Using NEAT to teach a crawling robot to follow a light

By Roby Velez

Abstract

For a previous project NEAT was used to evolve an artificial neural network (ANN) to control a crawling robot and get it to crawl either forwards or backwards. This paper extends the previous research to a more complicated task. This paper will discuss using NEAT to evolve an ANN to control a crawling robot and get it move *both* forwards and backwards in order to follow a light. Different manipulations were tried to obtain the most successful results. NEAT evolved ANNs to solve the task, the fastest when the robots, controlled by the ANN, were seeded with ANNs which already knew how to crawl.

Table of Contents

Abstract 1

Introduction 2

NEAT 3

Robot Introduction 4

Previous Work 6

Current Work 7

- **Experiments 1 and 2** 8
- **Experiments 3 and 4** 10

Discussion 12

- **Porting to the Real Robot** 13

Conclusion 13

References 13

Introduction

Nature has created organisms capable of doing incredible tasks, from humans who can think, create, and change their world to simple organisms that adapt and fend for themselves in harsh environments. Organisms have intelligence and autonomy that scientist have tried to incorporate into robots. Initially robots and intelligent systems were modeled like computers and all the focus was on creating good models and thinking algorithms. More recently there has been a shift to looking at approaches that limit the amount of information and control the designer has and allows the system to learn and adapt for itself.

One way to incorporate learning into a system or robot is with artificial neural networks (ANN). ANNs are modeled off of neural networks found in real organisms. ANNs have found success in pattern recognition and data generalization, but there doesn't seem to be much work done on using ANN to control locomotion. Researchers have found neural networks that control locomotion in real organisms that use oscillators. These oscillating neural networks, called Central Pattern Generators, have been modeled and used by researchers in Switzerland to create slithering(1), swimming(2), and crawling(3) robots.

While the researchers in Switzerland produced very cool robots they used models instead of artificial neural networks. For a previous project I used NEAT to evolve ANNs to control a crawling robot. The learning and control system used an ANN to read in the sensors of the crawling robot and control its servos. NEAT then altered the weights and topology of the ANN to produce the crawling behavior. In the end

NEAT produced ANNs that used oscillators to control the movement of the crawling robot just like the Central Pattern Generators.

In this paper I will expand on the previous work by looking at a more difficult task will be looked at: following a light. The rest of the paper will introduce NEAT and how it works, the robot used in the experiments, briefly go over the results of the previous experiments, and then introduce the current experiment and results.

NEAT

NEAT is a genetic algorithm that evolves artificial neural networks. It works by encoding ANN into their connections and nodes. This is the genotype of the ANN. Figure 1 shows an ANN and a partial genotype of the connection genes. The numbers in each box, from top to bottom, are the innovation numbers of the connection, which nodes the connection spans, and whether or not the connection is disabled. The connection genes also encode the weight of the connection, but this is not shown. Much of the following discussion is summarized from the NEAT paper(4) or re-worded from my previous paper(5).

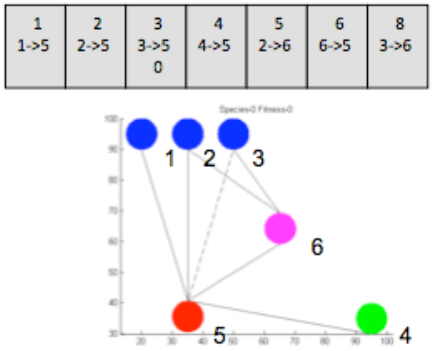

Figure 1: How the genotype of ANN is represented in NEAT. Inspired by figure 2 in NEAT paper.

Mutation in NEAT works by adding connections or nodes, or changing the weights of connections. Adding a connection is fairly simple. Connections can be added from any node to any other node. This is very important because it will enable things like memory and time dependence to emerge from the ANN. Figure 2 shows how nodes are added.

