An Application of NEAT:

Recurrent Neural Networks and a Communication Task

Final Project Paper
CS 81: Adaptive Robotics
May 13, 2010

1. Abstract

This paper explores the role of recurrent, NEAT derived neural networks as well as the role of communication in a team-based robot task where knowledge of time is important. For the purposes of this work, the task was implemented in simulation using Pyrobot a robot simulator written and developed in Python by Douglas Blanks and Lisa Meeden [5]. The task requires a team of two robots to first collide with a red circular puck object and then a blue circular puck object. It is complicated by the fact that all members of the team must collide with a red object before any member may collide with a blue one. Unfortunately, the results obtained from the experiment do not demonstrate that robots with recurrent neural networks perform any better than robots with feed-forward networks. In particular, robots in the population as a whole do not learn to reliably perform the task, implying that the task may be too difficult to evolve a fit population within 100 generations. This is most likely due to the fact that fitness is awarded in a very discrete way such that networks learn to avoid blue and pursue red but never to make the transition from colliding with red pucks to colliding with blue pucks. These problems were not resolved within the window of possible experimentation.

2. Motivation

The primary motivation for this experiment is to provide further evidence to show how neural networks can solve complex problems geared towards real-world applications. As such, the task was devised to illustrate that through the utilization of network recurrence and communication, NEAT evolved neural networks could not only be applied to robotic team tasks but also quickly and effectively learn to solve them. Learning for robotic teams is a necessity because many challenges that cannot be easily overcome by a single robot are solved with greater facility by a team. Thus a large area of neural network research has been in its application to robotic teams, as this serves as an area with great promise for future scientific advancement [2].

3. Related Work

In his paper *A comparison of approaches to the evolution of homogeneous multi-robot teams*, Dr. Matt Quinn demonstrates that in some nontrivial applications, a team of robots with homogeneous morphologies perform better with heterogeneous neural networks as brains. For his experimental setup, he uses two Khepera robots in simulation, each separated by a distance on the order of the diameter of a single robot. For the first task, robot teams start in one of 45 predefined configurations specified by discrete heading values in combination with their distances from one another. The brains of each robot are recurrent neural networks with eight inputs and four outputs.

In his paper, the author presents two distinct approaches for the creation of the teams: clonal and aclonal. In the context of the clonal approach, namely the one where both robots share an instance of the same brain, fitness is the average of the team's score for each of the 45 possible configurations. For the the aclonal approach, the one in which robots in the team have heterogeneous brains, each individual is evaluated twice for every configuration for a total of 90 evalutions, each one with a distinct partner. An individual's fitness is thus the average of the fitnesses obtained from each of the 90 teams of which it participated. This provides a good heuristic for evaluating an individual's performance because every robotic brain is tested not only in every possible configuration but serves as a team member with usually a statistically representative portion of the population.

The task itself was intuitively very simple but far from trivial. Both robots were given 10 simulated seconds to move 25 centimeters from their original positions. Although one would expect robots with homogeneous neural networks to on average

achieve higher fitness, as has been almost universally postulated and shown experimentally, this was not the case. Aclonal teams on average performed much, much better than clonal teams. Approximately twice as many aclonal teams scored greater than 95 percent of the maximum possible fitness score when compared to clonal teams.

A second task was then presented in which aclonal teams performed worse. This task required the two robots to turn in the same direction. Because this task does not require any specialization, Quinn argues that neural network heterogeneity was a detriment. He then concludes the paper by hypothesizing that for tasks that require different kinds of behavior, specialization of robots is helpful. Thus since heterogeneity between team members' brains allows for such specialization, he argues that the aclonal approach is indeed useful and should be used in cases where there are distinct subtasks [4].

The article *Recurrent neuronal circuits in the neocortex* by Douglas and Martin discusses the nature of recurrence within the neocortex, a part of the brain used heavily in social interaction and in processing sensory stimuli. Douglas and Martin find that neurons of similar types tend to be located close to one another and that almost all recurrent and excitatory connections are between neurons that are in relative proximity to one another. Consequently, the authors conclude that the majority of computation occurs within local, recurrent, neuronal circuits. They also suggest that the connections between individual circuits most likely serve as a means of information transfer but do not directly affect the lower level computation that occurs in the local recurrent circuits.

This article is of importance because it suggests that real brains make use of recurrent subnetworks that have specialized to form a specific type of computation. Therefore, the use of application specific, recurrent neural circuits in artificial neural networks will almost certainly allow for more complex computation than that that occurs in purely feed-forward networks. Since the neocortex is used for processing both auditory and visual stimuli, using a recurrent neural network to process information originating from the sensory inputs of a developing robot seems like the natural artificial analog. Although this article addresses primarily the underlying structure of biological neuronal circuits, it is also of significant importance to the field of artificial intelligence. In particular, it is quite conceivable that a synthetic adaptation of many of these biological networks could be beneficial in many robotic applications [1]. For these reasons, the next reviewed paper describes the genetic algorithm NEAT, which allows for the evolution of recurrent neural networks in conjunction with other appealing features [2].

Probably one of the best papers on NeuroEvolution of Augmenting Topologies (NEAT) is *Competitive Coevolution through Evolutionary Complexification* by Stanley and Miikkulainen, its creators. In their work, they give a very detailed description of the NEAT genetic algorithm and give strong experimental support for its superiority over fixed topology approaches. Through their comparison of the best neural network with fully connected, recurrent fixed topology neural networks with the same number of nodes, they demonstrate the superiority of NEAT. Specifically, even though the fully recurrent networks contain as a directed subgraph the network evolved by NEAT, mutation on the weights of the connections never produces an individual who is as fit as the winning genotype evolved using NEAT.

