A CONTROL SYSTEM USING BEHAVIOUR HIERARCHIES AND NEURO-FUZZY APPROACH

Dilek Arslan, Ferda N Alpaslan
Middle East Technical University, Computer Engineering Department, 06531 Ankara, Turkey

Keywords: Behaviour hierarchy, behaviour-based robotics, neuro-fuzzy systems, autonomous navigation.

Abstract: In agent-based systems, especially in autonomous mobile robots, modelling the environment and its changes is a source of problems. It is not always possible to effectively model the uncertainty and the dynamic changes in complex, real-world domains. Control systems must be robust to changes and must be able to handle the uncertainties to overcome this problem. In this study, a reactive behaviour based agent control system is modelled and implemented. The control system is tested in a navigation task using an environment, which has randomly placed obstacles and a goal position to simulate an environment similar to an autonomous robot's indoor environment. Then the control system was extended to control an agent in a multi-agent environment. The main motivation of this study is to design a control system, which is robust to errors and is easy to modify. Behaviour based approach with the advantages of fuzzy reasoning systems is used in the system

1 INTRODUCTION

Since the growing interest in agent based systems, many methods were developed for controlling autonomous intelligent agents, which are widely used for problem solving in Artificial Intelligence (AI). These methods can be categorized as deliberative and reactive approaches.

Deliberative approach, which is the classical way of controlling autonomous agents, relies on global planning method. A deliberative agent decides on which actions to take, by considering information about previous experiences and an overall goal as well as information from its current perception/situation.

However, deliberative approach has some drawbacks. For example, in a dynamic environment, some of the information, that agent remembers from a previous experience may become invalid as the environment changes. If a task is highly structured and predictable it makes sense to use a deliberative approach. But in complex, real-world domains where uncertainty cannot be effectively modelled, agents must have a means of reacting to an infinite number of possibilities.

In reactive approach, actions of the agent are based completely on the changes of its environment. Reactive agents don't use planning or internal models of the environment. Instead, they respond to apperception of the real world around them by using stimuli-response mechanism. Thus, in reactive approach, there is a direct connection between agent's inputs and actions. This causes the main drawback of reactive approach; uncertain inputs lead reactive agents into wrong actions.

After subsumption architecture was proposed by Brooks (Brooks, 1986), behaviour based approaches became very popular in solving perception errors and uncertainties of the environment for autonomous mobile robots.

Use of fuzzy logic (Zadeh, 1965) for dealing with uncertainties is also proved to be useful in recent years (Hagras, 2001), (Tunstel, 1996), (Tunstel, 1997), (Saffiotti, 1997), (Tunstel, 2002). Fuzzy inference systems, unlike classical inference systems, can express human expert knowledge

Arslan D. and N Alpaslan F. (2005).
A CONTROL SYSTEM USING BEHAVIOUR HIERARCHIES AND NEURO-FUZZY APPROACH.
In *Proceedings of the Second International Conference on Informatics in Control, Automation and Robotics*, pages 222-229
DOI: 10.5220/0001158802220229
Copyright © SciTePress

naturally without a need for an analytical model of the system. Since they don't need exact mathematical models, fuzzy inference systems are powerful tools to be used in uncertain and not completely known environments.

In fuzzy inference systems, there is not always expert knowledge available to define the proper rules and membership functions. To solve this problem, hybrid methods like neuro-fuzzy systems and genetic-fuzzy systems were proposed (Jang, 1993), (Lin, 1995), (Ahrns, 1998), Bonarini, 1996), Godjavec, 2000), (Hagras, 2000). These systems combine the advantages of fuzzy logic and neural networks.

In this study, a reactive behaviour based agent control system is modelled and implemented. The control system is tested for a navigation task in an environment, similar to an autonomous robot's indoor environment. As a second phase, the control system is extended to a multi-agent domain were the agents' tasks are to search a goal as well as avoide obstacles and other agent(s). The system uses a neuro-fuzzy system called Adaptive Network Fuzzy Inference System (ANFIS) to hold the rule bases of the behaviours (Jang, 1993). Behaviour hierarchies proposed by Tunstel (Tunstel, 1997) was used for the behaviour coordination.

