Neuro-evolving Maintain-Station Behavior for Realistically Simulated Boats

Nathan A. Penrod
Evolutionary Computing System Lab
Dept. of Computer Science and Engineering
University of Nevada
Reno, NV 89557
npenrod@cse.unr.edu

Sushil J. Louis
Evolutionary Computing System Lab
Dept. of Computer Science and Engineering
University of Nevada
Reno, NV 89557
sushil@cse.unr.edu

David Carr
Evolutionary Computing System Lab
Dept. of Computer Science and Engineering
University of Nevada
Reno, NV 89557
carrd@cse.unr.edu

Bobby D. Bryant
Evolutionary Computing System Lab
Dept. of Computer Science and Engineering
University of Nevada
Reno, NV 89557
bdbryant@cse.unr.edu

***Abstract*— We evolve a neural network controller for a boat that learns to maintain a given bearing and range with respect to a moving target in the Lagoon 3D game environment. Simulating realistic physics makes maneuvering boats difficult and thus makes an evolutionary approach an attractive alternative to hand coded methods. We evolve the weights of simple recurrent neural networks trained with a fitness function designed to combine multiple fitness objectives based on speed, heading, and position to create a robust maintain station behavior. Results with an enforced subpopulation neural-evolution genetic algorithm indicate that we can consistently evolve robust maintain controllers for realistically simulated boats in Lagoon.**

I. Introduction

The naval exercise known as maintain station consists of a following boat matching the heading and speed of a lead boat while staying in a specific position relative to the lead boat. Typically, maintain station takes the form of naval vessels moving together in a military formation. The physical limitations placed on the maneuverability of boats comprise the greatest challenge in coding the maintaining behavior. Boats can take a long time to accelerate, have a limited ability to turn, and are subject to drift when turning or decelerating. The existence of these various physical constraints results in a controller that requires careful tuning for each type of boat that executes the behavior because of the differences in maneuverability created by variations in size, mass, and power.

Previous work using neural networks to solve the inverted pendulum problem has shown that neural networks can learn to cope with complex physical situations without any explicit knowledge about the physics of a given system [1]. This ability of neural networks to learn physics implicitly provides an attractive alternative to hand coding because it allows for the creation of a robust controller without the need to understand the complex physics that govern the boat's movement, and these controllers could be evolved for different classes of boats with minimal programmer interaction. Because our goal involves creating a controller that both performs the maintain station behavior and performs the behavior in a way that a human observer would find believable, we performed our experiment with three different fitness functions to determine the method of calculating fitness that produces the most convincing results. We determine the best fitness function by making a quantitative comparison of the growth of fitness over time and also by making a qualitative evaluation to determine which behavior would be most appropriate for use in a game or simulation.

In this experiment, we successfully evolve a robust maintain station controller. We find that a fitness function that factors in both the relative heading to the postion to be maintained and the relative rotation between the maintaining boat and the lead boat performs best in producing robust, believable behavior.

The next section describes related work in neuro-evolution and behavior based robotics. We then define the maintain station behavior and the issues that arise in realistic simulations in greater detail. Section IV presents the neuro-evolutionary approach that we use to evolve the maintain station controller. We then introduce Lagoon, our 3D simulation gaming environment, describe the experiments conducted, and show the results. The last section presents conclusions and future work.

II. Related Work

Reseach on controlling the behavior of simulated entities draws on the influence of several related fields. Behaviors

like maintain station that provide the ability for groups to move together in formation are important because of the advantages formations provide such as the ability to share sensor information about the environment among multiple agents. Work by previous researchers to solve this problem have included techniques such as behavior based robotics [2]. Behavior based robotics have been applied to the Lagoon simulation as well, showing that these robotics based strategies translate well into a simulated environment [3].

However the goals for simulation programming can be somewhat different than the goals of robotics. The controllers used to drive the boats in the lagoon simulation will never be used for controlling real boats. For controllers constrained strictly to the virtual world we can make a number of assumptions including infinite power supplies, perfect sensors and complete control of the conditions in the world. We can also crash or mishandle unlimited numbers of virtual boats giving us the ability to test and evaluate thousands of controllers without damaging expensive equipment. Taking advantage of the computational resources at out disposal we would like to create controllers using more automated means such as an evolutionary method that could generate robust behaviors with a minimum of programmer intervention.