NEAT works primarily on a process of elaboration. At the start, all genotypes of a given population start off as feed-forward networks with random connection weights linking every input unit to every output unit. The population is then evaluated for a given task, each individual receiving a fitness score determined by the user-defined fitness function. From this point, several forms of mutation can occur, changing both the weights of the artificial synapses and additionally the structure. For instance, between

any two unconnected nodes a new connection may be added as a mutation. Additionally, a new node may be added between two existing nodes. The old connection is disabled and two new connections are formed.

Another form of innovation in NEAT is through sexual recombination, in which two brains that are represented as genetic chromosomes are crossed over to form a new one. This also presents a form of mutation because there is a chance that disabled genes may become re-enabled. In addition to artificial meiosis, NEAT makes use of speciation as a means of protecting innovation. Genotypes that are within a parameter-defined genetic neighborhood define a species. When an individual mutates and become sufficiently genetically dissimilar from its parent species, a new species is defined. As a result, NEAT benefits from the fact that members of a species compete primarily against one another. This allows new species to become refined and encourages genetic diversity.

The algorithm is also useful because the presence of recurrent connections allows for a conception of time and hence memory. Stanley and Miikkulainen argue that for the task of dueling robots, having a memory of the past is necessary because much of the information relating to the task is rooted in past events and actions. Thus for a robot player to maximize its success, it must remember what it has done and additionally its opponents actions. Only given this information, can a robot realistically make the best decision as to the course of action it should take. This is intuitive given the fact that almost all competition-based strategy centers around reacting to and countering an opponent's actions. Without memory, a robot can only respond to its opponent's behavior at the current time step. Consequently recurrent connections are a necessary part of sophisticated interaction between robots [2].

4. The Task, its Implementation and How it Relates

On the lowest level, the task is done in simulation. This is because there are number of complications that arise when using actual robots. Real robots have a certain degree of uncertainty about them in that experimental parameters are much more difficult to control and modulate. Of further note, experiments run in simulation do not risk damaging the equipment and can be run much faster and in much greater number. This allows for rapid prototyping and evolution that would be infeasible in the real world where time and resources would be at a premium. As such, the design decision was made to use Pyrobot, a two dimensional robot simulator which allows users to model robot interactions with a planar environment.

Pyrobot has many, nice, built-in features that allow for easy design and testing of robot-based scenarios. For the purposes of the experiment, many of these key features proved invaluable. This included built-in support for cameras including the use of blobify and match filters as well as support for sonars. Pyrobot also permits simulation with multiple robots, which is essential for the task as it is related to communication, an activity involving more than one individual. By using Pyrobot, the experimenter managed to focus more of his efforts on the experimental setup and its execution and less on the low-level, hardware-dependent details [5].

The task itself is rather intuitive and has a simple setup. Two purple Pioneer robots are placed at opposite corners of a square box which is approximately ten times longer and wider than the length of a single robot. In the world, there are additionally four circular pucks which have a diameter approximating a fifth of the length of a single robot. Two of the pucks are red and two are blue. At the start of each trial, pucks spawn randomly at points of a two dimensional lattice which occupies the bulk of the central part of the world. In this way, pucks are sufficiently distant from one another that rudimentary robotic sensory can distinguish them from one another. Robots are then given 100 time steps to work together as a team in order to collide with all of the

pucks. However, there are some minor caveats. In particular, once a robot collides with a red puck, it may no longer collide with any other red pucks if it and its partner are to continue to accrue fitness. Instead its partner must collide with the remaining red puck. Once this happens, both robots may proceed to collide with blue pucks. Yet as before, each robot can collide only with one blue puck. When this happens, the robots achieve the maximum fitness (See Figure 1 for a screenshot of the world).

Although this task sounds simple at first, it presents some major challenges. If the neural network is the control model of choice, then several design issues immediately arise. Firstly, one must decide the inputs and outputs of the network and then select a genetic algorithm. For the purposes of this experiment, a relatively rich set of sensors were chosen in order to give the robot the most information possible. This was a design decision based on research by Nolfi and Floreano in [3], strongly supporting the claim that an increase in the complexity of the sensory inputs leads to an associated increase in the complexity of behavior that the neural network can evolve. Thus since this task requires substantial knowledge of the domain at the current time step, the robots were given both cameras and sonars.

The robots each had a front sonar input unit and six additional units related to color perception using the camera. Front sonars were given to the robot to allow it to determine the distance to pucks and to reduce the amount of collisions with walls. The units related to color consisted of three grouping of two units for each of the colors red, blue and purple. These inputs fed in the area of the bounding box for largest blob of each color and their positions in the horizontal direction with respect to the midpoint of the field of view. These two kinds of color inputs would allow robots to get an idea of the proximity of the largest blob of a certain color and also its heading relative to the robot. In this way, a robot would be able to adeptly maneuver its way to a puck of a certain color to make the required collision. (See Figure 2 for a description of the robot's sensors).