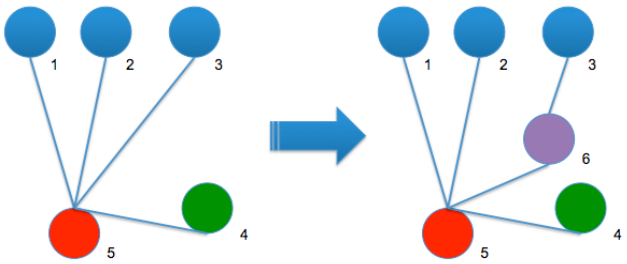

Figure 2: Schematic of how nodes are added in NEAT. Image inspired from figure 3 in NEAT paper.

Nodes are added between connections. As seen in figure 2 node 6 is added in the middle of the connection between 3 and 5. The original connection is split in two. The new connection from the new 6 to 5 retains the weight of the old connection. The new connection between the node 3 and node 6 has a weight of one. This ensures that the addition of a node doesn't radically change the performance of the ANN. The new node and connection can slowly come into its own.

As connections and nodes are added they are given innovation numbers. These innovation numbers are the order in which the connection or node was added. They give a history for the connection or node. The innovation numbers are used during reproduction.

When performing crossover NEAT takes the connection genes of each ANN and lines them up based on their innovation numbers as seen in figure 3. When crossover occurs only the connections or genes with a common history are crossed. This ensures that the general structure of the parents stays maintained during the reproduction process.

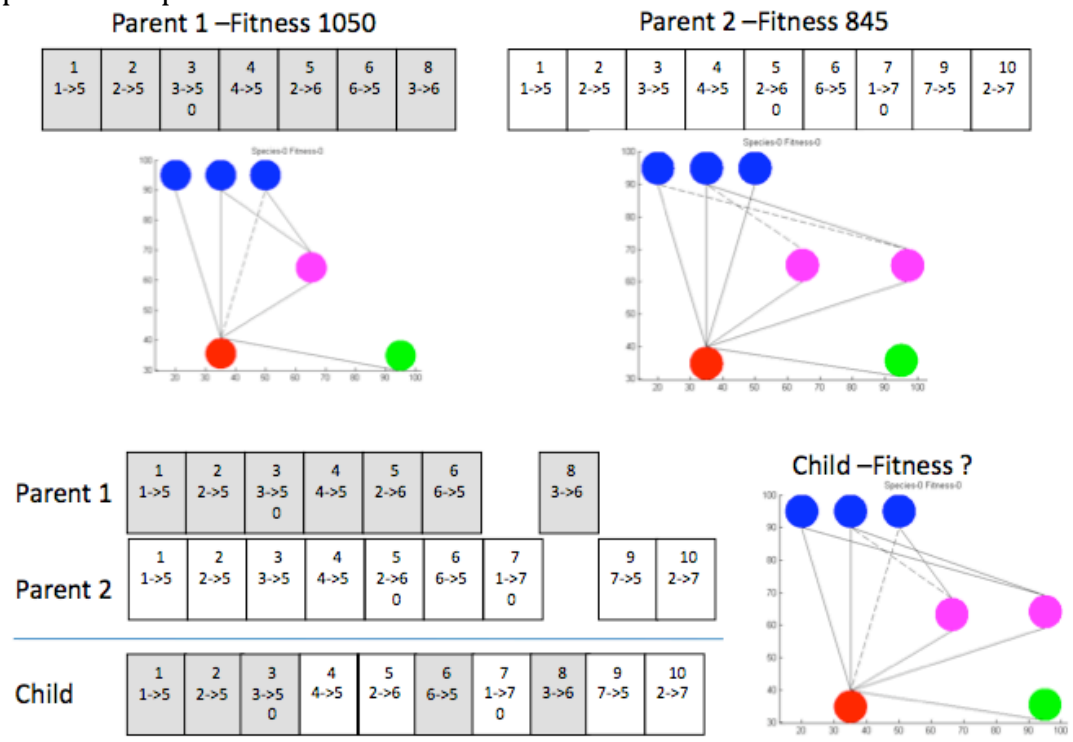

Figure 3: Crossover in NEAT. Image inspired from figure 4 in paper on NEAT.

