Soft Computing Paradigms for Hybrid Fuzzy Controllers: Experiments and applications*

M. R. Akbarzadeh-T., E. Tunstel[†], K. Kumbla[‡] & M. Jamshidi
NASA Center for Autonomous Control Engineering (ACE)
University of New Mexico, Albuquerque, NM 87131, USA
E-mail: akbazar@eece.unm.edu

Abstract

Neural Networks (NN), Genetic Algorithms (GA), and Genetic Programs (GP) are often augmented with fuzzy logic-based schemes to enhance artificial intelligence of a given system. Such hybrid combinations are expected to exhibit added intelligence, adaptation, and learning ability. In this paper, implementation of three hybrid fuzzy controllers are discussed and verified by experimental results. These hybrid controllers consist of a hierarchical NN-fuzzy controller applied to a direct drive motor, a GA-fuzzy hierarchical controller applied to a flexible robot link, and a GP-fuzzy behavior-based controller applied to a mobile robot navigation task. It is experimentally shown that all three architectures are capable of significantly improving the system response.

1 Introduction

Traditional methods which address robotics control issues rely upon strong mathematical modeling and analysis. The various approaches proposed to date are suitable for control of industrial robots and automatic guided vehicles which operate in *structured* environments and perform simple repetitive tasks that require only end-effector positioning or motion along fixed paths. However, operations in unstructured environments require robots to perform more complex tasks for which analytical models for control can often not be determined. In cases where models are available, it is questionable whether or not uncertainty and imprecision are sufficiently accounted for. Under such conditions fuzzy logic control is an attractive alternative that can be successfully implemented on real-time complex systems. Fuzzy controllers and their hybridization with other paradigms are robust in the presence of perturbations, easy to design and implement, and efficient for systems that deal with continuous variables. The control schemes described in this paper are examples of approaches that augment fuzzy logic with other soft computing techniques to achieve the level of intelligence required of complex robotic systems.

Three soft computing hybrid fuzzy paradigms for automated learning in robotic systems are briefly described and experimentally verified. The first scheme concentrates on a methodology that uses NNs to adapt a fuzzy logic controller (FLC) in manipulator control tasks. The second paradigm develops a two-level hierarchical fuzzy control structure for flexible manipulators. It incorporates GAs in a learning scheme to adapt to various environmental conditions. The third paradigm employs GP to evolve rules for fuzzy-behaviors to be used in mobile robot control. Experimental results of fuzzy controllers learned with the aid of these soft computing paradigms are presented.

2 Neuro-Fuzzy System

Neural networks exhibit the ability to learn patterns of static or dynamical systems. In the following neuro-fuzzy approach, the learning and pattern recognition of NN are exploited in two stages: first, to learn static response curves of a given system; and second, to learn the real-time dynamical changes in *a* system to serve as *a* reference model. The neuro-fuzzy control architecture uses the two neural networks to modify the parameters of an adaptive FLC. The adaptive capability of the fuzzy controller is manifested in a rule generation mechanism and automatic adjustment of scaling factors or shapes of membership functions. The NN functions as a classifier of the system's temporal responses. A multi-layer perceptron NN is used to classify the tem-

*This work was supported in part by Waste Education and Research Consortium grand under award #WERC/NMSU/DOE Amd 35 and by NASA contract # NCCW-0087

[†] Currently with Jet Propulsion Laboratory, Pasadena, CA.

[‡] Currently with Kirtland Airforce Base in Albuquerque, New Mexico.

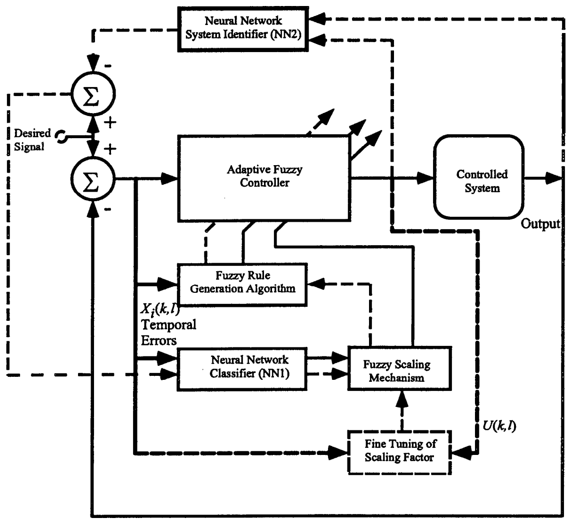
Neural Network System Identifier (NN2)
Desired Signal
Adaptive Fuzzy Controller
Controlled System
Output
Fuzzy Rule Generation Algorithm
$X_i(k,l)$ Temporal Errors
Neural Network Classifier (NN1)
Fuzzy Scaling Mechanism
$U(k,l)$
Fine Tuning of Scaling Factor