The article is organized as follows. Chapters 2 and 3 give the background about behaviour-based robotics, and neuro-fuzzy systems. Chapter 4 gives details of single-agent control architecture and its experiment results. Chapter 5 gives details of multi-agent control architecture and its experiment results. Chapter 6 concludes the study and gives future work.

2 HIERARCHICAL FUZZY BEHAVIOUR CONTROL

Controlling agents by using behaviour hierarchies by Tunstel (Tunstel, 1997) like many other works, is basically inspired by Brooks' subsumption architecture (Brooks, 1986). In this reactive approach, main idea is to divide a robot's task into a finite number of task-achieving behaviours and arrange these behaviours as a hierarchical network of distributed rule bases each responsible from a different part of the overall task.
There are two types of behaviours in the hierarchy: primitive and composite. Primitive behaviours are, at the bottom of the hierarchy and they are simple and self-contained behaviours, which serve a single purpose. Primitive behaviours are independent from other behaviours and they focus on a part of the complex task.

Only primitive behaviours themselves are not sufficient to perform a complex task. Coordination among them is needed. Composite behaviours are used for behaviour modulation. A composite behaviour controls two or more primitive behaviours and decides how true it is to let them affect the overall result of the agent. For example, in a navigation task, goal seeking can be considered as a composite behaviour and it may control primitive behaviours such as “go to a given coordinate” and “avoid obstacles”.

For behaviour modulation, composite behaviours use a concept called degree of applicability (DOA), which is a weighted control decision-making concept (Tunstel 1997), (Tunstel, 2002). Composite behaviours produce degree of applicability values for each primitive behaviour they control. These DOA values are a measure of instantaneous level of activation of primitive behaviours. Outputs of each primitive behaviour are multiplied with its degree of applicability value before adding this output into the overall result. Since degree of applicability values are used as percentages for desirability of the corresponding primitive behaviours, their values are between 0 and 1.

DOA values are determined dynamically for each step of the given complex task. This feature allows primitive behaviours to influence the overall behaviour to a greater or lesser degree as required by the current situation and goal. It serves a form of adaptation since it causes the control policy to dynamically change in response to goal information and inputs taken from the agent's environment (Tunstel, 1997).
Behaviour hierarchies can easily be extended to work in a multi-agent domain by adding some behaviour to the hierarchy for coordination and communication with the other agents.

3 ANFIS

ANFIS (Adaptive Network Based Fuzzy Inference Sytem) is a fuzzy inference system implemented in the framework of adaptive networks by using a hybrid learning procedure.

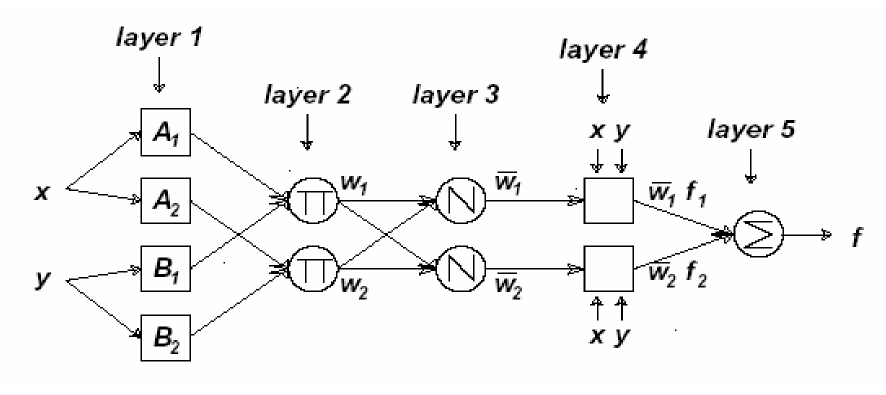

Figure 1: An example ANFIS architecture

ANFIS was proposed by Jang (Jang, 1993) in 1993. By using the hybrid learning method of neural networks and fuzzy inference systems, ANFIS constructs an input-output mapping based on human knowledge (by using fuzzy if-then rules which captures human knowledge easily) and stipulated input-output data pairs.
ANFIS is a feed-forward network whose nodes are connected through weightless links. Some of the nodes in an ANFIS network are adaptable which has adaptable parameters. The other type of nodes in ANFIS architecture is fixed nodes, which have no adaptable parameters. An example ANFIS architecture is shown in Figure 1. Adaptive nodes are shown as square nodes and fixed nodes are shown as circular nodes in the figure.