One challenge of using an evolutionary approach like a genetic algorithm is that these methods often solve problems in unexpected or unintuitive ways [4] [5]. We add a new level of complexity to the problem of evolving a controller when we demand that it not only perform a task but perform it in the same way that a human might perform it. Previous papers such as work on the evolutionary A* variant called GAMMA discus the importance of considering the believability of evolved agents for use in simulations [6]. Along these same lines a number of complex training regiments have been designed to encourage the believability of evolved agents including user modeling and training by example using lamarkian neuroevolution [7] [8] [9].

III. Problem

The maintain station behavior can be divided into two separate phases. In the first phase, the maintaining boat has to navigate to a target point in the world defined as an offset from the lead boat. In most cases a moving lead boat implies a moving target point. In the second phase, the maintaining boat has reached its target point and now must maintain its position on that target point by matching the heading and speed of the lead boat. If the lead boat alters course in any way, the maintaining boat must automatically adjust to the new heading and speed.

Figure 1 shows the maintain vector, the combination of target heading and target position that describes the desired position and heading of the target boat. The combination of a projected location of the target point used to calculate fitness called the projected point with the maintain vector describes all the information needed to evolve a controller. The base of the maintain vector, referred to as the target point, gets positioned at a location defined by a desired direction and desired distance relative to the lead boat. We

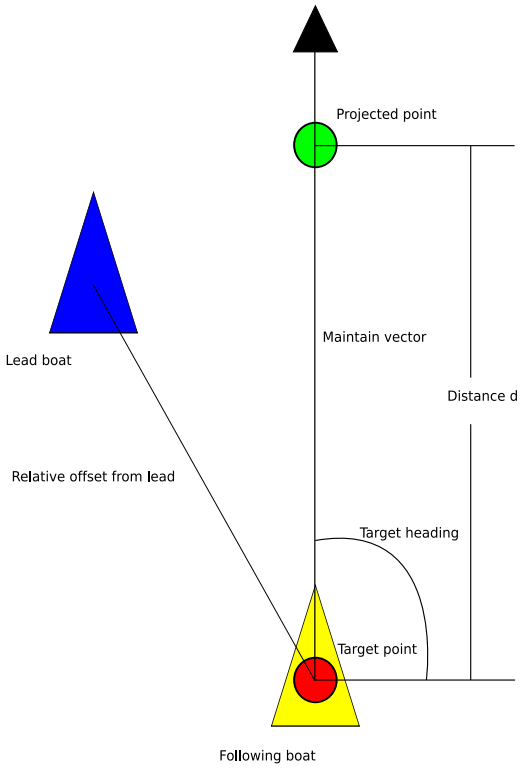

Fig. 1. The Maintain Vector

rotate the maintaining vector so that it matches the heading of the lead boat (target heading). Since the magnitude of the maintain vector changes the fitness landscape, we set the magnitude of the maintain vector (d in Figure 1) to be 100, a value that works well for our experiments. All sensor and fitness values are based on the maintain vector and not on the location of the lead boat. Training boats based on the maintain vector increases the flexibility of the behavior because the maintaining boat does not need to be aware of the lead boat's actual position allowing the behavior to be adapted to any arbitrary arrangement of lead and maintaining boats.

In the general case of the maintain station behavior the boats would also need a strategy to deal with the avoidance of obstacles in the environment. Both land and other boats could pose a threat to the integrity of the formation. We will not discuss the complex problem of obstacle avoidance in this paper but will consider this to be part of our future work.