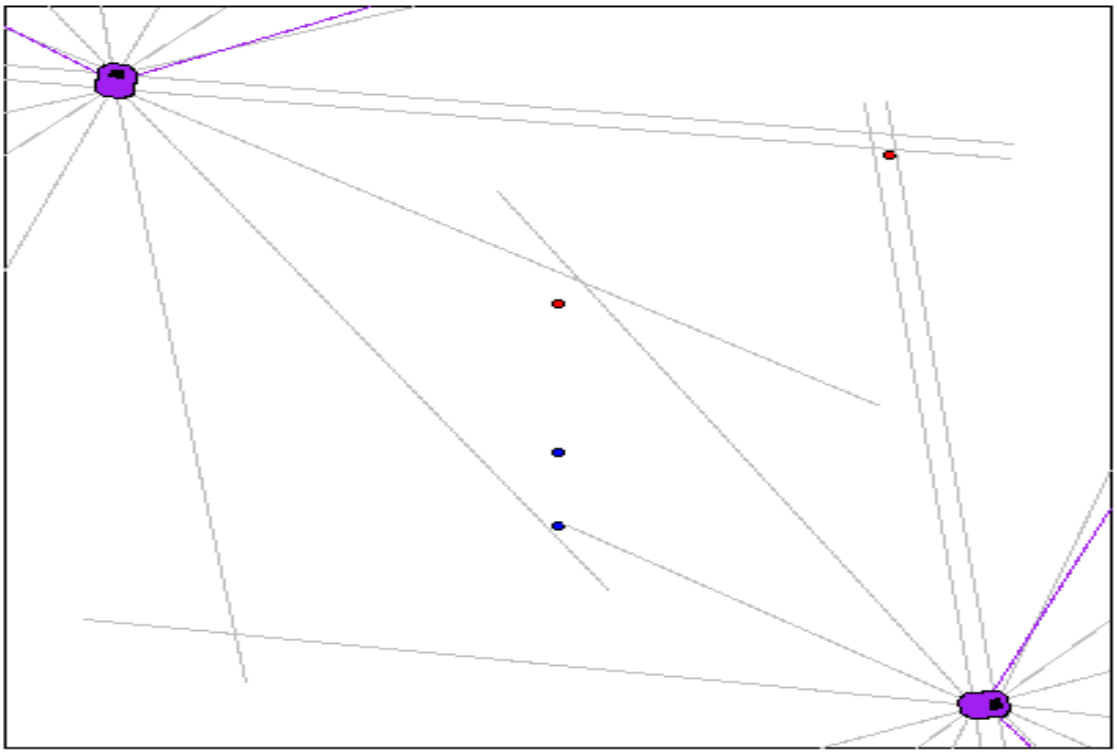

***Figure 1**: Shown here are the two robots at their starting positions in the upper left*

and lower right hand corners. The pucks spawn randomly in the rectangular region between the robots. In this example, the blue pucks spawn a distance of 1 from each other, which is the closest they can be. The gray lines emanating from the robots represent sonar sensors and the purple lines represent the farthest extent of the field of view for the camera.

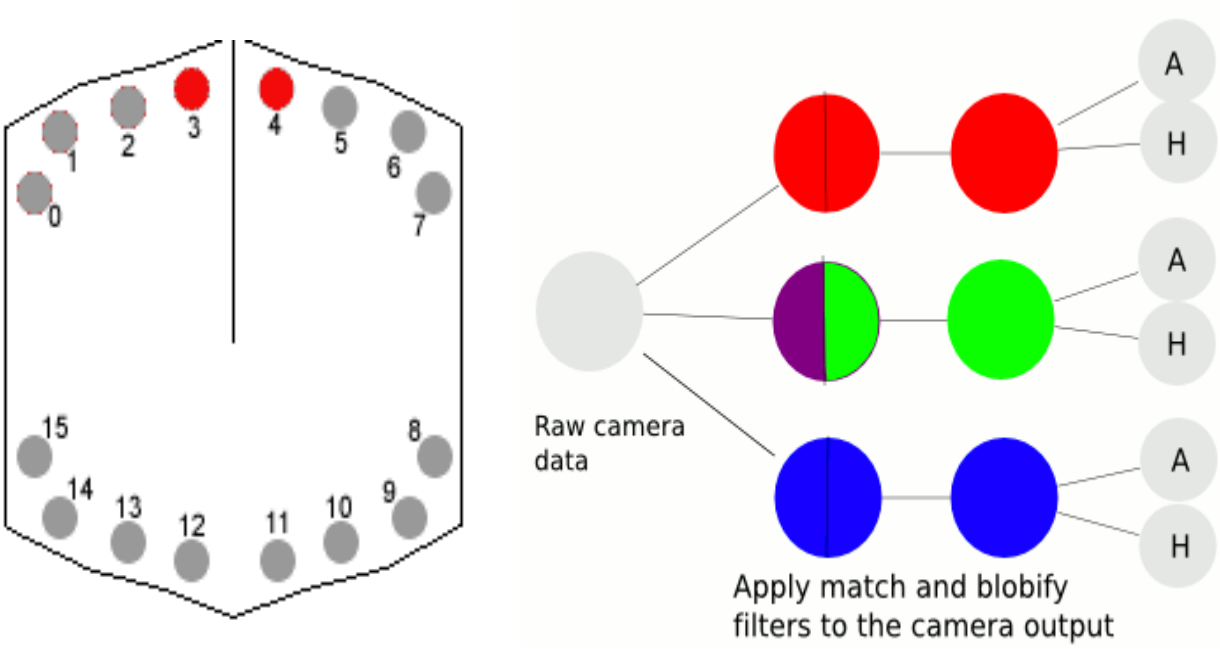

***Figure 2**: The above left graphic shows the sonar sensors available to the Pioneer robot. For the purposes of clarification, the two front sonars have been highlighted in red. The minimum sensor reading from sonars 3 and 4 is fed into the neural network's sonar input unit. The right graphic shows how the camera data is first sent through a match filter that matches red to red, purple to green, and blue to blue. Then a blobify filter is applied to each of the matched colors, and the areas of the biggest blobs of the colors red, green (purple) and blue each go to an input unit in the neural network (represented by the units labeled A). The H input units receive as input the scaled horizontal midpoint of the largest bounding box for a given color. This is computed as (xMin + xMax)/(2 * w), where this is the midpoint divided by w, the width of the viewing window of the camera.*

Additional inputs were added to enable communication. Each robot in the team received an input from the other robot, allowing robots to pass messages. The goal of this feature was to allow each robot to let the other one know if it had already collided with a red puck or if it still needed to do so. Ideally, the robots would develop a dialogue, allowing them to coordinate their collisions with the pucks, an integral part of completing the task and achieving maximum fitness. In later experiments where network recurrence was permitted, robots were given an additional sensor input which told them whether they had already collided with a red puck. In the opinion of the experimenter, this sensor models real-world sensors which tell robots if they have successfully stored an object. Consequently its addition is perfectly legitimate.

