Robot Soccer With GNG & NEAT

Kwame Osei
Department of Computer Science
500 College Avenue, Swarthmore College, PA 19081
kosei1@swarthmore.edu

Abstract

Complexification is an essential concept in evolutionary robotics that develops dynamic neural networks over generations with the aim of increasing their capabilities. Artificial networks networks are complexified by altering, adding and removing nodes and connections from the neural network structure. The dimensionality of the input layer in a neural network also plays a significant role in the development of intelligent behavior in robots. Reducing dimensionality is proven to increase the speed of evolution by categorizing the robots environment. Our aim is to evolve a robot that can perform basic soccer skills by reducing the dimensionality of its input vector through categorization of its environment, and gradually complexifying the neural network using a fitness function. In phase one of our experiment, we use a low-dimensional Growing Neural Gas (GNG) to categorize the robots environment into nodes that represent high-dimensional sensory inputs. In phase two, NeuroEvolution of Augmenting Topologies (NEAT) is used to evolve an intelligent brain by complexifying the neural network over evolutionary generations based on a specified fitness function. Our results showed significant increases in average and best fitness over multiple generations of evolution and supported the viability of a combination GNG and NEAT for evolving intelligent robots.

Introduction

In many sports, the current objective can change drastically due to a small change in game conditions. Whether a team defends its goal or attacks its opponent's largely depends on the team the player with the ball belongs to. Objectives in sports are also carried out in a particular sequence often times and a disruption of the order usually requires starting all over. Teams cannot score without getting the ball first and transferring it to a target area first, and once lost a team must retrieve the ball before moving onto a higher objective.

The universality of soccer, and the soccer player's reliance on game conditions in determining the current objective and appropriate behaviors, has made it a popular medium for artificial intelligence research (Ribeiro 2005). The best example of soccer AI is probably found in Electronic Arts' FIFA Soccer, the most popular sports video game franchise in the world. Each annual instalment in the franchise is critiqued primarily for the AI the computer exhibits and the latest, FIFA 2010, is widely considered the best sports game of all-time because of its superior AI.

It is not surprising then that soccer is fairly popular in the study of evolutionary and developmental robotics (Ribeiro 2005). It also an interesting platform because soccer is the most widely played sport by kids in the U.S. and abroad. Young athletes gain basic decision-making skills and an understanding of game objectives from soccer before moving on to other sports such as basketball and American football, where they still apply. Therefore we decided to incorporate soccer into our study of the development of robots that make decisions based on current game conditions, including the current objective.

More specifically our experiment focuses on developing a robot to perform very basic soccer tasks and adapt successfully to various environments. The procedure can be divided into two phases. In phase one we use a hard-coded brain to explore the robot's environment, allowing it to experience various game conditions and categorize them. This phase is carried out using a Growing Neural Gas (GNG). *Categorization* reduces dimensionality by creating a single node for each category of similar game conditions experienced by the robot as a multidimensional vector. We specify the desired sensory inputs that the robot keeps track of and the GNG determines the appropriate categorization of each input vector (Fritzke 1995).

In phase two we use the GNG nodes as input nodes in a neural network that is modified by the NeuroEvolution of Augmenting Topologies (NEAT) using a *complexification* algorithm. With the *complexification* algorithm and a fitness function, NEAT is able to develop complex network topologies that are capable of solving difficult tasks.

In this paper, we discuss our experiment in detail including our experimental setup, the categorization process, the complexification algorithm and our fitness reward system. Finally we present our results along with an analysis and possible ideas for future research based on this experiment. The robot's task is to retrieve a ball and approach a goal area with the ball while avoiding the goalkeeper. The goal is to develop a robot that can not only find the ball and goal but also understand the sequence of objectives depending on the state of the ball. We expect that by using GNG to categorize the environment and reduce dimensionality, and NEAT to develop complex neural networks, we can evolve adaptive robots that successfully perform this task.