IV. Approach

We use the ESP evolutionary algorithm to train maintaining boat controllers within the Lagoon simulation [10]. A type of genetic algorithm created specifically for use in neuro-evolution, ESP stands for enforced sub populations. One sub population of neurons is created for each neuron position in the neural network architecture. At each evaluation, networks are created by randomly combining one neuron

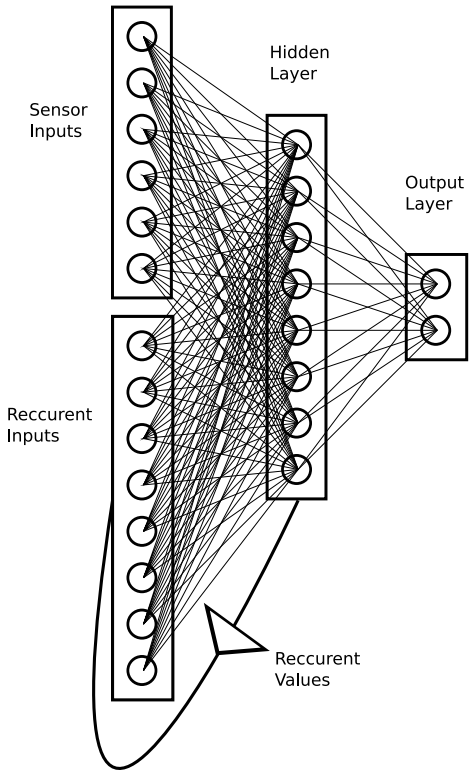

Fig. 2. Fully Connected Recurrent Network

Fig. 3. Lagoon Screenshot

from each of the individual sub populations into a complete network. At the end of the evaluation, each neuron receives a fitness score equal to the fitness score of the network that it participates in. To protect good neurons from incurring an unfair bias by participating in poor networks, we evaluate each neuron in three different networks in each generation of the genetic algorithm.

The neural networks we use are fully connected simple recurrent neural networks with a single layer of recurrence [11]. Figure 2 shows the design of the network used for the experiment. In a recurrent neural network, the values in the hidden layer of the network are retrieved from the previous forward propagation of the network and then are used as inputs to the network in the current propagation. This allows information from past observations to influence the current behavior of the network. The position of the maintaining boat relative to the position of the maintain vector defines the neural net input. The outputs of the neural network represent a desired heading and a desired speed for the maintaining boat.

We evaluate an evolving controller by placing a lead boat and a maintaining boat in random positions and with random headings in the open ocean. The lead boat travels along its heading and the maintaining boat must maneuver toward it and then follow at the specified offset. How well the boat maintains its relative position defines fitness within our simulated environment. The next section describes Lagoon, our 3D, naval combat simulation.

V. Lagoon

For our experiment we use the Lagoon real-time 3D naval combat simulation game [12] [13] [14]. Lagoon has several features that make it a good platform for AI research. It has been designed so that it can be run without graphics and can be simulated at speeds faster than real time allowing for quicker evaluations. It also has a flexible controller architecture that made it simple to incorporate neural network based controllers that can be substituted for the normal hand coded controllers. Figure 3 shows a screenshot from Lagoon.

Lagoon provides a complex physics model for moving boats in the simulated environment. For our experiment each boat has linear and rotational velocity, linear and rotational inertia and a friction equation that governs the way that boats move through the water. Each boat is moved by means of a throttle that provides forward thrust and a rudder that applies torque. We controll the rudder and throttle indirectly through the use of a class in Lagoon called the helmsman. Each boat has a desired speed and a desired heading that the helmsman class uses to determine an appropriate throttle and rudder setting for the boat.

VI. Representation

The inputs for the neural network are generated from three vectors based on sensor points surrounding the target point. Figure 4 illustrates the arrangement of the three sensor points around the target location. Sensor point A exists 100 units