The outputs of the each homogeneous robot's neural network were much simpler. There were two motor outputs, one for the left motor of the robot and one for the right motor as well as the aforementioned communication unit to the other member of the team. Sending values directly to the left and right motors to control the robot is a

practical choice because it makes interpretation of movement simple and it can be used to describe essentially any possible translational and rotational action undertaken by a car-like robot. As far as the communication output, only one unit was deemed necessary. This is because a single output unit can emit multiple values, each of which potentially mapping to distinct semantics for a given message (See Figure 3 for a generalization of the neural network).

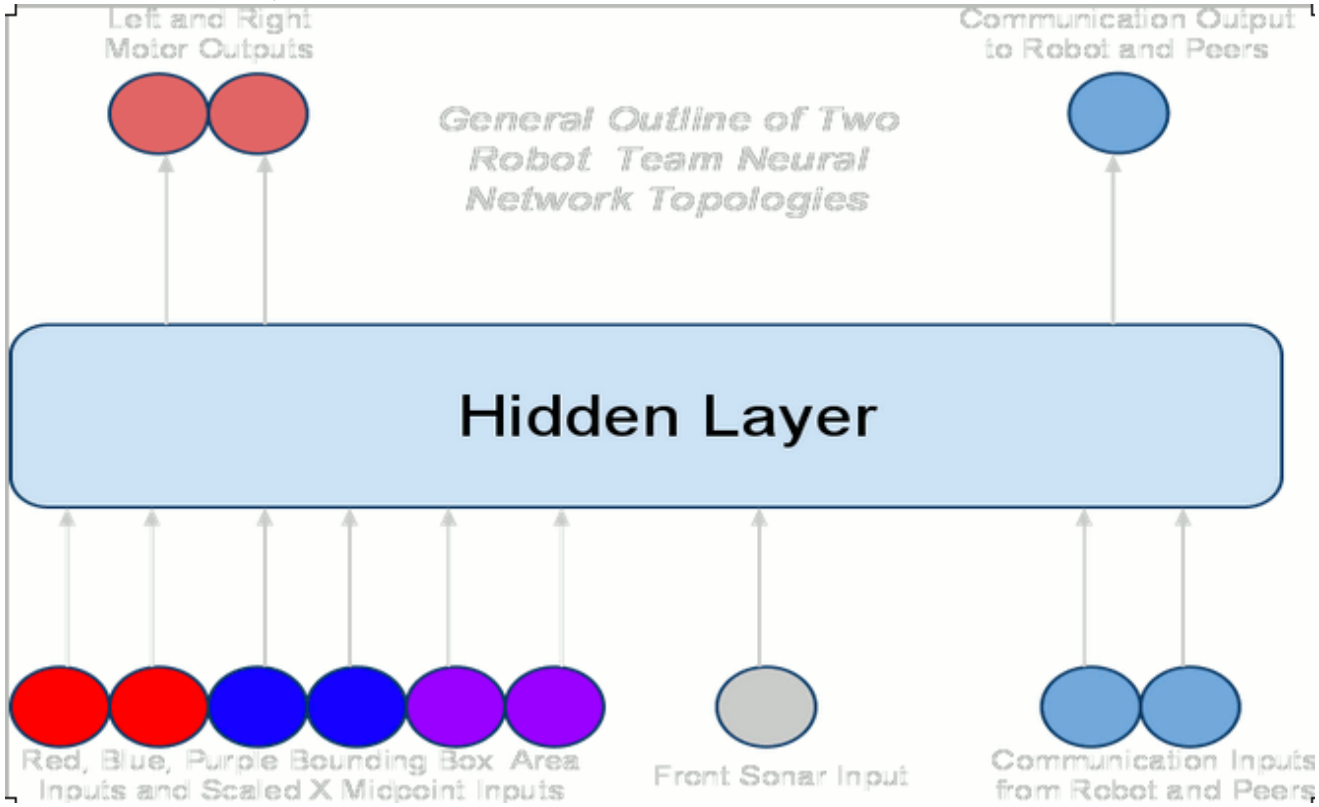

***Figure 3**: Illustrated above is general mock-up of the neural network. It contains nine input units, and three output units. For part one of the experiment, recurrent connections are disabled and for part two they are turned on. Thus there can be circuits between inputs, outputs and the hidden layer. This will be further explained in the next section.*

NEAT was chosen as the genetic algorithm. This is because NEAT efficiently searches the domain space by incrementally building on already existing solutions. Additionally, NEAT allows for the evolution of recurrent connections. The benefit of the evolution of recurrent loops, as hinted to previously, is that it mimics some of the computational centers within actual brains. Since Douglas and Martin infer recurrence increases the computational capacity of biological neuronal circuits, the synthetic analog is the class of recurrent neural networks. Thus by using NEAT, the robots will be able to increase the computational power of their neural networks via recurrent circuits and thus ideally develop more interesting behavior. Additionally, NEAT was selected as the evolutionary algorithm of choice because it allows for multiple species, which has been shown to be beneficial when attempting to reduce the number of generations it takes to evolve a fit population.

Of perhaps equal importance was the choice to use clonal teams over aclonal teams of robots. The main reason for doing this was not based on merit but unfortunately based on limitations to the existing NEAT library implementation inclued with Pyrobot.

Specifically, the current version does not support the evolution of multiple populations. As such, an aclonal implementation as Quinn advocates is not a practical option. However, the forced use of clonal teams should not be seen as too much of a detriment. Since all robots must perform the same kinds of actions, e.g. finding red and blue lights and synchronization, it makes sense for them all to have an instance of the same brain. So even though it would be interesting to evaluate the use of heterogeneous brains for control of the two robots, hopefully the use of homogeneous brains would not significantly affect the results.