Figure 1: Block Diagram of the Adaptive Neuro-Fuzzy Controller

poral response of the system into different patterns. Depending on the type of pattern such as "response with overshoot", "damped response", "oscillating response" etc. the scaling factor of the input and output membership functions are adjusted to make the system respond in a desired manner. The rule generation mechanism also utilizes the temporal response of the system to evaluate new fuzzy rules. The non-redundant rules are appended to the existing rule base during the tuning cycles. This controller architecture is used in real-time to control a direct drive motor. Figure 1 illustrate the architecture of the Neuro-fuzzy controller where the two NNs and the fuzzy control architecture are integrated for adaptive control of nonlinear systems [1].

2.1 Real-Time Adaptive Control of a Direct Drive Motor

In order to perform real-time control, it is necessary that the controller to stand alone with the sole task of calculating the output needed to control the object system. This means the task of communicating data for storing as well as acquiring controller parameters (if the controller is adaptive) should be performed by external processors. In this way a real-time control can be achieved with required sampling rate for high bandwidth operation.

The FLC algorithm requires processing of several functionalities such as fuzzification, inferencing and defuzzification. This means the computation time taken by the FLC itself does not leave any room for an adaptive algorithm such as rule generation, calculating the scale factor of the membership function, or NN algorithms. In order to implement all these functionalities, a multi-processing architecture is needed. This can be achieved by combining a sufficiently fast processor specifically designed for real-time processing, such as a TMS320C30 digital signal processor (DSP) combined with a PC Intel processor (Pentium or 486). Figure 2 shows the hardware built to interface and control a direct drive motor.

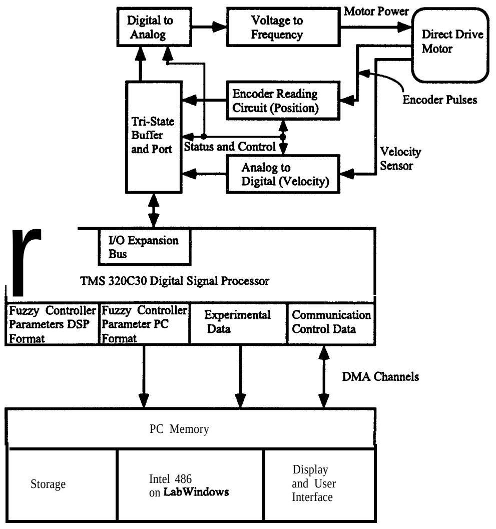
Digital to Analog
Voltage to Frequency
Motor Power
Direct Drive Motor
Encoder Reading Circuit (Position)
Encoder Pulses
Tri-State Buffer and Port
Status and Control
Analog to Digital (Velocity)
Velocity Sensor
I/O Expansion Bus
TMS 320C30 Digital Signal Processor
Fuzzy Controller Parameters DSP Format
Fuzzy Controller Parameter PC Format
Experimental Data
Communication Control Data
DMA Channels
PC Memory
Storage
Intel 486 on LabWindows
Display and User Interface

Figure 2: Hardware for Implementing Neuro-Fuzzy Controller for Real-Time Control of a Direct Drive Motor Using a Digital Signal Processor

The dynamics of a direct drive brushless DC Motor exhibits nonlinear characteristics. This is evident in the dead-zone regions of operation and frequency load characteristics. A *digital to analog converter* of 12-bit resolution is used to interface to the DSP board through a memory mapped register I/O port. The position of the motor is measured by an optical encoder (which generates 8000 pulses per revolution of the motor). This is converted to digital 12-bit position data by encoder circuitry. The DSP's expansion memory is accessible to the PC with which the data communication is carried out using *direct memory access* (DMA). The experimental data as well as fuzzy controller parameter and communication control data are sent back and forth using this DMA.

Figure 3(a) shows the result of the experiment. Initially there are no rules in the fuzzy controller. Hence for the first two cycles when the motor was commanded to go from +1000 to -1000 encoder readings, there is no action and the motor is stationary (O-1000 sampling time; each cycle corresponds to 500 sampled data).