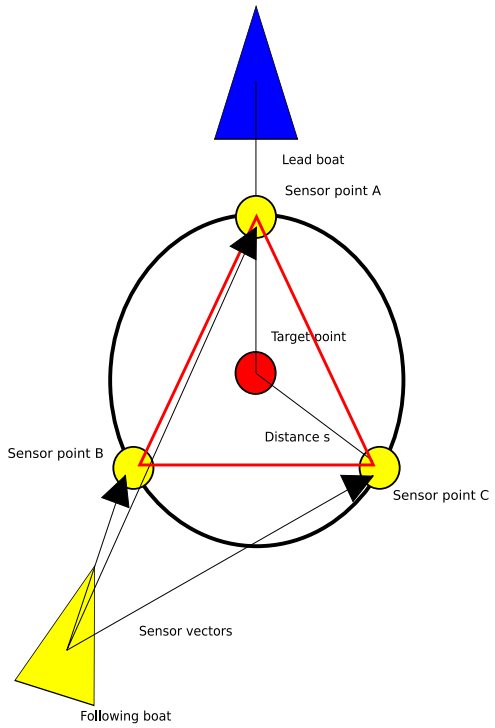

Fig. 4. The Three Vector Sensor

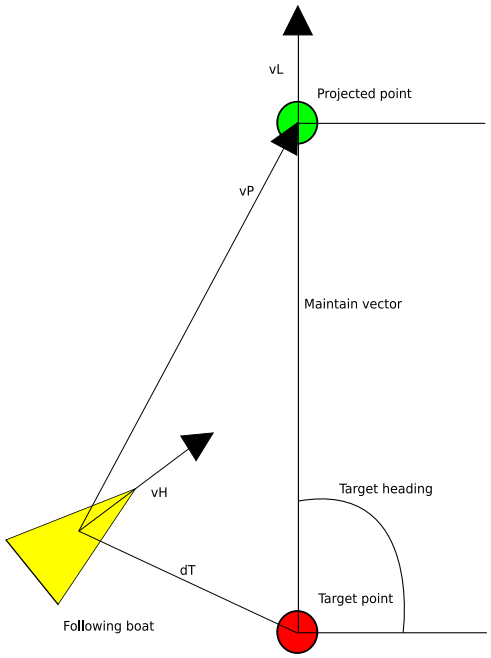

Fig. 5. Fitness Components

from the target point in the direction of the target heading. Sensor points B and C are positioned 100 units from the target point and rotated so that the three points form an equilateral triangle.

To compute the three egocentric sensor vectors, we first compute three angles finding the number of degrees from the location of the following boat to each of the three sensor points. Subtracting the maintaining boat's desired heading from each of these three values now gives us egocentric (with respect to the following boat) rotation angles. These angles are then converted into three normalized vectors. The x and y components of these three normalized vectors compose the six sensor inputs of the network 2. This three vector design was chosen because it not only provides information about the direction to the target point but also provides information that can be used to deduce the rotation of the maintaining boat relative to the maintain vector. Finally, the values of the three vector sensor are continuous from 1 to -1 and back without the break between 0 and 360 degrees that the sensors would report if the values were given directly as angles instead of as vector components.

The outputs of the network 2 control the desired heading and desired speed of the boat. The first output value is scaled so that it represents a value between 36 and -36 degrees. We add this value to the current desired heading of the boat. Lagoon then determines the best rudder position for achieving the desired heading. The second output is scaled between 0 and the maximum full speed of the boat. We use this value directly as the new desired speed for the maintaining boat. Since the neural network has no direct knowledge of it's speed, heading or it's current desired heading it must learn to navigate based entirely on the egocentric angles provided by its sensor.

VII. Fitness

We compared three potential fitness functions to determine which of the three produced the most convincing maintain station behavior. The behavior has three primary objectives

- Match the speed of the lead boat
- Match the heading of the lead boat
- Maintain position on a specified point relative to the lead boats position

However, not all of these objectives are equally important. For example, a boat could keep very close to the target point by driving around it in circles, but this behavior would lack the appearance of intelligence to a human observer making the result undesirable for use in a game or simulation. Conversely, if the following boat perfectly matches the heading of the lead boat it will appear much more intelligent even if it lags some distance from the target point. Thus, each of the three fitness functions was designed to emphasize heading as the most important objective and distance as the second. Matching speed gets rewarded implicitly because the sooner a boat reaches the target point the higher its maximum potential score becomes. Passing the point due to excessive speed results in lowered fitness. Fitness values are calculated at each time step in the simulation and then summed over the course of an entire evaluation.

To determine fitness, three vectors are first calculated based on the location of the boat and the maintain vector. Figure 5 illustrates the three vectors that are used in the fitness calculations. We project vector $v\vec{H}$ in the direction of the following boat's desired heading, vector $v\vec{P}$ from the following boat's position to the projected point and vector $v\vec{L}$ projected in the direction of the target heading. Finally, these three vectors are normalized before use in the fitness function equation. In addition to the three vectors, we calculate a distance dT representing the Euclidean distance between the following boat and the target point. Combining this information together we create three different fitness functions.