5. Evaluation of Fitness

Each team of robots which are clones of the same individual from the population is given five trials of 100 time steps to gain fitness. Total fitness is the summed fitness from each trial. In this way, a fitter chromosome is more likely to have a more consistent level of fitness, as total fitness is a measure of performance from five separate trials. Max fitness is defined as 60, 12 being the max subscore from each of the five trials.

For fitness evaluation, the following metric is used for each trial:

```
When there is a collision between a robot and a puck:
    if there are still red pucks left at the time of the collision:
        if robot1 collides with a red puck and robot2 has yet to do so:
            award the team one point
        else if robot2 collides with a red puck and robot1 has yet to do so:
            award the team one point
        else: # One robot has collided with two pucks or a blue puck
            award no additional fitness and start the next trial

    else if there are still blue pucks left at the time of the collision:
        if robot1 collides with a blue puck and robot2 has yet to do so:
            award the team two points
        else if robot2 collides with a blue puck and robot1 has yet to do so:
            award the team two points
        else:
            award no additional fitness and start the next trial

    else: # The robots have completed the task, give them a bonus
        double the current fitness for the trial
```

Although this fitness function is relatively discrete, i.e. fitness is linked to the number of red and blue pucks, it should promote the evolution of fit individuals. Individuals that manage to correctly collide with more of the pucks in the right order are given more fitness than those that do not. Additionally, by awarding teams a bonus for completing the task (doubling the fitness they receive from a trial), means that robots that complete the task some of the time will be more fit than robots that never complete the task. This is ideal because completion of the task is the desired outcome, and so robots that take some risk to complete the task should be rewarded. Otherwise robots that take no risk but finish the bulk of the task will become the dominant members of the population, regardless of whether they are ever able to achieve the actual objective.

6. Experimental Setup to Measure the Effects of Recurrence on Behavior and Hypotheses

The task was designed to test the role of recurrence in neural networks and its effect on behavioral complexification. As such, neural networks will be evolved using

NEAT in two ways, first with the evolution of entirely feed-forward networks and second with the evolution of recurrent networks. In the former, one would expect that robots will not be able to learn the task, as they will not know anything about previous time steps. Consequently, they will most likely only learn to pursue the red pucks but will not learn to wait for their teammate to collide with the other red puck. This is because the action of waiting requires a notion of time, and entirely feed-forward networks only have knowledge of their current inputs.

For the latter case where recurrence exists, the robots should be able to learn the task within a time window of 50 generations. Once useful, recurrent subnetworks evolve, the two robots should be able to develop a notion of past sensory inputs. This will permit the team members to evolve the tactic of waiting for one another and to make better use of their communication. Additionally since the team-based problem of synchronized collisions is related to past events, namely collisions, the ability to store this task relevant information would be invaluable. Consequently, it is the opinion of the experimenter that the robots with recurrent networks should drastically outperform teams evolved with recurrent neural networks.

7. Analysis of Preliminary Results

Initial results do not support the hypothesis of the author that recurrent networks would drastically outcompete feed-forward networks. This was unfortunate because it meant both types of neural networks performed poorly, not just the feed-forward networks as predicted. After 100 generations of evolution with a population totaling 50 neural networks, the individuals in the feed-forward networks failed to learn the task well. This result was not unexpected because, as mentioned previously, the task requires knowledge of prior time-based events. However, the recurrent network population managed to learn the task only marginally better. These results at first seem counterintuitive, but there are a number of factors that could have contributed to the stagnation of learning that occurred within the recurrent neural network population. An explication will be provided in the subsequent subsections 7.1 and 7.2.

7.1 Analysis of Feed-Forward Network Population

Figure 4 shows the average and max fitness for a run of the experiment for a total of 30 generations. In this case, the networks are feed-forward. Although the experiment was run for longer than 30 generations a number of times, this graph is representative of the behavior found in previous runs. As one can see, average fitness, shown as the blue line, plateaus after about five generations. The fitness achieved by the fittest individual of the population for each generation, shown by the red line, exhibits very noisy behavior. This is due to the fitness function because members of the population that chance on completing the task successfully achieve much higher fitness than members of the population who only manage to learn to collide with red. With a population of 50 individuals, this occurrence happens on average with approximately one individual during each generation. As predicted, these data show that the team members were unable to learn the task. Neither the average nor the best fitness curves have an upward trend but appear purely noisy.

Figure 5 shows the genotype of the most fit member of the feed-forward population from generation 29. As one can see, the network topology is simple. Since this network outperforms other more complex networks, it is likely that given no capacity for memory, additional complexification through the expansion of connections and nodes is not beneficial. However, one does note that in the winning neural network that both hidden nodes are related to additional processing of information related to the color blue. This fact makes sense because the evasion of blue pucks followed by the active pursuit

of blue pucks requires complex behavior. Additionally, the hidden unit which receives its input from the blue bounding box position input unit is connected to the communication output. From this, it makes sense to conclude that communication between robots in the team is driven largely by the color blue, as blue pucks present both a danger and potential bonus depending on where the team finds itself in the execution of the task.