Robot Introduction

The robot used in these experiments was created for a senior design project in engineering. It consists of a 3 Degree of Freedom arm mounted onto a wooden chassis. The robot moves by pushing or pulling itself. A concept drawing and actual image of the robot are shown in figure 4.

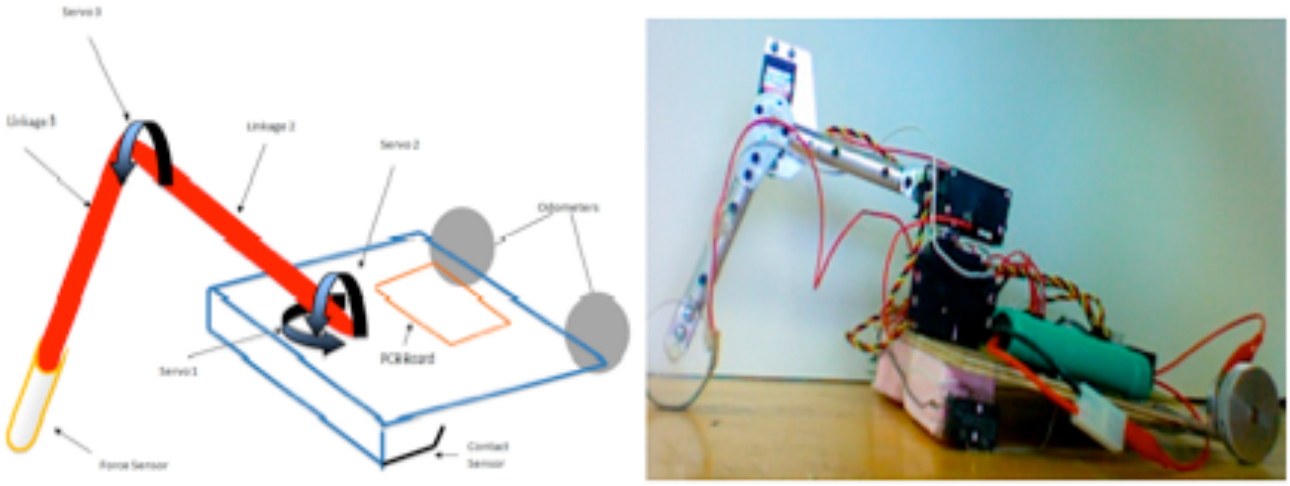

Figure 4: Concept drawing and actual drawing of the crawling robot.

The artificial neural network used to control the robot is shown in figure 5. It has six input nodes corresponding to the 6 sensors of the robot. It has two output nodes that control the speed of servo 3 and servo 2. Servo 1 was not used in the previous or current experiments. See the schematic to the left of figure 4 which identifies which servo is which.

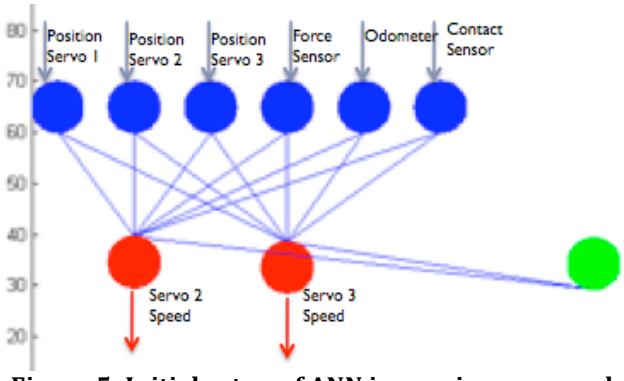

Figure 5: Initial setup of ANN in previous research.

