Chapter 4

Evolutionary Computational Techniques

4. Evolutionary Computational Techniques

In previous chapters, we have discussed various methods of designing intelligent control and then briefly discussed the fundamentals of Evolutionary Computations or Evolutionary Algorithms. In this chapter I am to discuss various evolutionary techniques used by researchers for design of real time control systems.

Before we get into the discussion of various methods, let me first discuss, the actual real time constraints with reference to intelligent control.

4.1 Real Time constraints in Intelligent Control

During the recent past field of integrated control design and real time scheduling has been object of several works. In this field a closer interaction between control design phase and control implementation (scheduling) is used to improve the control performance. The development of scheduling techniques and control theory considering both aspects permits the definition of new flexible scheduling schemes, where control design methodology takes the availability of computing resources into account at the design phase and allows optimization of control performance and computing resource utilization. This integration is usually considered at the design stage128.

In digital control, it is well known that the system is behaving in open loop between two sampling periods. Thus, the control performance degrades as far as the sampling period increases and the degrading depends on the control efforts applied to the plant. It is also generally the case that delays between the measurement sampling and the control signal updating deteriorates the control performances. A classical limitation in the reduction of the sampling period is determined by the complexity of the control algorithm and the time needed by the CPU to compute the result. But in the case of complex systems, there are many limitations:

1. The same computer is controlling many loops. Thus, the multitasking environment forces to enlarge the theoretical sampling period to allocate time for all the tasks.
2. The computation time is not fixed, usually depending on the data. The worst case execution time should be considered.

3. The operation under different modes like normal, abnormal and emergency, changes the overall computing load.
4. Different control algorithms, implying different computing loads and leading different control performances, could be foreseen. In particular, better performances can be achieved by performing additional pieces of code, denoted by optional tasks.

Above are applicable to intelligent control systems where an integrated design of control and scheduling, will improve the performance of the system.

Periodic talks are clock synchronized to fulfill the period timing requirement. However other synchronizing signals can be used to establish periodic activities. For instance, speed of a guiding vehicle can be used as a criterion to establish the period of the set of tasks. So it the speed of the vehicle is constant the period is constant, but if the speed decreases the period increases in time but is constant with respect to the vehicle advance. This period variation demands use of more or less CPU time and requires meeting some criteria to maintain the system schedulability.

4.1.1 Intelligent control

By intelligent control, it is usually understood those control systems designed using AI techniques such as Neural Networks (NN), Fuzzy Logic (FL), Expert Systems or Genetic Algorithms (GA).

In NN, each neuron can be conceptually considered as a parallel computing unit. Thus, from the point of view of timing requirements, the number of nodes does not matter. The processing time depends on the number of layers, as well as on the explicit delays involved in Recurrent Neural Network (RNN). In general, time required in any case is known in advance and fixed. Another source of trouble is the NN tuning. The learning of the correct weights requires back propagating the information through the network, in order to obtain the required derivatives of the network error with respect to network weights. Afterwards the weights are modified following some adaptation rule. The online learning can be carried out following several time schedules.

1. Learning at each sampling period. This is the most time demanding case.
2. Periodic Learning, if there is an accumulated learning over a fixed number of periods. The period can be changed according to the operating conditions.

3. Sporadic Learning: Whenever a degraded performance of the control system requires retuning of the network weights, the backward propagation and adaptation rules are executed. Sporadic learning can also be carried out, whenever time resources are available.

But most of the times NNs are implemented as a piece of software code to be run on a sequential processor. In this case the processing time also depends on NN size.

Fuzzy controllers have become the one of the most successful applications of Fuzzy logic. They can be easily applied to control processes of which we lack a useful mathematical model or in which vagueness plays a fundamental role and we have qualitative models. The knowledge supplied by an expert human operator is coded in the form of IF-THEN production rules, where the premise and consequent variables are categorized in terms of fuzzy sets. At the controller output, the control action inferred by a set of fired rules is obtained by means of interpolative reasoning methods.

