Genetic Algorithm and Neural Network Hybrids for Controlling Mobile Robots

Jimmy Secretan
University of Central Florida
Orlando, FL
jsecreta@pegasus.cc.ucf.edu

Guy A. Schiavone
Institute for Simulation and Training
Orlando, FL
guy@ist.ucf.edu

ABSTRACT

As the hardware capabilities of unmanned battlefield robots, such as Micro Aerial Vehicles (MAVs) and Unmanned Ground Vehicles (UGVs), increases, so to must the intelligence of the software controlling them. Genetic Algorithms (GAs) and Genetic Programming (GP) have proven effective in preliminary MAV and UGV simulations for evolving simple tracking and surveillance behaviors. However, the reactive approach that most robotic GAs provide falls short of demonstrating a comprehensive range of intelligence. If for instance, an object becomes occluded from a robot's view, GAs usually must evolve to considerable complexity before they can effectively handle such situations. In this paper, we suggest an approach whereby we augment the GA with a neural network predictor as one of its inputs. The robot's task consists of following another moving object and maintaining a certain distance. The neural network system is trained with the behavior of the robot's intended target, and feeds this as an input to the GA. We present simulation results of how well this method achieves its task, as well as suggestions for adapting these techniques for implementation on advanced mobile cluster computers.

ABOUT THE AUTHORS

Jimmy Secretan is currently a PhD student at the University of Central Florida. He specializes in neural networks, genetic algorithms and Beowulf parallel supercomputing.

Guy Schiavone is an Assistant Professor with the Computer Engineering Department at the University of Central Florida. He has served as PI and Co-PI on several research projects at UCF in various areas of Modeling and Simulation, and Computer Engineering. Dr. Schiavone held previous positions as Visiting Assistant Professor in the UCF Computer Science Department, Senior Research Scientist at University of Michigan, and Research Scientist at Institute for Simulation and Training. He has lead projects and published in the areas of electromagnetics, distributed and parallel processing, terrain databases, 3D modeling, analysis and visualization, and interoperability in distributed simulations for training. He received the Ph.D. in Engineering Science from Dartmouth College in 1994, and the BEEE from Youngstown State University in 1989.

Genetic Algorithm and Neural Network Hybrids for Controlling Mobile Robots

Jimmy Secretan
University of Central Florida
Orlando, FL
jsecreta@pegasus.cc.ucf.edu

Guy A. Schiavone
Institute for Simulation and Training
Orlando, FL
guy@ist.ucf.edu

INTRODUCTION

As robotic navigation begins to move from an experimental to a directly practical domain, intelligence and robustness are becoming more vital. These goals are highlighted by research thrusts such as the DARPA Grand Challenge, in which an autonomous robotic vehicle must travel several hundred miles without human intervention. To achieve tasks such as these, two systems with potential to control groups of mobile, battlefield robots are explored. One, known as Fuzzy ARTMAP (FAM), has been shown to be very effective as a neural network architecture in prediction, classification and function approximation. The other, called SAMUEL, has demonstrated its ability to produce rule-based agents which can effectively enact complex behaviors (e.g. predator/prey type relationships). SAMUEL uses genetic algorithms, a biologically inspired means of "evolving" new solutions to a problem, to find the best set of rules to accomplish its given mission. These systems appear to have two different, but complimentary strengths. FAM has the capability of providing predictions even in the absence of reliable data. This makes it capable of dealing with the uncertainties of sensory data in a complex terrain. SAMUEL is particularly capable with empirically adjusting the parameters of the rules that an agent must be supplied. These aspects suggest that together, the two systems may produce a more effective and robust multiple-robot control system than ones previously presented.

In this paper, we work to build and evaluate such a system. Combining the best aspects of the software control approaches presented by SAMUEL and FAM may give autonomous and semiautonomous robots previously unseen capabilities. The amalgamation of the systems can cover a wider range of intelligent behaviors than either can alone.

