Mid-Semester Report

My name is Travis Hite, and I am a student working with the Research Experience for Undergraduates program funded by the National Science Foundation. My home college is Kennesaw State University, however I plan on transferring to UGA sometime in the next year. The specific REU I am attending is being held at the Auburn campus, and is in the general field of computer science and engineering.

I have been working for the past five weeks with Dr. Dozier on an evolutionary robot. The robot we are using is a Khepera robot. It is 55 mm in diameter and 30 mm tall. It has a Motorola 68331 processor, eight infra-red proximity and light sensors, and two DC motors.

What we are trying to do with this Khepera robot is to build a non-biased interactive evolutionary algorithm in order to teach it not to hit walls. While this has been done before, this process can take many hours and up to several days to evolve to a suitable outcome. What is revolutionary about our approach is that we have come across a method to evolve our robot in merely a matter of minutes.

Our Khepera uses a general regression neural network. A physical concept of the function of the network is to picture a round disc, which would represent the robot. Divide this robot straight down the center. One wheel is on either side of the robot. On the top hemisphere of the robot, 6 sensors are equally distant from each other, as well as 2 on the bottom hemisphere. The top sensors are set to wall detection. Light bounces back to the sensors which take a reading. When a reading of 1023 is received (chosen because it is the highest possible reading), the sensors activate with a value of 1.

The sensors then feed through a connection to the wheel, with the GRNN being the focal point. This is where our algorithm comes in to play. We have defined a set of 4 weights. The first three weights (each corresponding to the front sensor, the mid-front sensor, and the mid-back sensor) have a range of -30 to 30. The fourth value (the sigma value) is between 1 and 512. An example value would be {18,-26,-4,354}.

The sigma value controls the overall wheel speed of the robot. The higher the value, the slower the robot goes. The other three weights either speed up or slow down the wheel on the opposite side of the sensor it is responding to. An optimal result is to have the robot go as slow as possible and turn as fast as possible to avoid walls and decrease potential damage. This result would have the value of {-30,-30,-30,512}.

The use of only four weights is another part of speeding up the process. Many algorithms will use 40 or more weights in their experiment. Using less weights decreases the amount of

processing time required for each individual set of weights.

For our experiment we are using a simple white poster board stage with size 30x30 cm. The small size of the robot and the area, as well as the small size of the stage and the cushioned poster board are designed to keep the robot safer. A smaller robot will have a smaller overall velocity for the same comparative results, which leads to better safety for the robot.

For a user-defined number of cycles, Interactive Evolutionary Computation is used (From here on referred to as the IEC). The beginning of IEC is to create a population of individuals are randomly generated, weighted at 0 to keep the population from being biased.

After a population is created, two candidates are taken at random to be parents. If one of the candidates has been aborted in the past, the sign of one of the first three weights is randomly flipped (Obviously, from the initial population, this is not a problem since at this point none of the candidates have been evaluated yet). This is called sign mutation. Each parent is allowed to control the robot for four seconds. During this time, the user interacts with the robot through a webpage connected to a server to the computer hosting the experiment. For each button click, or when the button is held down, hits are assigned to the individual. The winner of each tournament is the individual with the least number of hits. If the robot uses a behavior that is extremely dangerous (erratic, hits walls often at a high velocity, lunges back and forth, etc.), the candidate is aborted and assigned the maximum number of hits.

The difference between the two individuals is then memorized for later use. For instance, if we had an individual p1{-14, -25, -17, 136} and another individual p2{-20,-25,-22,388} the distance between the two would be <1,0,1,-1,1>. A value of 1 means the first number was higher. A value of 0 means the values were equal. A value of -1 means the value was less. The last value, which can only be -1 or 1, represents the winner (either the first value or the second). Since the second value is both slower and will turn faster, the second value won.

After both individuals are tested and a difference is set, one of our evolutionary operators come in to play. If both individuals were aborted, a random individual from the population is selected and mutated twice to replace both parents. If there is a case where the winning behavior is feasible and the losing behavior is infeasible, uniform-bounded mutation is used. For instance, take a behavior {-14,-25,-22,418} to be the winner. And say the range of mutation is between -12 and 12 for the first three weights and -50 to 50 for the last weight. Any results between {-26,-30,-30,368} and {-2,-13,-10,468} would be entirely possible. The mutation then overwrites the loser, the winner is put back into the population, and a new individual is selected to compete against the loser.

Another scenario occurs when a winner is selected that has not been aborted but has not won at least two tournaments. When this occurs, uniform-crossover is used. What happens with uniform-crossover is a new individual is created from the parents with a probability of .5. For instance, take two individuals {-22,-28,-18,464} and {-2,8,-24,316}, where the winning parent is the previous parent. The offspring generated could potentially look like {-22,8,-24,464}. Any combination of the two parents is possible. The offspring then replaces the losing parent. The winner is placed back in the population, and a new candidate is selected to go against the newly created individual.

The third scenario occurs when both parents have been deemed to be feasible (not aborted, two wins). In this case, uniform-bounded mutation is used again (in the same way as previously discussed).

When the IEC is finished, the MEC begins. At this point, the robot becomes still, and "meditates" (thusly the M) on its previous results. Two individuals are taken from the population and generate a distance vector as before. A search is then conducted for the difference vector that most closely represents the difference vector it came up with. A winner is then selected based on which individual was selected before. Thusly, previous user preference is applied throughout the meditative process. Evolution continues as it occurred in the IEC.