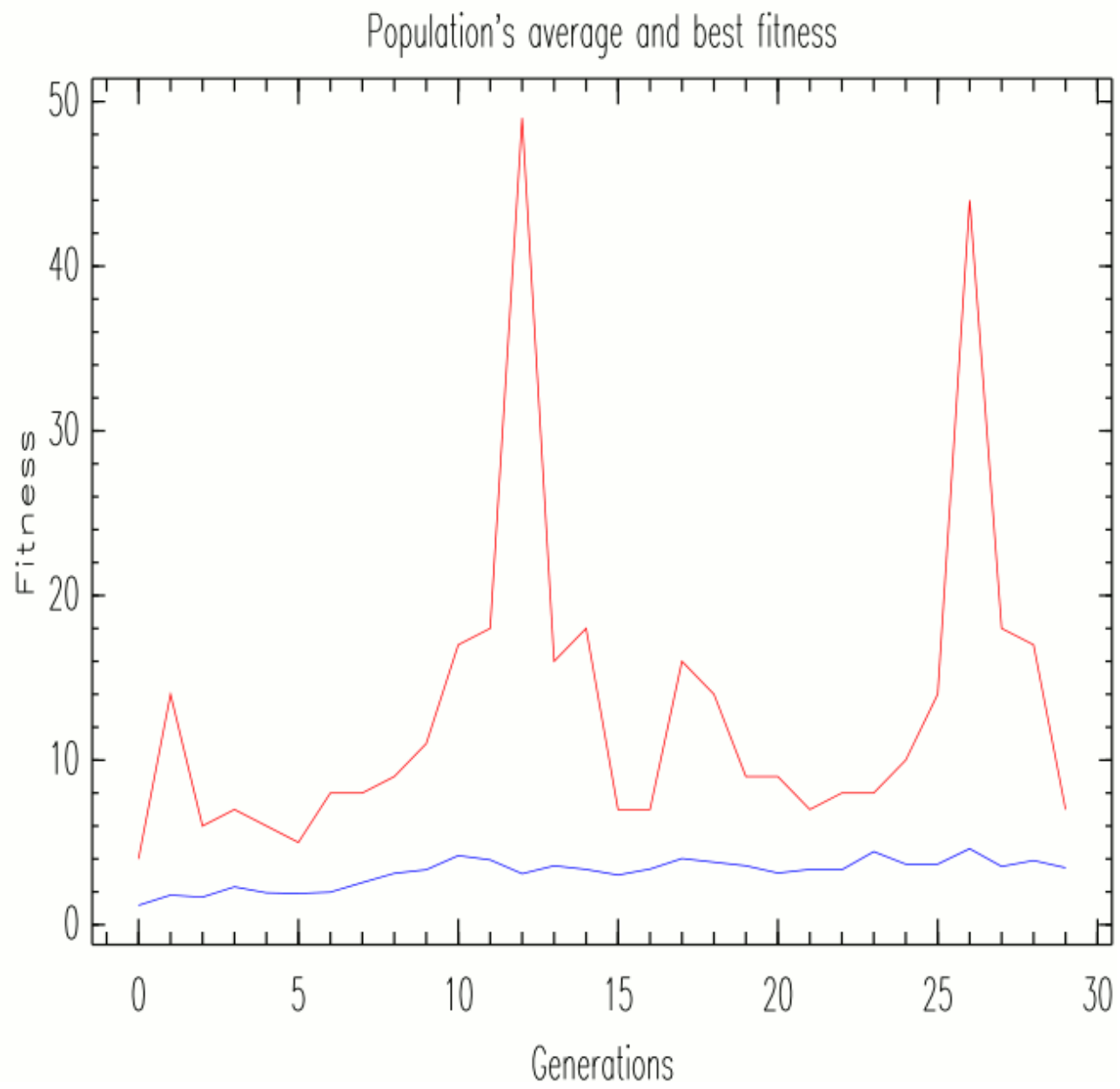

***Figure 4**: This graph illustrates the average and best fitnesses of the clonal, feed-forward network controlled teams. As one can see, there are spikes at generations 12 and 26 where the robots almost manage to finish the task every time. However, this achievement is merely a coincidence and when reevaluated, said genotypes attain on average fitnesses between 15 and 25.*

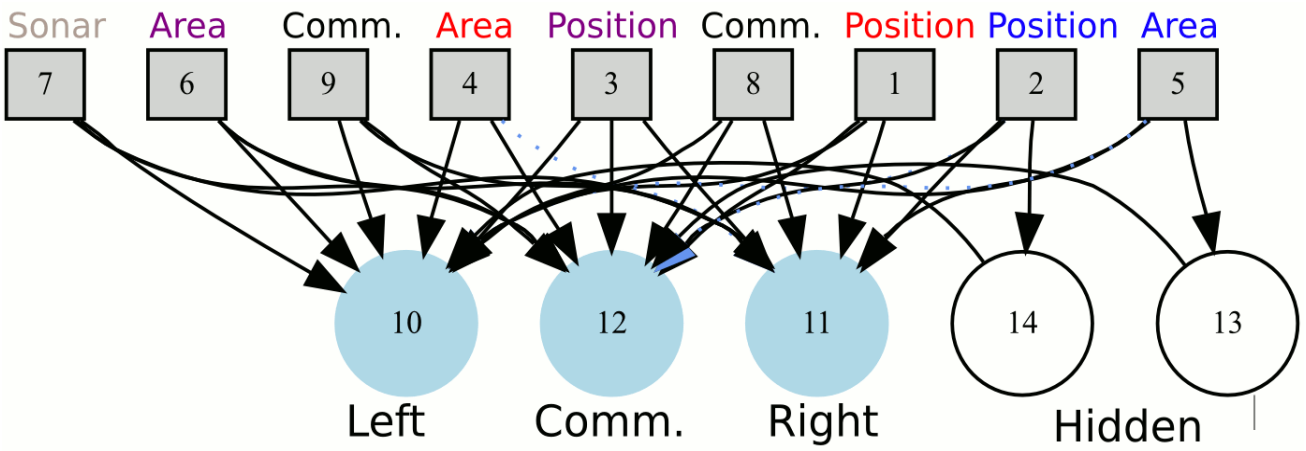

Figure 5*: The above neural network represents the most fit individual from the feed-forward population. Input units 1, 2 and 3 feed in the relative positions of the largest red, blue, and purple bounding boxes in that order. Units 4, 5, and 6 feed in the areas of the largest red, blue, and purple bounding boxes. Number 7 is the minimum reading from the two front sonars. Units 8 and 9 are the communication input units. Robot 1's messages are relayed through input 8 and robot 2's messages are relayed through input 9. The output units 10 and 11 are the signals to the two motors, and unit 12 is the communication output which is fed into either input 8 or 9 of both robots, depending on the robot's identifier of 1 or 2. The network has two hidden units 13 and 14.*

7.2 Analysis of Recurrent Network Population

As previously stated, the recurrent network population did not perform much better than the feed-forward network population. This was unexpected as the evolution of recurrent connections should have allowed the robots to develop the ability to store information from previous time steps. Thus they should have been able to better understand their environment and hence the task. Even when the experiment was run for 100 generations with a population of 100 individuals, no species managed to master the task.

