

# Soft Computing Techniques for Diagnostics and Prognostics

**Piero P. Bonissone & Kai Goebel**

GE Corporate Research and Development  
One Research Circle, Niskayuna, NY 12309, USA  
bonissone@crd.ge.com; goebel@crd.ge.com

## Abstract

This paper presents methods and tools which can be used within the framework of diagnostics and prognostics to accommodate imprecision of real systems. We outline the advantages and disadvantages of the different techniques and show how they can be used in a hybrid fashion to complement each other. We conclude the paper with a number of successful real world examples.

## Introduction

### Motivation

Diagnostics and prognostics has recently undergone revived interest in industry. Manufacturing companies are trying to shift their operation to the service field where they suspect an opportunity to yield higher margin. One embodiment is the long term service agreements with guaranteed uptime. The latter shifts the motivation to keep the equipment in working order more to the service provider. Since actually performing a maintenance action is generally undesired and costly, in the interest of maximizing the margin, service should only be performed when necessary. This can be accomplished by using a tool which measures the state of the system and indicates any incipient failures. Such a tool must have a high level of sophistication which incorporates monitoring, fault detection, decision making about possible preventive or corrective action, and monitoring of its execution. Because of the complexity of the task, AI and in particular also Soft Computing, has been called upon to help. Recently we have witnessed the application of many Soft Computing techniques in support of service tasks such as anomaly detection and identification, diagnostics, prognostics, estimation and control (Bonissone et al 1995, Chen & Bonissone 1998). In this paper we will briefly describe the components of Soft Computing and illustrate some of their most successful application to services and maintenance.

### Soft Computing

Soft Computing (SC), a term originally coined by Zadeh (1994) as "... *tolerant of imprecision, uncertainty, and partial truth*", is a sub-field of Artificial Intelligence.

According to Zadeh, Soft Computing is "*an association of computing methodologies which includes as its principal members fuzzy logics (FL), neuro-computing (NC), evolutionary computing (EC) and probabilistic computing (PC).*" (Zadeh 98). It should be noted however that we have not reached a consensus yet as to the exact scope or definition of SC (see for instance Dubois & Prade 1998).

The main reason for Soft Computing popularity is the *synergy* derived from its components. SC main characteristics is its intrinsic capability to create *hybrid systems* that are based on a (loose or tight) integration of these technologies. This integration provide us with complementary reasoning and searching methods that allows us to combine domain knowledge and empirical data to develop flexible computing tools and solve complex problems. Fig. 1 illustrates a taxonomy of these hybrid algorithms and their components. The interest reader should consult the work of Bouchon et al. (1995) and Bonissone (1997) for an extensive coverage of this topic.

### SC Components and Taxonomy

Fuzzy logics, introduced by Zadeh (1965), gives us a language, with syntax and local semantics, in which we can translate our qualitative knowledge about the problem to be solved. FL's main characteristic is the robustness of its interpolative reasoning mechanism. A comprehensive review of fuzzy computing can be found in Ruspini et al. (1998).

Probabilistic computing such as Bayesian Belief Networks, based on the original work of Bayes (1763) and Dempster-Shafer's theory of belief, independently developed by Dempster (1967) and Shafer (1976), gives us the mechanism to evaluate the outcome of systems affected by randomness or other types of probabilistic uncertainty. PR's main characteristic is its ability to update previous outcome estimates by conditioning them with newly available evidence.

Neural networks (NN), a major component of neuro-computing, were first explored by Rosenblatt (1959) and Widrow (1960). NN are computational structures that can be trained to learn patterns from examples. By using a

training set that samples the relation between inputs and outputs, and a learning method, for example a back-propagation type of algorithm introduced by Werbos (1974), neuro-computing (and in particular neural networks) give us a supervised learning algorithms that perform fine-granule local optimization. A comprehensive current review of neuro-computing can be found in Fiesler and Beale (1997).

Genetic Algorithms (GA), a major component of evolutionary computing, were first proposed by Holland (1975). GA give us a method to perform randomized global search in a solution space. In this space, a population of candidate solutions, encoded as chromosomes, is evaluated by a fitness function in terms of its performance. The best candidates *evolve* and pass some of their genetic characteristics to their *offsprings*, who form the next generation of potential solutions. For a current review of evolutionary computing the reader is referred to Back et al. (1997).