Fitness function 1 rewards the maintaining boat for pointing directly at the projected point. We scale the dot product of $v\vec{H}$ with $v\vec{P}$ to return a value between zero when the boat points in the wrong direction to two when the boat points in the correct direction. Scaling of the dot product ensures a positive fitness value. Fitness based on facing the projected point instead of the target point prevents a boat that passes slightly in front of the target point from receiving an inappropriately low fitness.

Multiplying the inverted square root of the distance dT to the dot product encourages the boat to come as near to the target point as possible. However, the distance component plays a secondary role to the heading component because a boat facing in the wrong direction will receive a fitness of zero no matter how close it comes to the target point.

$$(1 + (v\vec{H} \cdot v\vec{P})) * 1/\sqrt{dT} \quad (1)$$

Fitness function 2 rewards the maintaining boat for pointing either directly toward or directly away from the projected point. The lowest fitness values are awarded when facing perpendicularly to the projected point. This logic may seem counterintuitive but was shown to produce very good results in the first part of the experiment. We conjecture that the success of this fitness function results from the increased number of boats that achevie high fitness in the earlier generations of the experiment. In fitness function 1, boats that run from the projected point are severely punished and quickly leave the population. In fitness function 2, these individuals are rewarded and can remain in the population for a long time. Periodically, one of these fleeing individuals can mutate to become a follower instead. Selection pressure will be identical to fitness function 1 in later generations assuming that all fleeing boats have left the population and only followers remain.

This change in fitness results from using the absolute value of the dot product in the equation to ensure a positive value instead of scaling it from zero to two.

$$(abs(v\vec{H} \cdot v\vec{P})) * 1/\sqrt{dT} \quad (2)$$

Fitness function 3 adds a third component to fitness function 1 that considers how well the maintaining boat matches the target heading. In the first two functions, the boats gain reward for facing the projected point and rely on the fact that the projected point moves to encourage them to match the target heading. Of the fitness functions presented, only function 3 explicitly rewards the boats for aligning with the lead boat.

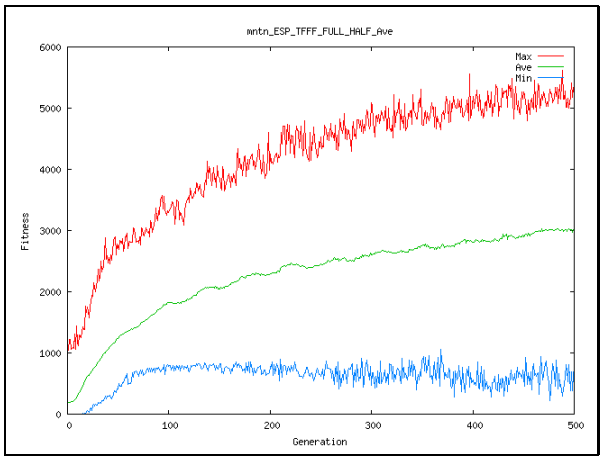

Fig. 6. Experiment 1 - Fitness 1

The new fitness component in function 3 comes from the dot product of $v\vec{H}$ with $v\vec{L}$ scaled to return a value from zero to two.

$$(1 + (v\vec{H} \cdot v\vec{P})) * (1 + (v\vec{H} \cdot v\vec{L})) * 1/\sqrt{dT} \quad (3)$$

VIII. Experiment

We perform two experiments in the paper. In the first experiment, the lead boat travels in a straight line in a random direction at half of its maximum speed. Each evaluation lasts for three minutes in order to give the maintaining boat enough time to be scored adequately for both the acquiring and maintaining phases of the behavior. The speed of the lead boat in the second experiment was selected at random at the start of each evaluation from between 25 and 75 percent of the lead boat's maximum velocity. Cases where the speed of the lead boat approaches either 0 or 100 percent of maximum velocity pose special challenges and are outside the scope of this paper. We perform a different set of experiments for each of the three fitness functions proposed below. Results for the experiments are averaged over ten random seeds.