Related Works

Evolutionary robotics has many advantages over human engineering and for this reason the field is growing as a research area in academia and corporate settings (Zagal & Ruiz-Del-Solar, 2007). The main advantage of evolutionary robotics over human engineering is its focus on developing more adaptive robots that can complete tasks in more dynamic environments.

The field is heavily influenced by more traditional fields such as neurobiology, genetics, and cognitive science. For instance artificial neural networks are modelled after neural systems in living organisms (Shultz 2003). From genetics the idea of a genetic propagation in offspring, mutation and crossing over have been borrowed and applied to the NeuroEvolution of Augmenting Topologies (NEAT). NEAT starts with a basic neural network with three layers - an input layer, a hidden layer and an output layer - and adds nodes and connections to the network ("Neat-python, A NEAT (NeuroEvolution of Augmenting Topologies) implementation in Python"). Unlike other alternatives, by doing so NEAT reduces initial dimensionality of the network. NEAT's complexification algorithm is capable of building on this and developing highly sophisticated neural networks that resemble neural networks in living organisms and can contain millions of interconnected neurons (Stanley Miikulainen, 2004).

Stanley and Miikulainen's experiment proved the viability of complexified neural networks for evolving intelligent, adaptive robots. They evolved robots to play Robot Duel, a game where two opposing robots must eat food to gain energy and strike their opponent once they have more energy. The robots in the experiment develop various unorthodox means on overcoming their opponents. It also showed that human bias misses out on solutions to tasks that only evolution can develop (Stanley & Miikulainen, 2004).

Evolutionary robotics and even NEAT have been previously used in experiments involving soccer. In Thiha's experiment a robot was developed to carry a ball past a defender and kick it past a goalie to score using NEAT. Thiha used a progressive fitness function that rewarded subtasks such as seeing the goal and the ball. It also reward additional fitness if the ball is seen straight ahead. The experiment's success proves that continous fitness functions are more effective than disjunct ones (Thiha 2009). It differs from our experiment in the complexity of the overall task and its use of two hard-coded robots.

Our project is inspired by our midterm project carried out previously in Professor Meeden's Adaptive Robotics course at Swarthmore College in Spring 2010. In that experiment we used NeuroEvolution of Augmenting Topologies (NEAT) to develop complex neural networks through evolution that could play defense or offense depending on which robot had possession of the ball. We did not develop a robot that completed the task satisfactorily and the best robot developed simply spun around in circles. This experiment taught us the importance of the allocation of fitness to developing intelligent behavior.

Experiments

We used a combination of GNG, NEAT and Pyrobot Simulator for our experiment. The world in Pyrobot resembled one-half of a soccer field and contained the evolving robot (red), a goalie robot (green), a goal represented by a red wall, and a ball represented by a blue puck (See Figure 1). Both of the robots used were Pioneer robots. The evolving robot was equipped with Camera and Gripper devices. The goalie robot was stationary in the center of goal.