SAMUEL

The SAMUEL system was developed initially in the late 80's and early 90's by John Grefenstette, working for the Navy Center for Applied research in Artificial Intelligence. The name SAMUEL is both an acronym for Strategic Acquisition Method Using Empirical Learning and a tribute to Arthur Samuel, one of the progenitors of the concept of automatic programming (Schultz and Grenfenstette, 1990). The system is designed for the autonomous, reactive control of different agents, potentially in a multi-agent environment (Marin, et al., 1999). One of the novelties behind SAMUEL is that it is specifically designed for the kind of multiple-step sequences of decision-making and action sets (Grefenstette, 1997) that are the backbone of many hunting and tracking scenarios. The system uses genetic algorithms among other competitive rule selection systems to evolve an acceptable control program for an agent whose constraints are specified (Grefenstette, 1997). The simple rule sets that are generated by SAMUEL are especially useful because they not only allow other learning algorithms and programs to easily interface to it, but they allow human programmers to both understand and manipulate these rules (Schultz and Grefenstette, 1990).

The SAMUEL software itself is quite robust. The latest iteration of SAMUEL, SAMUEL-97, provides an ANSI C interface and a PVM implementation of the program (Grefenstette, 1997). PVM (Parallel Virtual Machine) is a parallel environment that allows copies of programs to pass messages to each other. This affords SAMUEL the ability to perform its processing in parallel, either on an SMP machine or a Beowulf cluster computer. For each rule set, several different trials must be conducted on its effectiveness. An average fitness for the trials is then computed, which yields the overall fitness of the rule set. The SAMUEL program is parallelized by splitting up the trials among different processing nodes. This means that there is very little communication involved, and the algorithm can enjoy a nearly linear speedup.

SAMUEL is compatible with many versions of UNIX, including Linux. SAMUEL also has support for co-evolution (Grefenstette, 1997). Co-evolution is an interesting augmentation to the concept of genetic algorithms. In co-evolution, there is a certain problem set to accompany the solution set. As the solution set evolves to better surmount the problem set, the problem set is evolved via genetic algorithms to provide more of a challenge to the solution set (Dolin, Bennet and Rieffel, 2002). The guiding assumptions of SAMUEL are described in (Schultz and Grefenstette, 1990) as follows. First, a specified set of sensors will provide certain data to the program, either numerical or symbolic. Second, the agent generated by SAMUEL will be able to affect a distinct set of control variables, also either numerically or symbolically. Finally, the effectiveness of each agent will be evaluated at the end of a "problem-solving episode."

SAMUEL will create populations of rule sets, which within themselves contain the aforementioned Darwinian landscape of weighted rules. In a rule set, several rules are competing for their share of influence on their local system. Then these rule sets, as whole systems, are pitted against each other and globally evaluated on their effectiveness.

The SAMUEL system consists of three primary parts: a "performance module", a "learning module" and a "problem specific module" (Grefenstette, 1997). The learning module is the probably the most important and unique part of the SAMUEL system. It uses genetic algorithms to generate heuristic rule sets. The process begins with a pre-established set of rules, which may have been manually designed, randomly generated or produced by another learning algorithm. The performance module, known as the Competitive Production System (CPS) commands and controls the learning processes of SAMUEL. It is responsible for gathering sensor data and manipulating the control variables (Schultz and Grefenstette, 1990), to orchestrate the production of learning. It is also responsible for obtaining the payoff and resolving conflicts among rules in a rule set (Schultz and Grefenstette, 1990). If two rules do, in fact, compete to be activated during a cycle, it is the job of the CPS to choose the dominant rule based on its given strength (Grefenstette, 1997). The CPS is also given the charge of determining these rule strengths by both identifying how often each rule is active and how much it contributes to the solution in the simulation (Grefenstette, 1997).

Fuzzy ARTMAP

In the period from the 1970s to 1980s, a period that was relatively stagnant for the field of Neural Networks, Grossberg (1976) posited an approach to what is referred to as the *stability/plasticity* problem. This is the challenge that is faced by most machine learning algorithms. First of all, a machine learning system needs to give stable responses when asked to recall knowledge it has learned. An architecture that forgets knowledge too quickly may limit its utility. At the same time, there is a desire for a machine learning system to be plastic. That is, if it is presented with information that it has not seen before, it is able to learn the information relatively quickly, in order to make use of it as soon as possible. An example would be a system trained to recognize different kinds of aircraft. The recognition system starts training and eventually learns all of the aircraft in its training set. But suppose a new aircraft comes along, one that was not in the training set. It is desirable to have the system quickly add the new aircraft to ones it can already recognize.

