created, and each one is tested in a simulated environment – the ones that exhibit the desired behaviors are further examined and improved upon, until the optimal network has been discovered. A team of researchers at the University of Central Florida, led by Kenneth O. Stanley, have spent the last ten years developing algorithms for neuroevolution. Notably, they have devised two key methods for optimizing coordinated behaviors among multi-agent robotic teams. The first allows the neuroevolution of heterogeneous agents – each with their own unique neural networks,

---

[2] Russell and Norvig, p. 426
[3] Russell and Norvig, p. 597
[4] Clark, C. (2011) "Lecture 14 – Markov Localization" [PowerPoint slides]. Retrieved from http://www.cs.princeton.edu/courses/archive/fall11/cos495/COS495-Lecture14-MarkovLocalization.pdf
[5] Stanley, K. and Miikkulainen, R. (2002) "Evolving Neural Networks through Augmenting Topologies". *Evolutionary Computation* **10** (2), p. 99

allowing them to have unique responses to the same inputs[6]. The second provides a direct neural link between the networks of multiple agents, allowing them to share information with one another[7].

This paper seeks to determine if the combination of these two techniques – heterogeneous agents and inter-network connections – can result in a higher level of performance for multi-agent teams developed through neuroevolution. By combining the benefits of specialization and communication among the agents, it is believed that neuroevolution can produce teams that are able to reach a higher level of performance than either trait can achieve by itself. This will be tested through a series of simulations, each of which will require both individuality and cooperation from the robotic multi-agent teams.

2. Previous Work

a. Neuroevolution

The key to utilizing artificial neural networks in robotic agents is to properly train the weights and connections inside the networks in order to produce the desired behavior. Neuroevolution is the process of training these ANNs through the simulated Darwinian principles of natural selection and genetic evolution. It searches for ANNs that provide optimal behavior in the robots without any training from human input. Therefore, it is a process that is able to develop creative and original solutions to complex problems, even generating solutions that were unanticipated by the experimenter.

---

[6] D'Ambrosio, D., Stanley, K., et al. (2010) "Evolving Policy Geometry for Scalable Multiagent Learning" *GECCO-2010* (New York: ACM)

[7] D'Ambrosio, D., Stanley, K., et al. (2012) "Multirobot Behavior Synchronization through Direct Neural Network Communication" *ICIRA-2012I (*New York, NY: Springer-Verlag)

Neuroevolution was developed as an alternative to the traditional means of training neural networks, which occurs through supervised learning. Because it would be very difficult to generate input-output training data pairs for each desired type of robotic behavior, supervised learning is problematic in a robotic neural network. However, the ANNs can be developed through a different algorithm: genetic optimization. Although it is hard to predict what input-output data pairs will produce the network's desired behavior, it is simple to gauge the performance of the robot in a test environment. Therefore, a population of ANNs can be created, each one represented by a 'genome': a unique encoding of the network's structure and weights, with a similar function as DNA in humans. Each ANN is placed into a simulated environment and scored according to its fitness at accomplishing the desired task[8]. Just like an animal in the wild, the ANN's fitness determines its likelihood to reproduce, according to natural selection.

The genome of each selected ANN undergoes genetic reproduction: it is randomly paired with another reproducing ANN's genome, crossover occurs (each genome is split into two sections, one of which is exchanged with the other genome), and there is a small chance of mutation. Following the principles of Darwinism, over many generations the ANNs evolve to find a network that provides the optimal desired behavior. Therefore, with enough time, neuroevolution moves through the full search space of network behaviors to generate a robot with the highest possible fitness.

[8] Stengel, R. (2011) "Lecture 13. Monte Carlo Evaluation and Evolutionary Search" [PowerPoint Slides]. Retrieved from https://blackboard.princeton.edu/bbcswebdav/xid-338511_1

b. Neuroevolution of Augmented Topologies (NEAT)

An algorithm known as NEAT (Neuroevolution of Augmented Topologies) is able to provide significant performance improvements over other neuroevolution methods. This is accomplished through a focus on three key concepts. First, NEAT begins with a minimal starting neural network structure – all inputs are directly connected to all outputs, with zero hidden nodes. Second, new connections and hidden nodes can be created incrementally through mutation, allowing the structure to become more complex over generations. Natural selection will eliminate structure additions that result in poor performance, reducing the necessary search space of the algorithm[9]. And third, NEAT groups the ANN genomes into distinct species, according to their relative similarity to each other. This relative similarity is accomplished by comparing all genomes with each other, and counting the number of matching and non-matching genes (where a gene represents a network node or connection)[10]. Through this process of speciation, "topological innovations are protected in a new niche where they have time to optimize their structure through competition within the nice"[11] instead of competing with the entire population.

Focusing on a minimal starting structure, incremental growth, and speciation, NEAT provides a structure for neuroevolution that decreases the total search space while protecting innovation. This gives it a substantial performance gain over methods that lack these features[12], making it ideal as the fundamental neuroevolution algorithm of this research.

---

[9] Stanley, K. and Miikkulainen, R. (2002) p. 111
[10] Stanley, K. and Miikkulainen, R. (2002). p. 110
[11] Ibid.
[12] Stanley, K. and Miikkulainen, R. (2002). p. 121

c. Hypercube-based NEAT (HyperNEAT)

However, NEAT's algorithm may be further improved by evolving an *indirect encoding* of its neural network. In other words, the NEAT algorithm evolves and optimizes a network that is used to generate the robot's ANN, instead of evolving the ANN itself. This allows the algorithm to exploit the geometry and spatial relationships of nodes and connections in the ANN to discover larger and more complex solutions than would be possible through traditional NEAT[13]. This process is known as the Hypercube-based Neuroevolution of Augmented Topologies (HyperNEAT).

In HyperNEAT, the ANN is represented as a set of nodes and connections in an n-dimensional Cartesian space. The indirect encoding that is evolved by HyperNEAT is known as a Compositional Pattern Producing Network (CPPN). The CPPN takes as its inputs the coordinates of two nodes on the ANN, and outputs the value of the weight between them. This means that the position of a node in the neural network determines the weights of its connections. Additionally, neural networks evolved through HyperNEAT are scalable, with one test showing "an evolved ANN [scaled] *without further evolution* to a size of over *eight million* connections without loss of functionality"[14].

d. Heterogeneous Multi-Agent Teams

There are significant obstacles to be overcome in order to develop multi-agent teams using neuroevolution. First, if the algorithm produces the same ANN for each agent on the team, then every robot will perform in exactly the same way. This is not always

---

[13] Stanley, K., D'Ambrosio, D., and Gauci, J. (2009) "A Hypercube-Based Encoding for Evolving Large-Scale Neural Networks" *Artificial Life* (Cambridge, MA: MIT Press), p. 188
[14] Stanley, K., D'Ambrosio, D., and Gauci, J. (2009) p. 189

ideal, since specialization is important in many multi-agent tasks – for example, one robot may need to turn left upon entering a room, whereas another robot should turn to the right. However, if each robot's ANN is evolved separately, then the complexity of the simulation and reproduction computations increases exponentially with size[15]. However, by utilizing the HyperNEAT's ability to recognize spatial patterns, a single CPPN may be used to generate an unbounded number of unique ANN agents.

This is accomplished by altering the inputs of the CPPN so that, for each robot, in addition to the n-dimensional locations of a connection's starting and ending nodes, it also receives the robot's starting position relative to the rest of the team. In other words, each robot is initialized with a unique ANN determined by its starting position on the team. This is similar to the way that soccer players have different behaviors on the field based on their starting positions: those closer to the home goal play defense, those closer to the opposing goal play offense, and those at center field play a mixture of the two. Their actual position on the field may change from moment to moment, but their basic strategy remains constant[16].

Due to its basis in HyperNEAT, this method is also highly scalable without further evolution, allowing unique additional agents to be created by providing their relative positions to the CPPN.

[15] D'Ambrosio, D., Stanley, K., et al. (2010) p. 1
[16] D'Ambrosio, D., Stanley, K., et al. (2010) p. 4

e. Direct communication between agents

A second multi-agent neuroevolution experiment was performed by Stanley et al. to improve another key aspect of robotic teams: cooperation. In this experiment, the ANNs were expanded to include transmission and receiver nodes, which were directly connected with each robot's left and right neighbors (based on their starting position). The CPPN encodes the weights of these inter-agent connections by adding a third CPPN input *z* (similar to the heterogeneous multi-agent experiment). If the connection being generated connects the robot to its immediate left neighbor, *z* was set to -1; if it connects the robot to its immediate right neighbor, *z* was set to 1; and if it connects two nodes within the same ANN, *z* was set to 0. That way, the CPPN can evolve the optimal connection weights for the left and right communication links. This development allowed a team of initially disorganized robotic agents to synchronize their movements over time. However, each agent still receives the same ANN from the CPPN, so there is no heterogeneity between the agents, and no specialization can occur.

3. Approach

By combining the specialization of heterogeneous agents with the coordination achieved through direct communication, it is hypothesized that neuroevolution can produce multi-teams of robots that outperform agents that lack either of these qualities. This experiment began by adapting the open source C# code implementation of multi-agent HyperNEAT created by D'Ambrosio, Lehman, and Risi[17]. This project had a pre-existing implementation of heterogeneous multi-agent neuroevolution, but it did not have

[17] Retrieved from http://eplex.cs.ucf.edu/software/AgentSimulator_v1_0.zip

any means of direct communication between agents. Therefore, the code was expanded to generate agents with both of these qualities.

a. Artificial Neural Networks

The fundamental control system for the agents in these experiments are Artificial Neural Networks (ANNs), which receive input data from the robot's sensors, pass it through a hidden layer(s) of nodes, and then outputs control values to the robot's motors. Each node calculates its value by accumulating the values of its inputs, and then passing that sum through an activation function. A list of possible activation functions can be seen in Table 1. The resulting value is then output to all further connected nodes.

