Chapter 7

HYBRIDIZATION OF SOFT COMPUTING TOOLS

7.1 THE NEED FOR HYBRIDIZATION

Viewed from a much broader perspective, the Fuzzy Logic, Artificial Neural Networks and Genetic Algorithms are the constituents of a vigorously developing research area called soft computing and they have proved their efficiency in many complex problems. However, certain problems cannot be efficiently solved using a single one of them as each one has its own limitations and weaknesses. These limitations were the driving force behind the creation of intelligent hybrid systems, where two or more soft computing tools are combined, to overcome the limited power of each individual technique. Actually speaking, hybrid architectures research is an attempt to develop the next generation of intelligent systems.

Although the fundamental inspirations for each of the soft computing methodologies are quite different, they share the common ability to improve the intelligence of systems working in an uncertain, imprecise and noisy environment. Extracting models of complex phenomena and defining how to act on them to achieve a desired result is usually possible only through some hybridization such as Neuro-Fuzzy concept, Fuzzy-GA combination, Neuro-Fuzzy-GA approach etc. Since different soft computing tools are complementary in nature rather than competitive, it is advantageous to employ them in combination rather than exclusively. This integration provides complementary reasoning and searching methods that allows one to combine domain knowledge and empirical data to develop flexible computing tools and solve complex problems. Long term goals and challenges with respect to integration of various AI tools are:

- Constructing efficient, uniformly transparent mechanisms for representing large amounts of knowledge and data.
- Lessening the tension between speed and quality of action by continuing adaptation and extension of knowledge-based reasoning and learning techniques to real-time operation and control of complex real-world systems that involve hard deadlines.
- Making computers easier to use, more co-operative and customizable with interfaces that employ natural languages and other modalities to communicate in familiar and convenient ways.

7.2 CHARACTERISTICS OF HYBRID SOFT COMPUTING

◊ **Human expertise**: Hybrid soft computing utilizes human expertise in the form of fuzzy if-then rules, as well as in conventional knowledge representations, to solve practical problems.

◊ **Biologically inspired computing models**: Artificial neural networks inspired by biological neural networks are employed extensively in soft computing to deal with perception, pattern recognition and nonlinear regression and classification problems.

◊ **New optimization techniques**: Hybrid soft computing applies innovative optimization methods arising from various sources, such as genetic algorithms, simulated annealing, random search method and the downhill simplex method.

◊ **New application domains**: Because of its numerical computation, hybrid soft computing has found a number of new application domains besides that of traditional approaches.

◊ **Model-free learning**: Neural networks and adaptive fuzzy inference systems have the ability to construct models using only target system data.

◊ **Real-world applications**: Most real-world problems are large scale and inevitably incorporate built-in uncertainties, this precludes using conventional approaches that require detailed description of the problem being solved. Hybrid soft computing is an integrated approach that can usually utilize specific techniques within subtasks to construct satisfactory solutions to real-world problems.

◊ **Fault tolerance**: Hybrid techniques exhibit fault tolerance because of its parallel and redundant architecture.

◊ **Intensive computation**: Using hybrid soft computing techniques without assuming too much background knowledge of the problem being solved.

◊ **Goal driven characteristics**: Most of hybrid techniques are goal driven, i.e. the path leading from the current state to the solution does not really matter as long as one is moving towards the goal in the long run.

Various hybrid soft computing techniques which have been used in the present study are discussed here in brief.

7.3 Combining GA with Fuzzy Logic

GAs are search algorithms that use operations found in natural genetics to guide the trek through a search space. GAs use a direct analogy of natural behavior. GAs are theoretically and empirically proven to provide robust search, offering a valid approach to problems requiring efficient and effective search. Much of the interest in GAs is due to the fact that they provide a set of efficient domain-independent search heuristics which are a significant improvement over traditional methods without the need for incorporating highly domain specific knowledge. In spite of the robust search capability GA sometimes fails to find the optimum solution, especially when the optimum solution is very near to the constraint boundaries. In the traditional optimization techniques, these constraints must be strictly satisfied. However, it is not reasonable to discard those designs that slightly violate one or more constraints during the early design stage. These complexities and uncertainties encountered in the optimization and design of real structures provide the main motivation for examining a fuzzy integrated system.