We use ESP with a population of 200 individuals evolved over 500 generations. We create new populations at each generation using binary tournament selection with a 70 percent probability of single point crossover. Exponential mutation was used with a 10 percent proability and lambda value of 10. Each neural network has six sensor inputs, eight hidden nodes, and two output nodes.

IX. Results

We quantitatively evaluated each of the evolved controllers by comparing their fitness values. Referring to graphs of fitness over time (figures 6, 7, 8, 9, 10, 11) we can see that all of the experiments learned at virtually the same rate and achieved virtually the same maximum fitness (fitness values

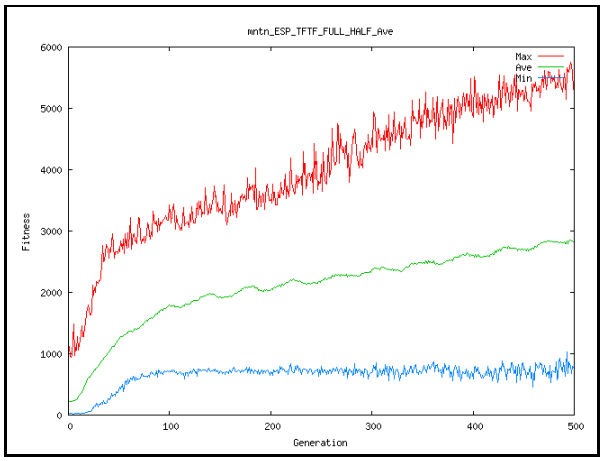

Fig. 7. Experiment 1 - Fitness 2

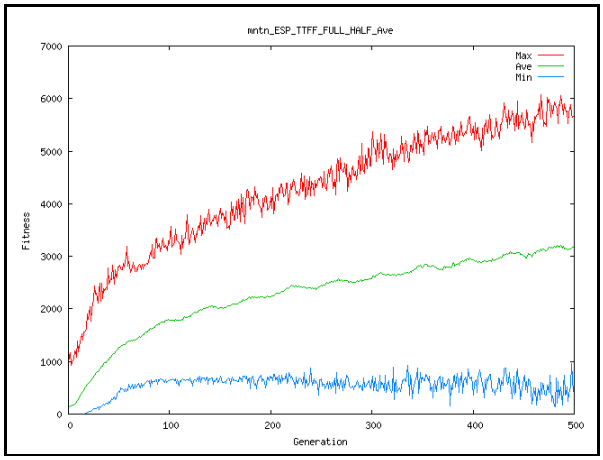

Fig. 8. Experiment 1 - Fitness 3

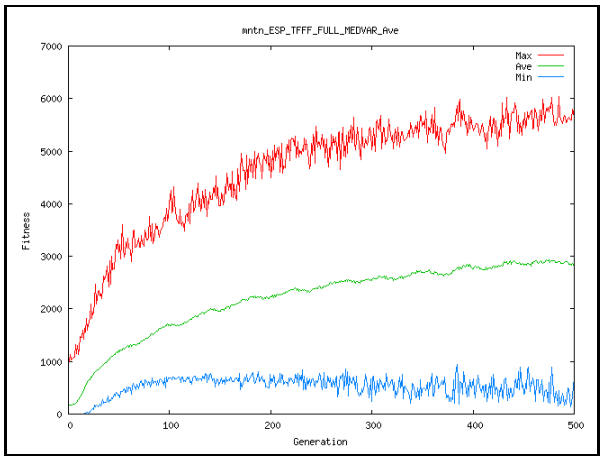

Fig. 9. Experiment 2 - Fitness 1

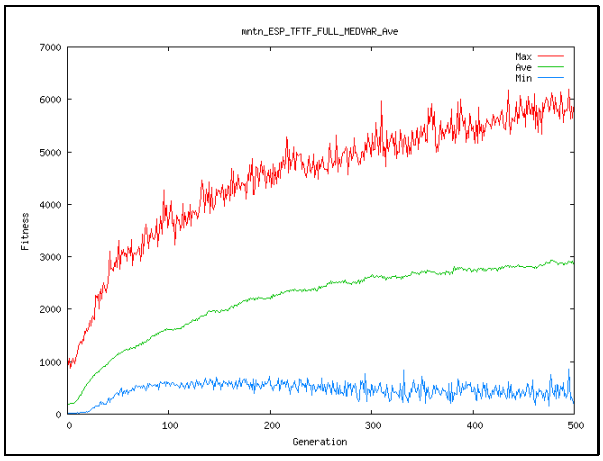

Fig. 10. Experiment 2 - Fitness 2

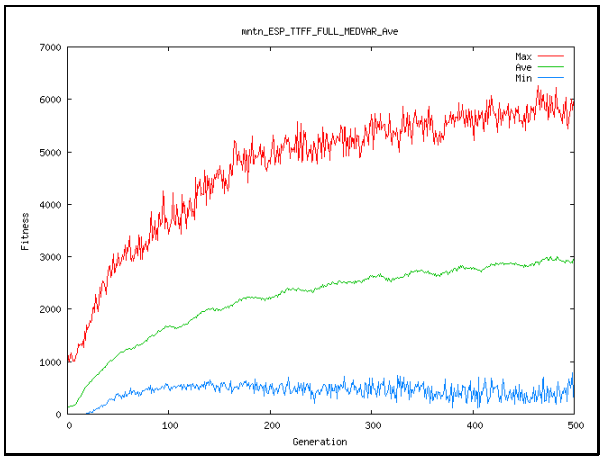

Fig. 11. Experiment 2 - Fitness 3

scaled for comparison) regardless of the fitness function or experimental setup. Each of the graphs displayed shows the average fitness values for an experiment run over ten different random seeds.

In addition to comparing the fitness values of the evolved controllers we visually inspected them to ensure that they could in fact perform the maintain station behavior. All of the controllers were observed starting from at least five random starting locations and in only one out of the three hundred of these random scenarios did a boat completely fail to find and follow the lead boat. This is a 99.67% success rate. The comparison of the graphs together with the observations of the learned behavior suggests that on average the evolved controllers are quantitatively equivalent.

Since all of our fitness functions generated controllers that passed the quantitative examination the next step is to qualitatively examine them to determine which of the controllers would be the best for use in a game or simulation. Qualitatively, we observed that at least 70 percent of the boats in each experiment behaved believably. Several boats were seen to pass the target point very quickly, stop to allow the target point to pass in front of them and then repeat this behavior to follow the lead boat. Other boats followed