Activation Function, $f(x)$	**Output**
`Linear`	x
`Bipolar Sigmoid`	$\frac{2}{1+e^{-4.9x}} - 1$
`Sine`	$\sin 2x$
`Gaussian`	$2e^{-(2.5x)^2} - 1$

Table 1. Activation functions utilized in ANN nodes[18]

The calculation of each node value proceeds in timesteps, so that at each interval the value of a node's input is equal to their calculated output from the previous step. This means that information flows through the network at a uniform and discretized pace. Pseudocode for the function of a node can be seen below.

[18] D'Ambrosio, Lehman, and Risi (2012)

```
function nodeOutput()
      inputs: n, the node being calculated
              connectWeights, the weights of all connections into n
              parents, the values of all nodes connected into n
              bias, some bias factor

      sum = 0;
      for i=0 to parents.length do
            sum += parents[i] * connectWeights[i];
      return (n.ActivationFunction(sum) + bias);
```

Pseudocode 1. Calculating node output values. This is called for each node in the network at each discrete timestep.

Using the HyperNEAT Neuroevolution methodology, each robotic agent is controlled by an ANN with a user-defined structure, known as the substrate. In the substrate, each node is given an x- and y-coordinate on a Cartesian plane, and the weights of the connections between nodes are determined by a CPPN. In this implementation, every node on the substrate has a linear activation function.

In addition to the standard input, hidden, and output nodes, two more types of nodes were added to the substrate ANNs: transmitter and receiver nodes. The purpose of these nodes is to communicate data from the output of one agent on the robotic team to the inputs of multiple other agents. This allows the entire team to effectively be connected in one meta-network, facilitating the transfer of information across the 'hive-mind'.

b. CPPNs and Connection Weights

The Compositional Pattern Producing Networks (CPPNs) – the networks that serve as indirect encodings for generating the weights on the ANN substrates – were also

expanded to facilitate transmitter and receiver nodes. The standard HyperNEAT implementation provides a heterogeneous CPPN with six inputs: a bias input, the *x* and *y* positions of the two values that are being connected, and the agent's relative (one-dimensional) starting position. A homogenous CPPN only has five inputs, since there is no need for an agent's relative starting position – they all are given the same ANN, regardless of their positions. The CPPN is optimized through genetic algorithms to produce ANNs that best accomplish the desired tasks. Therefore, a CPPN has two outputs: the weight of the connection between the two substrate nodes in question, as well as a bias factor for the connection. If the weight value output by the CPPN is below a threshold of 0.2, then the connection is disabled in the ANN. A CPPN can be conceptualized as the following six-dimensional function[19]:

$$CPPN(bias_{in}, x_1, y_1, x_2, y_2, z) = (w, bias_{out}) \quad (1)$$

As the CPPN sets the connection weights and bias among the ANN substrates of the robotic agents, the inter-agent connection weights must also be determined. Each agent's ANN is initialized with α transmitter nodes and β receiver nodes. Since agents have a finite number of receiver nodes, there is a limit to the number of robots that may communicate with each other. Specifically, for each agent,

$$Number\ of\ other\ agents\ connected = \frac{\beta}{\alpha} \quad (2)$$

There is no limit to the number of signals that may be transmitted by each robot, provided that there are a proportional number of receiver nodes to accept this information. Each

[19] D'Ambrosio, D., Stanley, K., et al. (2010) p. 8

agent is connected to its β/α closest teammates at the start of the simulation. Since the agent's relative starting position is also passed into the CPPN, this allows for the strongest amount of communication to occur between agents that are both physically and behaviorally close to each other.

c. Genomes and Speciation

In order to undergo genetic reproduction, each CPPN is represented by a genome, where each gene stores the structural details of either a node or a connection in the CPPN. A node gene contains the node number, its type (Input, Hidden, or Output), and its activation function. A connection gene represents a link between two nodes, and contains the source node ID, the target node ID, the connection weight, and an enable bit (to turn the connection on or off. Additionally, every gene has a unique innovation number, which allows the NEAT algorithm to track the genealogy and similarity between multiple genomes.

By examining the number of shared genes between multiple CPPNs, NEAT can divide the genomes into distinct species. A compatibility distance δ may be calculated using the number of genes, *D*, that are not present in both of the genomes, as well as the average difference in weight, *W*, between matching genes[20]:

$$\delta = \frac{D}{N} + W \tag{3}$$

N represents the number of genes in the larger genome. If the δ between two genomes is above a certain threshold, they are considered to be in separate species. If a genome *g* "is

[20] Stanley, K. and Miikkulainen, R. (2002) p. 110

not compatible with any existing species, a new species is created with g as its representative."[21] In this implementation, the program automatically shifts the threshold for δ to maintain between 6-10 unique species.

d. Fitness and Reproduction

When determining which genomes will reproduce in the next generation, the entire species fitness is considered. Each genome is tested in a simulated environment, where its performance is scored according to a fitness function. The fitness function can be altered for each test to generate the desired behavior in the agents. However, genomes within the same species must share their fitness with the entire group, so that the number of offspring given to each species in a given generation is dependent upon the average fitness of every genome in the species. This can be seen in the following equation[22], which determines the fitness for a species s with n genome members:

$$f_s = \frac{\sum_{i \in s} f_i}{n} \tag{4}$$

After a species has been allotted slots for offspring, the top individually performing genomes are chosen to reproduce. Reproduction occurs in these experiments according to the parameters[23] listed in Table 2.

[21] Ibid.
[22] Ibid.
[23] D'Ambrosio, Lehman, and Risi (2012)

Reproduction Parameter	Probability
Asexual Reproduction	50.0%
Sexual Reproduction	50.0%
Interspecies Mating	1.0%
Mutate Connection Weights	98.8%
Add new node	0.2%
Add new connection	2.0%
Delete connection	0.1%

Table 2. Probability of mutations and genetic crossovers during CPPN reproduction.

If the reproduction is sexual, then genetic crossover occurs with another genome: the genetic material of the two genomes are split and exchanged between them. If the reproduction is asexual, then no genetic crossover occurs. If a new node is added, it is randomly assigned an activation function (linear, bipolar sigmoid, Gaussian, or sine) with equal probability. If a new connection is added, it is randomly assigned a source and target node. After each species has produced its offspring, the fitness of the new generation is tested out in simulation, and the cycle continues.

e. Simulation Testing Environment

The project created by D'Ambrosio et al. generates an experiment by reading a number of customizable .csv files, each of which contain information about the placement of objects and robots in the environment, experimental parameters, and the positions of each of the nodes on the robots' ANNs. The program then runs for a default 1000 generations, storing each new best genome as it is found and logging the highest fitness score for each generation.

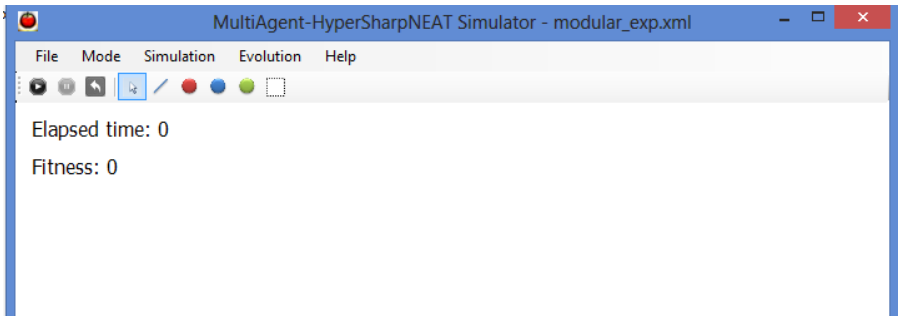


Figure 1. Experiment Simulator Interface

There is also a provided GUI, seen in Figure 1, which allows specific genomes to be loaded and visualized in the simulated environment. A collection of class interfaces also allow for the creation of any custom robots, sensors, fitness functions, or other experiment objects.

Three experiments were created to test the specialization and communication of the multi-robot teams: a room-clearing experiment, a predator/prey experiment, and a modular-movement experiment. For each of these experiments, the communicating, heterogeneous (CHet) teams were tested against both heterogeneous, non-communicating teams and homogenous, communicating teams. In this way, it can be experimentally shown for these representative test cases whether or not the combination of communication and heterogeneity is greater than either of these qualities by themselves.

4. Room Clearing Experiment

a. Experiment Parameters

The first experiment was created from an existing project designed to demonstrate the performance of heterogeneous multi-agent teams[24]. The experiment begins with seven robots lined up vertically, facing upwards towards the entrance to a square room. The

[24] D'Ambrosio, D., Stanley, K., et al. (2010) p. 5

agents are simulated as circular robots, each with eleven point sensors that detect the distance to the nearest wall, as well as two parallel wheels connected to the ANN outputs. The sensors are evenly distributed across the front hemisphere of each robot, and have a maximum range of 100.0 units. Two of the output nodes control the heading of the robot, and the third output node controls the robot's velocity.

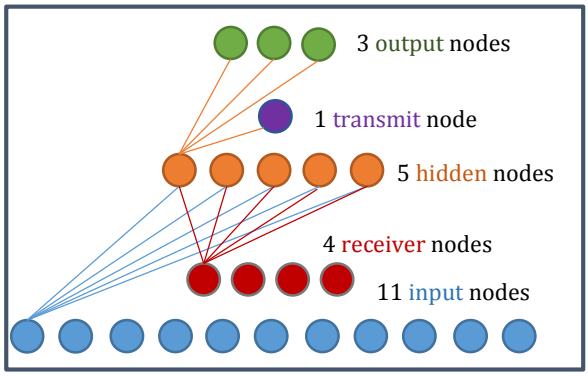


Figure 2. Room Clearing Agent neural network substrate. The network is fully connected between layers, connections are simplified here for clarity. Each node has a linear activation function. This network represents direct communicating robots – non-communicating robots lack transmit and receiver nodes.