From the literature survey it is found that hybridization of GA and FL yields two main approaches. The first approach, in which the robustness of GA is improved with the help of fuzzy logic concept is known as Fuzzy (controlled) Genetic Algorithms (FGA) and the second approach, in which GA tunes or learns various parameters of fuzzy systems used for decision making problems, is called Genetic Fuzzy Systems (GFS).

7.3.1 Fuzzy Genetic Algorithm

The use of Fuzzy-Sets based techniques permits GA behavior to be improved in different ways such as: representation models for dealing with more complex genotype-phenotype relationships, fuzzy logic control systems for introducing a self-control parameter process according to some performance measures, fuzzy operators and fuzzy connectives for designing genetic operators that introduce different population diversity levels and mechanisms, that allow GA performance to be analyzed from a human point of view. Also, some hybrid techniques consider variables with fuzzy values representing these in a chromosome and the second one considers non fuzzy variables with a fuzzy evaluation, having a fuzzy fitness.

It is important to note that the use of GAs may offer great potential for the Fuzzy-Sets-based optimization approaches for representing uncertainty and approximation in relationships between system variables, given the potential of GAs in a fuzzy environment as a flexible tool for optimization and search. In fact, GAs have been used for solving different fuzzy optimization problems and in which GA's parameters are fuzzified using fuzzy set concept.

7.3.2 Genetic Fuzzy Systems

A GFS is basically a fuzzy system augmented by a learning process based on a genetic algorithm. GAs are search algorithms that provide robust search capabilities in complex spaces, and thereby offer a valid approach to problems requiring efficient and effective search processes.

Genetic processes cover different levels of complexity according to the structural changes produced by the algorithm, from the simplest case of parameter optimization to the highest level of complexity of learning the rule set of a rule based system. Parameter optimization has been the approach utilized to adapt a wide range of different fuzzy systems, as in genetic fuzzy clustering or genetic-neuro-fuzzy systems. Analyzing the literature the most extended GFSs are *Genetic Fuzzy Rule-Based Systems* (GFRBSs) [74], where the genetic process learns or tunes different components of a fuzzy rule-based system (FRBS). Inside GFRBSs it is possible to find either parameter optimization or rule generation processes. The first step in designing a GFRBS is to decide which parts of the Knowledge Base (KB) are subject to optimization by the GA. The KB of an FRBS does not constitute a homogeneous structure but is rather the union of qualitatively different components. For example, the KB of a descriptive Mamdani-type FRBS is comprised of two components: a data base (DB), containing the definitions of the scaling factors and the membership functions of the fuzzy sets associated with the linguistic labels, and a rule base (RB), constituted by the collection of fuzzy rules. The decision which part of the KB to adapt depends on two conflicting objectives: granularity and efficiency of the search.

A search space of smaller dimension results in a faster and simpler learning process, but the obtainable solutions might be suboptimal. A larger, complete search space that comprises the entire KB and has a finer granularity is therefore more likely to contain optimal solutions, but the search process itself might become prohibitively inefficient and slow. With these considerations there is an obvious trade-off between the completeness and granularity of the search space and the efficiency of the search which offers different possibilities for GFS design. First of all, it is important to distinguish between tuning and learning problems.

Tuning is more concerned with optimization of an existing FRBS, whereas learning constitutes an automated design method for fuzzy rule sets that starts from scratch. Tuning processes assume a predefined RB and have the objective to find a set of optimal parameters for the membership and/or the scaling functions. Learning processes perform a more elaborated search in the space of possible RBs or whole KBs and do not depend on a predefined set of rules. Some of the possibilities are discussed below.

(a) Genetic learning of the rule base

A number of papers have been devoted to analyze the automatic generation of the knowledge base of a FRBS using GAs. The key point is to employ an evolutionary learning process to automate the KB generation, which can be considered as an optimization or search problem. The KB parameters constitute the optimization space, which is transformed into a suitable genetic representation on which the search process operates. When considering a rule based system and focusing on learning rules, there are three main approaches that have been applied in the literature: Pittsburgh approach [75], Michigan approach [76] and Iterative rule learning approach [77]. Pittsburgh and Michigan approaches are the most extended methods for rule learning developed in the field of GAs. The first one is characterized by representing an entire rule set as a genetic code (chromosome), maintaining a population of candidate rule sets and using selection and genetic operators to produce new generations of rule sets. The Michigan approach considers a different model where the members of the population are individual rules and a rule set is represented by the entire population. In the third approach, the iterative one, chromosomes code individual rules, and a new rule is adapted and added to the rule set, in an iterative fashion, in every run of the genetic algorithm.