Grossberg (1976) referred to this solution as Adaptive Resonance Theory (ART). In 1987, Grossberg provided the first implementation of an ART neural network known as ART1 (Christodoulou and Georgiopoulos, 2001). ART1 is considered an unsupervised machine learning algorithm. Unsupervised machine learning algorithms do not depend on an external teacher to aid in its classification. It clusters the data points in to categories of the architecture's own making. Several implementations of ART have followed, including Fuzzy ARTMAP (FAM). The diagram in figure 1 illustrates the design of a simplified FAM architecture only capable of solving classification problems. FAM embodies the stability/plasticity benefits of the original ART1, but augments this with the ability to utilize supervision and floating point inputs. With FAM, an external teacher can classify patterns and eventually have the architecture itself match those pattern classifications.

The FAM algorithm works as follows. Patterns are fed into the system at layer F_0^a, where they are complementary encoded. Complementary encoding involves taking each component of the input vector and subtracting it from one. Each complementary component is then concatenated to the original vector. For an input vector of dimensionality M, the F_0^a layer turns it into a vector of dimensionality 2M. The

complimentary encoding allows the absence of a feature to be as significant as the presence of one.

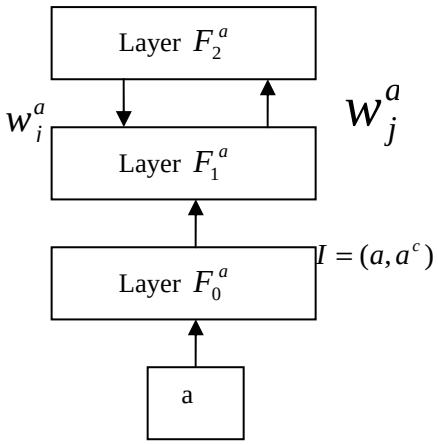

Figure 1. Simplified diagram of the Fuzzy ARTMAP architecture.

The complementary encoded input vectors are then fed to the layer F_1^a. Here they are compared against the architecture's collection of templates. Templates are compressed representations of categories that the architecture builds to classify the patterns. This category representation is somewhat different from the classification imposed by the teacher. All patterns represented by a single template belong to the same teacher imposed category. However, the architecture may produce several templates that all map to the same teacher-imposed category.

For each template the following metric is calculated:

$$T_j = \frac{|I \wedge w_j|}{|w_j| + \alpha}$$

Where "^" is called the fuzzy min operator. The fuzzy min of two vectors is simply a vector where each component is the minimum of the corresponding components in the first two vectors. The magnitude operator here finds what is called the "city block size." That is, it is the addition of every component in the vector.

The T-value, given by the previous formula, determines how similar the input pattern is to the other input patterns that have been previously encoded by the template. The input pattern iterates through the entire collection of templates, looking for the one that gives it the highest T-value. This template is chosen and additional checks are performed to determine if that template should indeed be chosen to encode that input pattern. First the vigilance is calculated between the input pattern and the template. The vigilance is a parameter that determines how closely patterns in the FAM architecture should be clustered. It allows the network to be "vigilant" about the creation of templates. The vigilance parameter is given by the equation:

$$\rho(I, w) = \frac{|I \wedge w|}{|I|}$$

If the input parameter does not meet the baseline vigilance requirement specified for the network, then the baseline vigilance is increased by a small amount and the search for a template begins again. During this next run of the search, the template that the pattern had initially chosen is disqualified. This continues until one is found that meets the vigilance, or what is called the uncommitted node is chosen. The uncommitted node is essentially a blank template that the architecture creates. This node (template) is eventually committed to represent a category of input patterns. When a node is found that passes the vigilance check, it then checks its mapping to the F_2^a layer. The F_2^a layer is a simple way of representing all of the available categories in the classification problem. Every template in the category representation layer (F_1^a) has its own set of weights to the F_2^a layer. These weights all take the form of a vector with one component equal to one and the rest equal to zero. This corresponds to the F_2^a layer, where every node represents a single category. The component of this vector that is equal to one signifies that the template is being mapped to that category.

