Extending Robot Soccer Using NEAT

Phyo Thiha
Department of Computer Science
500 College Ave. Swarthmore College, PA 19081
pthiha1@swarthmore.edu

Abstract

Robot soccer is an active research area that offers a challenging research domain to investigate a large spectrum of issues relevant to the development of complete autonomous agents. We used a two- and three-agent robotic soccer simulation to study a coevolutionary learning technique: NeuroEvolution of Augmenting Topologies (NEAT). Our aim is to figure out a good fitness function for the robot soccer player that can learn how to carry the ball, avoid the opponent (the defensive and the goalie) and kick it towards the goal. Through experimentation, we achieve two fitness functions that can be used for both the player and the defender robot. By using these fitness functions, NEAT autonomously evolved suitable neural network architectures for the robots to accomplish the task most of the time, proving that the fitness functions has roughly captured the learning process.

1. Introduction

Soccer is a rich domain for the study of learning issues. Teams of players must work together in order to put the ball in the opposing goal while at the same time defending their own. Learning is essential in this task since the dynamics of the system can change as the opponents' behaviors change. The players must be able to adapt to new situations.

Not only must the players learn to adapt to the behaviors of different opponents, but they must learn to work together. Soon after beginning to play, novice soccer players learn that they cannot do everything on their own: they must work as a team in order to win. However, before players can learn any collaborative or adversarial techniques, they must first acquire some low-level skills that allow them to manipulate the ball. These low-level skills—for example, dribbling and controlling the ball—though entirely individual in nature, are as critical to the development of a soccer player as others such as passing and receiving passes, which are more collaborative in nature.

Several research studies have been done to develop robots that can play soccer [RoboCup Research & Education]. However, because of the rich and dynamic nature of the environment in soccer, directly programming the robots to play soccer is impractical, if not impossible. Therefore, much of the focus has been shifted to treat robots as intelligent agents that can learn how to play soccer under different environments and conditions.

Our approach is based on NeuroEvolution of Augmenting Topologies [Stanley & Miikulainen, 2002], which trains the robot and develops increasingly complex neural networks using evolutionary learning technique. We identify appropriate input parameters and fitness function for the neural networks to be employed as the brain of the robots—the player and the defender. Next, we observe the learning process of the robots and modify the inputs and fitness function as necessary, to help one of them learn to carry and "kick" a soccer ball to the goal while avoiding the opponent—which also is run on the same brain but doing the opposite task of blocking the player robot to reach the goal. Our aim is to create a

full team of agents that use learned behaviors, at several different levels, to reason strategically and play an entertaining soccer match.

In this paper, we present detailed experimental results of our successful use of autonomously-evolved neural networks for learning a low-level behavior. This learned behavior of carrying and kicking a ball towards the goal is a crucial step in the direction of developing soccer robots that could perform more sophisticated strategies. We illustrate how evolving the robot's neural architecture using a relatively simple fitness function can achieve a somewhat sophisticated behavior in both the player and the defender robots.

2. Related Work

Evolutionary robotics is a technique for the automatic creation of autonomous robots. Inspired by the natural evolution principle "Survival of the Fittest", it views robots as autonomous artificial organisms that develop their own skills in close interaction with the environment and without human intervention.

In evolutionary robotics, an initial population of artificial chromosomes, each encoding the control system of a robot, is randomly created and put into the environment. Each robot is then free to act—move, look around, reach etc.—according to its genetically-specified controller while its performance on various tasks is evaluated by a fitness function, usually defined by the human engineer. The fittest robots then reproduce by swapping parts of their genetic material with small random mutations. The process is repeated until the birth of a robot that satisfies the performance criteria.

One of the evolutionary techniques employed in robotics is NeuroEvolution of Augmenting Topologies (NEAT)—developed by Stanley and Mikkulainen—which evolves network weights and structure, attempting to strike a balance between the fitness and diversity of evolved neural architecture of the robot. NEAT is based on three key ideas: tracking genes with historical markings to allow crossover between different topologies, protecting innovation via speciation, and building topology incrementally from an initial, minimal structure.

In particular, NEAT builds neural networks using a novel evolutionary algorithm, *complexification,* which starts with a small set of nodes and add more nodes as well as connections throughout the evolution to come up with more sophisticated solutions to the problem. NEAT, unlike other evolutionary algorithms, does not erase the architecture of the neural network each generation but rather builds up on existing strategy by adding new structures without dramatically changing the existing representation. Therefore, even if an optimum solution to a problem is reached, new nodes can be added to open up higher dimensional space where better solutions may exist. One of Stanley's experiments with NEAT includes simulation of robot duel, where robot controllers compete for a limited resource in a given environment for their survival. The experiment shows that robots exhibit sophisticated strategies that resulted from complexification of their genomes throughout the evolution process. Moreover, on simple control tasks, it was shown that NEAT algorithm often converges to effective networks more quickly than a variety of other state-of-the-art neuro-evolutionary techniques [Stanley et al., 2002d and Talyor et al., 2006].

Encouraged by NEAT's success in evolutionary learning, we employed it on a robot soccer player in a simulated environment for our midterm project [Thiha, 2009]. The goal was to evolve a brain that will drive the robot to carry the ball towards a colored wall, which is designated as the goal. In our midterm experiments, unlike Stanley's duel robots, we did not have a competitor for our soccer robot so there was no coevolution, and the robot rather evolved alone using the fitness function defined by us. In contrast, we decided to increase the difficult level of the learning by the player robot by having a goalkeeper robot in front of the goal as a distraction and have the player coevolve with the defender to perform different goals for their survival.