The common denominator of these technologies is their departure from classical reasoning and modeling approaches that are usually based on Boolean logic, analytical models, crisp classifications, and deterministic search. In ideal problem formulations, the systems to be modeled or controlled are described by complete and precise information. In these cases, formal reasoning systems, such as theorem provers, can be used to attach binary truth values to statements describing the state or behavior of the physical system.

When we solve real-world problems, we realize that they are typically ill-defined systems, difficult to model and with large-scale solution spaces. In these cases, precise models are impractical, too expensive, or non-existent. Our solution must be generated by leveraging two kind of resources: *problem domain knowledge* of the process/product that we want to improve/fix and *field data* that characterize the behavior of such process/product. The relevant available domain knowledge is typically a combination of first principles and experiential/empirical knowledge, and is usually incomplete and sometimes erroneous. The available data are typically a collection of input-output measurements, representing instances of the system's behavior, and are usually incomplete and noisy.

We can observe from Fig. 1 that the two main approaches in Soft Computing are *knowledge-driven* reasoning systems (such as Probabilistic and Multivalued Systems) and *data-driven* search and optimization approaches (such as Neuro and Evolutionary Computing). This taxonomy, however, is soft in nature, given the existence of many hybrid systems that span across more than one field, as we will see in a subsequent Section.

### Alternative Approaches to SC

The alternative approaches to SC are the traditional knowledge-driven reasoning systems and the data-driven systems. The first class of approaches are exemplified by first-principle derived models (based on differential or difference equations), by first-principle qualitative models (based on symbolic, qualitative calculi - see Forbus

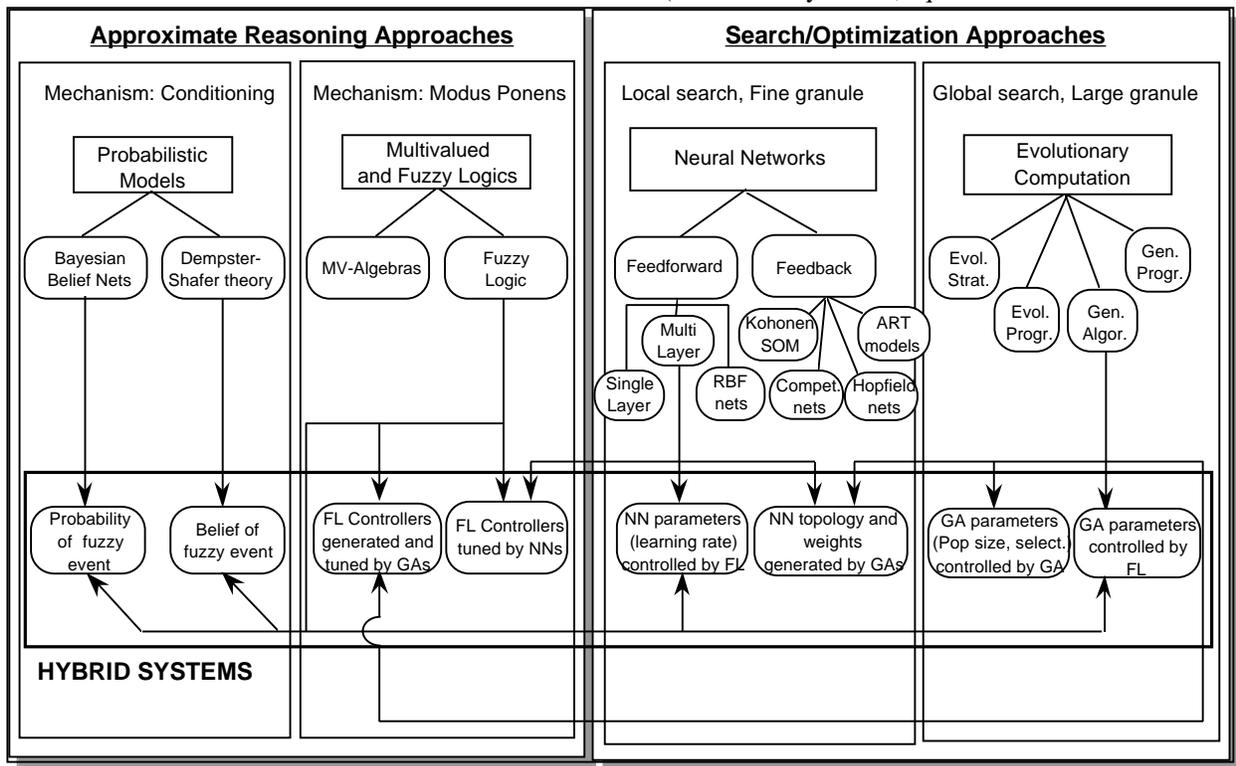


Fig. 1: SC Components and Hybrid Systems

(1981) and Kuipers (1985), by classical Boolean systems, such as theorem provers (based on unification and resolution mechanisms) or by expert systems embodying empirical or experiential knowledge. All these approaches are characterized by the encoding of problem domain knowledge into a model that tries to replicate the system's behavior. The second class of approaches are the regression models and crisp clustering techniques that attempt to derive models from any information available from (or usually buried in) the data.