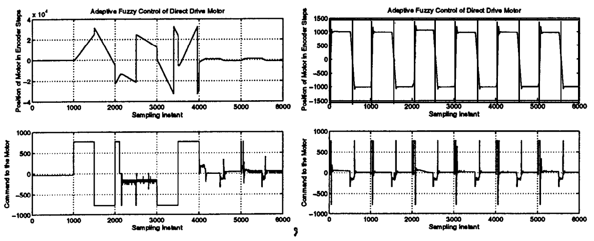
Figure 3: Position Control Direct Drive Motor: Response After (a) First and (b) Final Trial

However, in the third cycle when the motor is commanded to go to +1000 counts, the motor spins out of control clockwise. This is because the rule generation mechanism has produced 4 new rules in the last two cycles and they are in action. These rules are unstable because the rules generated are not the right ones or may be insufficient to control the motor adequately. This unstable behavior continues until after the 8th tuning cycle (after 4000 sampling instant). The corresponding motor command shows a bounded region., Figure 3(b) shows the stabilized response. Here, the fuzzy controller has completely learned to control the direct drive motor.

3 GA-Fuzzy Systems for Control of Flexible Robots

In this section, GAs are applied to fuzzy control of a single link flexible arm. GAs are guided probabilistic search routines modeled after the mechanics of Darwinian theory of natural evolution [2]. Genetic algorithms have demonstrated the coding ability to represent parameters of fuzzy knowledge domains such as fuzzy rule sets and membership functions [3] in a genetic structure, and hence are applicable to optimization of fuzzy rule-sets.

Several issues should be addressed when designing a GA for optimizing fuzzy controllers: the design of a transformation (interpretation) function, the method of incorporating initial expert knowledge, and the choice of an appropriate fitness function. Each of the above issues significantly influences the success of GA in finding improved solutions. These issues are briefly discussed below as they apply to design of a GA-fuzzy controller for a flexible link.

3.1 Application to Flexible Robot Control

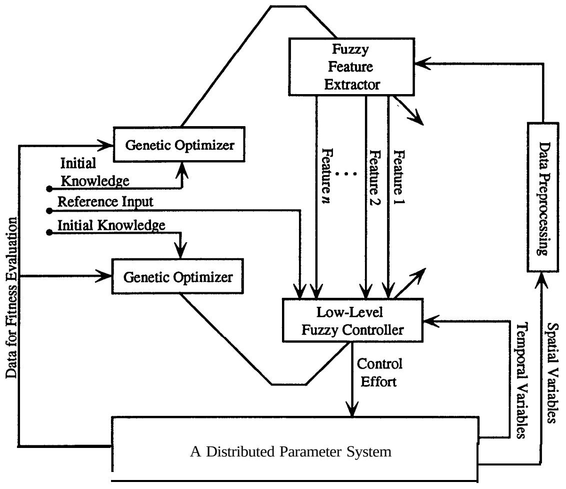

Figure 4: GA-Based Learning Hierarchical Control Architecture

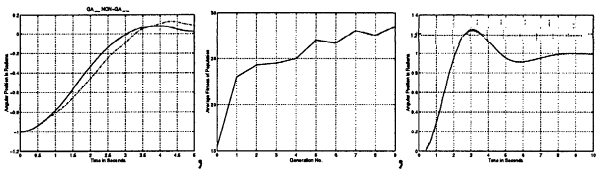
Figure 5: GA Simulation (a) Comparison of Simulation Responses (b) Plot of Average Fitness (c) Initial Experimental Results

The GA-learning hierarchical fuzzy control architecture is shown in Figure 4. Within the hierarchical control architecture, the higher level module serves as a fuzzy classifier by determining spatial features of the arm such as *straight, oscillatory,* curved. This information is supplied to the lower level of hierarchy where it is processed among other sensory information such as errors in position and velocity for the purpose of determining a desirable control input (torque). In [5] this control system is simulated using only a priori expert knowledge. In the given structure, a genetic algorithm fine tunes parameters of membership functions.

The following fitness function was used to evaluate individuals within a population of potential solutions,

$$fitness = \int_{t_i}^{t_f} \frac{1}{e^2+\gamma^2+1} dt, \tag{1}$$

where e represents the error in angular position and γ represents overshoot. Consequently, a fitter individual is an individual with a lower overshoot and a lower overall error (shorter rise time) in its time response.