In fact, NEAT has been applied to keepaway task in robot soccer (Whiteson et al., 2007). In keepaway, one team of agents, the keepers, attempts to maintain possession of the ball while the other team, the takers, tries to get it, all within a fixed region; the keepers try to complete as many passes as possible while preventing the ball from going out of bounds and the taker from touching it. In their paper, Whiteson et al. presented three studies that compare, contrast, and combine evolutionary and temporal difference (TD) methods for controlling robot actions with reinforcement learning. The first of their studies was related to soccer and NEAT, in which NEAT was applied to 3 vs. 2 keepaway task and compared against the performance—which is measured by the mean ball hold time of a 3-robot team—of a TD approach. In particular, they trained NEAT with initial population of 100 neural networks, each with 13 inputs, one for each state feature, and three outputs, one for each action such as passing and receiving the ball. The experiment demonstrated that NEAT can outperform TD method—namely, Sarsa (Sutton & Barto, 1998)—in difficult tasks, although NEAT was found to be more computationally intensive. Moreover, they also observed that NEAT's performance improved more when noise in robot actuators was removed; in other words, they found that NEAT learns much quickly and performs much better if the task is more deterministic.

While our experiment and Whiteson's first experiment share a common testbed—a robot soccer system—our goal is not to compare, combine and improve the machine learning techniques; instead, we are curious about the relation between the fitness function and the learning process, and the qualitative performance of NEAT. Another minute difference is that although our environment is relatively simple with at most three soccer robots in the environment, we do not eliminate the noise in our sensor, the camera. However, both Whiteson and we believe that evolutionary methods can optimize the learning and can find the best solution in the search space.

A similar work to ours has been done in the past [Ciesielski & Lai, 2001]. Their experiments were also conducted in a simulation environment but with one robot and a ball. The aim of their work was to learn a behavior in which a player, which is located to be able to see the ball, should go to the ball, dribble it towards the goal and score a goal—which is a lot similar to what we did in our midterm project. In their experiments, they also used composite fitness function which is different from a goal-only fitness function, where the number of goals scored in a fixed time period or the time taken to score a goal is used as the fitness measure. On the other hand, in composite fitness function, important features such as being near the ball, kicking the ball, being near the goal and kicking a goal are combined into one overall measure of fitness. The difference between our work and Ciesielski's is that they used a fixed neural network architecture—6 inputs, 3 hidden and 4 output nodes—which they obtained

through experimentation whereas we constructed neural network using NEAT, which dynamically adjust and build the appropriate architecture during the training process. Ciesielski et al. concluded that although their simulation robot show some rudimentary dribbling and scoring skills, the overall training period is long—and therefore, discouraging—and as a result, cannot be used for online soccer game where computation has to be done instantaneously; they even suggested finally that using goal-only fitness function could bode better. Although we did not aim to implement dribbling behavior in our robot, we found that composite function is enough to achieve the detecting, carrying, kicking and scoring behavior in the player robot.

3. Experiments

As in our midterm project, we used Python Robotics simulator (Pyrobot Simulator) [Blank et al., 2003] as our platform for experiments. We increased the size of the world in the experiments to make room for three robots. We used the red-colored puck as the soccer ball which is initially located at a fixed location in each experiment run.

The source code for NEAT was obtained from Google™ Code (Neat-python, 2008) and Professor Meeden modified it so that it can be run on the background of the Pyrobot simulator. In all of our experiments, we left most of the configurations of the NEAT intact. For example, several activation functions are available in NEAT but we decided to use "`tanh`" because it returns the value ranges between ± 1 and therefore, can be directly used as motor values. The maximum fitness threshold and population size are set to '100,000' and '30' respectively for all the experiments whereas the number of nodes varies for some experiments. Initially, the number of input node is set to 7 with no hidden nodes and 3 output nodes.

Our experiments can be divided into two major phases. In the first phase, we have two robots, one of them designated as a goalie and a player with the wall behind the goalie colored in blue as a goal (see figure 1 a). The purpose of the first phase is to see if our player robot can carry the ball, avoid the goalie and finally kick the ball towards the goal. We employed "Pioneer" robots attached with a camera and a gripper as both the goalie and the player. The gripper is supposed to act as substitute for the feet—to contain the soccer ball while carrying or in the case of the goalie, to contain the ball. The ball is always positioned in the middle of the world, and the robot is randomly positioned behind the ball with its head randomly oriented, but within some restrained degree, toward the ball[1]. This initial configuration of the player robot and the ball is necessary to make sure that the player—and therefore, NEAT—can easily see the ball and learn to detect the ball quickly.

In the case of the goalie, we did three different types of experiments with its initial configuration as well as how it moves. The first is a simple static position where the goalie is located at random location in front of the goal with random orientation just to keep the player distracted from the goal. The second is to move the goalie along the vertical line (see figure 1 b) in front of the goal. Next, is the most

1 Sample figures of the possible initial locations and orientation of the ball and the player robots can be seen in our midterm project paper [Thiha, 2009].

sophisticated of all where the goalie will look out for the ball once it enters the goal box area; if it passes the edge of the goal box, the goalie is repositioned back to the front of the goal. We hand coded the goalie's movement to achieve these behaviors.