Using the above scenario, the algorithm produces what it considers to be its best example. Generally we receive a good individual after the entire process, however several problems have been noted.

For one, since user interface time is condensed, any human error can create exponential errors through the MEC. The only real way to alleviate this problem with the current design is to be sure to pay close attention to the robot during the IEC time. This is not a major problem, since the IEC generally only takes a few minutes to go through all of its iterations.

Another problem has to do with the sigma value. Since the difference between 256 and 512 is hard to see with the human eye, often a bad sigma value can occur. The problem is also greater because hitting the robot just because it is moving faster can throw off the other weights. An alternative to this, though admittedly biased in nature, is to hit the robot every time it comes close to a wall regardless of turning or not. The concept here is that a robot that moves faster will come close to the wall more often.

What I have been attempting to do in the past few weeks is to do as much research as possible in order to catch up with the current research, and to come up with ideas to improve the robot. While I have been here five weeks, I can say I have only done

about three weeks of work, since the first week was Biaz's presentation on networking and an introduction to Graduate life and the second week Dozier was out of town leaving me to brush up on my programming skills.

The first idea we had on improving the robot was improving the interface itself. While Joe and Lacy are still working on their improvements, the idea we came up with for my improvement was found to be infeasible.

The idea was to build a steaming webcam into the webpage used to interact with the robot so the robot could be left on and interaction could take place from anywhere. I did some research on past experiments on this area and information about found out about the java media library. However, even the best results saw delays up to eight seconds. The problem is that how java steams video is that it writes the information received from the webcam to a buffer file. This file is then steamed to the receiving individual where it is decoded and displayed. Though we are sure there must be a way to bypass this delay on a large network such as Auburn's that would ensure maximum speed, it would have been too much work to attempt to implement.

In the meantime, I attended a series of lectures on Artificial Intelligence design as presented by Dr. Dozier. During this time Dozier made sure that material I was presented to read was tangential to the topic in class. The very first thing he had me do was read his paper “Evolving Robot Behavior via Interactive Evolutionary Computation: From Real-World to Simulation”.

In it he explains the basics for his robot and compares it to past experiments ran by other individuals. At the end he presents various examples of trial runs to show the results of the process. Though it makes for a very thorough explanation, it is rather difficult to read for someone new in the field. However, it gave me a good overview of the situation.

He then had me read “A Survery of Artificial Life and Evolutionary Robotics” which explains several past experiments in the field and compares information. It greatly aided in my understanding of the uses and history of the field.

I also read serveral other papers during this time including a chapter of a book written by Dozier and several others titled “An Introduction to Evolutionary Computation.” and several others that shall be listed on the website this essay is available on. Most of the other papers are mentioned are on evolutionary robotics or in related fields and were mostly meant to strengthen my knowledge of material within the field and give me a better understanding of what we are doing.

One of the most important things Dozier had me read during this time was the book “Evolutionary Robotics” by Stefano Nolfi

and Dario Floreano. In it they discuss design concepts and information on the fundamentals of genetic algorithms and neural networks. They also talk about several key experiments conducted and their impact. A short powerpoint presentation using information from the beginning of the book will be available on my website.

After the presentation he presented me with the code and went over it with me. After the meeting in which we went over the various evolutionary procedures mentioned earlier, I was asked to come up with some better evolutionary procedures to be used.

I then presented to him several ideas for changes that could be made to the way evolution takes place. One of the ideas was an observation that sometimes amongst two good candidates the user can end up picking the worse candidate and making a poor vector to use. My idea was to use an ELO rating system and modify the number of hits based on difference in rating. Without getting too far into how ELO works, instead of just counting the number of wins and losses to determine ranking, it modifies ranking based on a percentage of difference between the winner's ranking and the losers ranking, and adjusts accordingly. Using this addition would make the process more fair and prevent mistakes, but could lead to a bias towards older individuals.

Another idea I had was when both parents were aborted, use the mutated individual, but have the second offspring be an exact opposite mutation from the first, so at least we could come up with some better distance vectors between to use during the MEC.

In the end we decided instead to use naive bayesian modifiers instead to push mutation more towards the area we required. Originally this modifier was used on expert systems, which used a very basic if/then set of scenarios. However in real life situations this sort of black and white scenario leads to biased results.

Instead, this uses a set of hypothesis and evidence (either one to multiple or multiple to one, never multiple to multiple) to find the conditional probability of an occurence. This has not only been used effectively in expert systems, but also in data mining and even in categorizing text documents.

Our plan is to apply this concept to genetic mutation in order to bias mutation more towards user preference. For instance, if previous sigma values went in a positive direction at a probability of 25/35 and at a negative direction at a probability of 10/35, the channel would be adjusted at a rate of .75*(25/35). The .75 would be put in place in order to keep the rate from every reaching 100%, which would make mutation stagnant and keep us from being able to effectively navigate through the search space.

Currently I am evaluating data from previous runs to find how often negative user preferences are found. I have found that most of our problems seem to come primarily from closer to the end of the run, which would make sense since at this point most of the individuals would start looking similar and would be hard to distinguish better individuals from.

I have also found, as previously stated, that sigma values tend to be fairly poor across the board, which would be our biggest hinderenace to using the bayesian modifiers and applying them to mutation.

Our goal by the end of the summer semester is to contribute some overall change to the code and hopefully design a better algorithm that better represents user preference. Following this the team working on the robot is going to write out a paper to be submitted to conference on our findings.