The ANN also has five hidden nodes, which are all fully connected to the inputs and outputs. The weights of these connections are calculated by the team's CPPN. Agents with direct communication have one transmit node that they can use to send information to the four closest agents at the start of the simulation. Reciprocally, each directly communicating robot has four receiver nodes which can take a single transmit link from the four closest starting robots.

The test room is divided up into a discreet grid, and the fitness of a robot team is determined by the total number of grid squares within the field of view of each robot. The fitness is calculated at every timestep, and the total fitness is the sum of all of these. This

fitness can be calculated for the set of robots *R* for *T* timesteps, where v_{it}represents the total number of grid sections within the field of view of robot *i* at timestep *t*.

$$fitness_{tot} = \sum_{t=0}^{T} \sum_{i \in R} v_{it} \tag{5}$$

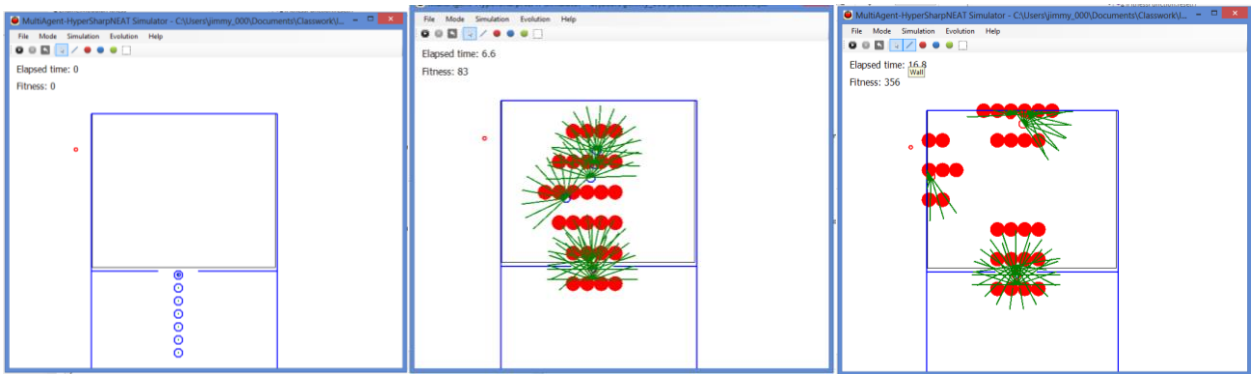
The evaluation lasts for a simulated total of 35 seconds, and is repeated for a population of 150 genomes over 1000 generations.

b. Results and Analysis

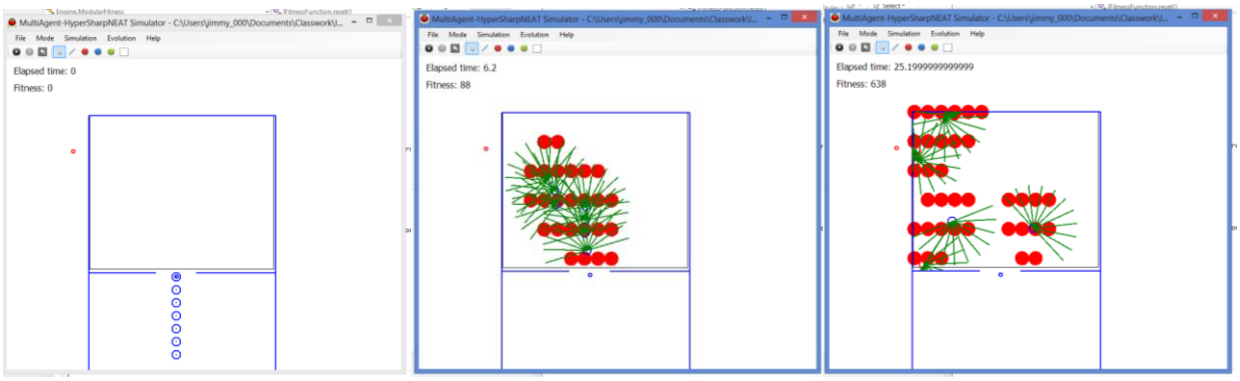
The first generation created by the Communicating Heterogeneous system had a maximum fitness of 1111.0. In the simulation, the team entered the room, with one agent breaking off to the left side of the room. Very quickly, however, the agents collided with the walls and with each other, ending up trapped against the top wall, the side wall, and the entrance to the room. Snapshots from the simulation of this team can be seen in Figure 3.

After 44 generations, the maximum fitness had been raised to 1389.0. By this point in the evolution, most of the agents in the simulation turn off to the left upon entering a room, resulting in good coverage of that half of the room. However, the right side of the room remains mostly unobserved, except for a single agent who moved towards the right after entering the room last.

0 Generations:



44 Generations:



928 Generations:

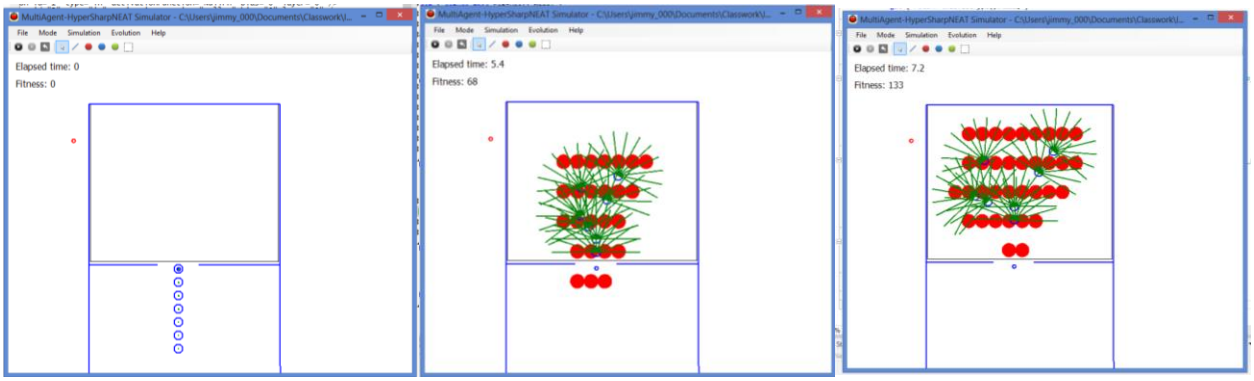


Figure 3. Simulation of best CHet team after 0, 44, and 928 Room Clearing generations. The blue lines represent walls. The blue circles represent the seven robotic agents. The green lines represent the rangefinder sensors for each robot. The red circles represent the grid sections within the robots' field of view.

Finally, after 928 generations, the CH system discovered the CPPN that generated the maximum fitness in the simulation. Upon entering the room, the agents fan out in both directions, moving to the left and right in alternating order. The agents that entered later

turn a little sharper to cover the lower section of the room, and the final agent to enter continues straight up to cover the top wall. This team had a fitness score of 1500.0.

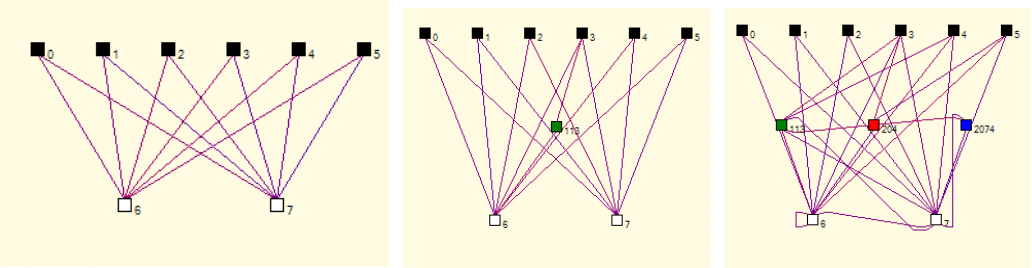
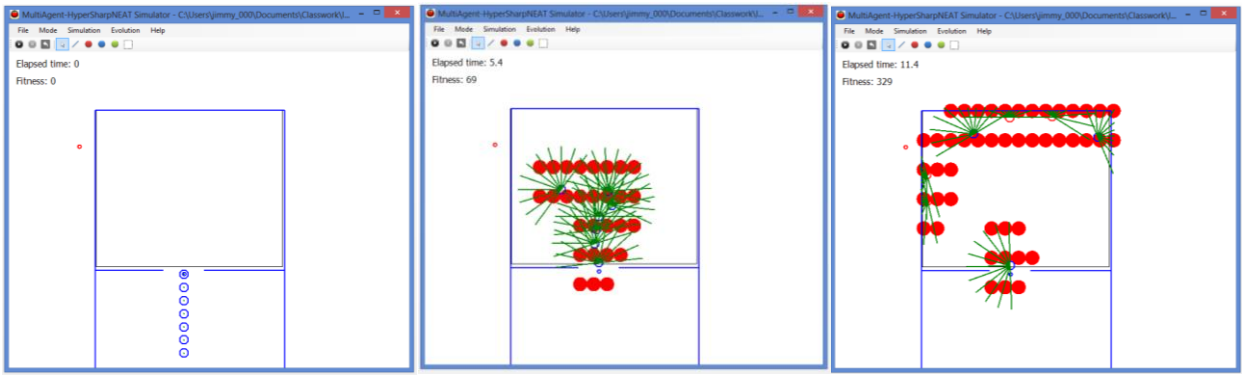


Figure 4. Best fitness Room Clearing CPPNs for CHet at Generations 0, 44, and 928, respectively. The black nodes (0-5) represent the CPPN inputs, which are fully connected to the weight output (6) and the bias output (7). The green node represents a Gaussian hidden node. The red node represents a Sine hidden node. The blue node represents a Linear hidden node.