Intuitively, we would like our player robot to be able to detect and track the ball; therefore, we decided to obtain center coordinates of the ball seen in the camera. To see the ball and track its position, we used image processing method in Pyrobot called "`blobify`", to form a blob when the camera sees the the ball, and retrieve the center of the blob in relation to the camera's frame of coordinates. We also wanted to know if the robot is approaching the camera, which is obtained by tracking the area of the ball and see if it is getting bigger. As the robot gets closer to the ball, we would like the robot to touch the ball. We were able to tell if the robot is in contact with the ball by using the gripper sensor. To detect the ball within the gripper, we used Pyrobot gripper device's built in method namely, "`getBreakBeam('inner')`". Once the robot has the ball or can keep track of it, we would like it to carry the ball towards the goal. Thus, we also check if the goal is visible in the camera. Since it is also important to know where the goalie is located in relation to the player robot, we calculate the distance between the two. These last three pieces of information that we collect are global: the distance from the robot to the ball, the ball to the goal and that from the goalie to the robot.

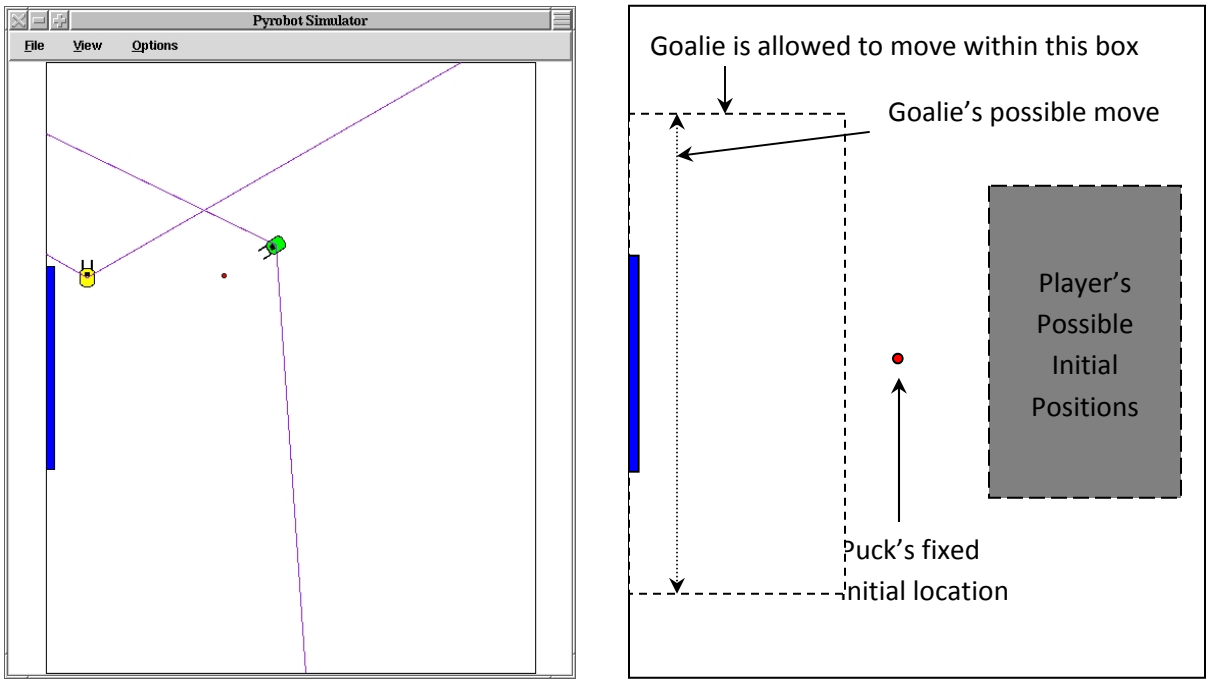
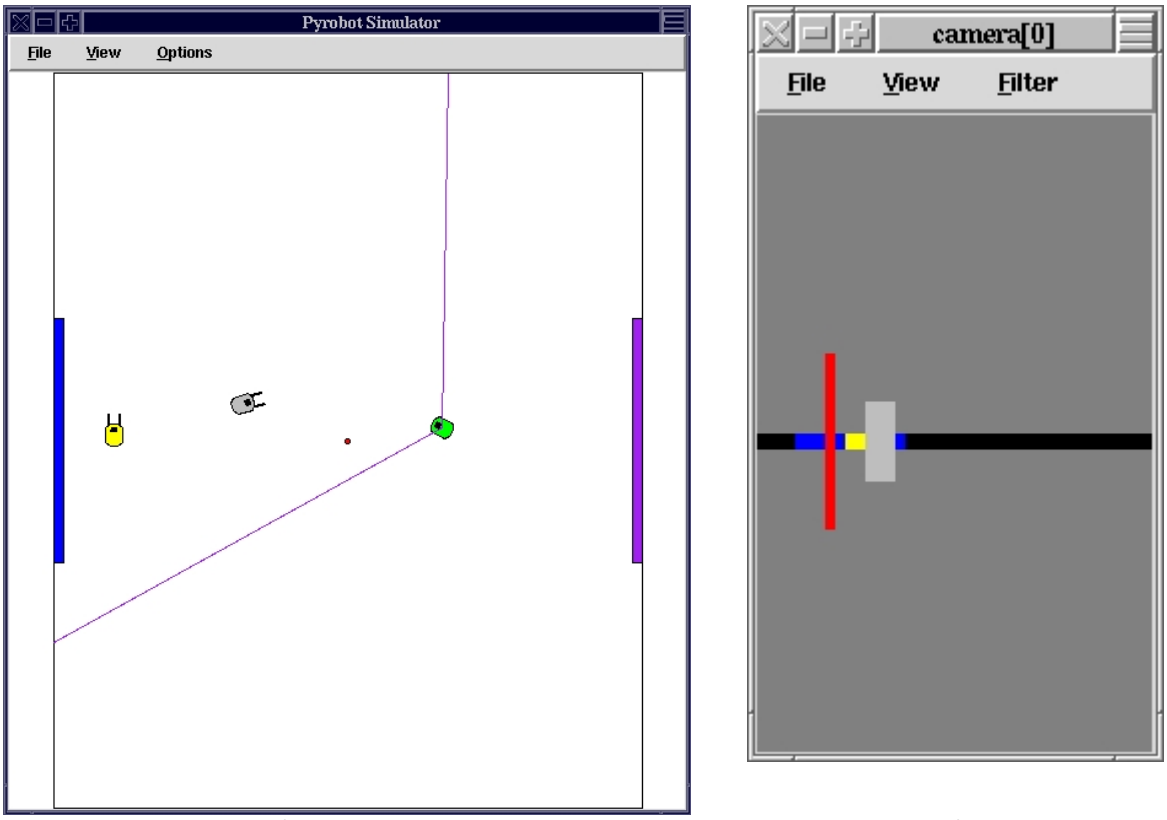