Knowledge driven systems, however, have limitations, as their underlying knowledge is usually incomplete. Sometimes, these systems require the use of simplifying assumptions to keep the problem tractable (e.g. linearization, hierarchy of local models, use of default values). Theoretically derived knowledge may even be inconsistent with the real system's behavior. Experiential knowledge, on the other hand, could be static, represented by a collection of instances of relationships among the system variables (sometimes pointing to causality, more often just highlighting correlation). The result is the creation of precise but simplified models that do not properly reflect reality or the creation of approximate models that tend to become stale with time and are difficult to maintain.

Purely data-driven methods also have their drawbacks, since data tend to be high dimensional, noisy, incomplete (e.g., DBs with empty fields in their records), wrong (e.g., outliers due to malfunctioning/failing sensors, transmission problems, erroneous manual data entries), etc. Some techniques have been developed to address these problems, such as feature extraction, filtering and validation gates, imputation models, and virtual sensors (which model the recorded data as a function of others variables).

The fundamental problem of these classic approaches resides in the difficulty to represent and integrate uncertain, imprecise knowledge in data-driven methods or to make use of somewhat unreliable data in a knowledge-driven approach.

Although it would be presumptuous to claim that Soft Computing *solves* this problem, it is reasonable to state that SC provides a different paradigm in terms of representation and methodologies which facilitates these integration attempts. For instance, in classical control theory the problem of developing models is decomposed into system identification and parameter estimations. Usually the first one is used to determine the order of the differential equations and the second one to determine its coefficients. Hence, in this traditional approach we have *model = structure + parameters*. This equation does not change with the advent of Soft Computing. However we now have a much richer repertoire to represent the

structure, to fine tune the parameters, and to iterate in this process.

For example, the knowledge base (KB) in a Mamdani type fuzzy systems (Mamdani 1975) is typically used to approximate a relationship between a state X and an output Y. The KB is completely defined by a set of *scaling factors* (SF), determining the ranges of values for the state and output variables; a *termset* (TS), defining the membership distribution of the values taken by each state and output variable; and by a *ruleset* (RS), characterizing a syntactic mapping of symbols from X to Y. The *structure* of the underlying model is the ruleset, while the model *parameters* are the scaling factors and termsets. The inference obtained from such system is the result of interpolating among the outputs of all relevant rules. This results in a membership distribution defined on the output space, which is then aggregated (de-fuzzified) to produce a crisp output. With this inference mechanism we can define a deterministic mapping between each point in the state space and their corresponding output. Therefore, we can now equate a fuzzy KB to a response surface in the cross product of state and output spaces which approximates the original relationship.

A Takagi-Sugeno-Kang (TSK) type of fuzzy systems (Takagi and Sugeno 1985) increase its representational power by allowing the use of a first order polynomial, defined on the state space, to be the output of each rule in the ruleset. This enhanced representational power (at the expense of local legibility) results in a model which is equivalent to Radial Basis Function (Bersini 1995). The same model can be translated into a structured network, such as ANFIS (Jang 1993), in which the ruleset determines the topology of the net (model structure), while the termsets and the polynomial coefficients are defined by dedicated nodes in the corresponding layers of the net (model parameters). Similarly, in the traditional neural networks the topology represents the model structure and the links weights represent the model parameters.

While NN and structured nets use local search methods, such as back-propagation, to tune their parameters, it is possible to use global search methods, such as genetic algorithms, to achieve the same parametric or to postulate new structures. An extensive coverage of these approaches can be found in Bonissone (1997). In the next section we will provide a brief summary.

## Hybrid Algorithms: The Symbiosis

Over the past few years we have seen an increasing number of hybrid algorithms, in which two or more of Soft Computing technologies have been integrated. The

motivation is to leverage the advantages of individual approaches to combine smoothness and embedded empirical qualitative knowledge with adaptability and general learning ability to achieve improvement of overall algorithm performance. We will now analyze a few of such combinations, as depicted in the lower box of Fig. 1. First we will illustrate the use of NN and GA to tune FL systems, as well as the use of GA to generate/tune NN. Then, we will see how fuzzy systems can control the learning of NN and the run-time performance of GAs.