The CHet system outperformed both the Non-Communicating Heterogeneous and Communicating Homogenous methods. The best team for Non-Communicating Heterogeneous team had a maximum fitness of 1427.0, achieved after 853 generations. This tem entered the room in a similar manner to the best CHet system, alternating turns to the left and right. However, these agents were unable to avoid hitting the walls of the environment, which significantly decreased their coverage of the room.

The Communicating Homogenous method achieved a team with a maximum fitness of 1008.0 after 295 generations. Each of the agents moved into the room in an identical fashion, curving to the left after passing through the entryway. However, this caused all of the agents to collide with each other after the first agent ran into the left wall, effectively immobilizing the entire team.

Non-Communicating Heterogeneous Best (853 Generations)



Communicating Homogeneous Best (295 Generations)

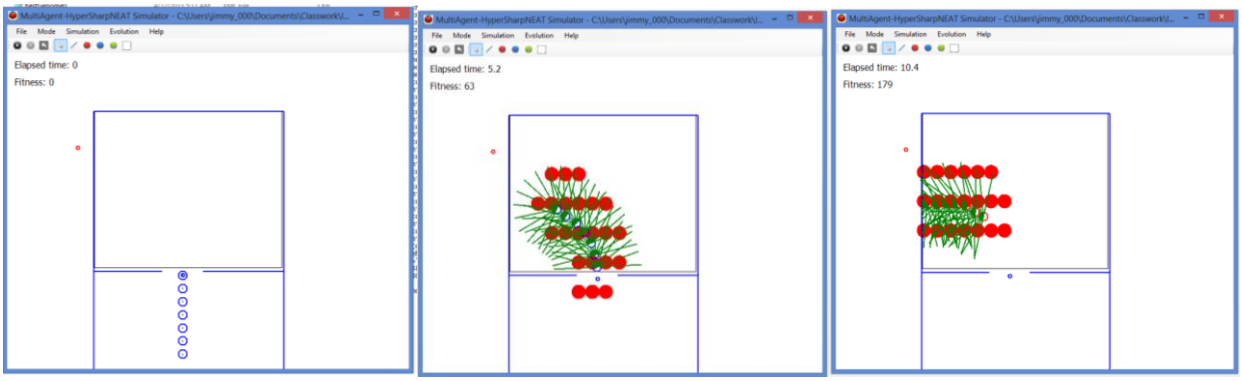
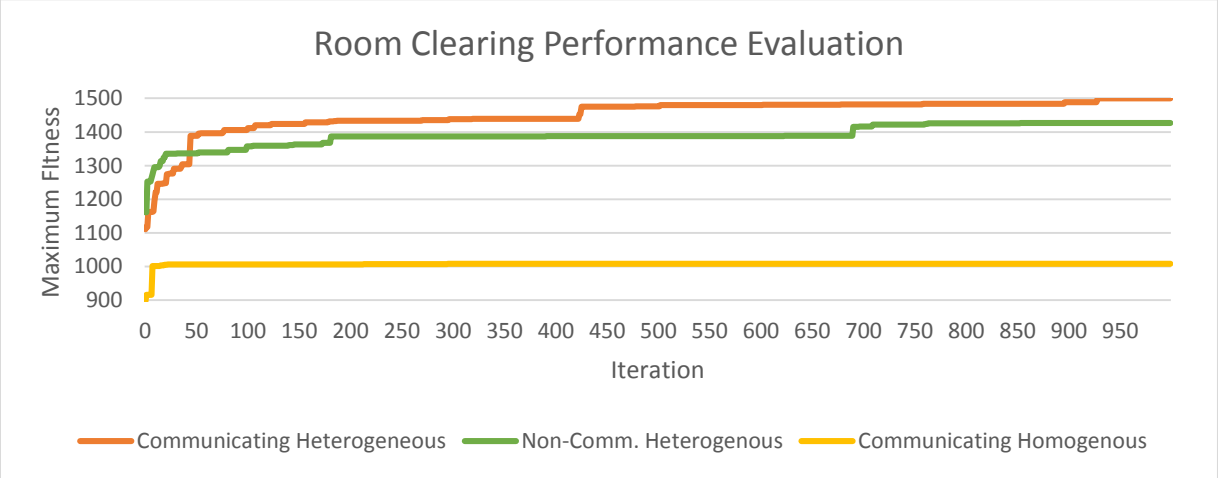


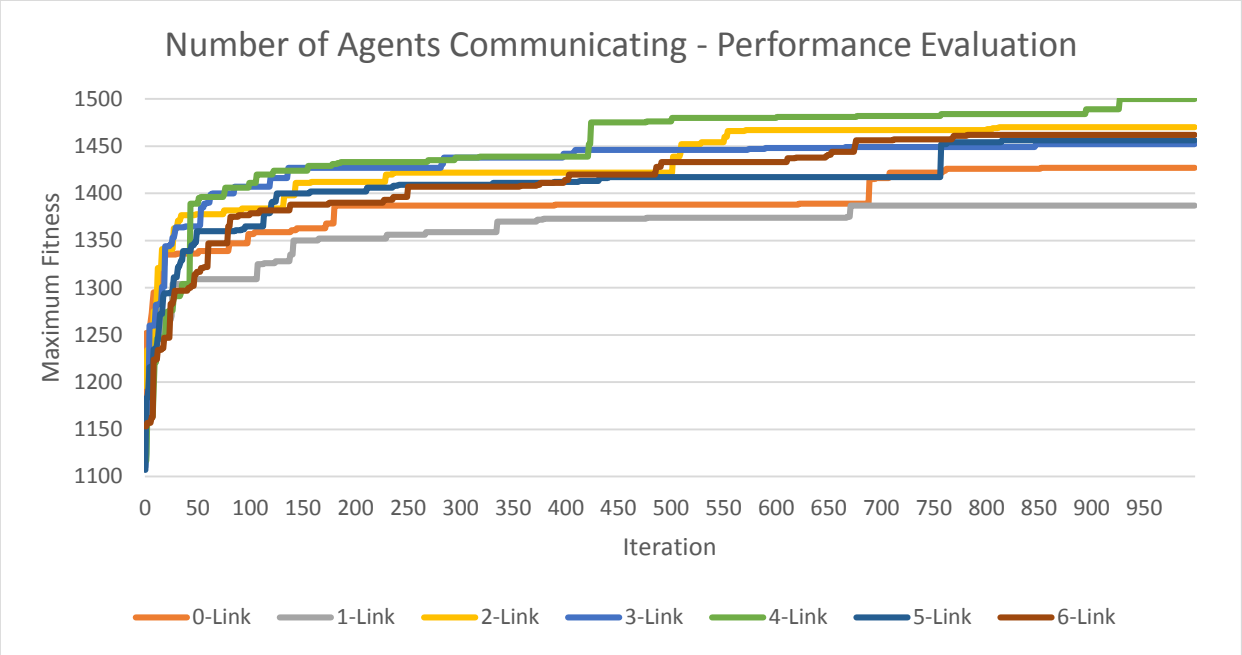
Figure 5. Simulation of best Heterogeneous and Homogenous teams after 853 and 295 Room Clearing generations, respectively. The blue lines represent walls. The blue circles represent the seven robotic agents. The green lines represent the rangefinder sensors for each robot. The red circles represent the grid sections within the robots' field of view.



Graph 1. Room Clearing Performance over Time. Each team consisted of 7 robots, trained over 1000 generations. The fitness represents the team's performance according the fitness fxn described in Equation 5.

The poor performance of communicating-only agents is understandable, as they lack any kind of individuality, and this experiment was designed to demonstrate the necessity of specialization between agents – as agents enter the room, they need to move in different directions to cover more of the space. This was seen in the heterogeneous agents, which performed well in both the communicating and non-communicating trials. However, direct communication did allow the CHet team to coordinate more to avoid hitting the walls while also avoiding collisions among themselves. This gave CHet the slight advantage necessary to outperform the non-communicating method.

Another series of experiments sought to discover the ideal ratio for connecting agents to one another. Given the seven robots on the team, trials were run connecting them to all, some, or none of the other agents, and examining the performance of the generated teams. It was found that connecting a robot to four out of its six teammates provided the highest amount of fitness. Interestingly, the trials with second highest fitness connected an agent to only two other agents, and the third highest fitness connected all six other agents. This suggests that communicating teams perform best when connected to an even number of neighboring agents. This is possibly related to the fact that these agents began the simulation spaced evenly apart, so that each robot has pairs of agents the same distances away, in opposite directions. Additionally, the CHet method performed at its worst when it was only connected to one other agent on the team – producing a maximum fitness even lower than not communicating with any other agents at all.



Graph 2. Number of Agents Communicating in the Room Clearing Experiment - Performance over Time. Each team consisted of 7 robots, with a varying number of connections between them. The fitness represents the team's performance according the fitness function described in Equation 5. The teams were each trained over 1000 generations.

5. Predator / Prey Experiment

a. Experiment Parameters

The purpose of this experiment is to test a team's ability to cooperate in order to mimic the pack cooperation of predators hunting their prey. This experiment takes place on an unbounded space with no walls. Nine robots are lined up vertically, facing towards the right, where eight simulated prey objects are arranged in a diamond formation.