Figure 1 a) A sample world configuration for phase 1 where the player is equipped with gripper (The red "dot" in the middle is the ball, the robot's camera visibility range is in "purple" and the blue wall on the left is the "goal") **b)** Approximate drawing showing possible initial configurations of the ball, the goalie, and the player for phase 1.

Based on the experience in the midterm project, we have the number of initial input nodes up to seven in our experiments including: an input with binary value to describe if the player robot sees the goal, an input carrying the information of the distance between the ball-to-player, and four inputs that carry the information of where the ball is seen in the camera of the player (see figure 2 a and b). Only one of the four camera inputs is activated at a time depending upon where the ball is located in the camera: left, right, middle or none. The value of that input is a ratio of the field's width to the ball-to-robot distance. The location of the ball in the camera is obtained by calculating the center coordinate of the ball seen. The input node for gripper is of binary nature, indicating whether the ball is in the gripper using '0' and '1'. The input node for the distance between the ball-to-player, was necessary to keep at least one input node that has non-zero values—and therefore, feeding the neural net with at least one input. As for output, we have three, one each for the left and right motors and one for the kicking action. The kicking action—which is also hand coded—is activated once the corresponding output node is above some arbitrary threshold—in our experiments, it was '0.5'.



a) b)

Figure 2 a) A sample world configuration for phase 2 where the defender is installed (The red "dot" in the middle is the ball, the robot's camera visibility range is in "purple" and the blue and purple wall on the left and right respectively are the "goals") **b)** Camera view as seen by the player robot. All three robots are equipped with camera as their main sensor. Here, the red rectangle represent the ball's area and location seen in the player's camera, the grey being the defender, the yellow the goalie and the blue the goal.

The second phase of the experiment is more complicated. We installed another "Pioneer" robot to act as a defender which we positioned in front of the goalie, near the edge of the goal box at random initial position. The defender's behavior is not hand coded but rather controlled by a brain that we evolve with a different fitness function that we derived from our midterm project. A minor edition to the environment is a wall, colored purple, that will acts as a target to which the defender robot is trained carry the ball towards (see Figure 2 a). The rest of the implementation is the same as in phase 1 with the goalie still operating with hand coded ball-tracking-and-hunting algorithm while the player is evolving its brain to drive and kick the ball towards the goal.

The most successful fitness functions of the player and the defender robots are as shown below.

FitnessFunction_Player

```
baseScore = 1.0
score = 0.0

if ball in the middle of the camera, then
   score = baseScore * 2

else if ball on the left of the camera, then
   score = baseScore * 1

else if ball on the right of the camera, then
   score = baseScore * 1

if ball in the gripper, then #excluded if gri
   score = baseScore * 3 #-pper doesn't exist

if robot is closer to the ball, then
      score = baseScore * 3

if ball is closer to the goal, then
      score = baseScore * 6

if goal area seen > 10, then
   score = baseScore * 2

else if goal area seen > 100, then
   score = baseScore * 4
```

(a)

FitnessFunction_Defender

```
baseScore = 1.0
score = 0.0

if ball in the middle of the camera, then
   score = baseScore * 2

else if ball on the left of the camera, then
   score = baseScore * 1

else if ball on the right of the camera, then
   score = baseScore * 1

if ball is closer to robot, then
   score = baseScore * 3

if ball closer to the goal, then
   score = baseScore * 6

if ball in the gripper, then
   score = baseScore * 4

   if goal in sight, then
      score = baseScore * 6

      if robot is closer to the goal, then
         score ← baseScore * 3
```

(b)

Figure 3 Two core fitness functions used in the experiment for **a)** the player robot and **b)** the defender robot