Table 1 lists all the sensors used and gives a brief description of the range of values they can take and how they were made.

Sensor Name	Description
3 Position Sensors	Made from monitoring a pin off of the Servo control board. They indicate the angle of each Servo, but are quite noisy. They range from a value of 80 to 400.
1 Force Sensor	Mounted at the end of the arm. The Servos are not very strong so the sensor was calibrated to be very sensitive, which results it in not being able to distinguish a wide range of different force. The robot pretty much knows if it is touching the ground or not. It is either 1023 when the arm is off the ground and around 600 when the arm is one the ground.
1 Contact Sensor	Two contact sensors are mounted on either side of the foam on the bottom of the chassis. The robot rests of them. They are wired together and act as one contact sensor. They report a 1 if one of them is released and a zero otherwise.
1 Odometer	When the robot rolls forward it is rolling forward on two wheels mounted at the back of the chassis. The two wheels are made up of continuous potentiometers which read a changing voltage as they turn. Only one Odometer reading is read. This is used to measure how far the robot has moved. It reads a value from 0 to 1023. If the potentiometers wiper reaches 1023 but continues to roll it will roll over to 0 and increment normally. If the potentiometer reads 0 and continues to roll backwards it will roll over to 1023 and continues decrementing normally.

Table 1: Servo descriptions.

Previous Work

This project is centered on an actual robot, but for running evolution experiments it was unrealistic to run them on the real robot. A simulator and simulated robot was created in MATLAB to run the experiments. For more detailed information on the simulator see the simulator section in reference 5. A still of the simulator is shown in figure 6.

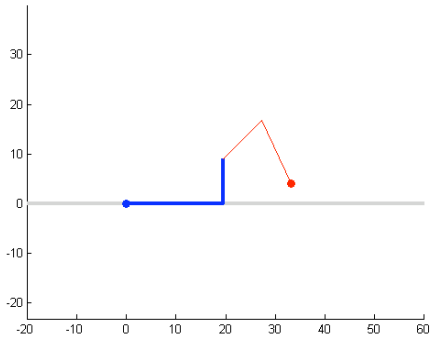

Figure 6: Simulator screen shot.

The environment is the simulator. Populations of ANN were created by NEAT and loaded onto the simulated robot. The simulated robot was given 450 time steps to do something. After 450 time steps the fitness of the ANN was the net displacement of the robot.

Twelve runs were made and let to go to 150 generations. ANNs evolved which produced forward and backward crawling motion. Figure 7 shows the average max fitness of 12 runs over the number of generations.

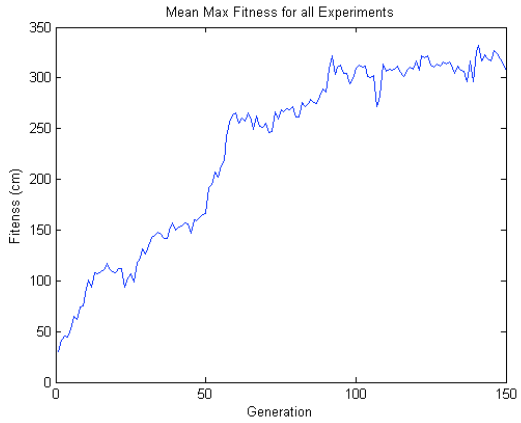

Figure 7: Average max fitness versus generation for previous research. Average from 12 different runs.

Current Work

Figure 7 shows that after 150 generations NEAT was able to evolve ANNs which produced motion. These ANNs learned either to move only forwards or to move only backwards.

A new experiment was designed to see if the ANNs could be evolved to move both backwards and forwards. A light was added to the simulator shown in figure 8. The robot would have to learn to chase the light. The light was stationary. If the simulated robot reached the light it would disappear and reappear on the other side of the simulated robot.