**FL tuned by NN.** Within the limited scope of using NNs to tune FL controllers, we want to mention the seminal work on ANFIS (Adaptive Neural Fuzzy Inference Systems), proposed by Jang (1993). ANFIS is a representative hybrid system in which NNs are used to tune a FLC. ANFIS consists of a six layers generalized network. The first and sixth layers correspond to the system inputs and outputs. The second layer defines the fuzzy partitions (termsets) on the input space, while the third layer performs a differentiable T-norm operation, such as the product or the soft-minimum. The fourth layer normalizes the evaluation of the left-hand-side of each rule, so that their degrees of applicability will add up to one. The fifth layer computes the polynomial coefficients in the right-hand-side of each Takagi-Sugeno rule. Jang's approach is based on a two-stroke optimization process. During the forward stroke the termsets of the second layer are kept equal to their previous iteration value while the coefficients of the fifth layer are computed using a Least Mean Square method. At this point ANFIS produces an output, which is compared with the one from the training set, to produce an error. The error gradient information is then used in the backward stroke to modify the fuzzy partitions of the second layer. This process is continued until convergence is reached. .

**FL tuned by GAs.** Many researchers have explored the use of genetic algorithms to tune fuzzy logic controllers. Cordon et al. (1995) have produced an updated bibliography of over 300 papers combining GAs with fuzzy logic, of which at least half are specific to the tuning and design of fuzzy controllers by GAs. For brevity's sake we will limit this section to a few contributions. These methods differ mostly in the order or the selection of the various FC components that are tuned (termsets, rules, scaling factors).

Karr, one of the precursors in this quest, used GAs to modify the membership functions in the termsets of the variables used by the FCs - see Karr (1991). In his work, Karr used a binary encoding to represent three parameters defining a membership value in each termset. The binary chromosome was the concatenation of all termsets. The fitness function was a quadratic error calculated for four randomly chosen initial conditions.

Kinzel et al. (1994) tuned both rules and termsets. They departed from the string representation and used a cross-product matrix to encode the rule set (as if it were in table form). They also proposed customized (point-radius) crossover operators which were similar to the two-point crossover for string encoding. They first initialized the rule base according to intuitive heuristics, used GAs to generate better rule base, and finally tuned the membership functions of the best rule base. This order of the tuning process is similar to that typically used by self-organizing controllers, as illustrated in Burkhardt and Bonissone (1992).

Lee and Takagi (1993) also tuned the rule base and the termsets. They used a binary encoding for each three-tuple characterizing a triangular membership distribution. Each chromosome represents a TSK-rule, concatenating the membership distributions in the rule antecedent with the polynomial coefficients of the consequent.

Bonissone et al (1996) followed the tuning order suggested by Zheng (1992) for manual tuning. They began with macroscopic effects by tuning the FC state and control variable *scaling factors* while using a standard uniformly spread termset and a homogeneous rule base. After obtaining the best scaling factors, they proceeded to tune the termsets, causing medium-size effects. Finally, if additional improvements were needed, they tuned the *rule base* to achieve microscopic effects.

This parameter sensitivity order can be easily understood if we visualize a homogeneous rule base as a rule table: a modified scaling factor affects the entire rule table; a modified term in a termset affects one row, column, or diagonal in the table; a modified rule only affects one table cell.

This approach exemplifies the synergy of SC technologies, as it was used in the development of a fuzzy PI-control to synthesize a freight train handling controller. This complex, real-world application could not have been addressed by classical analytical modeling techniques (without recurring to many simplifying assumptions). Furthermore, its solution space was too large for a pure data-driven approach. By using a fuzzy controller Bonissone (Bonissone et al., 1996) was able to translate locomotive engineers training procedures into an executable model that exhibited a reasonable performance. However, this performance, was far from optimal, even after manual tuning of the model. By using a genetic algorithm to tune the model's scaling factors and membership functions, Bonissone demonstrated that this approach was able to find much better solutions within a reasonable amount of time, under different train handling criteria. In addition, he showed that not all parameters needed to be treated equally, and that sequential

optimization could greatly reduce computational effort by tuning the scaling factors first. The scalability of this approach enabled him to tune the controller for each track profile, producing a library of offline-customized controllers.