Fuzzy controllers with proportional rules i.e. those in which consequent of a rule are never the antecedent of another rule may be treated as NNs with one layer129. Therefore, if the possibility of adapting the membership function, like weight, is assumed, all the consideration discussed above for NNs are valid. Therefore, their time requirements are equivalent to those described for NNs.

But if reasoning is expressed by means of a tree of rules, the time required to get the result will depend on the level of rules firing. Again, the quality of the result will be related to the time used to get it. In this case, the fuzzy controller may forward the last available result, if the reasoning is interrupted.

Other AI based controllers, such as expert systems or those using evolutionary computation methods like GA, having uncontrolled recurrence computations are generally not suitable for Real time applications. But if hybrid methods involving Fuzzy, Neural Network, Neuro-Fuzzy or GA, are implemented then their Real time performance can be comparable. Again, their applications to real time problem are required to be evaluated on case to case basis.

4.1.2 Real Time Control

In control systems the activities to be done at each sampling period may include: data acquisition, data analysis, pre-processing, control action computation and actuation delivering. It depends on each applications the relevance and complexity of each part. There are some other higher level control activities also, related to supervision, adaptation, and coordination without hard real time constraints under normal operating conditions. Also, some alarm activities may have variable priority depending on the operating conditions.

Now, let us have a brief introduction of different hybrid Evolutionary methods used by various researchers to implement real time control systems. As we discussed earlier and also in the previous chapter Genetic Algorithm based methods will not alone be suitable for real time problems due to their nature of uncontrolled recurrence. Hence, various hybrid evolutionary approaches are only discussed.

4.2 Structure and Parameter Tuning of Neural Network

We know that Neural Network has proved to be as good universal approximators. A three layer feed forward neural network can approximate any nonlinear continuous function to an arbitrary accuracy. Neural networks are widely applied in the areas of system modeling and control [130]. Owing to its particular structure, a neural network is very good in learning [131] using some learning algorithms such as Back Propagation and GA. In general the learning steps of a neural network are as follows. First, a network structure is defined with a fixed number of inputs, hidden nodes and outputs. Second, an algorithm is chosen to realize the learning process. However, a fixed structure may not provide the optimal performance within a given training period. A small network may not provide good performance owing to its limited information processing power. A large network, on the other hand, may have some of its connections redundant [132,133]. Moreover, the implementation cost for large network is high. To obtain the network structure automatically, constructive and destructive algorithms can be used [132]. The constructive algorithms start with a small network. Hidden layers, nodes, and connections are added to expand the network dynamically. The destructive algorithms start with large network. Hidden layers, nodes, and connections are then deleted to contract the network dynamically. The design of network structure can be formulated into search problem. GAs are employed to obtain the solution [134,135]. Pattern classification approaches are also used to design the network structure. An ANNA ELEONORA algorithm was proposed in [136], where new genetic operator and encoding procedures that allows an opportune length of the coding string were introduced.

Each consists of two parts: the connectivity bits and connection weights. The former indicates the absence or presence of a link, and the later indicates the weight associated with it.

A GNARL algorithm was also proposed in [137]. The number of hidden nodes and connection links for each network is first randomly chosen within some defined ranges. Three steps are then used to generate an offsprings: copying the parents, determining the mutations to be performed, and mutating the copy. The mutation of a copy is separated in two classes: the parametric mutations that alter the connection weights and the structural mutations that alter the numbers of hidden nodes and their presence of network links. An evolutionary system named EPNet can also be found for evolving neural networks [133]. Rank based selection and five mutation were employed to modify the network structure and connection weights. In [138], a three layer neural network with switches introduced in some link is proposed to facilitate the tuning of the network structure in a simple manner. A give fully connected feed forward network may become a partially connected network after learning. This implies that the cost of implementing the referred network, in terms of hardware and processing time can be reduced, which again is important for real time control applications. The network structure and parameters is tuned simultaneously using genetic algorithm. Using genetic algorithm and improved GA, the neural network is able to learn both the input – output relationships of an application and the network structure. The simulation results have an edge over traditional feed forward networks trained by the back propagation with momentum and adaptive learning rate.