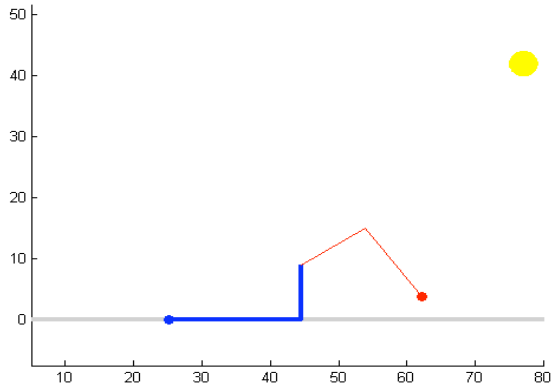

Figure 8: Simulator with light sensor added.

A light sensor was added to the robot. It fed into a 7th input node of the ANN shown in figure 9. The light sensor was modeled after a digital switch. It produced a 1 when the light was in front of the robot and a 0 when the light was behind the robot. For this set of experiments the robot's fitness was its net displacement when moving toward the light.

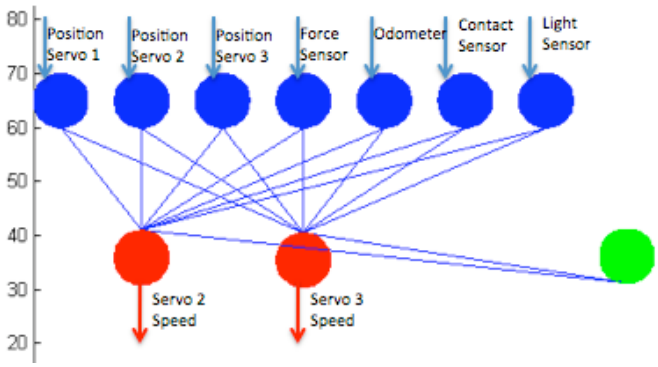

Figure 9: ANN setup with the light sensor added.

All the parameters for the pervious, crawling experiment, were reused for this experiment and are listed in the NEAT implementation section of reference 5.

Experiments 1 and 2

In the previous work both forward crawlers (pullers) and backward crawlers (pushers) evolved, but there wasn't an even distribution of pushers and pullers. Pushers were more common than pullers. It was discovered that pushers are inherently better at the task than pullers. This would affect how easy it is for the ANN to learn to track the light. If the light is initially placed behind the simulated robot it must learn to crawl backwards. If the light is initially placed in front of the simulated robot it must learn to crawl forwards. The first scenario is easier and it would finish crawling and move onto learning to track the light quicker.

Two experiments were created to explore this difference. In experiment 1 four evolution runs were made with the light initially placed 50 cm in front of the robot. In experiment 2 four evolution runs were made with the light initially placed 75 cm behind the robot. A mistake was made during the coding which caused the distance between the robot and light when it is behind and in front to be different. Unfortunately this was caught to late. It makes some of results harder to interpret, but there are strong trends in experiment 4 that would have occurred even if this error wasn't made.

The results for experiment 2 are easier to explain than the results for experiment 1 so they will be discussed first. Figure 10 shows the max fitness versus generation for 4 funs in experiment 2.

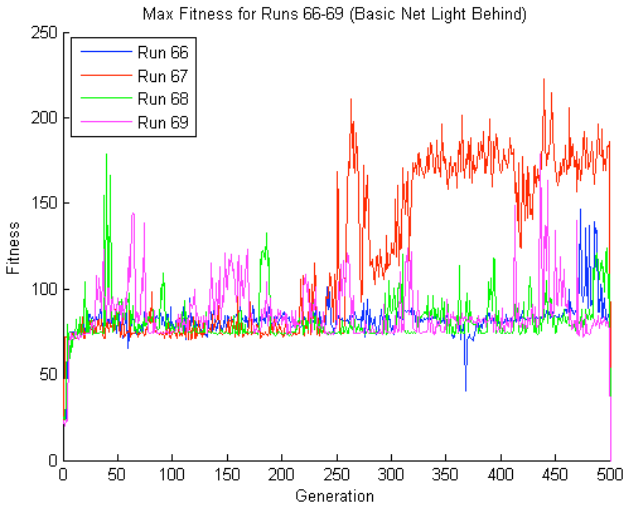

Figure 10: Max fitness versus generation for 4 runs in experiment 2.