**NNs Generated by GAs.** There are many forms in which GAs can be used to synthesize or tune NN: to evolve the network *topology* (number of hidden layers, hidden nodes, and number of links) letting the Back-Propagation (BP) learning algorithm tune the net; to find the optimal set of weights for a given topology, thus replacing BP; and to evolve the reward function, making it adaptive. The GA chromosome needed to directly encode both NN topology and parameters is usually too large to allow the GAs to perform an efficient global search. Therefore, the above approaches are usually mutually exclusive, with a few exceptions that rely on variable granularity to represent the weights.

Montana and Davis (1989) were among the first to propose the use of GAs to train a feedforward NN with a given topology. Typically NNs using Back-Propagation (BP) converge faster than GAs due to their exploitation of local knowledge. However this local search frequently causes the NNs to get stuck in a local minima. On the other hand, GAs are slower, since they perform a global search. Thus GAs perform efficient coarse-granularity search (finding the promising region where the global minimum is located) but they are very inefficient in the fine-granularity search (finding the minimum). These characteristics motivated Kitano (1990) to propose an interesting hybrid algorithm in which the GA would find a *good* parameter region which was then used to initialize the NN. At that point, BP would perform the final parameter tuning.

McInerney (1993) improved Kitano's algorithm by using the GA to escape from the local minima found by the BP during the training of the NNs (rather than initializing the NNs using the GAs and then tuning it using BP). They also provided a dynamic adaptation of the NN learning rate. For an extensive review of the use of GAs in NNs, the reader is encouraged to consult the work of Vonk et al. (1997).

**NN controlled by FL.** Fuzzy logic enables us to easily translate our qualitative knowledge about the problem to be solved, such as resource allocation strategies, performance evaluation, and performance control, into an executable rule set. This characteristic has been the basis for the successful development and deployment of fuzzy controllers.

Typically this knowledge is used to synthesize fuzzy controllers for dynamic systems - see Bonissone et al (1995). However, in this case the knowledge is used to

implement a smart algorithm-controller that allocates the algorithm's resources to improve its convergence and performance. As a result, fuzzy rule bases and fuzzy algorithms have been used to monitor the performance of NNs or GAs and modify their control parameters. For instance, FL controllers have been used to control the learning rate of Neural Networks to improve the crawling behavior typically exhibited by NNs as they are getting closer to the (local) minimum.

The learning rate is a function of the step size and determines how fast the algorithm will move along the error surface, following its gradient. Therefore the choice of the learning rate has an impact on the accuracy of the final approximation and on the speed of convergence. The smaller its value the better the approximation but the slower the convergence. The momentum represents the fraction of the previous changes to the weight vector, which will still be used to compute the current change. As implied by its name, the momentum tends to maintain changes moving along the same direction thus preventing oscillations in shallow regions of the error surface and often resulting in faster convergence. Jacobs (1988) established a heuristic rule, known as the *Delta-bar-delta* rule to increase the size of the learning rate if the sign of the error gradient was the same over several consecutive steps. Arabshahi et al (1992) developed a simple fuzzy controller to modify the learning rate as a function of the error and its derivative, considerably improving Jacobs' heuristics.

The selection of these parameters involves a tradeoff: in general, large values of learning rate and momentum result in fast error convergence, but poor accuracy. On the contrary, small values lead to better accuracy but slow training, as proved by Wasserman (1989).

**GAs controlled by FL.** The use of Fuzzy Logic to translate and improve heuristic rules has also been applied to manage GA resources (population size, selection pressure, probabilities of crossover and mutation,) during their transition from *exploration* (global search in the solution space) to *exploitation* (localized search in the discovered promising regions of that space - see Cordon (1995), Lee (1993).

The management of GA resources gives the algorithm an adaptability that improves its efficiency and converge speed. According to Herrera & Lozano (1996), this adaptability can be used in the GA's parameter settings, genetic operators selection, genetic operators behavior, solution representation, and fitness function.

In general we can state that the management of GA resources gives the algorithm an adaptability that improves its efficiency and converge speed. The crucial aspect of this approach is to find the correct balance

between the computational resources allocated to the meta-reasoning (e.g., the fuzzy controller) and to the object-level problem-solving (e.g., the GA). This additional investment of resources will pay off if the controller is extendible to other object-level problem domains and if its run-time overhead is offset by the run-time performance improvement of the algorithm.