4.3 Fuzzy Tuned Neural Network

Fuzzy tuned neural network consists of neural fuzzy network and modified neural network. In the modified neural network, a novel neuron model with two activation functions is employed. The parameters of the proposed networks are tuned with genetic algorithm with arithmetic crossover and uniform mutations.

It is well known that a 3 –layer feed forward neural network can approximate any nonlinear continuous function to an arbitrary accuracy. Neural Fuzzy Networks have been [130] used to deal with variable linguistic information. By processing fuzzy information, reasoning with respect to a linguistic knowledge base can be realized. GA is a directed random search techniques [121] that is widely applied in optimization problems. It can help to find the global optimal solution over a domain. GA has been applied in different areas such as fuzzy control, modeling and classification etc.

Fuzzy tuned neural network, consists of traditional neural fuzzy network and modified Neural network [139]. For the modified neural network, two different activation functions are used in the neurons of the hidden layer. By introducing the referred neuron, the freedom of the searched domain is enhanced. Some parameters of the activation functions in these neurons are tuned by the neural fuzzy network. The values of these parameters are therefore governed by some fuzzy rules and are trained by the Genetic Algorithm with arithmetic crossover and non-uniform mutation. Comparing with the traditional feed forward neural network, the fuzzy tuned neural network gives better performance with a similar number of parameters.

4.4 Design of Fuzzy Controller – A Genetic Algorithm Approach

Fuzzy control has been a hot research topic in recent past, despite the lack of a concrete theoretical basis; many successful applications on fuzzy real time control are reported in different areas. However, without in depth analysis, the design may come with no guarantees of system stability and good system performance. Stability analysis of fuzzy control systems based on the TSK (Takagi-Sugeno-Kang) fuzzy plant model is reported in [85,140]. The advantage of using the fuzzy model is that a nonlinear plant can be represented as weighted sum of linear subsystem, so that some linear or nonlinear control theories can possibly be applied to design the controller. Different stability conditions for this class of fuzzy control systems are derived. In [100, 83,117, 140,141, 142], the Lyapunov stability theory was employed to analyze the system stability. An LMI based design of fuzzy controller is discussed in [84, 102].

Genetic algorithm is powerful random search technique to handle optimization problems [100,83,86,134]. This is especially useful for complex optimization problems with a large number of parameters that make global analytical solutions difficult to obtain. A nonlinear controller is proposed in [143] to control a system represented by a TSK fuzzy plant model [85]. The controller has structure similar to fuzzy controller discussed in [86]. The main difference is that the weights in the nonlinear controller are signed but those in the fuzzy controller defined in [86] must be positive values, as they are considered to be as membership functions. Wang et al. derived a stability condition for a TSK fuzzy model based system using Lyapunov stability theory. A sufficient condition for the system stability is obtained by finding a common Lyapunov function for all the fuzzy sub-control systems. For the TSK fuzzy plant model with *p* rules, a fuzzy controller with *p* rules (*p* sub controllers) is used to close the feedback loop and p(p+1)/2 Lyapunov conditions are required. The number of sub controllers of the nonlinear controller need not be the same as that of the TSK fuzzy plant model. By allowing the positive and negative weighting values in the controller, the number of Lyapunov conditions ca be reduced to *p*. A way of designing the nonlinear gains and the feedback gains of the nonlinear controller is also discussed. The task of finding the common Lyapunov function can readily be formulated into a LMI problem. The GA with arithmetic crossover and non uniform mutation [144] is used to help finding the solution of the derived stability conditions and determine the feedback gains of the sub controllers.