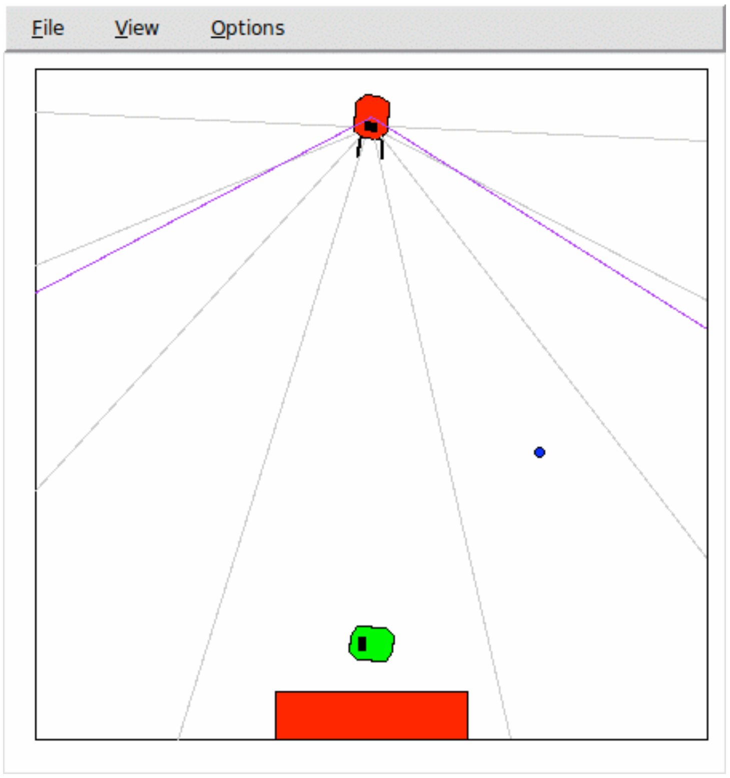

Figure 1. The Soccer Field world designed for our experiments. The world included 2 Pioneer robots, a ball (blue puck), and a goal (red wall).

Phase One: Categorization

We used GNG and a hard-coded brain to explore and categorize the robot's environment. The hard coded brain was set up to categorize the world in two stages: exploration and playing. During exploration the robot tries to cover as much ground in the environment and experience it from various angles. The aim is to expose the robot to as many game conditions as possible, making it highly adaptive. These categories are more general and broader and represent situations where only very general behaviors will be required later. In the next stage it focuses on the ball and tries to retrieve it and score. This stage targets exposure to game conditions it will most likely need to complete the task. By dedicating time to playing with the ball the robot creates more specific categories for conditions that may require more precise behaviors later. Categorization runs for a total of 1000 steps - the robot explores for 500 steps and plays for 500 steps.

The ball's position is reset every 50 steps and the robot is returned to the start position every 100 steps. This was done to create a more dynamic environment that requires adaptive robots (adapting to the ball's changing start position). However the robot always starts in the same position facing the ball, the goal and the goalie. The ball always starts in view and in the same vertical position but it can be anywhere in the horizontal plane.

The GNG received 7 input variables including inputs for the positions of the ball, goal and goalie, the distances to the ball, goal and goalie, and a boolean hasBall that started at 0.0 and switched to 1.0 when the ball was retrieved. All inputs are scaled between 0.0 and 1.0. The camera was used to determine all the variables except hasBall. The positions of objects relative to the robot were determined by their position in the robots camera. The left, right and center positions corresponded to inputs 0.0, 1.0 and 0.5. The distances to objects relative to the robot were calculated based on their size in the camera view (ie. objects got bigger as they got closer). The equation used to determine the distance to objects was:

```
Distance to object = object's height in camera view / camera height
```

hasBall was changed whenever the ball was between the robot grippers. The state of the grippers was determined using the `getBreakBeam('inner')` method. If the ball was between the grippers the robot stored the ball and closed its grippers, and the value of hasBall became 1.0. In order to keep the task simple we avoided letting the robot learn what to do when the ball was in its gripper.

We used an error threshold of 0.5 for our experiment. At the end of categorization the GNG made a file containing vectors corresponding to the categories it created.

Phase two: Evolution

In phase two we used individual GNG nodes as input nodes in a neural network and complexified the network in NEAT using a fitness function. Instead of plugging the 7 input values directly into the neural network, we compared the robot's current input vector with each GNG node and found the first, second and third closest matches to it. Next, the node corresponding to the first closest GNG node is given an activation of 1.00, the second, 0.66, and the third, 0.33.

The initial neural network that NEAT worked with contained 14 input nodes, one for every category made by GNG, two output nodes, and no hidden nodes. The probability of NEAT adding connections was set to 0.05 and the probability of adding new nodes was set to 0.03 as well. We used "tanh" as the neural network activation because it returns values from -1 to 1, the desired range for the robot's outputs. The two output nodes corresponded to translation speed and rotation angle. The specie size was set to 10 and the survival threshold was 0.2.