Finally, if the chosen template meets the vigilance requirements then the template is checked to see if maps to the correct label. If it does then the input pattern is agglomerated into the template by setting the template equal to:

$$w_j = w_j \wedge I$$

LITERATURE REVIEW

Both SAMUEL and Fuzzy ARTMAP neural networks have seen extensive application in the domain of robotic navigation. In (Schultz and Grefenstette, 1990), the Evasive Maneuvers (EM) program enjoys extensive discussion. The EM program simplifies a very ubiquitous military theme: the predator/prey relationship between a target and its aggressor. In

(Schultz and Grefenstette, 1990), it is a fighter aircraft and a heat-seeking missile. The fighter aircraft can sense the position and velocity of the missile. In turn, the missile is given a very basic homing algorithm, to find and destroy the aircraft. The object of the system is to generate evasion programs, through the use of evolutionary procedures, that help the aircraft to survive pursuit by the missile a maximum percentage of times. As an individual is generated through the evolutionary process, its ability to evade the missile is tested in a series of episodes. Its survival percentage becomes its cost function. Instead of simply beginning with a population of randomly generated evasion, the authors seed the population with an evasion program, which they themselves heuristically devised. This program, starting out with a 75% evasion percentage, was eventually evolved into a program with a 95% evasion percentage. It is interesting to note that seeding the initial population with an intelligently designed program like this will often evolve highly effective solutions in a minimum number of generations (Schultz and Grefenstette, 1990).

The SAMUEL learning system has been applied to the control of actual mobile robots, as evidenced by (Schultz and Grefenstette, 1996). The chosen behavior for SAMUEL to evolve was a shepherding behavior for another robot. This is very typical of the animal analogs to which SAMUEL is often applied. The "sheep" robot in this experiment is given a very simple behavior. It avoids other objects, but otherwise proceeds in a random walk. The "shepherd" robot must get within a certain range of the "sheep" without hitting anything, and push it in to a "pasture" area. The system simulates an episode, where it can see how well the current rule set performs with the random placement and action of the system. When the episode completes, the critic module of SAMUEL assigns credit based on how well the shepherd completed the task. If the shepherd pushed the sheep into the pasture within the time limit, then the full amount of credit is assigned as the payoff. If the shepherd did not reach the goal, then partial credit is assigned as the payoff, depending on how close the shepherd was. If the episode ends in a collision, zero credit is assigned. The information that the robots use is actually quite minimal. The shepherd knows the range to the sheep, the sheep's bearing from the current orientation, the range to the pasture and the pasture's bearing from the current orientation. In the 250^{th} generation of the genetic algorithm, SAMUEL reached an 86% success rate. The learned programs were run, with reasonably favorable results, on actual robots. Including errors due to communication and loss of contact with the sheep, the real robot was successful 67% of the time.

In (Bugajska, et al., 2002), a hybrid cognitive-reactive system is presented that uses a system called ACT-R as a high level cognitive framework to drive a low level reactive SAMUEL system. Reactive intelligence is a control architecture that consists of simply responding to inputs. Systems run off of reactive intelligence often have little or no memory and lack any higher level cognition. These systems simply react minute to minute. SAMUEL, when applied to robotics, has been used in an entirely reactive manner. The ACT-R architecture has its origins in theories of human cognition. In practical applications, it serves as a high-level scripting language for effecting intelligent behaviors. The test of intelligent behavior in this study was the control of simulated micro-aerial vehicles (MAVs). First, human participants interacted with the simulation. This yielded the data that would allow the researchers to build the necessary ACT-R models. Then with access to the same simulation, SAMUEL was able to build the necessary rule sets for obstacle avoidance and navigation. To test the effectiveness of the hybrid system, a 6 MAV and a 16 MAV simulation were used. During this experiment, the ACT-R component managed and dispatched the MAVs, while the SAMUEL component ensured that they did not collide with each other and that they were trained on the appropriate targets. As a comparison, a human subject participated in the same simulation. The performance of the hybrid system turned out to be comparable to the human controller.

In (Racz and Dubrawski 1994), FAM is utilized in a robot to give position estimates based on a vector of ultrasonic sensor readings. The robot used in the study was permitted to explore a doorway area. As it went along, added inputs to the FAM consisted of sonar readings for the input vector and odometry based mappings for the classifications. As the FAM completed training, it was found that the network could predict the robot's coordinates relative to the door with only about 3% major misclassification. The authors suggest that this system be used to allow robots to navigate through the use of geometric beacons that already exist in an environment.