As we know control actions of a fuzzy controller are described by some linguistic rules, making control algorithm easy to understand. To facilitate a systematic tuning procedure, a fuzzy controller implemented by a neural-fuzzy network was proposed in [145]. Through tuning fuzzy rules can be generated [138]. GA is applied to fuzzy systems to help generate the memberships functions and / or the rule sets [146]. These methods make the design simple, but do not guarantee the system stability. In order to investigate stability the TSK fuzzy plant model as discussed [143] is used. However, formulating the stability conditions into an LMI problem as in [143] limits the realm of the stability analysis. In order to have a systematic method to obtain a fuzzy controller derived from GA [134,143] is used. The stability conditions for fuzzy control systems are first derived and then based on these conditions; the parameters of the fuzzy controller are obtained using GA. GA with arithmetic crossover and non-uniform mutation is used to help find the solution to the stability conditions and the feedback gains of the fuzzy controller.

4.5 GA based methods for Constructing TSK Fuzzy Rules

The identification of dynamic nonlinear systems is a difficult task. Many identification techniques based on the ANN and the fuzzy inference systems has been available that can explain the behavior of an unknown system for which only a set of input-output data is available. Fuzzy modeling approach for system identification from the numerical data has a distinguished feature in that it can express complex nonlinear system linguistically using fuzzy inference rules. The premise part of a fuzzy rule defines a local fuzzy region while the consequent can be a fuzzy set, a constant or a linear equation, i.e. different consequents result in different fuzzy inference systems, but their premise parts are always the same. One of the most outstanding fuzzy inference model is the one suggested by TSK [85,147], whose rule base is comprised of rules with non-fuzzy consequents. This model has excellent capability to describe a given unknown system; however, it is too complex to implement in a digital computer and consumes much time due to its complexity, which keeps the model from being used in the practical real time control design applications. The conventional (Mamdani type) models are easier to implement in a digital computer and is intuitively mode persuasive toward human beings than the TSK model because its consequents are expressed by the linguistic variables, not linear equations but this has poorer capability of system description. In [148] a new identification algorithm is suggested by combining the merits and removing the demerits of the two of the most noticeable fuzzy models. This means that it has the same consequent structure as that of TSK model and its identification mimics the simple procedure of Mamdani type model. Each input – output data pairs gives rise to fuzzy rules; the collection of all such rules gives the rule base for the fuzzy model. A large rule base will result in computation complexity and memory overloading in the implementation of the fuzzy system.

To reduce the requirement of large memory and to minimize computation complexity, fuzzy modeling is concerned not only with a fuzzy inference system that can approximate the identified system correctly, but also with favoring the selected fuzzy inference system with fewer fuzzy rules. This is achieved by exploiting the clustering approach. In [149] Kim et al suggested a clustering approach for TSK models which, unfortunately, is quite complex. Also, the algorithm requires black box recursion in the number of clusters and in parameters that define the clusters. It leads to a significant computational burden when the data sets are large. Wang and Mendel [150] has taken an approach of independently clustering in input and output spaces and assigning a weight to each rule to capture the relation in input-output data. Compared to other clustering methods, this approach gives simpler function approximators requiring however larger number of fuzzy rules. The performance of a fuzzy model is much more sensitive to the choice of fuzzy partitions than to the use of complex clustering method for rule base generation. In [148] Ashwinikumar extended the work of Wang and Mendel and developed clustering approach for Mamdani type models, to TSK models, while doing that recursion is considerably reduced leading to a simpler function approximation approach. Direct application of the GA is not possible in the same because all the parameters are not independent; number of fuzzy rules in rule base is a function of fuzzy partitions of input variables. Parameter selection procedure in [148] wraps GA with another GA: in each iteration of GA that selects fuzzy partitions, another GA code is called that give optimum values of certainty factors of the rules. A genetic process that automatically designs the whole knowledge base; the database being evolved by a GA and the rule base being generated by the algorithm, is also developed.