4 SINGLE AGENT CONTROL

In this study, behaviour hierarchies and a hybrid learning method of neural networks and fuzzy inference systems are combined to implement an autonomous agent control method. This method obtains the advantages of fuzzy systems, numerical systems, and provides flexible control architecture.
Task of the agent is to reach a given goal position while avoiding obstacles on its way and following the shortest path to the goal as close as possible. Behaviour hierarchy used to achieve this task is given in Figure 2.
For learning all the primitive and composite behaviours in the hierarchy, except *Move Randomly* behaviour, ANFIS learning architecture is used in off-line learning mode.
A hybrid of gradient method and the least squares estimate is applied in each epoch. This procedure is composed of a forward pass and a backward pass. In the forward pass, input data goes forward to calculate each node's output and the overall error measure is calculated. Parameter set of the ANFIS network, S, is calculated by using the equation below:

$$S_{i+1} = S_i - \frac{S_i a_{i+1} a_{i+1}^T S_i}{1 + a_{i+1}^T S_i a_{i+1}}, i = 0,1,\ldots.,P-1$$

where S_i is called covariance matrix.
In the backward pass, the error rates propagate from the output towards the input layer, and the parameters in S are updated by the gradient method.
Assuming the given training data set has P entries, error measure for the *p*th entry of the training data is the sum of squared errors;

$$E_P = \sum_{m=1}^{\#(L)} \left(T_{m,p} - O_{m,p}^L \right)^2$$

where #(L) represents number of layers in the network, $T_{m,p}$ is the *m*th component of *p*th target output vector, and $O_{m,p}^L$ is the *m*th component of actual output vector produced by the ANFIS network. Hence the overall error measure is;

$$E = \sum_{p=1}^{P} E_P$$

In the training phases of all behaviours, the agent is placed in random coordinates on a board with obstacles placed randomly on to get the training data set. Behaviours in this hierarchy are explained below;
Avoid Obstacle: This behaviour has three inputs: distance from the closest obstacle on the left, distance from the closest obstacle on the right, and distance from the closest obstacle in front. Obstacle Avoidance behaviour tends to go to the direction where obstacle distance is the farthest.

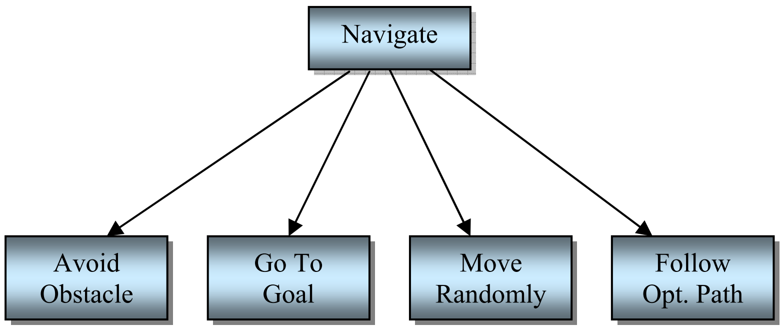

Figure 2: Behaviour hierarchy for controlling single agent

Go To Goal: Inputs to this behaviour are: goal distance on the left, goal distance on the right, and goal distance in front. This behaviour tries to go to the direction where goal distance is the smallest.

Follow Optimum Path: Inputs of this behaviour are: distance from the optimum path on the right, on the left, and in front. *Follow Optimum Path* behaviour, as its name implies, tries to follow the shortest path to the goal as close as possible.

Move Randomly: This behaviour does not use any learning technique. It simply produces a random speed and direction for the next movement. It is used when the agent gets stuck somewhere and cannot move.

All four behaviours explained above are simple primitive behaviours, which deal with only a single goal. For example, *Go To Goal* does not care if there are obstacles in the direction it chooses to go or *Avoid Obstacle* does not know if it gets closer to the goal or not while trying to escape an obstacle.