There were some results which were superior to those of the feed-forward networks. Networks with recurrence were more consistent in the fitnesses they obtained. Within each generation, a large minority of the individuals (approximately 30 percent) performed relatively well, consistently completing the first part of the task (colliding with the red pucks). However, the average fitness was not much better than the average fitness from the feed-forward population (See Figure 6 for the graph of average and best fitness for individuals of the recurrent network population for a representative run). This illustrates the fact that the feed-forward population produced more individuals that received outstanding fitness but additionally many more individuals with far worse fitness. The reason why this happened was that robot teams that attempted to collide with any puck received on average better fitness than those that chose only to collide with red. Robots that chose only to collide with red pucks achieved at best a fitness of 10 out of the maximum of 60, and more often than not obtained a fitness score of 6 to 8. However, robots that collided with any puck were able to occasionally receive points for correctly completing the task. Since the fitness function is heavily weighted towards rewarding task completion, this allowed robots with chaotic movement to outcompete robots that only went after red pucks. As a result, approximately 20 percent of species

adopted this strategy. These results were consistent across runs.

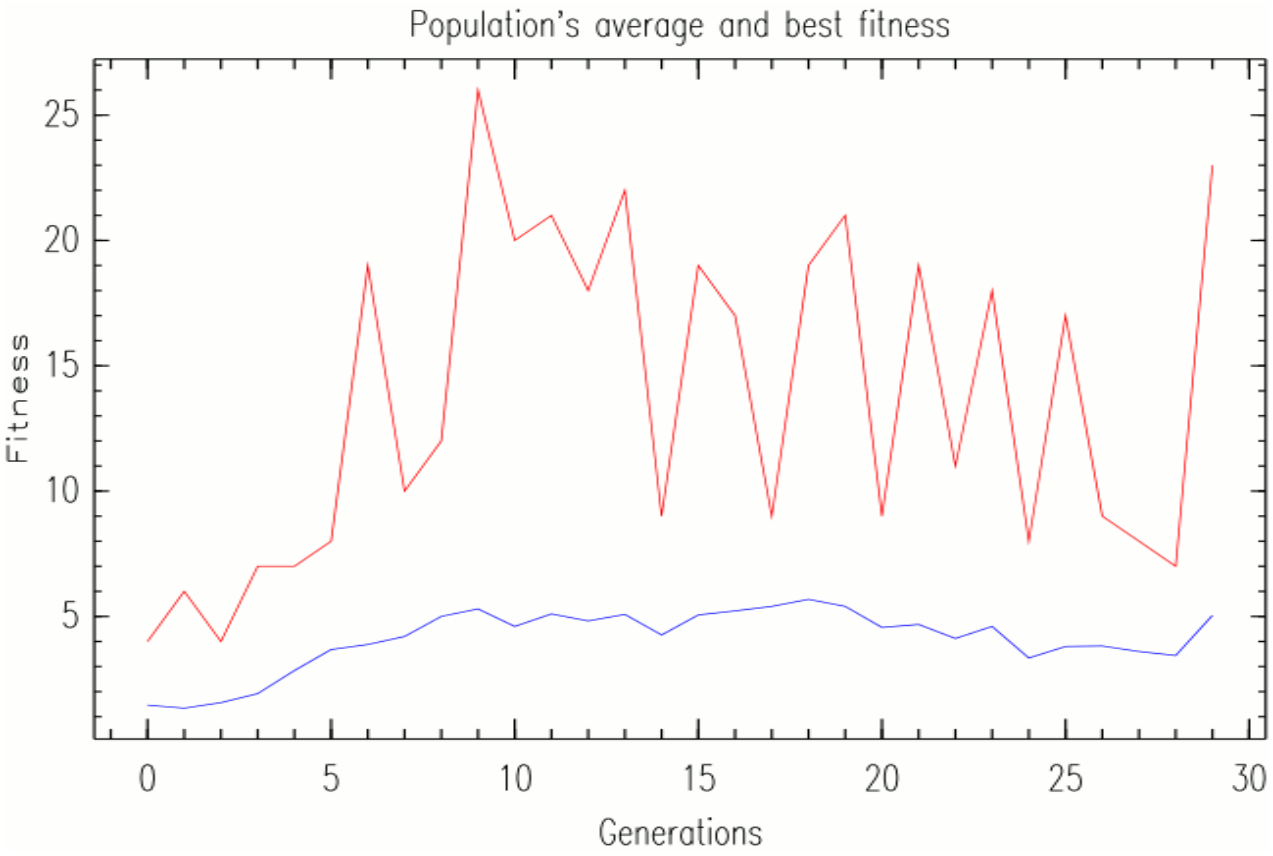

Figure 6*: The above shows the average and best fitnesses for individuals in the evolving recurrent neural network population of 100 individuals. As in the previous graph, red represents the fitness score of the fittest individual from the population for a given generation The blue curve is the mean fitness score for the entire population. It is quite evident that the max fitness is quite noisy, suggesting that much of this high fitness was obtained via luck.*