Here, results from previous simulations of the architecture are applied experimentally. The method of *grand-parenting* [6] was used to create the initial population. Members of the initial population are made up of mutation of the knowledgeable *grandparent(s).* As a result, *a* higher fit initial population results in a faster rate of convergence as is exhibited in Figure 5(a). Figure *5(a)* shows the time response of the GA-optimized controller when compared to previously obtained results through the non-GA fuzzy controller. The rise time is improved by 0.34 seconds (an 11% improvement), and the overshoot is reduced by 0.07 radians (a 54% improvement). The average fitness of each generation is shown in Figure 5(b). A total of 10 generations were simulated. Mutation rate for creating the initial population was set at 0.1. Mutation rate throughout the rest of the simulation was set to 0.01. Probability of crossover *was* set to 0.6. Initial experimental results demonstrate that the GA learned controller is able to control the actual experimental system as in Figure 5(c).

The hardware used to implement the above algorithms is the same *as* was explained in the previous section with few modifications pertaining to flexible robot control such as tip end position sensor and several strain gauges distributed evenly across the length of the flexible beam.

4 GP-Fuzzy Hierarchical Behavior Control

The robot control benefits to be gained from soft computing-based hybrid FLCs is not limited to rigid and flexible manipulators. Similar benefits can be gained in applications to control of mobile robot behavior. Autonomous navigation behavior in mobile robots can be decomposed into a finite number of special-purpose task-achieving behaviors. An effective arrangement of behaviors as a hierarchical network of distributed fuzzy rule-bases was recently proposed for autonomous navigation in unstructured environments [8]. The proposed approach represents a hybrid control scheme incorporating fuzzy logic theory into the framework of behavior-based control. A behavior hierarchy that encompasses some necessary capabilities for autonomous navigation in indoor environments is shown in Figure 6. It implies that goal-directed navigation can be decomposed as a behavioral function of goal-seeking and route following. These behaviors can be further decomposed into the lower-level behaviors shown, with dependencies indicated by the adjoining lines. Each block in Figure 6 is a set of fuzzy logic rules. The circles in the figure represent dynamically adjustable weights in the unit interval which specify the degree to which

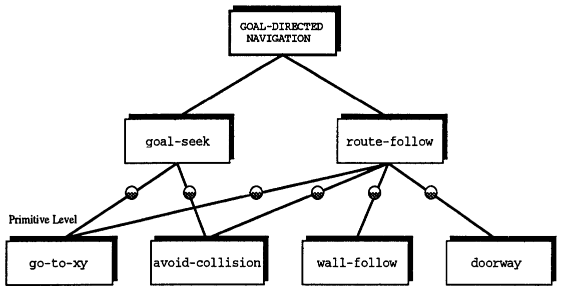

Figure 6: Hierarchical decomposition of mobile robot behavior.

low-level behaviors can influence control of the robot's actuators. Higher-level behaviors consist of fuzzy decision rules which specify these weights according to goal and sensory information. Each low-level behavior consists of fuzzy control rules which prescribe motor control inputs that serve to achieve the behavior's designated task.

The functionality of this hierarchical fuzzy-behavior control approach depends on a combined effect of the behavioral functionality of each low-level behavior and the competence of the higher-level behaviors which coordinate them. Perhaps the most difficult aspect of applying the approach is the formulation of fuzzy rules for the higher level behaviors. This is not entirely intuitive, and expert knowledge on concurrent coordination of fuzzy-behaviors is not readily available. We have successfully addressed this issue in [9] using GP to computationally evolve rules for composite behaviors. In this section, we describe the genetic programming approach to fuzzy rule-base learning. Next, we present a representative experimental result of applying the behavior hierarchy' to autonomous goal-seeking.

4.1 GP-Fuzzy approach

The GP paradigm [10] computationally simulates the Darwinian evolution process by applying fitness-based selection and genetic operators to a population of individuals. Each individual represents a computer program of *a* given programming language, and is a candidate solution to a particular problem. The programs are structured as hierarchical compositions of functions (in a set *F)* and terminals (function arguments in a set T). The population of programs evolves over time in response to selective pressure induced by the relative fitnesses of the programs for solving the problem. For the purpose of evolving fuzzy rule-bases, the search space is contained in the set of all possible rule-bases that can be composed recursively from *F* and *T.* The set, *F,* consists of components of the generic *if-then* rule and common fuzzy logic connectives, i.e. functions for antecedents, consequents, fuzzy intersection, rule inference, and fuzzy union [9]. The set, T, is made up of the input and output linguistic variables and the corresponding membership functions associated with the problem. A rule-base that could potentially evolve from F and T can be expressed as a tree data structure with symbolic elements of F occupying internal nodes, and symbolic elements of T as leaf nodes of the tree. This tree structure of symbolic elements is the main feature which distinguishes GP from GAs which use the numerical string representation.