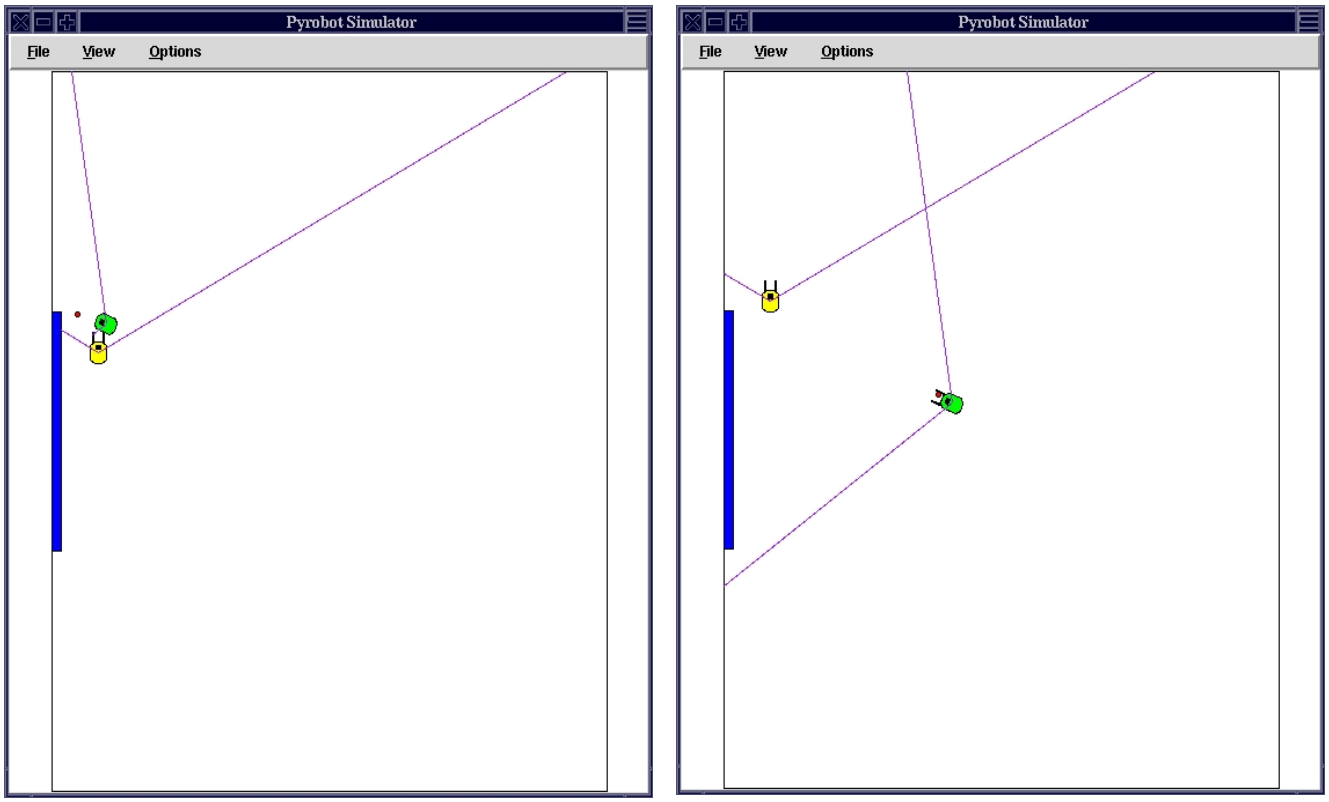
Each experiment has 30 individual chromosomes per generation, and each individual was run for 3 trials, each of them lasting 250 time steps. However, due to the steps given for the hand coded goalie's

movements, each generation took about an hour to finish. For the first phase, we did eight experiments in total to achieve the satisfactory performance from the player robot whereas in the second phase, it took as about ten experiments to see desirable improvement.

4. Results

Having had an experience with NEAT and Pyrobot in the midterm helped the speed of experimentation process. Based on our knowledge from the midterm, we knew that we need to feed non-constant values—the ratio between the field's width and the ball-to-robot distance—to the inputs of the player and the defender that has to do with tracking the ball in their cameras to make sure that the robots keeps moving and exploring their environment.

During the first phase experiments, the player was able to achieve the task of carrying the ball towards the goal almost all the time when the goalie is either static or just moving along a vertical trek in front of the goal. It's not surprising because we already accomplished carrying the ball towards the goal in our midterm project and the goalie is too naïve to provide a threat to the player's operations. Only at certain times, the goalie moving will knock the ball out from the player's view or block the player to deter it from going to the goal (see figure 4 a). But this happens with chance, and was not a strong proof that we achieved what we wanted.