Another method of robotic navigation using FAM is presented in (Bonarini, 1996), except with an emphasis on multisensory fusion. Simple robotic sensors have well-known shortcomings. For instance, sonar rangers often used in robotics will not give proper readings if it is ranging an object that has poor acoustic reflectivity. Optical sensors can likewise be fooled by poor optical reflectivity. If using infrared sensors in outdoor environments, there is a high likelihood that the infrared wavelengths present in sunlight could interfere

with the sensors. Researchers have long suggested integrated multisensory approaches to the problem. This study leverages FAM to integrates sonar sensors, light sensors and bump sensors in a robot known as Fuzzy CAT. The light and sonar sensors are each fed into respective FAM networks, which yield an interpretation as to the obstacles that they find. This output symbolically represents where it sees the obstacle. In the next layer of the architecture, these outputs are integrated with the bump sensor outputs (whose readings are much easier to interpret). Then a path is proposed through the obstacles, which is arbitrated by a final layer. Testing this sensor fusion approach in a 100 different trials, the robot only failed to find its way in 9 of those trials. By contrast, using the same system with one sensory system active at a time, the robot failed at up to 40% of the trails and bumped into the path up to 50% more.

In (Streilein, Gaudiano and Carpenter, 1998), a FAM neural network is used for object recognition. Simple sonar sensors are used to recognize objects such as a laboratory wall, an office chair, cardboard boxes, a trashcan and a water bottle. The input to the FAM was a sampling of the raw acoustic signal from the sonar sensor. This was matched with the classification of the object. An interesting aspect to note in this study is that the final system used to test the classification consisted of 5 different FAM neural networks using voting. Each of the networks utilized a different order of pattern presentation. Pattern presentation can have a significant effect on both the effectiveness of the classifier and its size. The accuracy of classification for this system ranged from 70% to 90%, for the different objects.

PROBLEM STATEMENT

For SAMUEL to have practical as well as academic interest, solutions that it generates through simulation must be transferable to real environments. To deal with the increased difficulty of its task, the genetic algorithm may need additional methodologies for acquiring knowledge about its environment.

The addition of the FAM neural network will augment SAMUEL's ability to generate effective rule sets for the specified experimental task. The rule sets generated by SAMUEL using the FAM motion predictor should have better fitness over the same number of generations than the same framework that does not utilize the FAM.

SYSTEM DESIGN

To test the hypothesis, the environment for SAMUEL was intentionally kept as a simple as possible. The robots are modeled as holonomic circles, capable of changing speed with some linear acceleration and turning with some angular acceleration. The environment in which they are simulated to operate contains no obstructions other than four walls at the North, East, South and West. In (Schultz and Grefenstette, 1996), the robot used to physically verify the effectiveness of the models which SAMUEL had generated was the Nomad 200. The Nomad 200 is a popular robot manufactured by Nomadic Technologies, Inc. It is equipped with a programmable onboard PC, along with 16 sonar sensors, 16 infrared sensors and a tactile "bump" sensor (Schultz and Grefenstette, 1996). The one in this particular study was outfitted with a structured light range finder. The environment in which the Nomad was utilized was also simple. The experiment was done in an indoor lab with flat, easily traversable doors and large obstacles.

The robot's responses remained the same in this study; that is, it was still only able to control its speed and wheel direction. Because in (Schultz and Grefenstette, 1996), the only obstacles in the environment were walls and the target, the sensory technology was kept relatively simple.

Figure 2 depicts the overall design of the simulation system. As previously specified, the robot model interacts with the environment model, both moving the robot through the environment and receiving observations. The robot is driven by the decisions from the SAMUEL module. In order to arrive at those decisions, SAMUEL consults with the FAM, from which it receives predictions and to which it provides inputs.

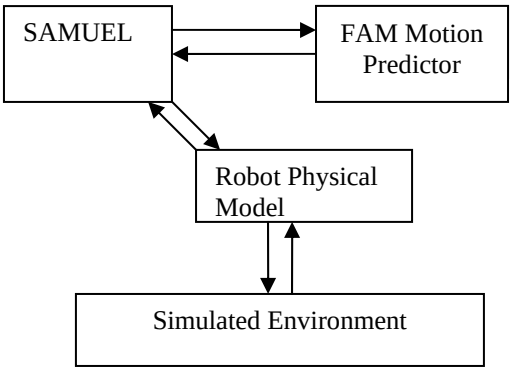

Figure 2. Design of the SAMUEL simulation.