Figure 10 shows that for experiment 2 most of the runs were able to produce ANNs that reached the light at least once. All the runs produced ANNs which get fitness of at least 75, which is the distance of the light to the robot when it is placed behind the robot. A fitness of 75 corresponds to the simulated robot completing one lap. When the light was placed behind the simulated it moved backwards until it reached the light. At this point the light disappeared and reappeared in front of the simulated robot about 50 cm away. It got fitness for reaching the light the first time. For doing one lap. But most of the ANN kept going and so the fitness obtained from reaching the light once, in this case around 75, is all the fitness they obtained.

One run was able to learn to track the light. It reached fitness value of 200 which about 3 laps. These ANN learned to crawl backwards and then forwards toward the light.

Figure 11 shows the max fitness versus generation for experiment 1.

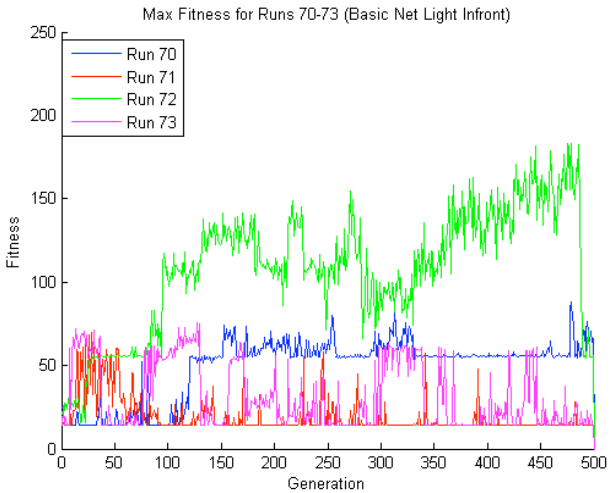

Figure 11: Max fitness versus generation for the 4 runs of experiment 1.

Unlike experiment 2 not all the runs were able to produce ANN that completed at least 1 lap. Not all of the runs were able to produce ANN that could crawl forward and reach the light at least once. This is because crawling forward is more difficult than crawling backwards. One run was able to begin to track the light. It only reached a fitness of 200 toward generation 450, but it was nonetheless successful.

Neither experiment is clearly better. Both experiments were only able to produce one run which reached a fitness of 200. Experiment 2 looks to be a little better because it reaches 200 faster and looks more stable, but there is so much variability in the runs its hard to say.

The issue with the initial distance of the light doesn't affect the ANNs which are successfully tracking the light. The coding error makes the pushers, who do one lap, look better then the pullers, who do one lap, because the former will get 75 cm instead of 50 cm as fitness. But once the ANNs learn to track the light this doesn't matter. If you can follow the light the only limit to your fitness is how fast you can move not how many times you have to turn around. This can be said because it doesn't take long for the simulated robots to change directions.

While experiment 1 and 2 didn't produce definite results another manipulation was tried. The initial ANNs in experiment 1 and 2 didn't know anything. They had to learn first to crawl before they could learn to follow the light. Two experiments were devised where the initial ANNs were bootstrapped with topologies which made it easy for them to learn to initially crawl.

Experiments 3 and 4

Experiment 3 was seeded with ANNs which have the topology of a puller. In this experiment the light was initially placed in front of the robot. In previous research it was discovered that pulling and pushing ANN were actually quite similar. Figure 12 shows two ANNs side by side. The ANN of the left is a pushing configuration and the ANN on the right is a pulling configuration. The only difference between the two ANNs is the connection between the two output nodes. This connection alters the pattern of oscillation of the ANN. For a more detail explanation of the oscillations, topologies, and how they differed for the pushers and pullers see the analysis section of reference 5.