Since these behaviours only consider their own simple goals, another more complex behaviour is needed to coordinate them. In the hierarchy given above, the composite behaviour, which coordinates and controls them, is *Navigate* behaviour explained below.

Navigate: Composite behaviour *Navigate* controls four primitive behaviours by finding their appropriate DOAs in each step of the execution such that the agent moves towards the goal without hitting obstacles and follows the optimum path towards the goal. As the overall task, *Navigate* behaviour uses all the information about obstacle distances, goal distances, and distances from the optimum path.

This behaviour can be thought as the combination of four parts, each controlling a primitive behaviour. For controlling *Avoid Obstacle* behaviour, inputs used are: the result produced by behaviour *Avoid Obstacle*, obstacle distance in the direction where *Avoid Obstacle* intends to go, and the distance from the goal position in the current position. This part of the behaviour tries to produce a Degree of Applicability (DOA) value for the *Avoid Obstacle* primitive behaviour such that DOA increases as the agent approaches to an obstacle and decrease as the agent approaches to the goal.

Second part of the *Navigate* behaviour controls DOA value of *Go To Goal*. Inputs of this part are: the result produced by the behaviour *Go To Goal*, obstacle distance in the direction where *Go To Goal* intends to go, and the distance from the goal location in the current position. DOA value for the *Go To Goal* behaviour tends to increase as the agent gets closer to the goal and decreases as the agent gets closer to an obstacle.

Third part of the behaviour controls *Move Randomly* behaviour and produce its DOA value. Inputs of this part are: distances between the current position and the position two steps ago, four steps ago, and six steps ago. DOA of *Move Randomly* behaviour tends to increase as these distances get smaller.

Fourth and the last part of the *Navigate* behaviour controls, DOA of *Follow Optimum Path*. Inputs of it are: the result produced by the *Follow Optimum Path* behaviour, obstacle distance in the direction where *Follow Optimum Path* wants to go, and distance from the goal location in the current position. DOA gets bigger if optimum path is far. If the agent is already on the optimum path, then DOA is negative.

Outputs produced by all behaviours are multiplied by their DOAs and vector summation is used to combine the results of all behaviours.

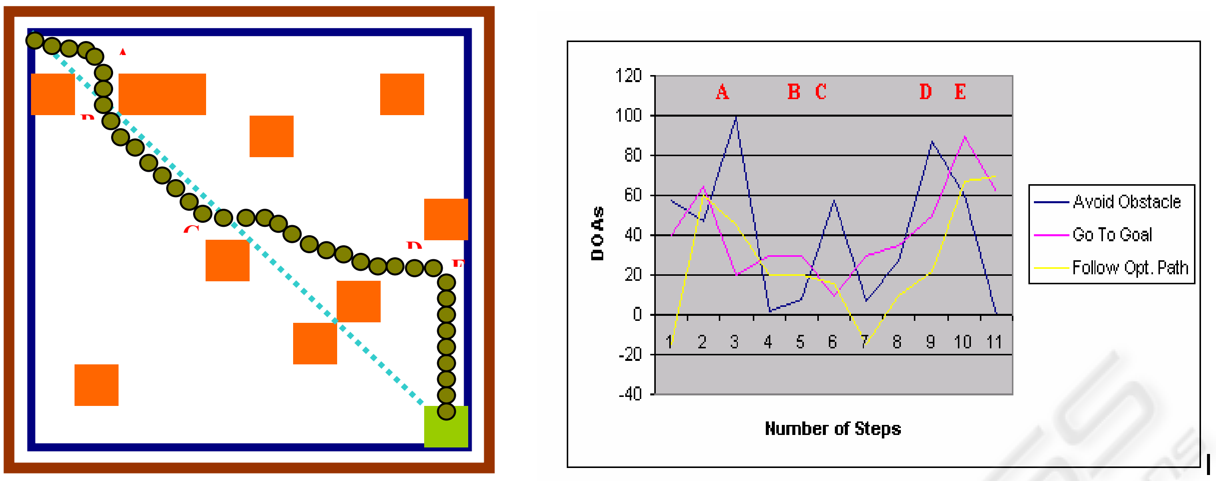

Figure 3: Experiment results for single agent control

Experiments were carried out in an environment of size 100x100. The agent's task is to reach a goal position without hitting any obsatcles and by following an optimum path to the goal as close as possible. Obstacles used in the environment are static obstacles and are represented by orangerectangles in Figure 3. The green rectangle on the figure is the goal. The dashed line on the figure is the optimum path to the goal when the agent starts its task from the upper left corner of the board. To test robustness of the control architecture 10% error was added to inputs of the behaviours (10% of the input values were added or subtracted randomly and these values were used as inputs).