We ran the evolution for 50 generations. The population size was set to 20 and the maximum fitness threshold was 10000. The threshold was set to be astronoically high because we wanted to run the experiment for all 50 generations. Each individual in the population had 3 trials lasting 300 steps. We ended a trial if the robot stalled and this significantly cut down the time required to evaluate individuals. The ball began at the same vertical position at the start of each trial but its horizontal position was random. The robot always began at the top of the world facing down so it could initially see all other objects no matter where the ball was placed.

The fitness function used in the experiment was progressive and awarded fitness for performing subtasks that led to completion of the overall task (See Figure 2a). The robot received continuous fitness for getting progressively closer to the ball, seeing it head on and retrieving it. It lost fitness for not seeing the ball when it should be looking for it. After storing the ball the robot received fitness for seeing the goal, for getting closer to the goal and for touching it. It lost fitness for approaching the goalie head on. The robot also received additional fitness for moving forward at all times. We implemented this to encourage robots not to drive backwards, which was common in earlier runs and prevented the ball from entering the grippers.

```
if ball is not seen and not hasBall:
  ballScore -= 1
if ball is seen in center:
  ballScore += 1
if ballDistance < 0.5:
  ballScore += 1
if ballDistance <0.75:
  ballScore += 0.5
if ball is in gripper:
  store ball
  ballScore += 50
if hasBall and goal is seen:
  goalScore += 0.5
if hasBall and goal is seen in center:
  goalScore += 1
if hasBall and goalDistance < 0.5:
  goalScore += 1
if hasBall and goalDistance < 0.75:
  goalScore += 0.5
if hasBall and goalDistance < 0.005:
  goalScore += 500
  end trial
if hasBall and goalie in center:
  goalieScore -= 0.5
if robot is moving forward:
  movingScore += 0.5
score = score + ballScore + goalScore + goalieScore + movingScore
```

Figure 2. Fitness function. Fitness was awarded progressively for completing subtasks that lead to the completion of the overall task.

Results

The results from the experiment were fairly encouraging. Our GNG categorize the environment into 14 unique categories.

The average fitness increased significantly over the 50 generations but the highest fitness attained in generation 50 was not as high as we expected. Performance increased quickly early on and can be attributed to early low-dimensionality in NEAT. Over the duration of evolution, 7 species were created and 3 died off (See Figure 3). The initial specie died off around generation 26. The second specie was created around generation 6 and died off in generation 40. The third was created in generation 39 and died off in generation 46. Three more species were created around generation 39 and one last specie was created in generation 45. Speciation became more frequent in later generations, specifically around generation 39.