the target boat by swinging back and forth in a sinusoidal motion, altering the shape of the wave to maintain proximity to the target point. While these strategies can achieve a high fitness they are not suitable to represent a human driver in a simulation because the behavior is not readily perceived as human like. We suspect that the largest problem across all experiments was that maintaining boats found it difficult to precisely control their speed. Bad controllers such as the stop and go followers and sinusoidal followers descried above may have evolved these behaviors as a way of avoiding the need to control their speed effectively. This result was not very surprising considering that the boats received no direct information about the relative speed of the lead boat. In the best results recurrent observations of the lead boat's position seem to have generated implicit information about the relative speed but this seems to have occurred in only about 10 percent of the observed behaviors.

Discussing the general trends observed in the behaviors created by different fitness functions the boats evolved with function 1 were dubbed the strict followers. These boats had a very optimal strategy for finding the maintaining position that included fast speeds and sharp turns. In general they maintained position on the target point very well but did not tend to approach the point gracefully.

Boats evolved with function 2 were dubbed the casual followers. These boats showed a very graceful approach to the targeted point with slower approach speeds and longer more graceful turns. However these boats are more likely to struggle while maintaining the target point with several of the evolved boats learning to drive either behind or even in parallel with the target point.

Boats created using function 3 were dubbed the sensible followers. These overall were the best looking of all of the results combining the more precise maintaining of the first function with the more graceful maneuverability of the second function. However the most believable of these behaviors were elusive with only about half learning to balance both approach and maintain effectively and the remainder focusing on either one phase of the maintain behavior or the other.