Initial direction of the agent is South. The agent knows only the distances from the goal, optimum path and closest obstacles to simulate perception of a mobile robot. At the starting position, because of the *Avoid Obstacle* behaviour, the agent chooses to turn east (that is agent's left hand-side). At this point, since the agent is already on the Optimum Path, composite behaviour *Navigate* chooses to produce a negative DOA for *Follow Optimum Path* behaviour as shown in Figure 3 and stay in the current position. When applying *Go To Goal* behaviour, both going forward and left are equally active. However since it is trained to favour going forward in this case, it chooses to go forward. Because the obstacle is close, *Avoid Obstacle* has the highest DOA and the agent turns left. It still goes forward slightly because of the *Go To Goal* behaviour. The points where the agent changes direction are marked on both the agent's path and the graph, which shows DOAs of the behaviours. Agent's behaviour at these points is explained below.

At point A, direction is East. *Avoid Obstacle* and *Follow Optimum Path* choose to turn right, but *Go To Goal* still chooses going forward. At this point, the distance from the obstacle in the direction where the agent is to go determines the DOA because output of the *Follow Optimum Path* is greater than the obstacle distance in that direction. DOA is chosen as the highest value possible to prevent the agent hitting the obstacle. DOA of *Go To Goal* also is determined according to distance from the closest obstacle in that direction. DOA of *Avoid Obstacle* is very high because of close obstacles.

At point B, the direction is South. Since there are no obstacles close to the agent on the left, right, and in front, DOA of *Avoid Obstacle* is relatively small. Again because there are no close obstacles, DOA of *Follow Optimum Path* is determined by the distance from the optimum path. Since the agent is not very far from the optimum path, DOA at this point is not high but it still affects the overall behaviour and causes the agent to go left. The distance from the goal also determines DOA of *Go To Goal* and it is higher than the previous step's DOA since the agent is getting closer to the goal. *Go To Goal* behaviour causes the agent to go forward at this point.

At point C *Avoid Obstacle* becomes dominant again as the agent approaches a new obstacle. Since distance from the optimum path remains the same, DOA of *Follow Optimum Path* does not change much. DOA of *Go To Goal* is small in this step because this behaviour chooses the agent to go towards the obstacle and its DOA is determined by the distance from the obstacle.

At point D again DOA of *Avoid Obstacle* increases because of decreasing distance from the obstacle. Since the goal is getting quite close, DOA of *Go To Goal* begins to increase.

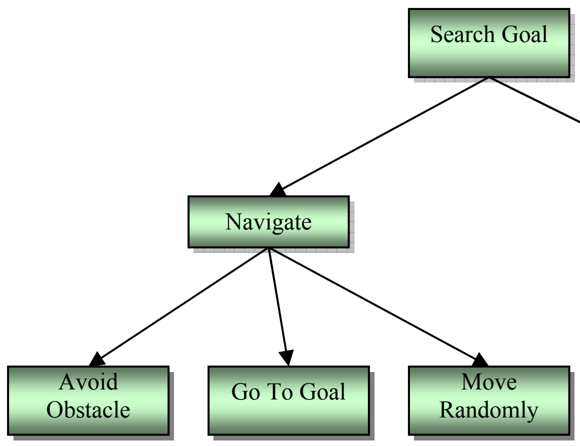

Figure 4: Behaviour hierarchy for multi-agent control

t point E the most dominate behaviour is *Go To Goal* since the goal is closer now. In spite of this fact, DOA is not increased fast in order not to hit the wall. Because the agent is getting close to the goal, DOA of *Avoid Obstacle* gets smaller. This is needed for the *Obstacle Avoidance* behaviour to not to prevent the agent from reaching the goal by moving it away from the walls of the board.