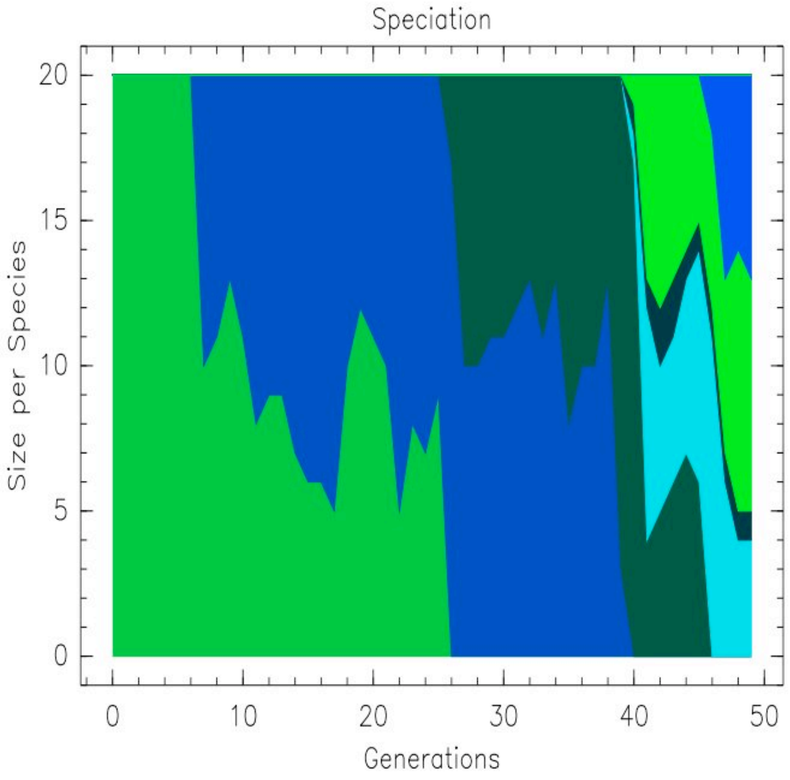

Figure 3. Speciation number and size. Seven species in total existed and three died off.

The best fitness fluctuated wildly throughout the experiment but had a high fitness period between generations 20 and 33. The best fitness in generation 1 was about 400 and the best fitness in generation 50 was 877, more than double that (See Figure 4). Although best fitness robot in generation 50 had a fitness of 877, when the same brain was used in further experiments it could not complete the task and only attained a fitness of 470. The overall highest fitness was about 1450 and happened in generation 27. Average fitness increased overall but not significantly. During the high fitness period the average reached its highest point of about 450. The average then fell drastically around generation 33. The average in the first generation was about 150 and the final average in generation 50 was about 300. Although the average more than doubled it would have been higher if evolution had successfully developed multiple high fitness individuals every generation. There was usually only one or two high fitness individuals in each generation. These were usually the robots that successfully retrieved the ball.

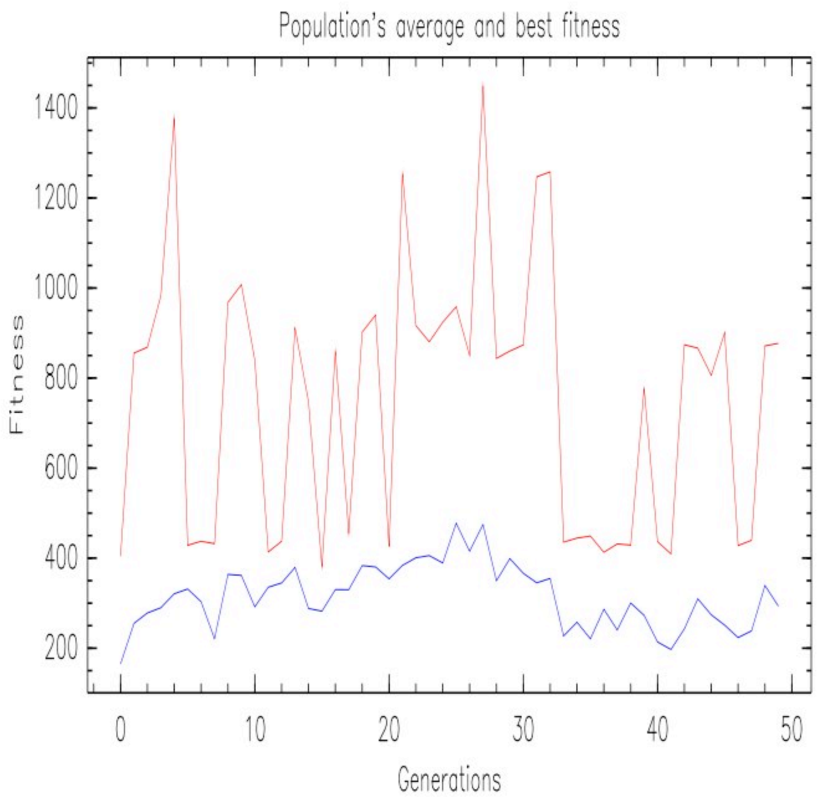

Figure 4. Best and Average fitness curves for each generation. The average fitness increased significantly but did not increase consistently.