X. Future Work

There are a number of variations of this experiment that could be attempted to try and create more consistant evolution of the maintain station behavior including changing the fitness function or varying the sensor configuration. Specifically relative speed sensors should be added to the network to test our conjecture that the presence of this information could allow maintaining boats to control their speed more precisely. Also the experiment could be expanded to include new behaviors. Now that we have established a good method of evolving a maintain station behavior in the open ocean a logical next step would be to evolve boats that also have some evasive capabilities. A behavior with the ability to maintain formation while avoiding collisions with other boats would be an extremely useful behavior for simulations like Lagoon. The basic problem of avoidance centers around the task of determining when to avoid a nearby boat and when to ignore it. For example boats in a formation should not try to avoid each other unless one of them falls out of position. Future experiments could include expanding the network to contain some radar sensors and then training the networks in the presence of other boats to determine whether or not they would be capable of making this determination.

XI. Summary

In conclusion we have observed that it is possible for a neural network to learn to perform the maintain station behavior in the open ocean. Also we have shown that a fitness function that scores a boat based on it heading and rotation relative to a maintaining vector can consistently evolve robust and believable solutions to this problem. The three fitness functions we tested showed that different solutions to the problem exist, and that while many different approaches to the problem are capable of maximizing a fitness function some of these solutions are more desireable for use in simulated worlds based on their resemblance to perceived human actions.

XII. ACKNOWLEDGMENTS

This material is based upon work supported by the Office of Naval Research under contract number N00014-05-0709.

References

[1] C. W. Anderson, "Learning to control an inverted pendulum using neural networks," *IEEE Control Systems Magazine*, vol. 9, pp. 31–37, April 1989.

[2] T. Balch and R. C. Arkin, "Behavior-based formation control for multirobot teams," *IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION*, vol. 14, pp. 926–939, 1998.

[3] A. Olenderski, M. Nicolescu, and S. J. Louis, "A behavior-based architecture for autonomous ship control," in *Proceedings of the 2006 IEEE Symposium on Computational Intelligence in Games*. New York: IEEE Press, 2006, pp. 148–155.

[4] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[5] J. H. Holland, "Genetic algorithms and adaptation," in *Adaptive Control of Ill Defined Systems*, O. G. Selfridge, E. L. Rissland, and M. A. Arbib, Eds. New York: Plenum Press, 1984.

[6] R. Leigh, S. J. Louis, and C. Miles, "Using a genetic algorithm to explore a*-like pathfinding algorithms," in *Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Games*. New York: IEEE Press, 2007, pp. 72–79.

[7] K. O. Stanley, B. D. Bryant, and R. Miikkulainen, "Evolving neural network agents in the NERO video game," in *Proceedings of the IEEE 2005 Symposium on Computational Intelligence and Games*, 2005.

[8] B. D. Bryant, "Evolving visibly intelligent behavior for embedded game agents," Ph.D. dissertation, Department of Computer Sciences, The University of Texas at Austin, 2006.

[9] B. D. Bryant and R. Miikkulainen, "Acquiring visibly intelligent behavior with example-guided neuroevolution," in *Proceedings of the Twenty-Second National Conference on Artificial Intelligence (AAAI-07)*. Menlo Park, CA: AAAI Press, 2007, pp. 801–808.

[10] F. Gomez and R. Miikkulainen, "2-D pole-balancing with recurrent evolutionary networks," in *Proceedings of the International Conference on Artificial Neural Networks*. Berlin; New York: Springer-Verlag, 1998, pp. 425–430. [Online]. Available: http://nn.cs.utexas.edu/keyword?gomez:icann98

[11] M. I. Jordan and D. E. Rumelhart, "Forward models: Supervised learning with a distal teacher," Center for Cognitive Science, Massachusetts Institute of Technology, Cambridge, MA, Occasional Paper 40, 1990.

[12] C. Miles, *Lagoon - The RTS*, www, http://lagoon.cse.unr.edu, 2006.

[13] C. Miles and S. J. Louis, “Towards the co-evolution of influence map tree based strategy game players,” in *Proceedings of the 2006 IEEE Symposium on Computational Intelligence in Games*. New York: IEEE Press, 2006, pp. 75–82.
[14] ——, “Co-evolving real-time strategy game playing influence map trees with genetic algorithms,” in *Proceedings of the 2006 IEEE World Congress on Computational Intelligence*. New York: IEEE Press, 2006.