All rule-bases in the initial population are randomly created, but descendant populations are created primarily by reproduction and crossover operations on rule-base tree structures. For the reproduction operation several rule-bases selected based on superior fitness are copied from the current population into the next, i.e. the new generation. The crossover operation starts with two parental rule-bases and produces two offspring that are added to the new generation. The operation begins by independently selecting one random node (using uniform probability distribution) from each parent *as* the respective crossover point. The subtrees subtended from crossover nodes are then swapped between the parents to produce the two offspring. GP cycles through the current population performing fitness evaluation and application of genetic operators to create a new population. The cycle repeats on a generation by generation basis until satisfaction of termination criteria (e.g. lack of improvement, maximum generation reached, etc). The GP result is the best-fit rule-base that appeared in any generation.

In the GP approach to evolution of fuzzy rule-bases, the same fuzzy linguistic terms and operators that comprise the genes and chromosome persist in the phenotype. Thus, the use of GP allows direct manipulation of the actual linguistic rule representation of fuzzy rule-based systems. Furthermore, the dynamic variability of the representation allows for rule-bases of various sizes and different numbers of rules. This enhances population diversity which is important for the success of the GP system, and any evolutionary algorithm for that matter. The dynamic variability also increases the potential for discovering rule-bases of smaller sizes than necessary for completeness, but sufficient for realizing desired behavior.

4.2 Real-time navigation

GP *was* used to evolve fuzzy decision rules to be used for goal-seeking by a mobile robot. Population sizes of 10–20 rule-bases were run for a number of generations ranging from 10-15. In GP, genetic diversity remains high even for very small populations due to the tree structure of individuals [10]. The experimental testbed is a custom-built mobile robot driven by a two-wheel differential configuration with two passive casters for support. The independent drive motors are geared DC motors. The robot stands about 75 cm tall and measures about 60 cm in width. Range sensing is achieved using a layout of 16 ultrasonic transducers; optical encoders on each driven wheel provide position information used for dead-reckoning. Its maximum speed was limited to $0.3m/s$. The vehicle is controlled using a 75MHz Pentium-based master processor (laptop PC) and Motorola MC68HC11 microprocessor slaves for sonar processing and low-level motor control functions. In the current implementation, the cycle time of the control system is 0.15 seconds (7Hz). This includes time spent acquiring and preprocessing sonar and encoder data, and commanding the motors. The overall inference of the behavior hierarchy consumes about 0.05 seconds of this time. That is, the hierarchy itself can run at a rate of 20Hz.

During operation, the robot is not provided with an explicit map of the environment. However, it is cognizant of the notion of a two-dimensional Cartesian coordinate system. Its paths are not pre-planned; they are executed in response to instantaneous sensory feedback from the environment. Therefore, we are essentially dealing with a local navigation problem as opposed to a global navigation problem which relies on a global map that is either provided a priori, or is acquired via exploration. The experiment was conducted in an indoor environment consisting of corridors and doors. The robot's task is to navigate from one location to another on the same floor of the building. The result of the navigation task is shown in Figure 7 as the path traveled in a portion of the indoor test environment which includes the start and commanded goal. The robot is displayed as an octagonal icon with a radial line indicating its heading. It *was* commanded to navigate from a start pose, $(x\ y\ \theta)^T = (9.5m\ \ 22m\ \ 3.0rad)^T$ to a goal located at $(21.5m, 37.5m)$. As shown in Figure 7, the robot successfully navigates in close proximity to the goal without prior map-based information. The fuzzy-behavior based motion control relied primarily on sonar and dead-reckoning information, both of which are known to be sources of uncertainty in mobile robot navigation. Throughout the navigation task, the fuzzy-behavior hierarchy continuously modulates the weights of low-level behaviors in order to appropriately coordinate their concurrent activity.