This brief review of hybrid systems illustrates the roles interaction of knowledge and data in SC. To tune *knowledge-derived models* we first translate domain knowledge into an initial structure and parameters and then use global or local data search to tune the parameters. To control or limit search by using prior knowledge we first use global or local search to derive the models (structure + parameters), we embed knowledge in operators to improve global search, and we translate domain knowledge into a controller to manage the solution convergence and quality of the search algorithm. All these facets have been exploited in some of the service applications described below.

## Diagnosics and Prognostics Tasks

The task of diagnosis is to find an explanation for a set of observations and – in the case of prognosis – to forecast the course of events. Diagnosis can further be broken down into anomaly detection and failure identification, depending on the desired granularity of information required. Prognosis is concerned with incipient failure detection, margin prediction, or overall performance prediction. The latter can be prediction of efficiency, current system status, margins, etc.

The outcome of diagnosis and prognosis will drive planning and execution. Possible planning includes corrective action which can be either reactive or proactive. Corrective action includes reconfiguration of the current system or sub-system, derating the setpoint, or changing the goal. Another possible plan is maintenance planning which has to take into consideration not only the current system status, but also the cost of maintenance (out of sequence vs. routine), disruption of service, and the cost of further damage. All these steps can be interim fixes or tactical decisions.

Some of the challenges that diagnosis and prognosis systems face are the ability to a changing environment which must allow the distinction between “normal” wear or desired system changes and the reportable system deviation. The transients are often times very similar and a proper distinction becomes important. Environments do not only change with time but also in space. Mobile diagnosis or remote monitoring and diagnosis systems are one possible answer. However, accessibility and

transmittability are limited by bandwidth and cost. Therefore, the system may be equipped with remote repair capabilities which is a step towards an autarkic system. This implies a more sophisticated decision maker which can reason about the information gathered and come to an optimal judgment within the constraints of the system.

Diagnosis and prognosis of complex systems will likely incorporate a number of different techniques and tools which are refined to deal with particular problems. That means also that some tools may provide a diagnosis for the same fault. It is to be expected that the diagnostic or prognostic findings of different tools are not exactly the same. In the best case, the fault magnitude is found to be different and one has to find a trade-off for the degree of failure. In the worst case, the tools completely contradict. It is then necessary to provide some conflict resolution capabilities and to weigh reliability, evidential manifestations, fault magnitude, etc. In addition, the domain in which the fault is being reported may be different, such as either a probabilistic or fuzzy domain, simple weights or binary state evaluations. A mapping from these different domains to a unified one is required. Finally, information acquired at different times, system states, sampling periods must be aggregated to a final evaluation. This is all task of the diagnostic information fusion.

## Case Studies of SC Applications in Diagnosics and Maintenance

### Distributed ANFIS models to predict voltage breakdown in power distribution network

**Problem.** Power blackouts have been traced in a number of cases to the spread of initially localized events. These events were caused by voltage instabilities which then extended to voltage collapses (Yabe et al., 1995). Typically, power systems consist of a complex web of interconnected components such as bulk transmission and local power generation. Problems for example at an urban subsystem can percolate through the system and lead to a blackout of a larger area. The vulnerability of the system to voltage collapse depends on system operating conditions, is driven by customer behavior, and is not very easily distinguishable from normal operating conditions. Conventional countermeasures are limited by the cost involved and the allowable operating points. It is therefore desired to monitor and predict voltage stability of power systems.

**Solution.** Most techniques for voltage stability calculation calculate the point of collapse or the reactive power margin. For a given system configuration, the interaction between voltage, reactive power and active power are

defined in a three-dimensional surface. Load variation results in a trajectory on that surface (Fig. 2). It is assumed that the collapse line is located at points of infinite voltage derivatives. Different system configurations (other than changes in load which is simply movement on the local surface) result in different surfaces and it is the goal to provide monitoring and prediction of local subsystems.

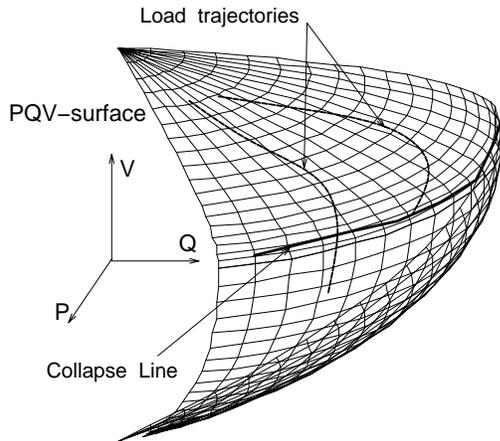


Fig. 2: Load-voltage surface with trajectories and collapse line

To start, each surface was represented via ANFIS (Yabe et al., 1995). Training was performed using a range of power system configurations and recording local observations while the system is driven to collapse. These surfaces represent a model base. In addition, an inference module was utilized which inferred the margin to collapse for the current state of the system. If the state lies on one of the surfaces, then several parameters can be computed such as directional vector, minimum  $dS$ , constant power factor (PF) as displayed in Fig. 3.

