## 16.5 Soft Computing Based Hybrid Fuzzy Controllers

Traditional methods, which address robotics control issues, rely upon strong mathematical modeling and analysis. The various approaches proposed till date are suitable for control of industrial robots and automatic guided vehicles, which operate in various environments and perform simple repetitive tasks that require end effectors positioning or motion along fixed paths. However, operations in unstructured environments require robots to perform more complex tasks for which analytical models for control is very difficult to determine.

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 section 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. The first scheme concentrates on a methodology that uses neural networks (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.

### 16.5.1 *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 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 multilayer perceptron NN is used to classify the temporal 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 is 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 nonredundant 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.

### 16.5.2 Real-Time Adaptive Control of a Direct Drive Motor

In order to perform real-time control, it is necessary for 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 multiprocessing 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).

### 16.5.3 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. GAs have demonstrated the coding ability to represent parameters of fuzzy knowledge domains such as fuzzy rule sets and membership functions 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.

#### 16.5.3.1 Application to Flexible Robot Control

The application of GA-fuzzy systems applied to flexible robot is discussed here.

The GA-learning hierarchical fuzzy control architecture is shown in Figure 16-18. 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 this, control system is simulated using only a priori expert knowledge. In the given structure, a GA fine-tunes parameters of membership functions.

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

$$\text{fitness} = \int_{t_s}^{t_f} \frac{1}{e^2 + \gamma^2 + 1} dt$$

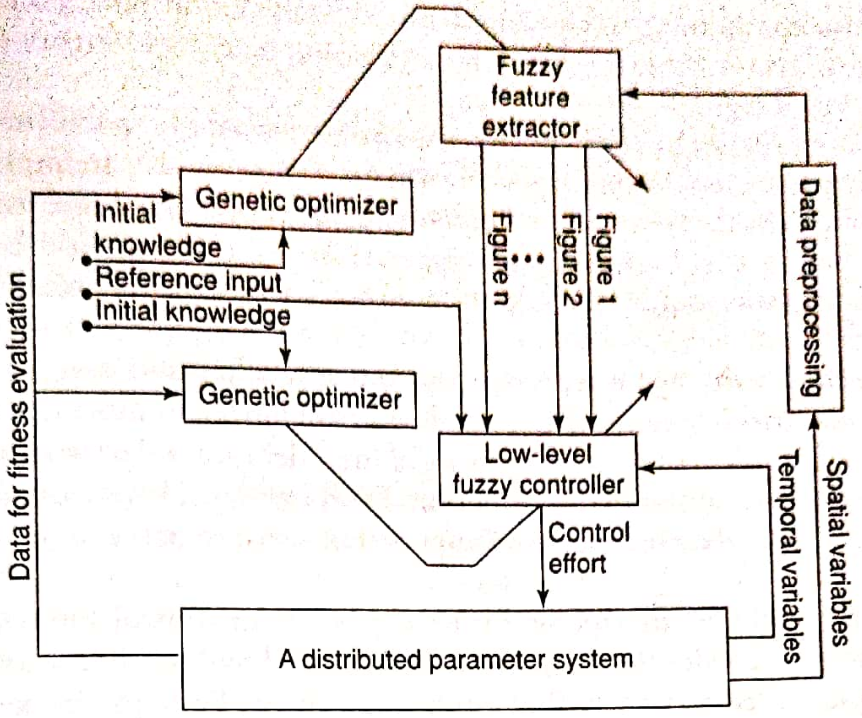

**Figure 16-18** GA-based learning hierarchical control architecture.

where $e$ represents the error in angular position and $\gamma$ 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 *grtd-parenting* was used to create the initial population.

Members of the initial population are made up of mutation of the knowledgeable grand-parent (sb). As a result, a higher fit initial population results in a faster rate of convergence as is exhibited in Figure 16-19(A). Figure 16-19(A) shows the time response of the GA-optimized controller when compared to previously obtained results through the non-GA fuzzy controller.