The main design principle of the FAM motion predictor was that given sequence of sensory inputs from previous time periods, the FAM should be able to output the approximate expected position of the other robot. The hope is that the target robot will follow some predictable pattern when traversing over the terrain. Since we fix the rule set of the target robot in the simulation, this assumption is well founded. It is then up the FAM predictor to determine how well it can predict this rule set based solely on observations from its interactions with the target. Were this system to be used in practice, we would not have *a priori* knowledge of this rule set, so it is not utilized directly by the evolutionary process or the FAM. Figure 3 depicts the high level design of the FAM motion predictor.

Three different sensory inputs to the robot were used to train the FAM. These include target range, target bearing and target heading. The input vector was constructed out of a number of these aggregate sensory readings. It was arbitrarily chosen for eight sets of points to comprise each input vector. The number of sets to use in the input vector could very feasibly have a significant effect on the effectiveness of the FAM. Analysis in further studies may reveal the effect of this choice.

To compose an input / output pair, capable of being read by a neural network, a corresponding output point is also needed. Though a number of different FAM predictors may be implemented concurrently within the same framework, it was decided that the experiment would only concentrate on the prediction of a single parameter: that is the target bearing. It is hoped the genetic algorithm will use this data to adjust its motion accordingly. Though FAM is capable of effective function approximation with real numbers, it was decided to simplify the task of the FAM by using discrete representations. This allowed ease of programming and speed of execution. Also, it allowed FAM to capitalize its greater proclivity for classification. Again, representation by 16 discrete values, representing the range of the target bearing variable were used.

The FAM predictor consisted of both a FAM neural network and what was labeled a Motion Cache. The Motion Cache served as a high level interface to SAMUEL simply allowing sensory values to be input from SAMUEL's inner loops and extracting a prediction thereafter. The Motion Cache was based primarily on a list that would add new sets of data points to end, and remove sets from the beginning, keeping the cache sized fixed. The cache, in reality, held nine items, though eight were used as the input vector. The ninth item, the last to be added, was used to classify the target bearing that was the result of the previous eight motion points. Immediately thereafter, that input vector lost its first set of points and concatenated the last set added. It was then asked for a prediction. This method of continuous training and use for FAM is known as on-line learning.

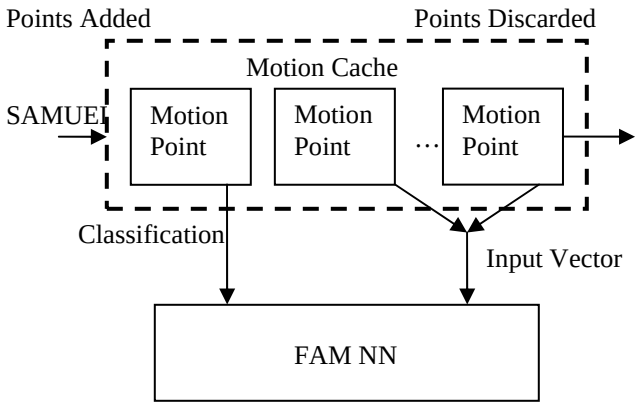

Figure 3. Design of the FAM motion predictor.

EXPERIMENTS

The experiment design mostly matched the work done in (Schultz and Grefenstette, 1996). The experiment consisted of two simulated robots. The first robot served as the target. It used a fixed rule set in the SAMUEL system. The rule set was the same one used in (Schultz and Grefenstette, 1996), slightly modified for the accommodation of a FAM prediction value.

The second robot operated utilizing a genetically modifiable rule set. Its only objective was to track the target robot and maintain a certain distance. The fitness function was calculated according to how well it could maintain its distance. The robot had points deducted if it got closer than half of the specified distance or farther away than one and half times the specified distance. The final metrics gathered by the system were the percentage of the time that the rule set was effective. If the rule set was effective throughout the entire episode, it received 100% of the credit. If it collided with the other robot, it received 0%. If it strayed outside of its accepted range, it would receive partial credit according to the amount of time it managed to stay within the range. The final fitness function was the average of these percentages of all of the rule set's episodes.

Two experiments were conducted. The metric that the experiments were designed to optimize was the highest percentage successful attempts at the task. The first experiment acted as a control experiment, the purpose of which was to see how well the SAMUEL algorithm worked with its new set of sensors in a more complex environment. The second experiment tested the same system with the addition of the FAM predictor. The robots in both experiments used the same set of starting positions for the robot.

Each of the experiments ran over 100 generations, where each generation had 100 rulesets being tested. For each of those 100 rule sets, 20 different episodes were used to assess its fitness. Each of those episodes had a maximum of 60 simulation steps, however, the simulation and credit assignment would end early if two robots collided. In the SAMUEL code, a simulation step is composed of 5 substeps of time. The mutation rate that was used in these experiments is 1%.