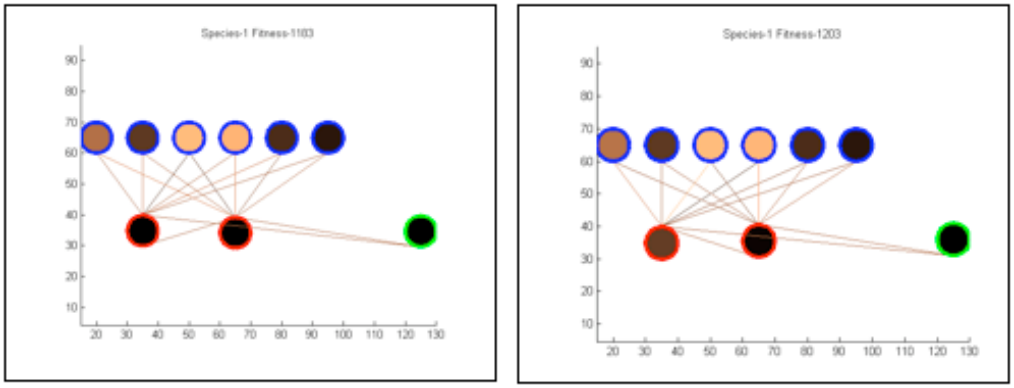

Figure 12: ANNs which produce basic pushing behavior (left) and basic pulling behavior (right).

Experiment 3 was seeded with the ANN on the right of figure 12. The weights were not optimized to produce crawling motion. This means that just because the ANN had the right topology they didn't necessarily crawl. The ANNs were simply pointed to the correct door. They had to learn to open it for themselves. The weights were not optimized in order to limit the amount of domain, outside knowledge that was put into the system by me. Experiment 4 was seeded with the ANN to the left of figure 12. In experiment 4 the light was initially placed behind the robot.

Figure 13 shows the max fitness versus generation for four runs in experiment 3. In these experiments more of the runs were able to accomplish simple crawling forward behavior and reach the light at least once. But one of the runs didn't, which shows that even though the ANNs have the right topology they still need to find the right weights.

The results from experiment 3 are a little more promising that those from experiment 1. By generation 500 two runs produces fitness values of around 200. But again the variability from run to run its hard to make an definite claim.

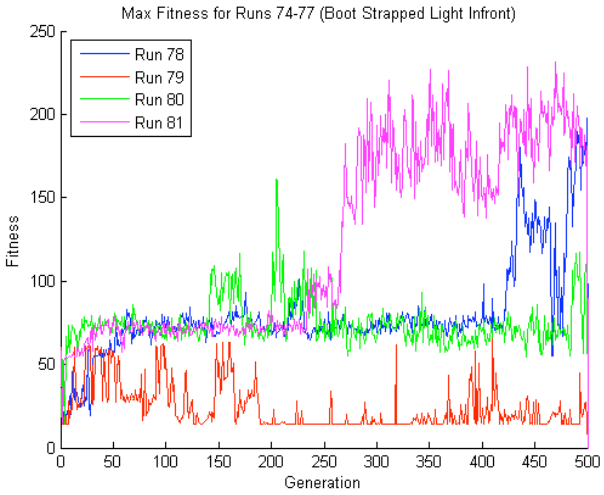

Figure 13: Max fitness versus generation for 4 runs in experiment 3.

Figure 14 shows the max fitness versus generation for experiment 4. Here all of the runs took to the initial bootstrapping and all the runs, at one point in their history, produced ANN which followed the light source to some degree. Also two of the runs looked to produced ANNs which got fitness values greater than 200 which no other run was able to do.