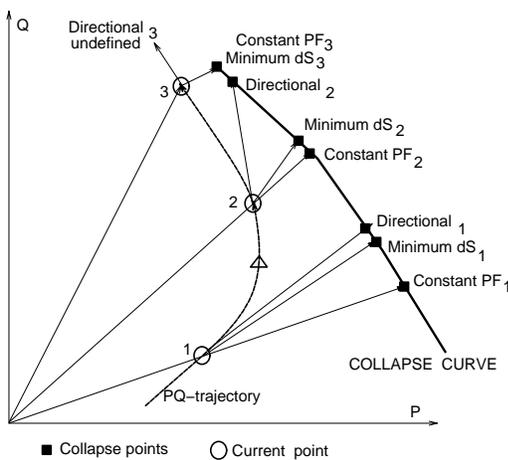


Fig. 3: Collapse points

In addition, a load prediction module was employed which used a fuzzy exponential weighted moving average predictor which utilizes an adaptive smoothing parameter (Khedkar & Keshav, 1992). The parameter changes its value based on the system status which is captured in fuzzy rules. Finally, a self-diagnosis module was employed which calculates several indices (equivocation, inapplicability, urgency) to evaluate how well the system performs its monitoring task. It uses these indices to assess its own reliability in predicting a voltage collapse.

**Results.** The system was tested on a variety of different settings ranging from small to large system. Noteworthy is the interpolative power of the overall system which compensates for a lack of being able to cover all possible scenarios with empirical data for training or to interpolate between rules. Fig. 4 shows the interaction of the different modules employed in this scheme. A small system test was set up where two power plants were modeled by a generator, an exciter, a power system stabilizer, a turbine, and a boiler. The system was used to test the predictive capability of the system which performed very well. A different set-up with a different amount of shunt compensation was a test for the fuzzy inference since the predictive system had never seen this configuration before. Collapse was predicted when subject to different load conditions about one minute before the actual event. For a large system test, 25 machines, 91 buses, and 109 branches were modeled to reflect a location between a bulk transmission and a large urban subsystem with both load and local generation. The load at all buses was ramped in this case. Test cases included changes in generation dispatch, strength of the transmission system, etc.

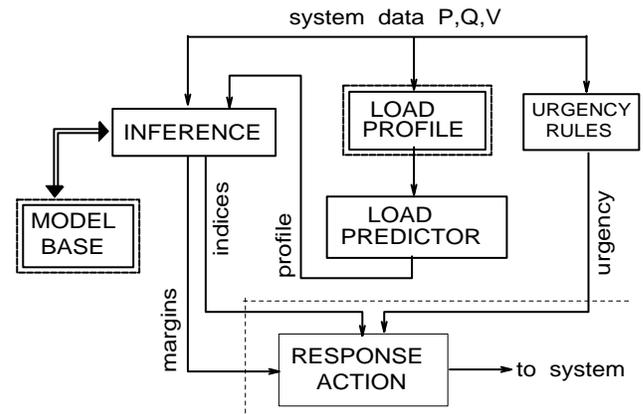


Fig. 4: Architecture of the System

## Prediction of Paper Web Breakage in Paper Mill Using Neural Nets and Induction Trees

Paper mills have enormous size of often times a hundred meters in length or more. Making paper involves running miles of wet and dry webs over and between large cylinders which convert pulp into the final product. The speed of the web can reach 30 mph. With those conditions, it is not surprising that the web breaks about once a day. Breakage results in a stand-still of the machine of about an hour and a half which in turn cause plant wide revenue loss of several million dollars per day. It is therefore desirable to prevent breakage by predicting imminent breakage to be able to take preventive action.

Breakage does occur because the roll draw is larger than permissible. Draw is necessary to stretch the paper to acquire desired properties. The force exerted is between certain minimum and maximum values which are determined by the tendency of the web to stick to a roll and the maximum force which depends on non-constant web properties (tear strength, tensile strength, and dry content) and external factors (e.g., temperature, lubricant, roughness, ash content, and retention). To increase yield, it is desirable to run the machines at the upper operating point (Chen & Bonissone, 1998).