The prey is a non-neural object designed to stay still until a predator comes within a range of 50 units. At that point, it moves at a constant velocity away from the predator. If there are multiple predators within its range, it moves in the direction away from the closest predator. The prey moves at a speed of 50 units/second, whereas the predators have a maximum speed of 40 units/second, therefore a single predator can never catch the

prey on its own. Instead, predators must work with each other to corral their prey. If a predator comes into contact with a prey object, it has been caught, and it stops moving. For simplicity, a prey object does not have any collision control among other prey – they can share the same physical space.

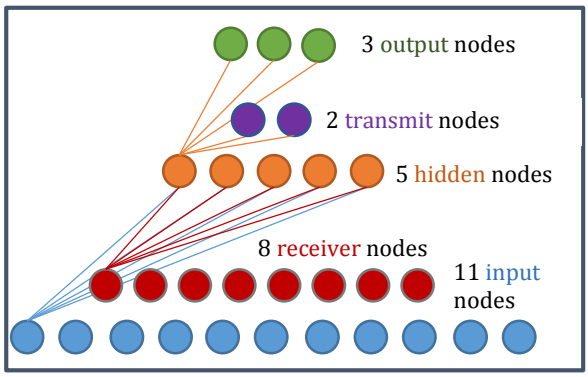


Figure 6. Predator Agent neural network substrate. The network is fully connected between layers, connections are simplified here for clarity. Each node has a linear activation function. This network represents direct communicating robots – non-communicating robots lack transmit and receiver nodes.

The predator robot has 11 input sensors on its front hemisphere, 5 hidden ANN nodes, and 2 output wheels. As in the room clearing experiment, two output nodes determine the robot's heading, and a third controls its velocity. The sensors on these robots detect the distance to the nearest prey object, and operate on a single line of sight, instead of a wide field of view. Therefore, the prey must be directly in front of the sensor and within its range (250 units) in order for its presence to be registered. The sensors are shown in the visualizer as green lines, and the prey objects as red circles. Agents with direct communication have two transmit nodes that they can use to send information to the

nearest four agents. This means that they have eight receiver nodes to collect information from the other agents around them.

The fitness function for this experiment is based upon the number of prey caught by the team, and the speed at which they were caught, where the final fitness is equal to

$$fitness = (100 * p) + 2 * (100 - t), \quad (6)$$

where p is equal to the total number of prey caught, and t is equal to the time it took to catch all of the prey[25]. If not all of the prey are caught, then t is equal to 100 (the runtime, in seconds, of the simulation). The simulation was repeated for 150 genomes over 1000 generations.

b. Results and Analysis

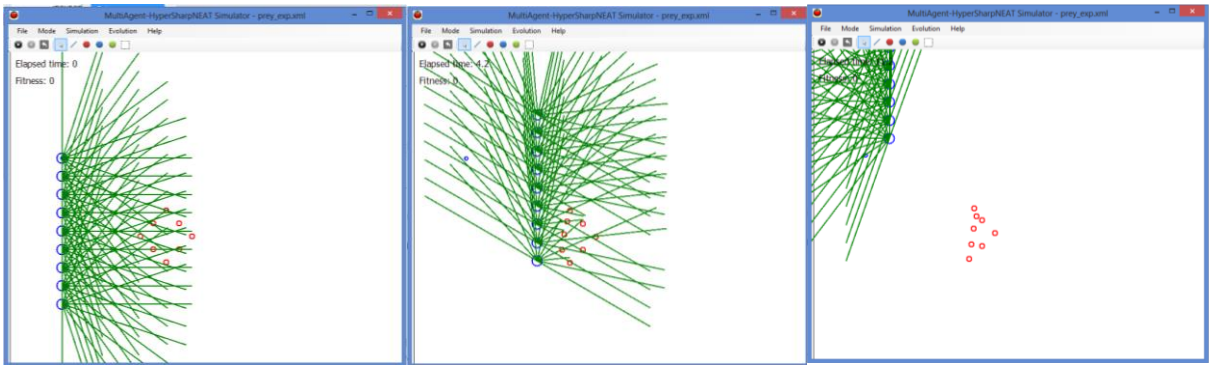
The first generation created by the CHet method failed to capture any prey, giving it a fitness score of zero. These agents each moved in a counter-clockwise circle, disregarding the prey completely. The prey were able to move out of their way, and the agents continued circling, never colliding with each other or with the prey. Snapshots of this simulation can be seen in Figure 7.

However, after 23 generations, a slight modification made the circling behavior successful in catching seven out of the eight prey, giving it a fitness score of 700. While completing the first counterclockwise loop, the agents move in slightly different arcs, which results in slight collisions among themselves so that when they finish the loop they are no longer in their original vertical formation. Instead, they are able to loop upwards to

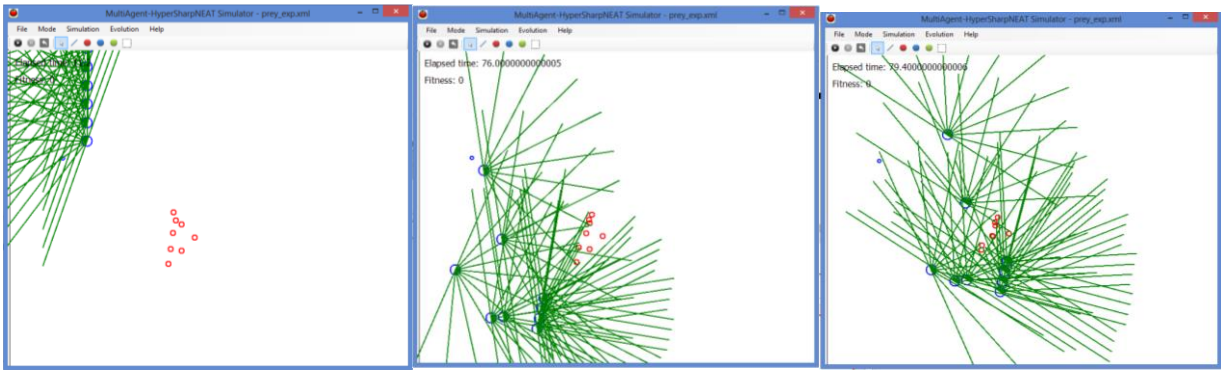
[25] D'Ambrosio, D., Stanley, K., et al. (2010) p. 5

approach the prey from multiple positions, allowing them to ensnare all but one of the prey.

0 Generations:



23 Generations:



905 Generations:

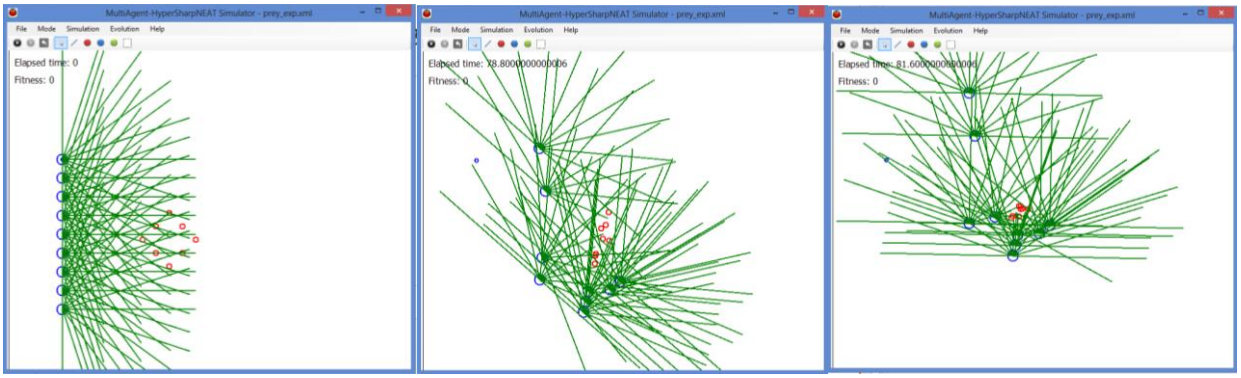


Figure 7. Simulation of best CHet team after 0, 23, and 905 Predator generations. The blue circles represent the seven robotic predator agents. The green lines represent the rangefinder prey sensors for each robot. The red circles represent the prey objects.

After 905 CHet generations, a genome was produced that had the maximum fitness in capturing the prey. The predators loop counterclockwise away from the prey as before, but when they swing back around to attack, they have positioned themselves on either side of the prey. As they corral the prey into a tight clump, agents close in from all three sides to capture the entire group. This gives the predators a fitness score of 981.2. As seen in Figure 8, by the 905th generation, the CPPN had evolved to a very high degree of complexity, utilizing sine, linear, and bipolar sigmoid hidden nodes, along with self-referential connections.