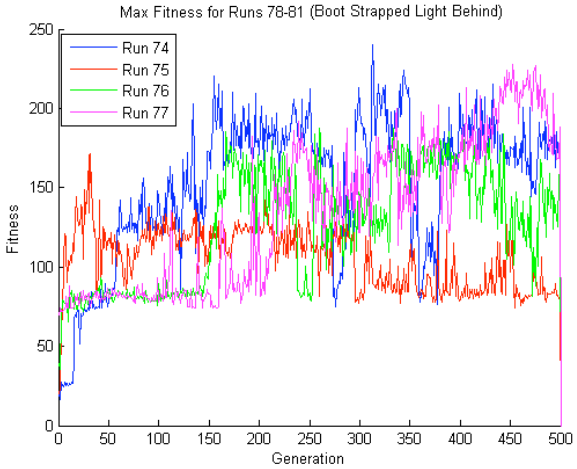

Figure 14: Max fitness versus generation for 4 runs in experiment 4.

With all the issues with these experiments the results of experiment 4 looked to be pretty definitive. Nearly all the runs produced light followers with fitness values above 150.

Discussion

Experiment 2 should have produced the type of results seen in experiment 4. As seen in figures 10 and 14 nearly all of the runs for experiments 2 and 4 produced ANNs which were crawling very early on. This means that the issue of starting from scratch should not have been a problem for the ANNs in the experiment 2. The only explanation is that the initially seeding of experiment 4 was too good. It apparently was unrealistic of a solution that would naturally occur and caused the ANNs in experiment 4 to do so much better. A much better manipulation would have been to just train the ANNs on crawling and then train them to follow a light.

Figure 13, for experiment 3 shows at the very end the fitness of one of the runs starts to pick up. This run was let to go for more than 500 generations, but is not showed here. The fitness did pretty well and became stable. Part of the reason many of the results from these experiments aren't conclusive is because they weren't run long enough. It may have been better to performed fewer runs and let each run go out to 1000 generations. Please note though that a 1000 generations takes a very long time and hogs a whole computer in Hicks, which is why many of these runs weren't set that high.

In the end the experiments were a success. Each was able to produce one or more runs with light following robots. This experimenter would have liked to have seen more stable, and higher fitness early on and clearer distinctions from the manipulations.

Porting to the Real Robot

A light sensor, to detect direction, is a very poor sensor for a mobile robot. This research was built around a light following robot to make the idea easy to conceptualize. The light sensor in the simulator acted like a binary switch that produces a 1 when the light was in front of the simulated robot and a 0 when the light was behind the simulated robot. When the ANNs were ported to the real robot the light sensor was replaced with a switch. In one position the switch makes the robot go forward. In the other position the switch makes the robot go backwards.

In the previous research nearly all of the ANN evolved worked well on the real robot. In this research not all of the evolved ANN worked well on the real robot. It's not known why some of the solutions break on the real robot, but ANNs were found which made the real robot move forward when the switch was in one position and backwards when the switch was in the other position.

Conclusion

This paper shows that ANNS evolved with NEAT can be used produce complex behaviors such as following a light. More, longer experiments need to done to explore how things like initial starting position and prior knowledge influence learning. From the results obtained in this project it can be said that bootstrapping the population with hand written ANNs does speed up the learning process.

References

1. Crespi, A.J.. " AmphiBot I : an amphibious snake-like robot." Robotics and Autonomous Systems 50(2008): 163-175.
2.Crespi, A.J.. "Controlling swimming and crawling in a fish robot using a central pattern generator." Autonomous Robots 25(2008): 3-13.
3.Crespi, A.J.. " From swimming to walking with a salamander robot driven by a spinal cord model" SCIENCE 315(2007): 1416-1420.
4.Stanley, Kenneth. "Competive coevolution through evolutionary complexification." Journal of Artifical Intelligence Research, vol 21. 2004
5. Velez, Roby. "Engineering Design: Design and Control of a Simple Robot" 2009