A lot of the robots showed very jerky movements and often made reasonable decisions but lacked accuracy. For example a lot of robots successfully approached the ball but failed to turn the right amount to get it between their grippers. Other robots approached it but not fast enough because to went one step back for every two steps forward. The robot's also did not learn to move fast when they are farther from the ball and slow down when they were close to it. They went pretty slow at all times but sometimes not slow enough to not bump the ball.

We were impressed by the fact that spinning was rarely seen in the robots. No robots that spun in one place were observed but some did drive around in circles. Most robots also seemed to realize it was important to keep object in the center of the view because they turned in one direction until the object was at the edge of the center and then turned in the other direction.

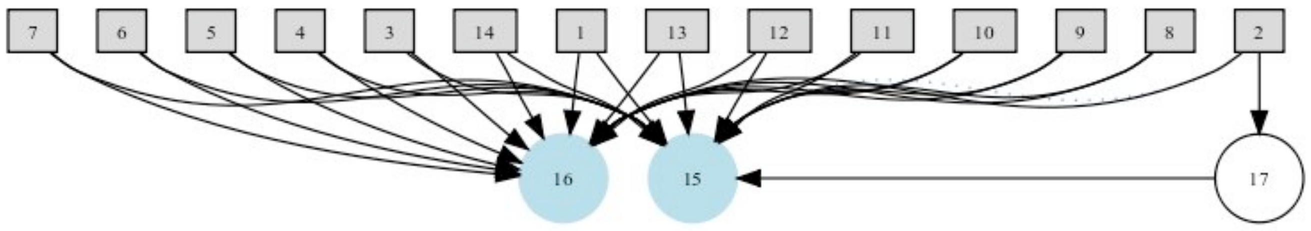

Figure 5. Phenotype of the best chromosome. This was the best chromosome after 50 generations of evolution and included one hidden node and 29 connections.

The best chromosome after 50 generations of evolution is show in Figure 5. It had 29 connections and 1 hidden node. The neural network was much smaller than we expected. This can be due to the small probability of adding nodes or connections used or a trait just particular to this specie.

Discussion

Our experiment was fairly successful and proved the viability of GNG and NEAT to evolve intelligent robots. The results showed that GNG and NEAT were able to increase fitness significantly during evolution but the considering the simplicity of the task and the length of the evolution we expected better results. The average fitness did not increase significantly from generation to generation even though it doubled over the entire evolution. We attributed the lack of consistent results to the the fitness function and identified areas that could be tweaked to improve performance. For instance we thought the ballScore was too heavily weighted compared to other fitness subscores and the movingScore was too small. This was caused by excessive rewards for approaching the ball and seeing it and insufficient reward for moving forward. The fact that robots still drove backwards after more than 30 generations showed that the movingScore was not playing a significant role.

In further experiments we adjusted our fitness function to the one in Figure 6 below and were able to get significantly better results. In subsequent runs about 4 robots in each generation consistently found the ball by the 20th generation compared to only one or two after 50 generations. We achieved this by reducing the weight given to ballScore and goalScore and increasing that of movingScore and goalieScore. The reward for retrieving the ball was reduced so that the fitness level was not too different between a robot that stored the ball and one that came close but failed.

```
if ball is not seen and not hasBall:
  ballScore -= 1
if ball is seen in center:
  ballScore += 1
if ballDistance < 0.30:
  ballScore += 1
if ballDistance <0.60:
  ballScore += 0.5
if ball is in gripper:
  store ball
  ballScore += 50
if hasBall and goal is seen:
  goalScore += 0.5
if hasBall and goal is seen in center:
  goalScore += 1
if hasBall and goalDistance < 0.30:
  goalScore += 1
if hasBall and goalDistance < 0.60:
  goalScore += 0.5
if hasBall and goalDistance < 0.010:
  goalScore += 150
  end trial
if hasBall and goalie in center:
  goalieScore -= 2
if robot is moving forward:
  movingScore += 1
score = score + ballScore + goalScore + goalieScore + movingScore
```

Figure 6. The Adjusted Fitness function. BallScore was slightly deemphasized and movingScore was given significantly more weight.