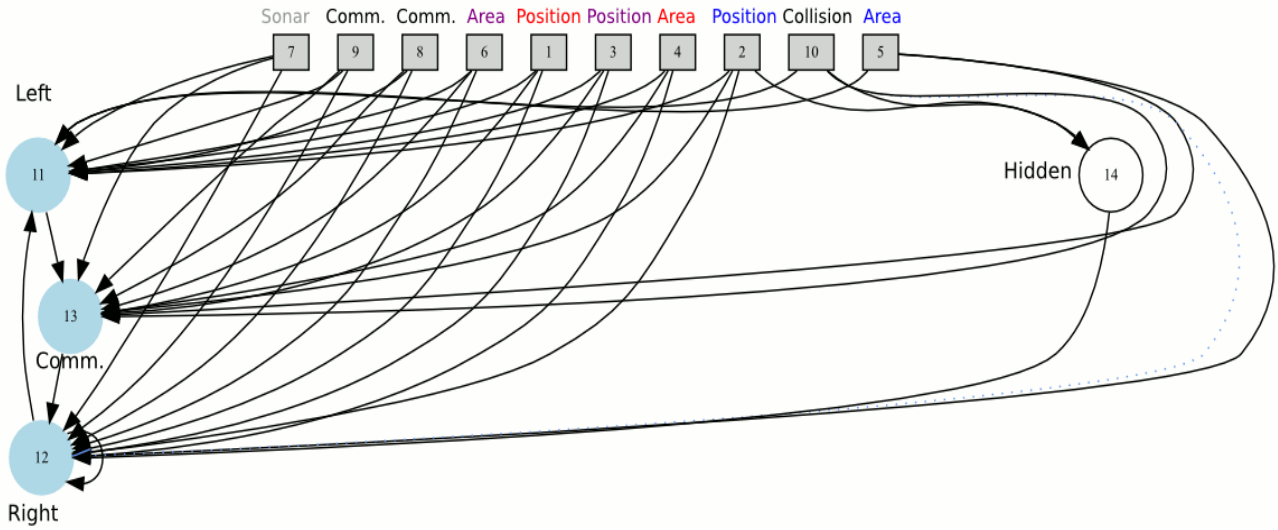

Figure 7*: This is the winning phenotype from generation 29 of 29 for the recurrent network population evolved above. As one can see, there are recurrent*

connections between the output nodes. However, it is difficult to conclude if these connections actually cause the network to obtain greater fitness.

8 Conclusions and Future Directions

It is the opinion of the author that the initial hypothesis of whether recurrence would aid in the solving of this task cannot be verified given the data collected from the experiments. If one could determine if a longer evolution of the neural networks would be helpful, it would be logical to rerun the experiments for more generations. Unfortunately, it is the conclusion of the author that this is not a reasonable option. This is because there is no clear method for determining the number of time steps the evolution would require, as it is difficult to determine the size and dimensionality of the domain space. However, the experiments are useful because they demonstrate some of the difficulties with neural evolution and genetic algorithms. Specifically, there was no way for the author to know how many generations he would have to evolve the population for. It is quite possible that NEAT would evolve recurrent neural networks to accomplish the task in much shorter time frame than if it were to evolve simple feed-forward networks. Thus in the opinion of the author, it is quite conceivable that had it been plausible to evolve the populations of neural networks over many more generations, there would have been data supporting the hypothesis. However, due to the computational cost of running robots in simulation using Pyrobot, it was only feasible to run populations of 100 individuals for 100 generations of 100 time steps. These problems could have been resolved by using a simulator written in a compiled language (faster) or a simulator that would not require explicit rendering of the simulation. The result would be that it would have been plausible to run the evolution for many more time steps and with a larger population. Yet due to the rapid prototyping nature of Python and the Pyrobot simulator, it was an obvious first step to use this set of tools, even with their associated costs.

Another possible problem could have been with relation to how fitness is awarded. In particular, the design choice to stop the current run when a robot hits a puck out of order could have made the task much more difficult to learn. Robots that go for the approach of colliding with blue and red under this scheme are severely penalized, and therefore robots that only pursue the color red have a competitive advantage from an early stage. The result was that after a few generations of evolution, robots were observed to avoid blue at all costs even after all red pucks had been removed through collisions. Therefore it is the opinion of the author that the task would have been more doable if the robots gained fitness for colliding with pucks in the right order but were allowed to continue if they made a mistake. This would have hopefully created individuals that could have consistently solved large portions of the task if not complete it in its entirety.

In the future, the author looks to attempting to implement these two changes (changing the fitness function and perhaps then the platform) and verify that more definitive results are indeed obtained. Although the experimenter could not explicitly answer the initial question, this experiment illustrates one of the major difficulties of developmental robotics: namely that current algorithms frequently fail to yield anticipated results. Consequently, it is the view of the author that this experiment and other failed experiments like it are important because they drive further research into the improvement of the fundamental algorithms which aim to create artificial intelligent systems. Further scientific inquiry into this discipline is bound to produce more efficient and effective forms of neural evolution. The more the better.

References

[1] Douglas, Rodney J., and Kevan A.C. Martin. "Recurrent neuronal circuits in the neocortex." *Current Biology* 17.13 (2007): 495-500. Web.

[2] Mikkulainen, Risto, and Kenneth O. Stanley. "Competitive Coevolution through Evolutionary Complexification." *Journal of Artificial Intelligence*. 21. (2004): 63 - 100. Print.

[3] Nolfi, Stefano, and Dario Floreano. "Co-evolving predator and prey robots: Do 'arms races' arise in artificial evolution?." *Artificial Life* 4.4 (1998): 311 – 335.

[4] Quinn, Matt. "A comparison of approaches to the evolution of homogeneous multi-robot teams." *Evolutionary Computation* 1. (2001): 128 - 135.

[5] Pyrobot. http://pyrorobots.org.