(b) Genetic tuning

The tuning of the scaling functions and fuzzy membership functions is an important task in FRBS design. The parameterized scaling functions and membership functions are adapted by the GA according to a fitness function that specifies the design criteria in a quantitative manner.

(c) New trends in genetic fuzzy rule-based systems

In addition to those presented in previous section, it is possible to explore new directions in applying genetic (evolutionary) techniques to FRBSs. Two possibilities are:

(i) Designing fuzzy rule-based systems with genetic programming [78]

(ii) Genetic selection of fuzzy rule sets

(d) Genetic fuzzy clustering algorithms

There are several references in the literature proposing the application of GAs in fuzzy clustering, most of them devoted to improve the results of FCM-type algorithms [79] by using the GA to optimize some parameters of these kinds of algorithms. The use of GAs to optimize the parameters of an FCM-type algorithm generates two different GFSs. *Prototype-based algorithms* encode the fuzzy cluster prototypes and evolve them by means of a GA guided by any centroid-type objective function [80]. *Fuzzy partition-based algorithms* encode, and evolve, the fuzzy membership matrix [81]. A second possibility is to use the GA to define the distance norm of an FCM-type algorithm. The system considers an adaptive distance function and employs a GA to learn its parameters in order to obtain an optimal behaviour of the FCM-type algorithm. Finally, a third group of genetic approaches are based on directly solving the fuzzy clustering problem without interaction with any FCM-type algorithm.

7.4 COMBINING GA WITH NEURAL NETWORK

Artificial neural networks (ANN) modeling is now established as one of the more practical branch of machine learning, and is being widely used in most scientific, engineering and business modeling applications. Modeling real-life data with large number of input variables, which can be done by training an artificial neural network (ANN), is a complex and a time consuming task. The developer of the ANN has to answer two central questions: which are the relevant inputs and which dimension of the hidden layer is appropriate for a good out-of-sample prediction. The number of inputs and outputs, which are related to the desired model, determines the architecture of the ANN. The ANN model developer selects the number of hidden neuron layers and the number of neurons in each layer. There is a consensus, based on theoretical considerations, that one hidden layer is sufficient. It is also known that too many neurons degrade the effectiveness of the model, as the number of connection weights in the ANN model may cause overfitting and loss of the generalization capacity. On the other hand, too few hidden neurons may not capture the full complexity of the data.

Many heuristic rules have been suggested for finding the optimal number of neurons in the hidden layer. Most of them employ trial-and-error methods, in which the ANN training starts with a small number of hidden neurons, and additional neurons are gradually added until some performance goal is satisfied. In real-life modeling, in which sometime a small number of training examples are available, choosing large number of hidden neurons in an ANN leads to the following undesirable consequences: Large number of connection weights in the model; long training times, local minima, small generalization capacity etc. The result is a resistance to accepting the ANN as reliable modeling tool. Genetic Algorithm can be used to resolve these problems.

There are a few reasons why it can be beneficial to use GAs for training neural networks. With regard solely to the problem of weight (and bias) selection for networks with fixed topologies and transfer functions there is the issue of local optima. As the number of exemplars and the complexity of the network topology increase, the complexity of the search space also increases if the error function obtains more and more local minima spread out over a larger portion of the space. Gradient-based techniques often have problems getting past the local optima to find global optima. However, GAs are particularly good at efficiently searching large and complex spaces to find nearly global optima. As the complexity of the search space increases, GAs present an increasingly attractive alternative to gradient-based techniques such as backpropagation. Even better, GAs are an excellent complement to gradient-based techniques such as backpropagation for complex searches. A GA is run first to get to the right hill and the gradient-based technique climbs the hill to its peak.

A second advantage of GAs is their generality. With only minor changes to the algorithm, they can be used to train all different varieties of networks. They can select weights for recurrent networks, i.e. networks whose topologies have closed paths. They can train networks with different transfer functions than sigmoids, such as the Gaussians used by WPNN or even step transfer functions, which are discontinuous and hence not trainable by gradient techniques. Furthermore, GAs can optimize not just weights and biases but any combination of weights, biases, topology, and transfer functions.