As we discussed earlier the fuzzy system design is divided into two steps: the structure identification and parameter identification. In the structure identification step, the input space can be partitioned to describe the inherent structure of an identified system so that the number of fuzzy rules and the shapes of the fuzzy sets in the premise part are determined. Subsequently, in the parameter identification step, a parameter estimation method can be applied to fine tune the parameters of the obtained fuzzy system, in the structure identification step. In [151], each individual in the population is considered to partition the input space to determine a rough fuzzy system structure and its premise part, then the recursive least square method is applied to determine the consequent part of the constructed fuzzy systems. Wong and Chen developed a system, where we only have input-output data of the system under consideration. A method based on GA and recursive least square, each individual in the population is represented to determine a rough fuzzy system, then the recursive least squares method is applied to determine the consequent parameters of the fuzzy system so that the constructed system has a high performance. A fitness function is also proposed such that it can effectively guide the search procedure to choose an appropriate fuzzy system that satisfied the predetermined acceptable performance and has a low number of fuzzy rules. This has overcome the limitation of conventional fuzzy design system where the number of fuzzy sets of each input variable must be defined in advance.

4.6 Self Learning Fuzzy Logic Controllers using GA with reinforcement

The current research trend is to design a fuzzy logic system that has the capability to learn itself, starting with self organizing control techniques of Mamdani [89,90]. It is expected that controller performs two tasks: 1) it observes the process environment while issuing appropriate control decision and 2) it uses the results of the decision for further improvement. Wong and Chen [151], Lin and Lee [152] proposed a fuzzy logic control / decision network, which is constructed automatically and tuned by learning the training examples. The learning task includes the tuning of the fuzzy membership functions used in the control rules and identification of the main control rules.

Unsupervised learning, as connectionist learning methods, do not rely on the network teacher that guides the leaning process; like the supervised learning class, the teacher associates the learning system with desired outputs for each given input. Leaning involves memorizing the desired outputs by minimizing the discrepancy between the actual outputs of the systems and the desired output. In reinforcement learning class, the teacher provides the learning system with scalar evaluation of the system's performance of the task according to a given index. The objective of learning system is then to improve its performance, as evaluated by critics, by generating appropriate outputs.

If supervised learning is used in the control, when input – output data is available, it is more efficient than the reinforcement learning. However, many control applications are required to select the control actions. This has the consequences emerge over uncertain periods for which input/output training data sets are not readily available. In such cases, the reinforcement learning systems is used to provide the unknown desired outputs through system with suitable evaluation of system performances. In such cases reinforcement learning is more appropriate than the supervised learning. GA is used to design [153] fuzzy logic control rules and seek the optimal linguistic values of the consequences of the fuzzy control rules in reinforcement learning. Chiang et al presented a novel General Reinforcement Fuzzy Logic Controller (GRFLC), which has capability to learn its own control rules without expert knowledge. Because GA can seek the optimal membership functions in the consequence of control rules, GRFLC compensate for inappropriate definitions of membership functions in the antecedent of control rules. It is applicable to control problems for which analytical models of the process are unknown, since GRFLC learns to predict the behavior of a physical system through its action evaluation network.

4.7 Evolutionary design of Fuzzy Rule Base

Recent research and applications employing non-analytical methods of soft computing such as fuzzy logic and evolutionary computation has demonstrated the utility and potential of these paradigms for developing intelligent control system. Fuzzy logic and evolutionary computation have proven to be convenient tools for handling real-world uncertainty and knowledge representation, and the design of intelligent control. Jamshidi et al [154] presented an approach that exploits the combined attributes of these paradigms for the purpose of developing intelligent algorithms for controlling autonomous dynamic systems that interact with the real world. The genetic programming paradigm is applied to the problem of learning rules for use in a fuzzy rule based control system. In Genetic programming, a population is comprised of computer programs that are candidate solutions to a particular problem. The programs are structured as hierarchical compositions of functions and arguments of various sizes and shapes. These individuals participate in the simulated evolutionary process wherein the population evolves over time in response to selective pressure induced by the relative fitnesses of the individuals in a particular problem environment.