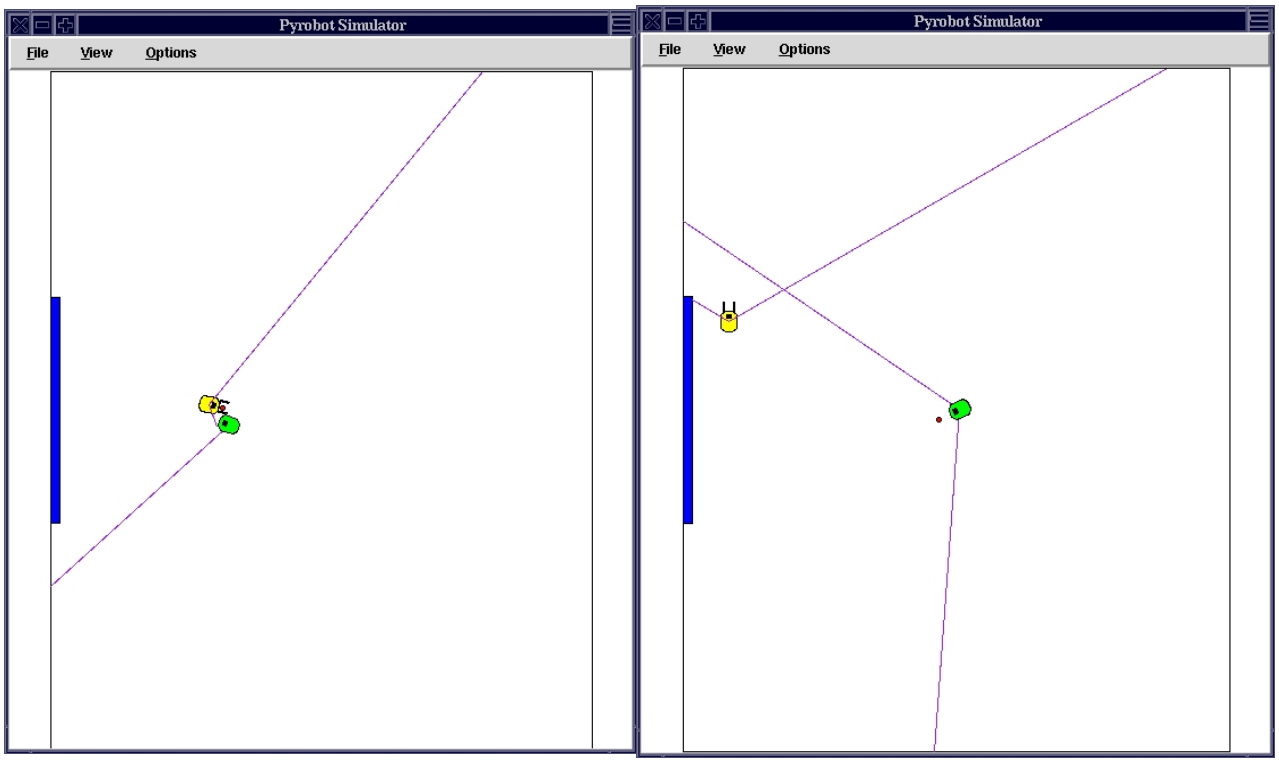


a) b)

Figure 4 a) The goalie blocking the player from reaching the goal **b)** The player getting stuck with the ball within its gripper

Besides, we found that the player robot is stalled whenever it tries to turn when the ball is within its gripper or just next to the gripper (see figure 4 b). Moreover, the robot is sometimes fixated on having the ball within its gripper more so than driving it towards the goal. Therefore, we finally decided to take the gripper off which we can afford to do now because our goal has changed from carrying the ball towards the goal to kicking it. This method worked satisfactorily but when the ball is next to the robot's body, it didn't seem to work. To work around this, we inserted a hand-coded action in each time step, which checks if the robot is blocked for a certain amount to steps and if so, back off and turn the player robot a bit to get out of the situation.

However, when the goalie is "alive"—that is, when it runs along the vertical trek when the ball is outside of the goal box, but actively hunts the ball if it enters the goal box, we find more interesting behaviors. Sometimes, the player robot would lose track of the ball and slip the ball inside the goal box, which is easily retrieved and pushed outside of the goal box by the goalie (see figure 5 a). Sometimes, the player robot would sometimes carry the ball in the box and when it cannot see the goal area fully in its camera—because the goalie is blocking part of it—the player will turn its direction a bit to stay away from the goalie; then depending on the amount of direction it turns, the player may or may not successfully score the goal.



a) b)

Figure 5 a) The goalie reaching the ball before the player **b)** The ball is not inside the goal box, and therefore, the goalie is not activated and if the player carries it inside the goal box and kicks the ball, it has a lot of potential to create a goal

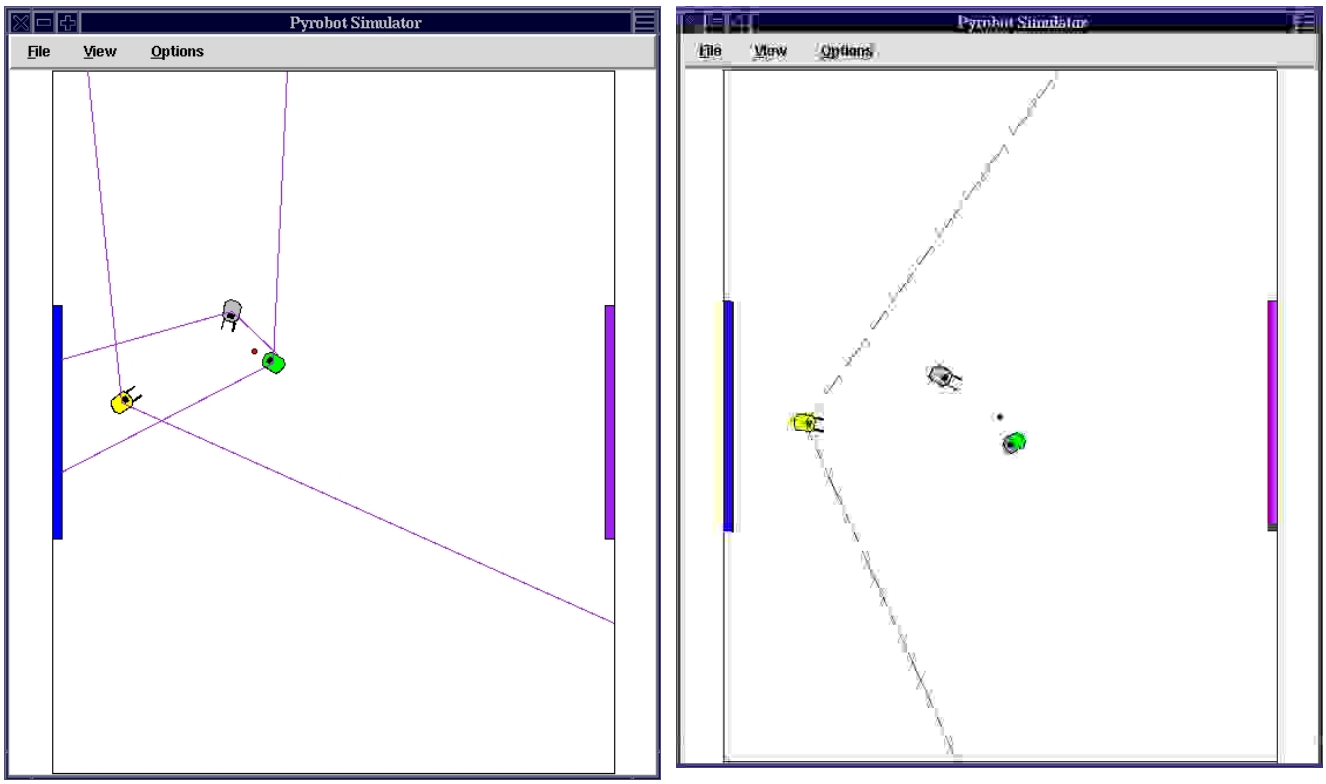
Kicking action was achieved by adjusting the threshold for the third output during multiple experiments. The disadvantage of kicking action was that it can happen anytime—we have not found a way to control when to let it occur—and if it happens at the beginning of the trial, the player will most likely lost the ball since it has already pushed the ball inside the goal box and the keeper will sure try to grab it first. There can be times when the kicking helps, usually if it happens within the goal box, depending on where the player robot is headed (see figure 5 b).