A third reason to study GAs for learning neural networks is that this is an important method used in nature. While it is not clear that evolutionary learning methods are used in individual organisms it is relatively certain that evolutionary learning at a species level is a major method of neural network training. For example, in parts of the brain such as the early visual processing component, the neural network weights (in addition to the topology and transfer functions) are essentially "hard-wired" genetically and hence were "learned" via the process of natural selection.

Genetic algorithms have been used together with neural networks in a variety of ways, each with their own body of literature. One such way is to use GAs for the reprocessing of NN data, usually using the GA to perform feature selection. A second use of GAs with NNs is to use the GA to select the NN topology. A third such application is to employ GAs for selecting the weights of NN. Some approaches optimize both the weights and the topology simultaneously. A fourth application is to use GAs to learn the NN learning algorithm.

There are two basic differences between the different approaches for using GAs for weight selection. The first difference is that the weight selection is performed on different architectures. Examples of the different network architectures to which genetic weight selection algorithms have been applied are: feedforward sigmoidal, cascade-correlation, recurrent sigmoidal, recurrent linear threshold, feedforward with step functions, and feedback with step function. The second difference is the difference in the genetic algorithms.

Another possible combination of GA with NN is Neuro-GA in which ANN is being employed for predicting the fitness of the candidate solution, when fitness evaluation is computationally intensive as in case of optimization of continuum structures where fitness evaluation requires Finite Element analysis to be carried out. Once trained, ANN can help in calculating fitness values faster to accelerate the search process.

7.5 COMBINING NEURAL NETWORKS WITH FUZZY LOGIC

Hybrids of ANNs and FL are referred Neuro-Fuzzy systems [82]. It results in a hybrid intelligent system that synergies these two techniques by combining the human-like reasoning style of fuzzy systems with the learning and connectionist structure of neural networks. Thus, Neuro-Fuzzy models incorporate, from fuzzy models, the facility of understanding the model in realistic way and on the other hand, from neural networks based models, the capability of adaptation and learning. Neuro-Fuzzy hybridization is widely termed as Fuzzy Neural Network (FNN) or Neuro-Fuzzy System (NFS).

The performance of the tuned fuzzy system is highly dependent on the initial fuzzy sets, while the optimal initial fuzzy system is usually unknown even for the expert. In most of the cases membership functions are fixed and predefined, this approach lacks flexibility. Furthermore, as the number of variables and membership functions increases, the interpretability of the fuzzy system becomes questionable. However, while neural networks based model can be considered as "black box", Neuro-Fuzzy models allow incorporating expert knowledge in different parts of the modeling process.

Fuzzy BPNN (FBPN) can be constructed by modifying the structure of a conventional crisp BPN to accept fuzzy-valued inputs. In the process some problem-specific experts are requested to evaluate the importance of each input parameter with linguistic terms, which can then be converted into fuzzy numbers, aggregated, and multiplied to the corresponding input parameter. Fuzzy-valued inputs are propagated through the FBPN, and finally the network output is generated. The fuzzy-valued network output is then defuzzified and applied to predict the actual output. After that, the deviation between the defuzzified output and the actual output is propagated backward to adjust the thresholds and connection weights in the FBPN. With respect to prediction accuracy (effectiveness), the minimal RMSE achieved by applying the FBPN is slightly lower than that of the crisp BPN. On the other hand, the learning of FBPN starts with a considerably smaller value of the initial RMSE than that of the crisp BPN. Also, much fewer epochs are required to converge to the minimal RMSE with the FBPN than with the crisp BPN. Experience of researchers confirms that the FBPN outperforms the crisp BPN in the respect of efficiency represented with the two indexes.

7.6 Combining GA, FL and ANN

Genetic fuzzy neural networks are the result of adding genetic or evolutionary learning capabilities to systems integrating fuzzy and neural concepts. The result will be a genetic-neuro-fuzzy system (or a genetic-fuzzy-neural network) [74]. The usual approach of most genetic-fuzzy-neural networks found in the literature, is that of adding evolutionary learning capabilities to a fuzzy neural network that usually is a feed-forward multilayered network to which, previously, some fuzzy concepts where incorporated. The result is a feed-forward multilayered network having fuzzy and genetic characteristics. Genetic-fuzzy-neural networks incorporate fuzzy numbers to represent the weights, perform fuzzy operations in the nodes of the network, and/or incorporate fuzzy nodes that represent membership functions. Also, the learning process applies GAs to obtain the weights of the neural network, to adapt the transfer functions of the nodes and/or to adapt the architecture of the net.