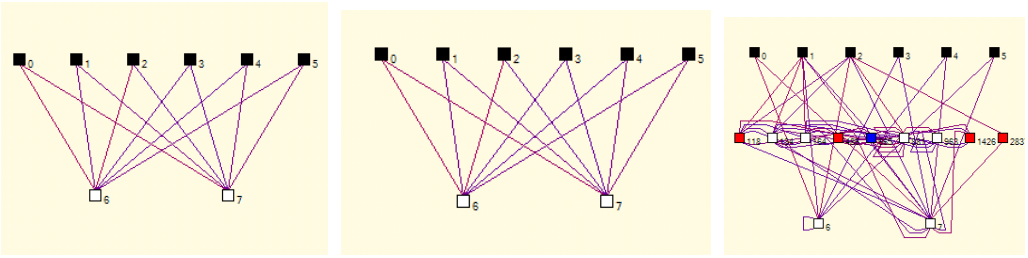
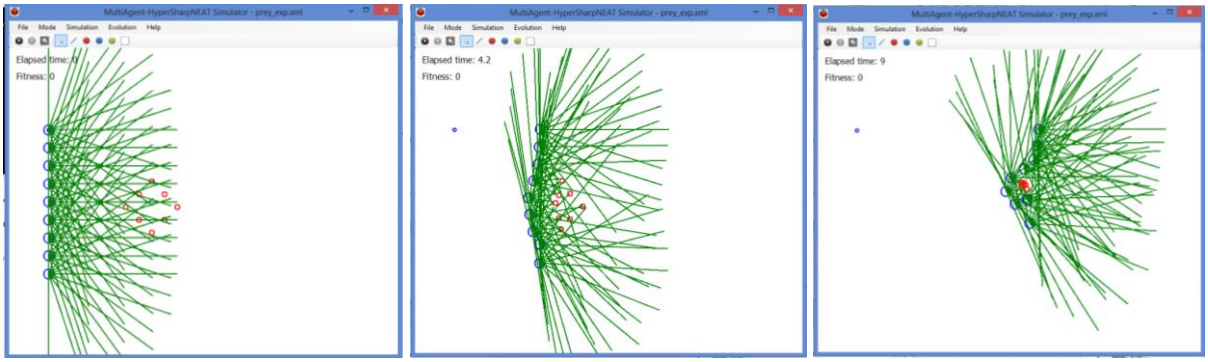


Figure 8. Best fitness Predator CPPNs for CHet at Generations 0, 23, and 905, respectively. The black nodes (0-5) represent the CPPN inputs, which are fully connected to the weight output (6) and the bias output (7). The white hidden nodes represent Bipolar Sigmoid hidden nodes. The red nodes represent Sine hidden nodes. The blue nodes represent Linear hidden nodes.

In this experiment, the Non-Communicating Heterogeneous method managed to generate a team that very slightly outperformed the CHet agents. Instead of utilizing the CHet team's looping strategy, this team moved directly towards the prey, pushing them to the left. As the team moved in a vertical line, the ends of the formation slowly began to wrap towards the center. The line of agents enfolded the prey, approaching them from the left, above, and below. Because this method captures all of the prey slightly faster than the CHet looping method, it resulted in a higher fitness score of 984.4.

The Communicating Homogenous team attempted a similar strategy, but the agents were unable to move in the exact same pattern. Instead, the top agents were more spread out, which allowed a single prey to slip through the net and escape capture. The team ignored this lone remaining prey, and continued moving to the left. Because this team only captured seven of the prey, it received a fitness score of 700.0.

Non-Communicating Heterogeneous Best (84 Generations)



Communicating Homogenous Best (10 Generations)

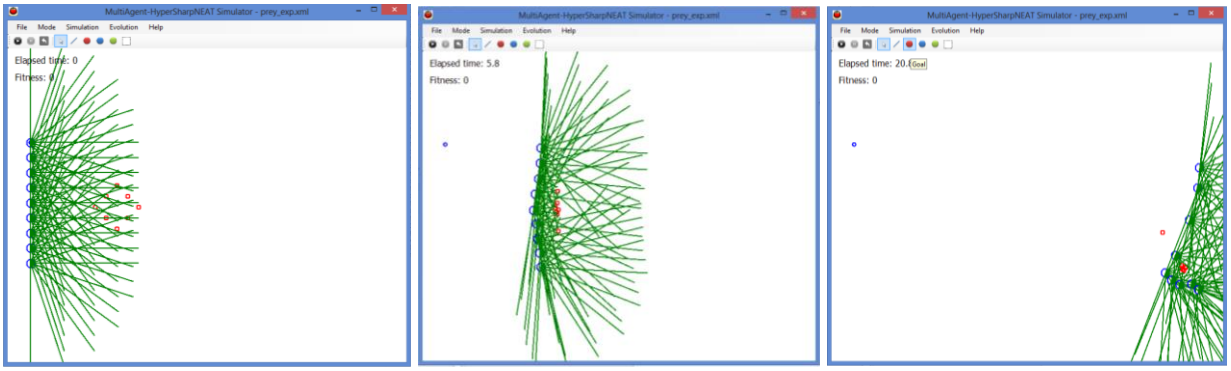
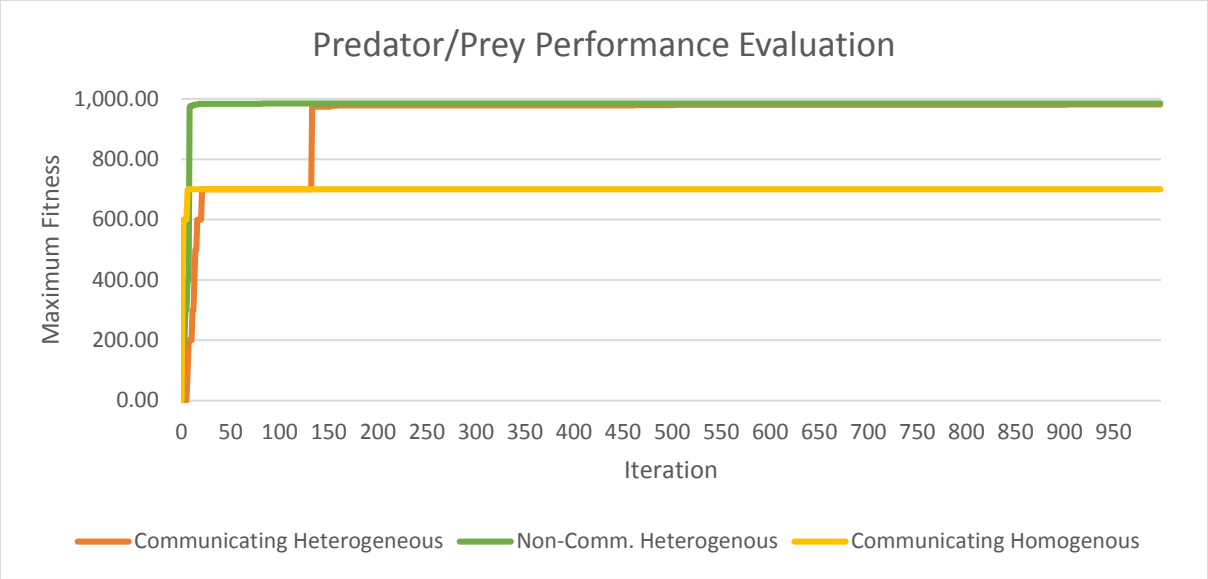


Figure 9. Simulation of the best Heterogeneous and Homogenous predator teams after 84 and 10 generations, respectively. The blue circles represent the seven robotic predator agents. The green lines represent the rangefinder prey sensors for each robot. The red circles represent the prey objects.



Graph 3. Predator Performance over Time. Each team consisted of 9 robots, trained over 1000 generations. The fitness represents the team's performance according the fitness function described in Equation 6.

Because this experiment was adapted from a test designed to demonstrate the performance of heterogeneous agents[26], it is understandable that the heterogeneous teams developed the best strategies. However, it is interesting that the two methodologies, CHet and non-communicating heterogeneous, both arrived at relatively similar fitness scores while taking two very different approaches to the problem. However, the two approaches represent two distinct optimized uses of communication and heterogeneity. The looping method is a complex movement pattern, and a close amount of coordination is required between the agents to align themselves properly to attack the prey when they exit the loop. However, the pincer-like strategy of flanking the prey from top and bottom also requires each agent to move in a specific pattern relative to its neighbors. It is unclear why the CHet method favored optimizing its quality of communication instead of heterogeneity, although this required a great deal of heterogeneous specialization among the team members.

[26] D'Ambrosio, D., Stanley, K., et al. (2010) p. 5

6. Modular Motion Experiment

a. Experiment Parameters

This final experiment was designed to test the ability of multi-agent teams to act as independently moving modules of a single, larger entity. For this experiment, seven robots are placed horizontally in an unbounded space. They are each physically connected to each other by a 'wire' 40 units long. Each robot can only move along the arc of the circle made by rotating itself around the neighboring robot to its left, so that it is never more than 40 units away from its neighbor. The length and direction that it travels is determined by its two motor outputs: it moves in the direction of whichever output is highest. This creates a structure similar to a bicycle chain, where each link in the chain is a distinct module able to exert its own motor output. Each robot has one input – the time elapsed in the simulation – and five hidden nodes. Agents with direct communication have two transmit nodes that they can use to send information to all other agents. Therefore, an agent has twelve receiver nodes to collect information from the other six agents in the chain.

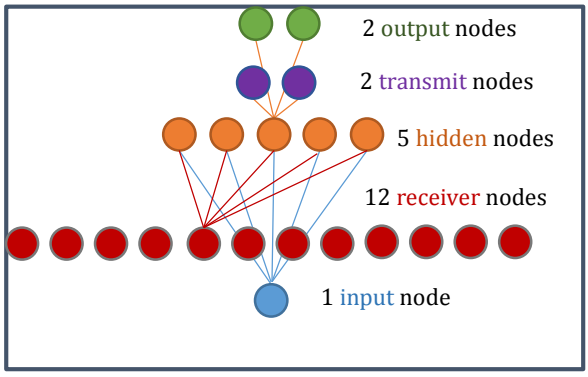


Figure 10. Modular Motion Agent neural network substrate. The network is fully connected between layers, connections are simplified here for clarity. Each node has a linear activation function. This network represents direct communicating robots – non-communicating robots lack transmit and receiver nodes.