5 MULTI-AGENT CONTROL

As the second phase of the study, the control method for a single agent explained in the previous sections was extended to control the agents in a multi-agent architecture. Task of the agents is to search the goal while avoiding obstacles on their way. This time the agents must learn to avoid the other agents too to prevent collisions and keep the agents apart so that they can search different parts of the board to find the goal. Behaviour hierarchy used is given in Figure 4.

The newly added behaviours are as follows:

Avoid Agent: This behaviour prevents the agents to collide and get close to each other so that they can search different parts of the board. This primitive behaviour allows agents to share the search space somehow.

Search Goal: This composite behaviour has two parts to control behaviours *Navigate* and *Avoid Agent*. The first part controls *Avoid Agent* behaviour and has four inputs; output produced by the primitive behaviour *Avoid Agent*, obstacle distance in the direction *Avoid Agent* wants to go, goal distance and the distance from the closest agent. Second part of the *Search Goal* produces a DOA value for *Navigate*. DOA it produces is complement of the *Avoid Agent*'s DOA.

Experiments were carried out in the same environment defined in section 4. The obstacles used in the environment are static obstacles but the agents move around the board and they can be considered as dynamic obstacles for the other agent(s). Two agents were used for this experiment. The agents start at locations (40, 0) and (80, 0) of the board. The goal location they are supposed to find is the lower right corner of the board. The results of the experiments are shown in Figure 5.

The path shown in pink is the path Agent-1 follows through the experiment. The path shown in green is the path of Agent-2. The experiment stops when at least one agent reaches the goal position.

Initial directions of both agents are South. Numbers on the figure shows number of steps of the agents. Green numbers belong to Agent-1 and red numbers belong to Agent-2. At the starting position (point-1), because the agents barely see each other, *Avoid Agent* behaviour is not much

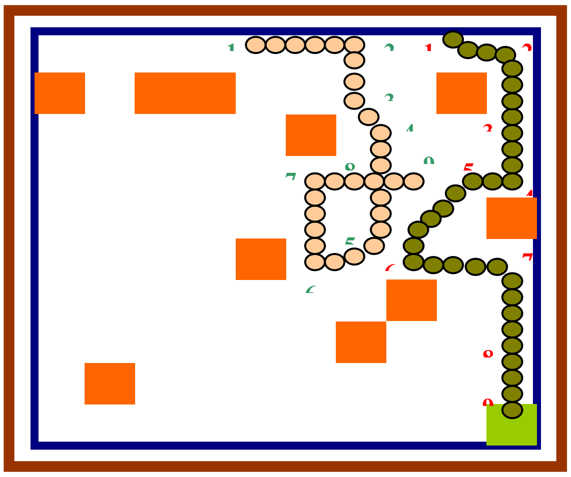
Figure 5: Experiment results for multi-agent control

active. In this position, the most dominant behaviour is *Move Randomly*.

At point-2, since the agents get closer, they begin to see each other and *Avoid Agent* behaviour gets more dominant. Because of this behaviour, both of the agents change their direction to go away from each other.

At point-3, Agent-1 is oriented by the *Avoid Obstacle* behaviour and changes its direction to move away from the obstacle but Agent-2 keeps going towards the only direction it would not approach Agent-1 and hit walls of the board.

At point-4, Agent-2 is guided by the *Avoid Obstacle* behaviour and Agent-1 goes forward to not to approach Agent-2 on the left and the obstacle on the right.

At point-5, both *Avoid Obstacle* and *Avoid Agent* behaviours dominate Agent-2. So the agent goes towards a direction which is a composition of these two behaviours and ends up going South to avoid Agent-1 and going West to avoid the obstacle. For Agent-1, both *Avoid Obstacle* and *Avoid Agent* behaviours choose to go towards West.

At point-6, both agents are controlled mostly by *Avoid Obstacle* behaviour. At point-7 while Agent-1 is still controlled by *Avoid Obstacle* behaviour, Agent-2 begins to see the goal. Because of both *Avoid Obstacle* and *Go To Goal* behaviours, it turns towards south.

At points 8 and 9 Agent-1 is controlled by *Move Randomly* behaviour because there are no close obstacles and agents around. Agent-2 is now very close to the goal and it is controlled by only *Go To Goal* behaviour.