Jamshidi et al revealed that [154] genetic programming as a tool for designing rule bases for fuzzy logic controllers. For the purpose of evolving rule bases, the genetic programming implementation has some advantage over the simple GA and neural networks. The advantages are, it facilitates manipulation of linguistic variables directly associated with the problem and it allows for population of rules bases of various sizes. The performance because of the rule bases evolved using genetic programming has been favorable compared to that of complete rule bases. Here, Genetic programming is able to produce fuzzy controllers that required fewer rules than necessary for rule base completeness. We can also tailor out the fitness functions in order to emphasis the desired performance attributes.

Kang et al [155] presented an approach to evolutionary design of an optimized fuzzy rule base for modeling and control. Evolutionary programming is used to evolve the structure and the parameter of fuzzy rule base. Since they are codependent, the simultaneous evolution with no predefined assumption about the rule base structure can result in a more appropriate rule base for a given task. These controllers are better in tracking problems in comparison with the optimized PID controllers in terms of overshoot, decay time, and the steady state error.

4.8 Hierarchical Evolutionary approach to Design Fuzzy Systems

Most of the evolutionary methods presented here as well in literature encode all possible rules into the chromosome. This procedure has some disadvantages: the effective adequacy of a fuzzy system can not be properly measured under the presence of a destructive element, one part of the chromosome that cancels the effect of very promising ones, very common in the early stages of evolution and hard to be identified; the genetic operators provide the same treatment to completely different kinds of information, membership parameters, number of fuzzy rules, and the type of fuzzy rules, encoded into the chromosomes.

To avoid the uniform treatment of distinct entities Delgado et al [156] presented a modular and hierarchical evolutionary design approach of fuzzy systems, where the design parameters of the fuzzy systems are distributed along different modules that evolve in an iterative way. The approach considers simultaneously many important aspects that must be addressed when designing a fuzzy system:

- the system is automatically designed by means of GA techniques, providing a way to decide on memberships function parameters – type, shape and location, overlapping between fuzzy sets, fuzzy relation associated with each individual rules and the number of rules of the resulting fuzzy model,
- the evolutionary process is structured in a hierarchical configuration composed of the following modules: membership functions, population of individual rules and population of set of rules.
- The modules interact during the evolutionary process, the inference mechanism and reasoning operators are arbitrarily defined. In Delgado et al [156], the hierarchical structure is also used, but an additional module added to increase its flexibility.

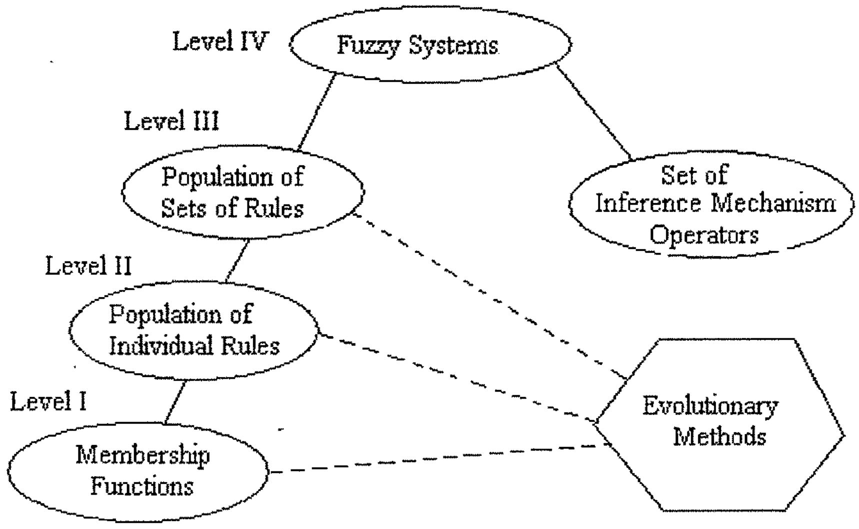
Level IV
Fuzzy Systems
Level III
Population of Sets of Rules
Set of Inference Mechanism Operators
Level II
Population of Individual Rules
Level I
Membership Functions
Evolutionary Methods

Figure 4.1: The hierarchical evolutionary structure

The modular and hierarchical structure is depicted in the figure 4.1. The model has flexibility to control: the parameters of membership functions – type, shape and localization of every term defined on the universe of each variable; the granularity of the universe associated with each variable; the overlapping degree among the fuzzy sets in each universe; which variables are relevant in each fuzzy rules; how many and which rules will take part in the fuzzy systems; the parameters of inference mechanism.