When we installed the defensive player in the second phase, we expected an easy transition since we thought that as long as we used the fitness function from the midterm to develop the brain for the defender and create a colored goal at the opposite end of the field, we'd achieve the task easily. However, it turned out to be an underestimation. First of all, the defender and the goalie were tightly packed in the goal box since it is their starting position. Therefore, the movements of both were sometimes constrained space-wise; we tackled the problem by extending the width of the field and therefore, the defender had enough space to move around while training.

Moreover, since both the defender and the players were evolved at the same time with relatively similar fitness function, they both seemed to have equal chance of getting the ball. As a result, the training process of both seems unable to overcome the distraction from each other. We solved the problem by only allowing the defender to move once every four time steps so that the player has a priority to move, train and learn first how to maneuver with the ball. Although we did all of these fixes and other minor adjustments such as trying out different weight for fitness scoring scheme and different input and output threshold, the player never learned how to dribble—which is carry the ball around the opponent. This is not surprising to us since we no longer use grippers—which we initially hope will help us carry the ball but turned out to be hindering the robot's movement.

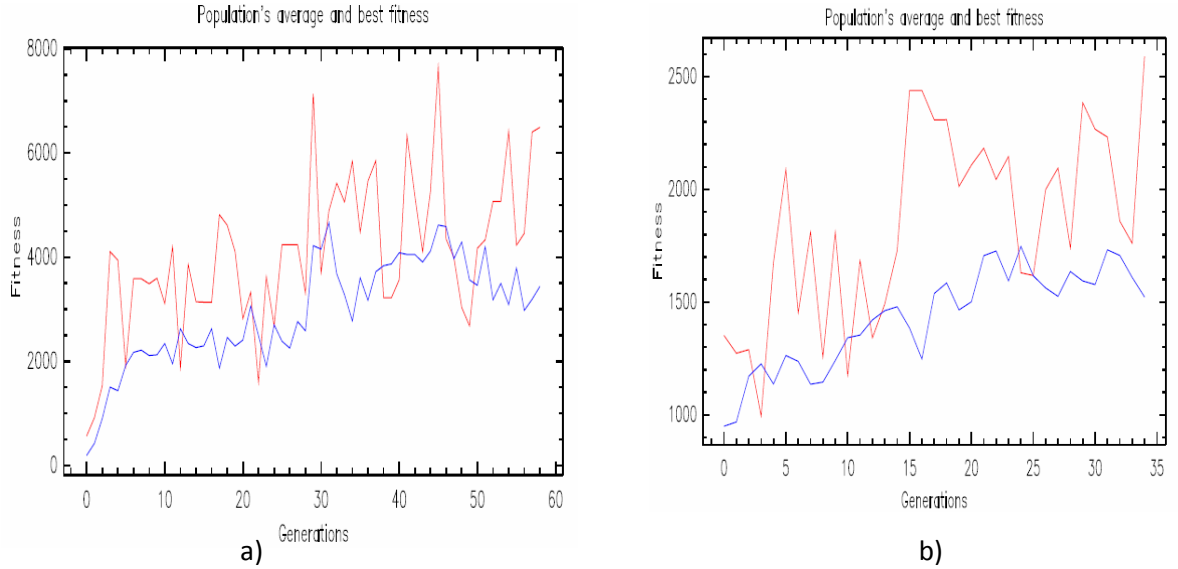
An interesting behavior displayed by the player robot in the second phase experiments is that it tends to activate the kicking action a lot, albeit at arbitrary places. Therefore, if it touches the ball first (before the defender), carries it towards the goal box and activates the kicking action, the chance of scoring a goal is highly probable (see figure 6 a and b). On the other hand, in some few cases, the player may never kick and will depend on carrying the ball to the goal. Though it may seems arbitrary, we empirically find that the player can score a goal with minimum of 60% accuracy if chromosomes 26 and above of later experiments were used.

We also present a couple of interesting average fitness and phenotype diagrams in figure 7 and 8 respectively.