The fitness function for this experiment is equal to the total distance moved by all agents in the positive *x* direction (to the right). This can be represented by the following equation, where the fitness is calculated a set of robots R over T timesteps, and Δx_i represents the lateral movement of robot i during the last timestep.

$$fitness_{tot} = \sum_{t=0}^{T} \sum_{i \in R} \Delta x_i \tag{7}$$

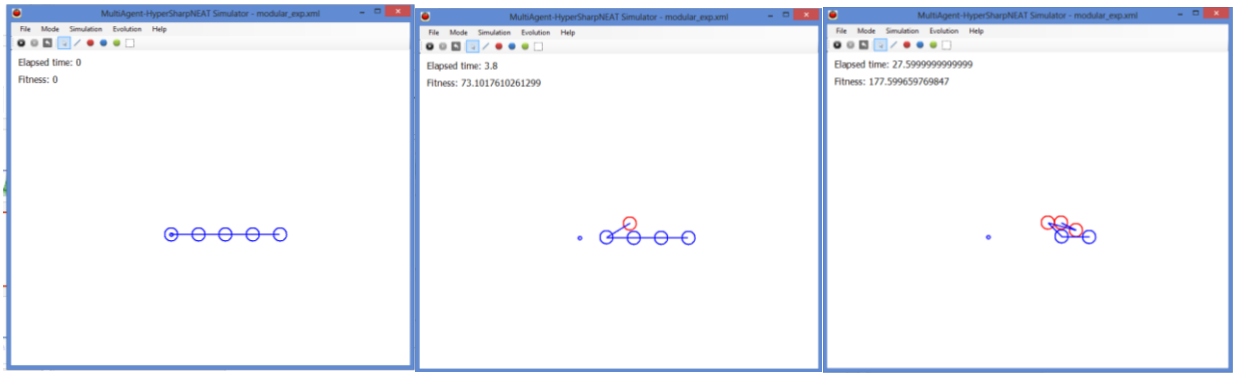
Since all agents are initially positioned to move in a primarily vertical direction, significant coordination is required to move the team in a lateral direction. This simulation lasted for 60 seconds, for 150 genomes over 1000 generations.

b. Results and Analysis

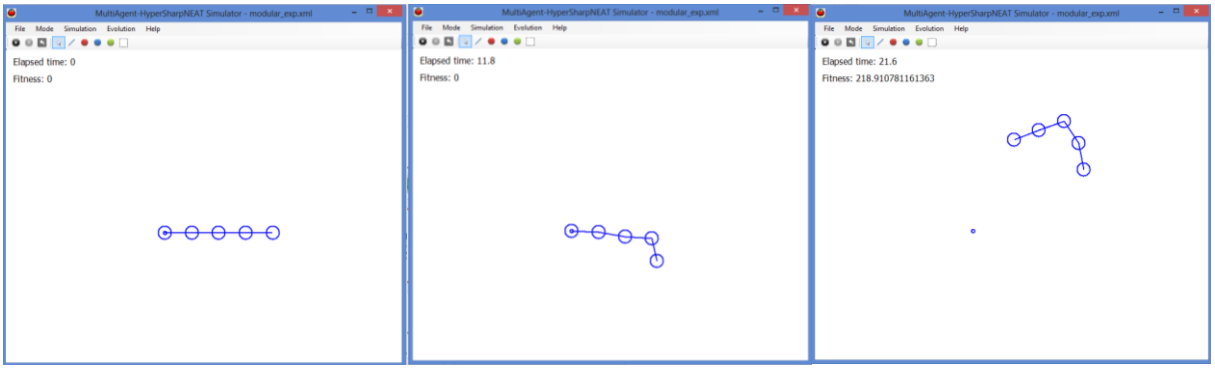
The first generation created by the CHet method failed to find a method of chain locomotion. The leftmost agent rotates around its neighbor, colliding with the top side of the chain. After several seconds, the second and third agent rotate clockwise as well, colliding with the other modules. The chain becomes tangled in itself, and fails to move any further. This gives it a small fitness score of 177.6, accumulated from the slight rightward movement of the first three modules.

However, after 58 generations, the CHet method has found a functional means of locomotion. Mirroring the tactic tried by the first generation, the right most module swings downwards below the rest of the chain. After pausing for a few seconds, the entire chain moves all at once in a clockwise swing, and manages to project itself towards the top-right corner of the simulation. The chain ends up taking a 'V' shape, moving almost like a flock of geese. This method returns a fitness score of 6812.27.

0 Generations:



58 Generations:



730 Generations:

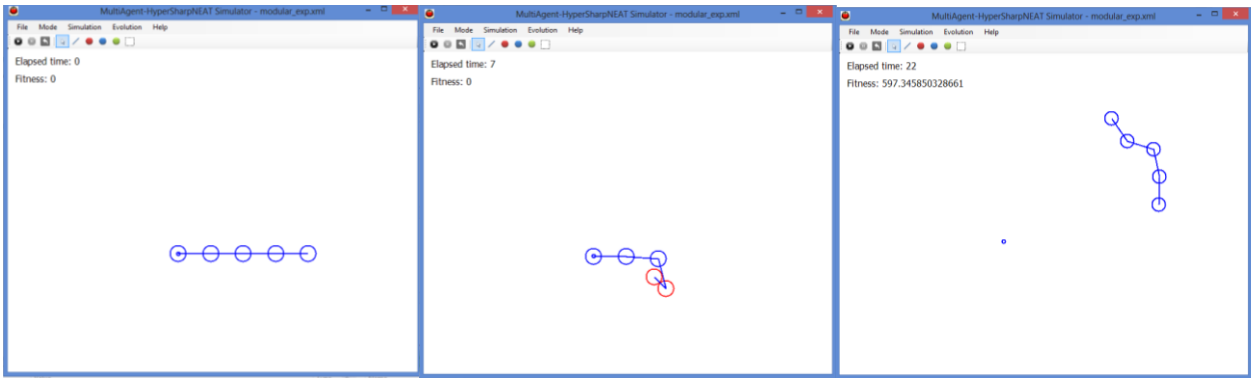


Figure 11. Simulation of best CHet teams after 0, 58, and 730 Module Motion generations. The blue circles represent the five robotic modules in the chain. The blue lines connecting these modules represents the physical link that binds the modules together into the greater chain. Red circles represent modules that have collided with another module.

The maximum fitness strategy for the simulation was discovered by the CHet after 730 generations. In this strategy, the two rightmost chain modules swing clockwise and collide with the center module. The two colliding modules continue to exert motor

pressure, and they are stopped only by the presence of the center module. After a few seconds, the chain almost seems to fire like a gun, as the three leftmost modules swing clockwise, freeing the two right modules and allowing them to launch the system at high speed towards the left. The 'V' shape from the previous generation is still somewhat visible, but the force of the launch moves the formation into a more open, linear stance. After 60 simulated seconds, the modules in the chain moved an accumulated total of 9455 units along the positive x-axis.

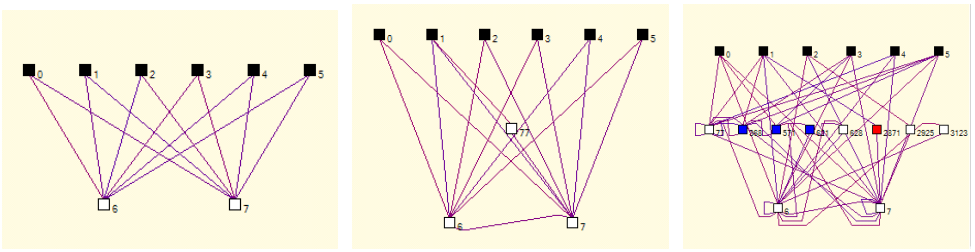


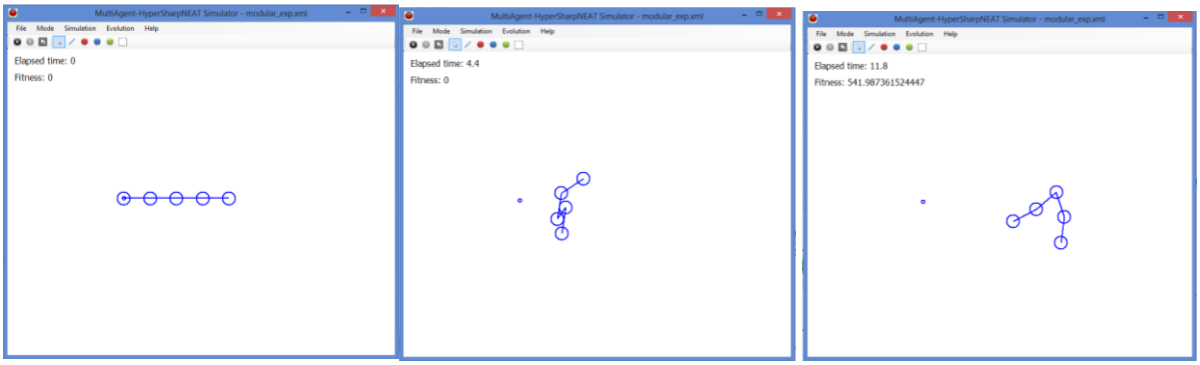
Figure 12 Best fitness Modular Motion CPPNs for CHet at Generations 0, 58, and 730, respectively. The black nodes (0-5) represent the CPPN inputs, which are fully connected to the weight output (6) and the bias output (7). The white hidden nodes represents a Bipolar Sigmoid hidden node. The red node represents a Sine hidden node. The blue nodes represent Linear hidden nodes.

The CHet performed significantly higher than the Non-Communicating Heterogeneous and Communicating Homogenous methods of chain locomotion. The non-communicating heterogeneous team began moving by simultaneously swinging the left two modules downwards and the right three modules upwards – both in the counter-clockwise direction. The two swinging groups move themselves into a vertical position, before the second rightmost module passes in between the third and fourth module. There was no collision detection implemented for the chain connectors, and the modules are spread far enough apart to allow one to pass in between two others. After this complex maneuver, the rightmost two modules complete their counter-clockwise swing, and are now located on the left side of the team. The chain, now shaped like an inverted 'V', moves off towards the

left. This highly specialized formation moves slower than the CHet formation, and results in a final fitness score of 5504.