4.9 Evolutionary fuzzy Systems

One of the most important considerations in designing any fuzzy system is the generation of the fuzzy rules as well as the membership functions for each fuzzy set. In most existing applications, the fuzzy rules are generated by experts in the area, especially for control problems with only a few inputs. With an increasing number of variables, the possible number of rules for the system increases exponentially, which makes it difficult for experts to define a complete rule set for good system performance. An automated way to design fuzzy system might be preferable.

There are many ways to attack the referred problem. A straight forward approach is to use clustering algorithm [149] or similar methods to partition the pattern space into many subspaces with or without overlaps among them, then map the center of each cluster into a rule according to the definition of fuzzy variables [150]. One disadvantage of this approach is that the extracted rules are independent of the membership functions so there is no guarantee that the fuzzy systems obtained will have sufficiently good performance, especially for a complex system problem with a large number of input variables. In many cases, however, the system's performance can be improved by further tuning the membership functions and selecting suitable fuzzification and defuzzification methods.

The design of fuzzy system can be formulated as a search problem in high dimensional space, where each point represents a rule set, membership functions and the corresponding system's behavior. The performance of the system forms a hypersurface in the space. Developing optimal fuzzy system design is equivalent to finding the optimal location of this hypersurface. These hypersurface is …

- Infinitely large since the number of possible fuzzy sets for each variable is unbounded
- Non-differentiable since changes in the number of fuzzy sets are discrete and can have a discontinuous effect on the fuzzy system's performance

- Complex and noisy since the mapping from a fuzzy rule set to its performance is indirect and dependent on the evaluation method used.
- Multimodal since different fuzzy rule sets and/ or memberships functions may have similar performance.
- Deceptive since similar fuzzy rule set and membership functions may have quite different performances.

GA's are commonly used evolutionary algorithm that provides a way to search poorly understood irregular spaces. One of the key issues in the evolutionary design of fuzzy systems using GA's is their genotype representation meaning what is encoded into chromosomes.

Thrift [157] and Hwang et al [158] encode all the rules into the chromosome while fixing the membership functions. Using several critical points to represent each membership function, while using all the possible rules, Karr and Gentry [159] use GA to evolve these critical points. Since in a fuzzy system the membership function and rule set are codependent, they should be designed or evolved at the same time. Homaifar and McCormik [160] used GA to tune the membership functions and evolve the rule set at the same time. The base length of each triangular membership function and all possible rules are encoded into the chromosomes. Similar to [160], Lee and Takagi [161] also encode membership function and all the rules into the chromosome, but have a different way to encode the triangular membership functions. They restrict the adjacent membership function to fully overlap and also constrain one membership function to have its center resting at the lower boundaries of the input range. By using this coding, only *n-1* membership function centers needed to be encoded, here *n* is the maximum number of partitions for a given dimension. These methods encode all possible rules into the chromosome. There are certain drawbacks by doing so: first the computational efficiency associated with fuzzy logic is lost using a high number of rules and second, the robustness decreases with increasing the number of rules. This is especially true when the dimension of the inputs and the number of fuzzy sets for each input variable become great since the number of possible rules exponentially increases with these numbers.

In most applications, not all possible rules need to be used; only a portion of the rules are needed. So only this portion of rules should be encoded into the chromosome and evolved. By doing so length of the chromosome is reduced greatly and therefore is suitable for larger as well as real time problems. Karr [162] considers a very special case where the number of rules provided by an expert, together with many complete rules forming the rule set and the antecedents for the remaining ones so only the consequent parts of the latter type need to be evolved and, therefore, need to be encoded in the chromosome. This is not the case for many applications. Most of the time, it will be difficult, if not impossible, to know a priori exactly how many rules are required to be included in the rule set; only maximal number can be guessed or estimated.