6 CONCLUSION

Fuzzy controllers have been widely used in robotics applications in recent years, because there is usually uncertainty in the inputs and it is not possible to obtain a model of the environment. Another advantage of using fuzzy logic in robot controllers is the convenience it provides to represent human knowledge without a need for analytical model of the system.

In this study, a behaviour-based control strategy using ANFIS neuro-fuzzy learning approach is presented. Fuzzy behaviour hierarchies are used to combine the behaviours in the system. It resulted in a system robust to errors in input data, and easy to modify by adding new behaviours to the hierarchy. The agents using this control architecture successfully navigate in simulated indoor-like environments with both static and dynamic obstacles in it and find and reach goal positions.

This study has an advantage over the previous studies, which apply fuzzy behaviour hierarchies [4, 6] in finding and tuning the membership functions, which is usually done this by trial. In this study, the membership functions are found and tuned by ANFIS automatically.

As a future work, multi-agent control architecture in the second phase of the study can be improved by adding new behaviours to the system. For example, a new behaviour can be added for the agents to share the information they have with the other agent or share their tasks. As another improvement ANFIS system can be used in on-line learning mode to adapt the agent to the changes in the environment.

The control architecture presented in this study is tested in a simulated environment. As another future work, the study can be tried on a real mobile robot and in real world problems like tasks of finding a target location in an unknown environment.

REFERENCES

Ahrns, I., J. Bruske, G. Hailu, and G. Sommer, 1998. *Neural fuzzy techniques in sonarbased collision avoidance*. Soft Computing for Intelligent Robotic Systems, pages 185–214. Physica.

Bonarini, A., 1996. *Evolutionary learning of fuzzy rules: competition and cooperation*. Fuzzy Modeling: Paradigms and Practice, pages 265–284. Kluwer Academic Press, Norwell, MA.

Brooks, R. A., 1986. A Robust Layered Control System for a Mobile Robot. *IEE Journal of Robotics and Automation*, Vol. RA-2, No.1, pp 14-23.

Godjavec, J., N. Steele, 2000. *Neuro-fuzzy control for basic mobile robot behaviors.* In Fuzzy Logic Techniques for Autonomous Vehicle Navigation, pages 97–117.

Hagras, H.,V. Callaghan, and M.Colley, 2000. *Learning fuzzy behavior co-ordination for autonomous multi-agents online using genetic algorithms and real-time interaction with the environment.* Fuzzy IEEE.

Hagras, H., V. Callaghan, 2001. A Hierarchical Fuzzy-Genetic Multi-Agent Architecture for Intelligent Buildings Online Learning, Adaptation and Control. *International Journal of Information Sciences*.

Jang, Jyh-Shing R., 1993. ANFIS: Adaptive-Network-Based Fuzzy Inference System. *IEEE Trans. Systems, Man & Cybernetics*, Vol. 23, pp 665-685.

Lin, Y., G. Cunningham, 1995. A New Approach to Fuzzy-Neural System Modeling. *IEEE Transactions On Fuzzy Systems*. Vol.3, No.2.

Saffiotti, A., 1997. The Use of Fuzzy Logic for Autonomous Robot Navigation. Soft Computing, Vol. 1(4), pp 180-197.

Tunstel, E., M. Jamshidi, 1996. On Genetic Programming of Fuzzy Rule-Based Systems for Intelligent Control. *International Journal of Intelligent Automation & Soft Computing*, Vol.2 No.3, pp. 273-284.

Tunstel, E., T. Lippincott, M. Jamshidi, 1997. Behaviour Hierarchy for Autonomous Mobile Robots: Fuzzy-Behaviour Modulation and Evolution. *International Journal of Intelligent Automation & Soft Computing*, Vol.3, No.1, Special Issue on Autonomous Control Engineering, pp. 37-50.

Tunstel, E., M. Oliveira, S. Berman, 2002. Fuzzy Behaviour Hierarchies for Multi-Robot Control. *International Journal of Intelligent Systems*, Vol.17 449-470..

Zadeh, L. A., 1965. Fuzzy Sets. *Information and Control*, No. 8, pp. 338-353.