Despite evolving for 1000 generations, the Communicating Homogenous method failed to discover a meaningful solution to the problem. Its movements end up in a tangled chain, similar to the first generation of the CHet method. This failed strategy results in a final fitness score of 68.

Non-Communicating Heterogeneous Best (942 Generations)



Communicating Homogenous Best (810 Generations)

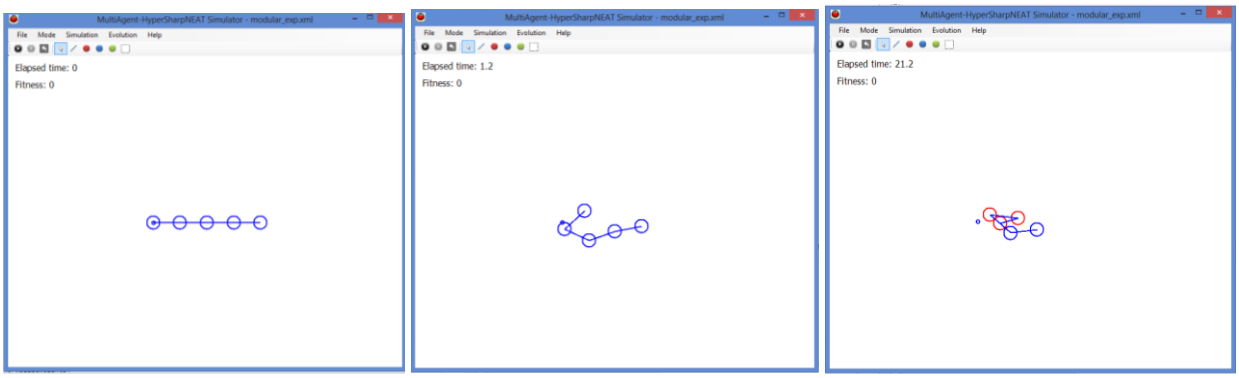
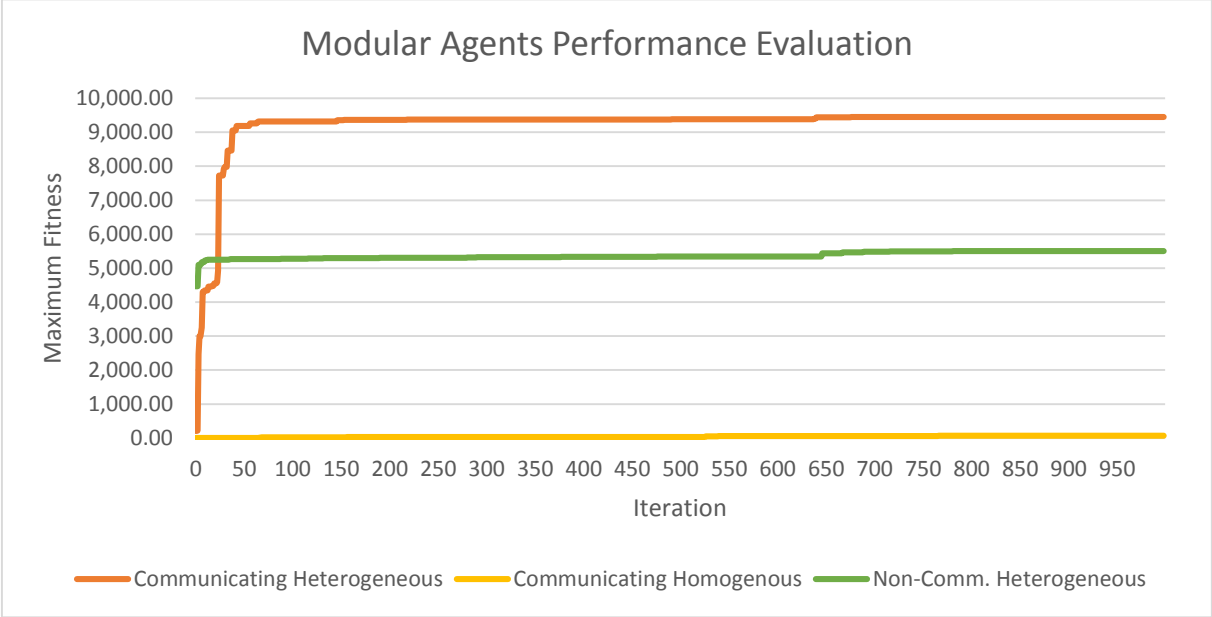


Figure 13 Simulation of the best Heterogeneous and Homogenous Modular Motion teams after 942 and 810 generations, respectively. The blue circles represent the five robotic modules in the chain. The blue lines connecting these modules represents the physical link that binds the modules together into the greater chain. Red circles represent modules that have collided with another module.

This experiment demonstrates the great creative power that can be generated by Neuroevolution algorithms, as the different methods each generated unique and

unexpected strategies to solve a complicated locomotion problem. By utilizing the direct communication between the nodes, the CHet team was able to coordinate its actions to allow simultaneous action, resulting in a spring-like launch across the simulation plane.



Graph 4. Modular-Motion Performance over Time. Each team consisted of 5 robots, trained over 1000 generations. The fitness represents the team's performance according the fitness function described in Equation 7.

7. Conclusion

Neuroevolution is a field that will continue to receive a lot of research in the coming decades, and its applications to multi-team robotics are clear. It has been shown that by combining direct communication with heterogeneity among the agents, robotic teams are able to act in a manner that is at once independent and unified. The Room Clearing experiment showed the impact that communication between multiple agents can have on a geometric movement task. The Predator experiment demonstrated the differences between optimizing based on communication and optimizing based on heterogeneous specialization. And the Modular Motion experiment shows the power that robotic teams

can possess when the two qualities are combined, and the agents may move together with independent, yet synchronized movements.

An interesting observation from this research is the manner in which the CPPN search state is explored in order to optimize the agent's behavior. All of the performance evaluation charts demonstrate the maximum fitness of the HyperNEAT methods moving in discrete steps. Usually, the first generations are marked by a rapid increase in fitness. Because NEAT evolution begins from a minimal structure, the algorithm is able to very early on identify what structural features are most important to the team's performance. However, as the structure becomes more sophisticated, the algorithm must explore a wider search space to identify improvements. That is why the fitness growth usually slows down after roughly 100 generations, and the rest of the search is characterized by sharp jumps as new structural improvements are finally discovered.

However, it should also be stated that the complexity of a genome is not necessarily an indication of its performance. When HyperNEAT finds a solution, it continues to search for optimizations on that solution by creating more complex structures. However, very minor increases in the fitness will still result in this complexification. For example, the Communicating Homogenous method for the Modular Motion experiment never found an adequate solution that allowed the chain to move, yet it continued to make small, inconsequential improvements over the course of 1000 generations. The final, highest fitness CPPN (seen in Figure 14) is incredibly complex, yet produces roughly the same fitness evaluation as the first generation CHet result (seen in Figure 12).

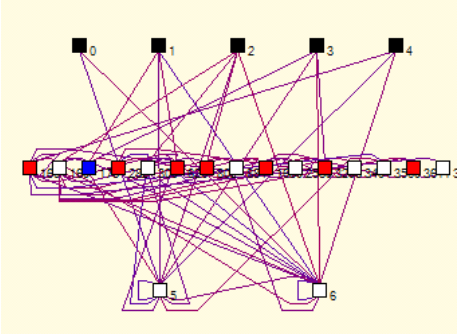


Figure 14. Complexification of the Communicating Homogenous CPPN in the Modular Motion experiment. This CPPN was the highest fitness result after 810 generations, yet the resulting team of agents could not generate valid motion in the chain.

Additionally, it is not explicitly clear that some of these agent teams are utilizing the information provided by their sensors, or if they are simply moving according to highly specialized optimizations inside of their ANNs that control their motor outputs. This could be an area for further research, where a greater emphasis is placed on the sensor inputs in determining the success of a communicating heterogeneous robot team.

8. References

[1] Clark, C. (2011) "Lecture 14 – Markov Localization" [PowerPoint slides]. Retrieved from http://www.cs.princeton.edu/courses/archive/fall11/cos495/COS495-Lecture14-MarkovLocalization.pdf

[2] Russell, S. and Norvig, P. (2010) *Artificial Intelligence*, *3rd Edition* (New Jersey: Prentice Hall), p. 425

[3] D'Ambrosio, Lehman, and Risi (2012). "HyperSharpNEAT-Compatible Mutliagent Simulator and Experimental Platform *Now Including the Evolvable Substrate* (Version 1.0)" Retrieved from http://eplex.cs.ucf.edu/software/AgentSimulator_v1_0.zip

[4] D'Ambrosio, D., Stanley, K., et al. (2010) "Evolving Policy Geometry for Scalable Multiagent Learning" *GECCO-2010* (New York: ACM)

[5] D'Ambrosio, D., Stanley, K., et al. (2012) "Multirobot Behavior Synchronization through Direct Neural Network Communication" *ICIRA-2012I (*New York, NY: Springer-Verlag)

[6] Stanley, K., D'Ambrosio, D., and Gauci, J. (2009) "A Hypercube-Based Encoding for Evolving Large-Scale Neural Networks" *Artificial Life* (Cambridge, MA: MIT Press), p. 188

[7] Stanley, K. and Miikkulainen, R. (2002) "Evolving Neural Networks through Augmenting Topologies". *Evolutionary Computation* **10** (2), p. 99

[8] Stengel, R. (2011) "Lecture 13. Monte Carlo Evaluation and Evolutionary Search" [PowerPoint Slides]. Retrieved from https://blackboard.princeton.edu/bbcswebdav/xid-338511_1