It is better to encode the number of rules to be included in the rule set together with rules and/ or membership functions into the chromosome to be evolved. There are several ways to do this Lee and Takagi [163] proposed encoding membership functions and fitness functions in chromosomes. Shimojima et al [164] and Inoue et al [165] defined membership functions for each rule and encoded effectiveness information of each rule and membership functions. Shimojima et al used fitness functions that encouraged minimizing the number of rules; Inoue et al used a method they called "forgetting".

Due to highly complex and nonlinear characteristics of the problem space, uniform distribution of the fuzzy sets is usually not optimal. The performance of a fuzzy classification system based on fuzzy if-then rules depends on the choice of a fuzzy partition. If a fuzzy partition is too coarse, the performance may be low. If a partition is too fine, many fuzzy if-then rules cannot be generated because of the lack of training patterns in the corresponding fuzzy subspaces. For a problem, some parts of pattern space might require fine partition, while the other parts require only coarse partition. Therefore, the choice of an appropriate fuzzy partition is important and difficult. To cope with this difficulty, Ishibuchi et al [166] introduce the concept of distributed fuzzy if-then rules. They encode all fuzzy if-then rules corresponding to several different fuzzy partitions into a tri-value string {-1, 0, 1} and apply GA to remove the unnecessary rules from fuzzy if-then rules corresponding to the different fuzzy partitions. Since each possible rule for each subspace is coded into the chromosome, the length of the chromosome is very large when the number of input dimensions and/or of different partitions is large. Other ways to tackle the nonlinear distribution problem should be sought. A natural and better way to employ nonlinear functions in addition to linear functions as membership functions. Natural choices are *Gaussian functions, sigmoid functions*, etc. Through the inclusion of linear and nonlinear functions, the type of membership function for each fuzzy set will not be predetermined, but instead be evolved during the design process.

GA behavior is determined by the exploitation and exploration relationships kept through out the run [167]. Given fixed setting of the parameters such as crossover and mutation rates through out the run, the GA may have its exploitation / exploration relationship (EER) disproportioned and produce a lack of diversity in the populations [167]. Accordingly, GA parameter setting should be adapted through the run. Since the interaction between GA parameter settings and GA performance is complex and unknown, finding algorithm to achieve optimal adaptive parameters setting is very difficult, if not impossible. This suggests the use of fuzzy systems for adapting GA parameters and whose outputs are GA control parameters [167]. Lee and Takagi [161] propose an automatic learning technique to design fuzzy rules for tuning GA. Due to the heavy computation requirement, they first apply this technique to design a fuzzy system to tune a GA, and then apply the obtained fuzzy system to tune the GA to solve other different and more complex problems. This implies that the robustness of the obtained fuzzy rules strongly depends on the problems to be solved and the performance measures used by the technique. Shi et al [168] evolutionary fuzzy expert system have been discussed in which the membership function shapes and types and the fuzzy rule set, including number of rules inside the rule set, are evolved using GA. In addition, the genetic parameters of the evolutionary algorithm are adapted via a fuzzy expert system.

4.10 Summary

As far as design of real time nonlinear control system design is concerned, there are number of different methods are available but to determine about which method will be suitable for particular problem / application on hand is difficult. Researchers have tried different methods for different applications like control of spindle motor of a CD ROM drive [169], under water remotely operated vehicle [170], inverted pendulum [171], cart pole balancing [153], Robot control [154], obstacle avoidance [172] etc.

GA has limitation of recursive convergence, which limits its application to real time system in general. But hybrid approach of fuzzy logic, neural network and GA has proved to be better for real time application. Here, their usage will depend on the application. In the application referred above, researchers have found that results obtained with the hybrid approach are far better compared to traditional nonlinear controllers.