### 16.5.4 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

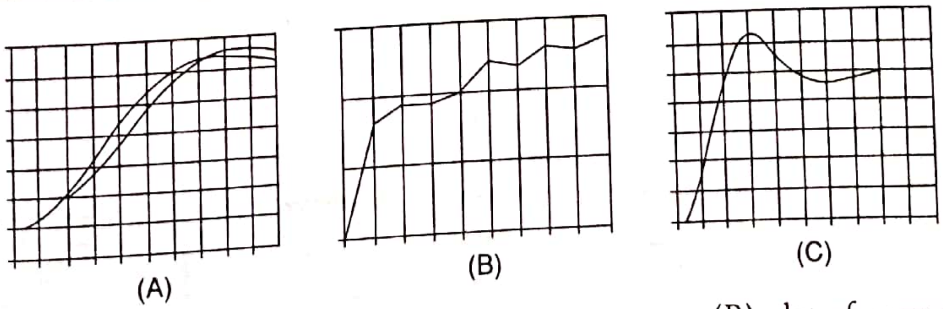

Figure 16-19 GA simulation: (A) Comparison of simulation responses; (B) plot of average fitness; (C) initial experimental results.

proposed for autonomous navigation in unstructured environments. 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 16-20. 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 16-20 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 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 that 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. This issue is addressed using GP to computationally evolve rules for composite behaviors. The forthcoming section describes the genetic programming approach to fuzzy rule-base learning.

### 16.5.5 GP-Fuzzy Approach

The GP paradigm 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

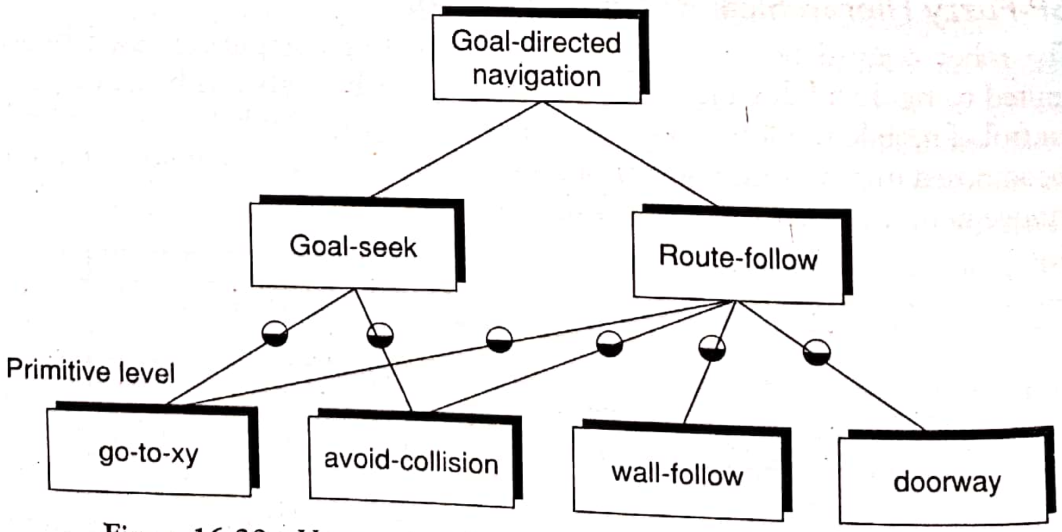

**Figure 16-20** Hierarchical decomposition of mobile robot behavior.

over time in response to selective pressure induced by the relative fitness's 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. 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 on the basis of 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 offsprings 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 subtending from crossover nodes are then swapped between the parents to produce the two offsprings. GP cycles through the current population perform fitness evaluation and apply 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.

In this section, the soft computing approaches in handling complex models and unstructured environments are studied. Neuro-fuzzy, GA-fuzzy and GP-fuzzy hybrid paradigms can be successfully implemented to solve the prominent robot control issues, namely, control of direct drive robot motors, control of flexible links and intelligent navigation of mobile robots. This in near future allows us to combine soft computing paradigms for more intelligent and robust control.

## 16.6 Soft Computing Based Rocket Engine Control

Many of the rocket engine programs initiated by NASA's Marshall Space Flight Center (MSFC) in Huntsville, Alabama, have been successful as evident by success of the Space Shuttle Main Engine, ground testing of the former X-33 engine and Fastrac X-34 engine for