Since FAM, has the two aforementioned adjustable parameters, an effort was made to optimize the ones used in the simulations. The choice parameter has relatively little effect when kept within a reasonable range. For the simulations, the choice parameter was fixed at 0.01, a value suggested reasonable by previous literature. For the vigilance parameter, experiments were conducted in the range from 0 to 1, as shown in figure 4. The optimal vigilance of 0.85 yielded by the experimentation was used in all subsequent simulations.

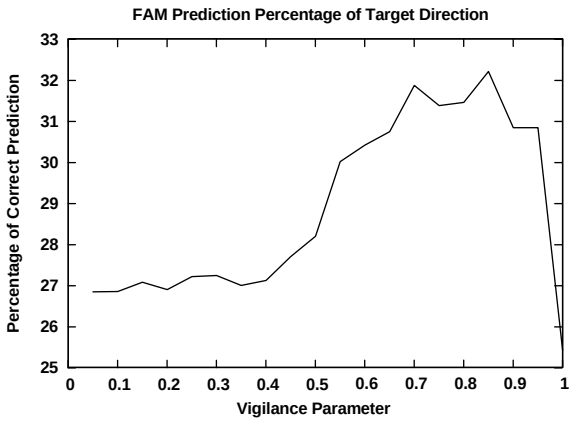

Figure 4. FAM correct prediction percentage for different vigilance values.

RESULTS

The FAM predictor in the preliminary tests appeared to work relatively well. It was able to guess exactly the bearing of the target robot about 32% of the time with the optimal vigilance. Since there are 16 possible discrete bearings, this is obviously more accurate than a random guess. This accuracy is evaluated with the realistic scenario of an online training mode. That is, the robot is making predictions in real time. Thus, it is required to make predictions even when it is starting out and its network is relatively untrained. As the amount of training increases, it is reasonable to expect the accuracy to increase. Predictions that are close, but not exact may also assist the robot in achieving its goals.

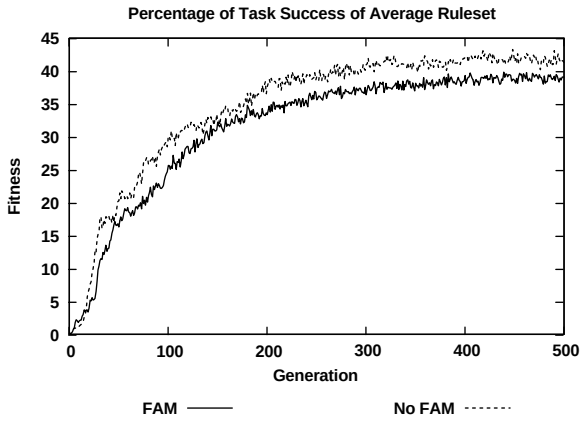

Figure 5. Performance of the system with FAM and without FAM.

Figure 5 provides an average of 5 different runs of each version of SAMUEL. In the trial runs, it appears that the FAM only provided an advantage during about the first 15 generations. At that point, the rule sets that did not incorporate FAM stayed a few percent ahead. This could be for a number of reasons. The task may have been too simple to incorporate a prediction usefully. Also, the prediction algorithm may not have had the necessary accuracy to properly guide the robot.

FUTURE WORK

Future work may focus on a more difficult navigational task, which will likely include a 3-d terrain. Future versions will incorporate different classification algorithms, such as backpropagation neural networks and Bayesian Belief networks, in the same framework. If it is possible to find an algorithm that yields better prediction results, then it may be postulated that this would allow better performance from the genetic algorithm.

To evaluate the utility of the models used by the system, a robot running in an actual terrain would ideally be used. Results from previous studies suggest that the models that SAMUEL evolves are robust enough to provide significant correlation between simulations and experiments. Future work may center around further validation of this system in complex terrains and on actual robots.