Another suspected cause of the inconsistencies is the categorization process. It was very difficult to get the robot to adequately explore its environment and we believe certain categories can be improved upon. This would improve accuracy of the robot's behavior and allow it to improve on its performance in task that require accuracy such as trapping the ball. Our categorization is most likely responsible for the jerky movements as the robot moved from one category to a very different one because no in-between category had been developed. We would like to cap the size of the GNG to restrict how many categories it can make and instead focus on perfecting the definition of each category. We could also remove the exploration phase of the categorization because it creates nodes that the robot doesn't use in the evolutionary task. Also robots that put themselves in extremely rare conditions can just receive bad fitness and get eliminated so there is really no need to categorize useless conditions. In a sense, at the risk of human biases, we would select for robots that find the simplest, straight-forward solution to the task.

We we're impressed with certain low-level behaviors the robots exhibited. The robots showed that they had learnt to keep targets in the center of their camera view because the turned in one direction until the object was almost out of the center region and then reversed their rotation direction. This was a big improvement upon our midterm experiment where robots tended to spin in circles. As previously mentioned, we awarded excessive fitness for tracking the ball and this may be responsible for this achievement.

In future experiments we would like to compare the performance of our combination of GNG and NEAT to other similar evolution techniques and settings. We would like to compare it to the performance of NEAT on its own. We would also like to compare various fitness functions. As discussed previously, we believe our fitness function is to blame for some awkward persistent behaviors. We hope that by comparing different fitness functions we can isolate the specific causes for these behaviors. We also realized quite late how important the categorization phase is on its own. Therefore we would be interested in improving on this phase and conducting a deeper analysis of it. We are also very interested in trying this experiment in a real-world setting with physical robots. We expect this to present new challenges and complication to overcome.

Although we did not develop robots that could perform the task repeatedly our results did show that GNG and NEAT can be used to evolve soccer skills in robots. They also showed that the robots could understand a change in game objectives once they had retrieved the ball. A high percentage of robots that got the ball were also able to reach the goal. Our further experiments showed that we could improve upon our model by isolating small changes and observing the resulting evolutionary effect. We have many ideas for improving on our model and look forward to testing out some of our stated hypothesis on what went wrong.

Acknowledgements

We would like to acknowledge Professor Meeden for helping us develop our idea for the experiment, providing the modified version of NEAT, and making us aware of the similar experiment Phyo Thiha conducted as part of the same class the previous year. We also thank our classmates in Professor Meeden's Adaptive Robotics seminar for giving us feedback at various checkpoints throughout the experiment.

References

Fritzke, Bernd (1995). A growing neural gas learns topologies. *Advances in Neural Information Processing Systems,* vol. 7, 1995.

Neat-python, A NEAT (NeuroEvolution of Augmenting Topologies) implementation in Python. Google^{TM} Code 2008 November. World Wide Web, URL is http://code.google.com/p/neat-python/.

Ribeiro, Fernando (2005). RoboCup – The evolution of a Robotic Scientific Challenge.

Shultz, T.R. (2003). Introduction and A neural network primer. *Computational DevelopmentalPsychology*, 2003.

Stanley, K.O. & R. Miikkulainen (2004). Competitive Coevolution through Evolutionary Complexification. *Journal of Artificial Intelligence Research,* vol. 21, 2004.

Thiha, P., Extending Robot Soccer Using Neat. In *CS81 Final Projects*, 2009. World Wide Web, URL is http://www.cs.swarthmore.edu/~meeden/cs81/s09/finals/Phyo.pdf

Zagal, J.C. & J. Ruiz-Del-Solar (2007). Combining Simulation and Reality in Evolutionary Robotics. *Journal of Intelligent and Robotic Systems,* vol. 50, 2007.