The hierarchy of fuzzy-behaviors provides an efficient approach to synthesis of behavioral capabilities neces-

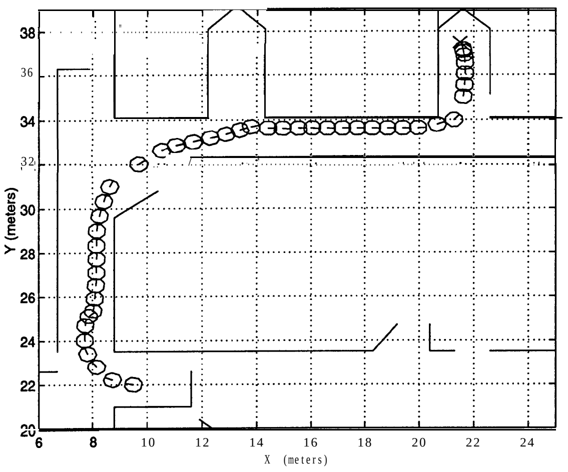

Figure 7: Goal-seeking navigation using fuzzy-behaviors.

sary for autonomous navigation by mobile robots. This hybrid control scheme incorporates fuzzy logic into the a behavior-based control framework for which coordination rules can be discovered using GP. For the purpose of evolving fuzzy rule-bases, GP has certain advantages. Namely, it facilitates manipulation of the linguistic variables directly associated with the problem, and it allows for populations of rule-bases of various sizes. The navigation result demonstrates robustness of the fuzzy-behavior hierarchy to uncertainty in real world sensor-based control of mobile robots. In addition, the result shows that the approach is useful in situations where maps are not available, or are perhaps unreliable.

5 Conclusion

In this paper, three experiments illustrate the utility of soft-computing approaches in handling complex models and unstructured environments. Neuro-fuzzy, GA-fuzzy, and GP-fuzzy hybrid paradigms are successfully implemented to solve three prominent robot control issues, namely: control of direct drive robot motors, control of flexible links, and intelligent navigation of mobile robots. In the future, as these paradigms mature, we will gain more knowledge of their exact nature and advantages. This will allow us to combine soft computing paradigms for more intelligent and robust control. Not long ago, a hybrid combination of these paradigms could not be applied to a real-time system. However, as shown in this paper, with the current advances in increase of speed of processing and DSP parallel processors, various combination of hybrid soft computing paradigms are now realizable.

References

[1] K. K. Kumbla, "Adaptive Neuro-Fuzzy Controller for Passive Nonlinear Systems," *Ph.D. Dissertation*, University of New Mexico, April, 1997.

[2] D. E. Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning," *Addison-Wesley*, MA, NY, 1989.

[3] A. Homaifar & E. McCormick, "Simultaneous Design of Membership Functions and Rule Sets for Fuzzy Controllers Using Genetic Algorithms," *IEEE Transactions on Fuzzy Systems*, Vol. 3, No. 2, pp.129, 1995.

[4] M. A. Lee and H. Takagi, "Integrating Design Stages of Fuzzy Systems Using Genetic Algorithms," *Proceedings of the 1993 IEEE International Conference on Fuzzy Systems*, San Francisco, CA, pp. 612-617.

[5] M.-R. Akbarzadeh-T. "A Fuzzy Hierarchical Controller For A Single Flexible Arm," *Proceedings of the 1994 International Symposium on Robotics and Manufacturing, ISRAM'94*, Maui, Hawaii. 1994.

[6] M.-R. Akbarzadeh-T. and M. Jamshidi, "Incorporating A Priori Expert Knowledge in Genetic Algorithms," *Proceedings of the 1997 IEEE Conference on Computational Intelligence in Robotics and Automation*, pp.300-305, Monterey, California, 1997.

[7] E. H. Mamdani, "Twenty Years of Fuzzy Control: Experiences Gained and Lessons Learnt", *IEEE Intl. Conf. on Fuzzy Systems*, pp. 339-344, 1993.

[8] E. Tunstel, "Mobile Robot Autonomy via Hierarchical Fuzzy Behavior Control", *6th Intl. Symp. on Robotics and Manufacturing, 2nd World Automation Congress*, pp. 837-842, May, 1996.

[9] E. Tunstel, T. Lippincott and M. Jamshidi, "Behavior Hierarchy for Autonomous Mobile Robots: Fuzzy-behavior modulation and evolution", *Intl. Journal of Intelligent Automation and Soft Computing*, Vol. 3, No. 1. pp. 37-49, 1997.

[10] J. R. Koza, *Genetic Programming: On the programming of Computers by means of natural selection*, MIT Press, Cambridge, MA, 1992.