The computational requirements for generating the SAMUEL rulesets and for training the FAM are significant. However, operating both systems is less intensive. This leads to two different options when employed in a production system. Training could be done before hand on a cluster computer, and the final rulesets and FAM network uploaded the UGV. However, this leaves the UGV incapable of the flexibility that is one of the primary advantages of SAMUEL. An alternative is the use of mobile Beowulf clusters. There are already a few examples of mobile, low-power Beowulf clusters in the literature being used for intensive computation in mobile environments. Recently, (Chung et. al., 2002) have built mobile cluster powered robot known as the "Beobot." The Beobot uses two industrial dual processor motherboard cards, with Pentium III processors. The robot is in use for evaluating machine vision algorithms. Sandia National Laboratories has a "Lunchbox Beowulf", consisting of four PC/104 form Pentium IIs (Williams and Armstrong, 2002). In (Schiavone, et al, 2003) three candidate systems were developed to evaluate the feasibility of mobile cluster computing. The systems included one comprised of a host PC and daughter PC boards, one utilizing 9 StrongArm processors, and one based on low power VIA boards, common in network appliances. The systems had power consumption in the range of about 20W to 300W, making mobile computing a very real possibility.

REFERENCES

Bonarini, A. (1996). "Symbol grounding and a Neuro-Fuzzy architecture for multisensor fusion," *Intelligent Automation and Control: recent trends in developing applications (Proceedings of the WAC-ISRAM '96)* (Albuquerque, NM.).

Bugajska, M., Schultz, A., Trafton, J., Taylor, M., Mintz, F. (2002), "A Hybrid Cognitive-Reactive Multi-Agent Controller," Proceedings of the 2002 IEEE/IRSJ International Conference on Intelligent Robotics and Systems, EPFL, Switzerland, 2002.

Christodoulou, C. and Georgiopoulos, M. (2001), *Applications of Neural Networks in Electromagnetics*, Artech House, Boston.

Chung, D., Hirata, R., Mundhenk, T. N., Ng, J, Peters, R. J., Pichon, E., Tsui, A., Ventrice, T, Walther, D,. Williams, P., & Itti,, L. (2002), "A New Robotics Platform for Neuromorphic Vision: Beobots", *Proc. 2nd Workshop on Biologically Motivated Computer Vision* (BMCV'02), Tuebingen, Germany, pp. 558-566, Nov 2002.

Dolin, B., Bennet, F. and Rieffel, E.G. (2002), "Co-Evolving an Effective Fitness Sample: Experiments in Symbolic Regression and Distributed Robot Control," Proceedings of the 2002 ACM Symposium on Applied Computing.

Grefenstette, JJ., *The Users Guide to SAMUEL-97: An Evolutionary Learning System*, Retrieved March 25, 2003 from the World Wide Web: http://www.aic.nrl.navy.mil/~schultz/samuel/docs/.

Grossberg, S. (1976). "Adaptive Pattern Recognition and Universal Recording II: Feedback, Expectation, Olfaction, and Illusions," *Biological Cybernetics*, vol. 23, pp. 187-202.

Marin, J.A., Radtke, R., Innis, D., Barr, D., Schultz, A.C. (1999), "Using A Genetic Algorithm to Develop Rules to Guide Unmanned Aerial Vehicles," IEEE Systems, Man and Cybernetics '99 Conference Proceedings.

Racz, J. and Dubrawski, A. (1994), "Mobile Robot Localization With an Artificial Neural Network," International Workshop on Intelligent Robotic Systems IRS '94, Grenoble, France, July, 1994.

Schiavone, G., Dolezal, M., Tracy, J., Secretan, J., and Mangold, L. (2003). "Beowulf Supercomputing for Mobile Applications," *Proceedings of the I/ITSEC Conference*, Orlando, FL, Dec. 2003.

Schultz, A.C. and Grefenstette, J.J. (1990), "Improving Tactical Plans with Genetic Algorithms," Proceedings of the 2nd International IEEE Conference on Tools for Artificial Intelligence, November 6-9, 1990.

Schultz, A.C., Grefenstette, J.J., Williams, A. (1996). "RoboShepherd: Learning a complex behavior," in *Proceedings of the International Symposium on Robotics and Automation* (Washington, D.C. pp. 763-768).

Streilein, W., Gaudiano, P., Carpenter, G. (1998), "A Neural Network for Object Recognition through Sonar on a Mobile Robot," IEEE ISIC/CIRA/ISAS Joint Conference, Gaithersburg, MD, September 14-17, 1998.

Williams, M., & Armstrong, R. (2002). "Building a Linux Minicluster using commodity components". Retrieved June 25, 2003 from http://www.linuxdevices.com/articles/AT2143783710.html