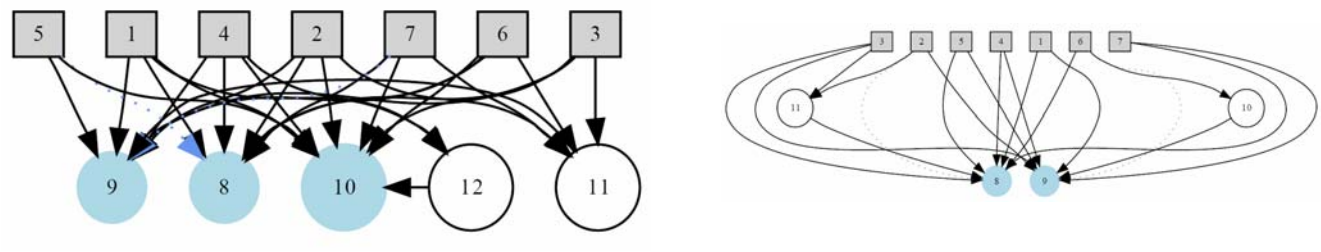
a) b)

Figure 6 a) The player carries the ball into the goal box, the goalie got activated and the defender lost sight of the ball while turning around; however, this event may not produce a goal since the player robot—the green one—is facing in the wrong direction from the goal **b)** The player lost sight of the ball and the defender approaching it; no chance of creating a goal



a) b)

Figure 5 Average fitness for a) the player robot and b) the defender robot. We can see that the score of player robot fluctuate too much especially because it has kicking action which doesn't guarantee the best fitness score all the time.



(a) (b)

Figure 6 Phenotype diagrams of the best chromosomes. None of the Phenotypes--both (a) and (b)—were originally configured to have hidden nodes. Although phenotype (a) looks simple, it is one of the best successful experiments where the player robot can score the goals well. (b) is a sample of phenotype for the defender robot, which only has two outputs because there is no kicking action implemented in it.

5. Discussion

Defining the output for the neural network for the player was fairly straightforward but tricky: It would signal the right and left motors to move either forward or backward, allowing the robot to move in any direction whereas move back and forth to simulate the kicking action when the third output is higher than the threshold—0.5. Deciding how to represent the world state, i.e. the inputs to the neural network, also turned out to be easy because of our background knowledge from the midterm project. However,

It required some amount of thinking, justification and experimenting to decide whether to use a particular value as part of the inputs or the fitness function. According to our experience in the midterm and this project, we found that it is important to come up with a good set of input values first and then explore the suitable fitness function. Next is to adjust the parameters and threshold values. It usually takes around 10 or so experiments—each lasting 30 generations—to observe a satisfactory behavior from the robot.

Although the smart choice of the input nodes and the fitness scores are critical to some extent, the entire project definitely required very little human involvement than it would have taken if we were to also decide the neural network architecture and the excitation values. Writing up the fitness function as a whole was pretty straight forward for us although coming up with a good one is somewhat harder and is solely our responsibility. However, given the simplicity of the fitness function we finally attained and somewhat sophisticated behavior the network was able to achieve, we can't deny that evolutionary techniques are powerful and could play a huge role in developmental robotics.

As a practical note, we noticed in our experiments that the performance of the robots reach the best around 26th generation even though we ran a total of 30. Since the most noticeable weakness of the evolutionary learning is the amount of time required to train, we could improve it a little if we can develop a way to detect and stop the runs once the performance—measured, of course, in terms of fitness score—reach the plateau.

The next step for us is modify the defender robot's fitness function so that it does not have to depend on seeing the opposite goal as well as fine tuning the kicking behavior in the player robot and if possible, induce purposeful kicking actions instead of arbitrary ones as it is doing currently. More critical and helpful improvement would be to find ways to speed up the training process especially now that we have multiple agents which are using different brains and requires non-negligible amount of calculation power and time cost for each trial. Although it is surprising to find out that evolving neural network architecture with NEAT will yield us a network that will perform somewhat sophisticated soccer game actions, the question of whether NEAT will be the solution for more realistic a complex task—such as dribbling—is yet to be seen.

Acknowledgements

We are grateful to Professor Meeden for her mentorship, specifically for porting the NEAT code to work with Pyrobot, showing us how to use and troubleshoot some issues related to the Pyrobot simulator, and finally and most importantly, providing ideas on deciding the input nodes for the neural net. Rachel and Madeleine, our classmates, sparked our interest to begin this project. Moreover, Rachel Lee and Andrew Pace provided us with samples of the work they did for their midterm projects, from which we figured out how to simulate two robots with different brains to coevolve. Therefore, both of them deserve our thanks as well as our classmates who gave us insightful feedback during our class presentation.

References

Blank, D., Meeden, L., & Kumar, D. (2003). Python robotics: An environment for exploring robotics beyond LEGOs. In *Proceedings of the SIGCSE Conference*, *Reno*, *Nevada*.

V. Ciesielski and S. Y. Lai. Developing a dribble-and-score behaviour for robot soccer using neuro evolution. In P.Wigham, editor, *Proceedings of the 5th Australia-Japan Joint Workshop on Intellignet and Evolutionary Systems*, pages 70–78, Dunedin, New Zealand, 2001.

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

RoboCup Research & Education. Publications related to Soccer Simlation League 2005 July. World Wide Web, URL is http://www.robocup.org/research/5.html.

Stanley, K. O. & R. Miikulainen (2002). Competitive coevolution through evolutionary complexification. *Journal of Arificial Intelligence Research*, vol. 21, 2004.

Stanley, K. O., & Miikkulainen, R. (2002d). Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10 (2), 99–127.

Sutton, R. S., & Barto, A. G., *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

Thiha, P., Evolving a Robot Soccer Player Using NEAT. In *CS81 Midterm Projects*, 2009.

Taylor, M. E., Whiteson, S., & Stone, P. (2006). Comparing evolutionary and temporal difference methods in a reinforcement learning domain. In *Proceedings of the Genetic and Evolutionary Computation Conference* (GECCO 2006) (pp. 1321–1328). Seattle, WA.

Whiteson, S., Taylor, M. E., and Stone, P. Empirical studies in action selection for reinforcement learning. *Adaptive Behavior*, 15(1):33–50, March 2007.