**Solution.** Two approaches were chosen because of their complementary nature. Neural nets with fuzzy accelerators and induction trees derived from decision trees. Both help in predicting some aspect of the web breakage. The former is more accurate and provides a finer estimate of the break tendency while the latter is more useful for diagnosis and for finding relevant features related to breakage. The two approaches were then combined to give different types of information about the system state (Fig. 5). The neural net output was used to indicate the possibility of a break in the short term using a continuous number. The output of the induction tree was interpreted as a description of the sensitivity relationships between the current break tendency and the current values of the input sensor readings.

Other approaches were considered as well, such as fuzzy causal modeling, principal component analysis, and learning vector quantization. However, they did not produce the same results as the methods chosen.

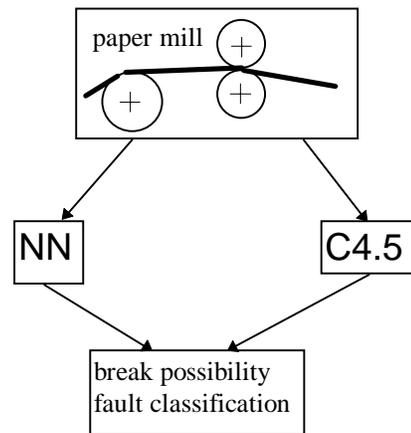


Fig. 5: Hybrid System for Breakage Detection

**Results.** The network was trained via backpropagation algorithm using normalized input sensor data. Output was a quantized measure before breakage such that the last ten readings before breakage were assigned a value of 0.9, the next last 20 readings a value of 0.5 and the 10 readings before that a value of 0.1. The interpretation could be low, medium, and high time to breakage. Training was done using fuzzy accelerators which adaptively adjust momentum, learning rate, and steepness parameters after each iteration. They react to the rate at which learning commences. For example, if the error derivative is the same over several learning epochs and the decrease of the error is small, the learning rate is increased. The other parameters are adjusted in a similar fashion whereby the condition is evaluated using fuzzy reasoning. During training a correct classification of 99.1% was achieved for the neural network. The validation set achieved 86.8% correct classification.

C4.5 was used as the induction tree and the same data as used for the neural network were employed for training and testing. Here, the classes were assigned the linguistic label “low”, “medium”, and “high”. A subset of the classification rules was then used in the on-line detection as a fault isolator to explain the current situation. A typical classification rule is:

IF *Wire water pH* < 0.104  
AND *Retention aid flow* > 0.260  
THEN *classification* = *high*

Training data achieved a 95.3% correct classification while validation data came to 87.2%

The supplemental nature of neural nets and decision trees allows a high classification rate with simultaneous explanation of the outcome.

## Generator Diagnosis using BBN

**Problem.** Complex industrial systems such as utility turbine systems are often monitored to detect incipient faults and to prevent catastrophic failure. Generators are fairly large devices which consist of a few basic components: inlet system, compressor, combustor, turbine, and exhaust system. The goal is to operate the turbine at maximum allowable temperature and to keep emissions within desired range while delivering the maximum power. Generators can deliver power in a wide range and often are bundled in series of generators which are switched on and off according to demand while the process of the individual generator is ideally run at steady state.

Due to a large number of moving parts and large forces and speeds at work, there is always the potential for failure. Typical failures could be shaft cracks, bearing failures, blade failures, nozzle failures, etc. Often times, the individual component cannot be observed directly. For example, it is impossible to visually inspect the turbine shaft in operation. Indirect measurements, however, can give clues to a potential problem. Therefore, reasoning systems have to be built which use these clues to come up with an explanation. In complex systems, such as a generator, there is a large number of causal interconnections which make approaches like rule based expert systems hard to set up and to manage. What aggravates the problem is that the exact magnitude of the relationship is often times not completely understood. A link could be somewhere between strong or weak. Finally, there can be different types of links which could denote either the flow of information or an actual physical influence.

**Solution.** A modeling tool which addresses the needs outlined above is the Bayesian Belief Net (BBN). BBNs have been successfully used for generators (Morjaria, 1996, Longtin et al., 1994). For the generators considered, complex interdependencies were initially modeled using causal nets. Later, the network was amended with probabilistic information. Cause-and-effect relations such as shown in Fig. 7 can be set up to model particular behavior. In the example shown a blade fault may cause both an increase in boiler temperature and vibration. However, as indicated before, one cannot directly observe the blade fault. Therefore, it is necessary to use the measurable units to perform diagnostic reasoning. The

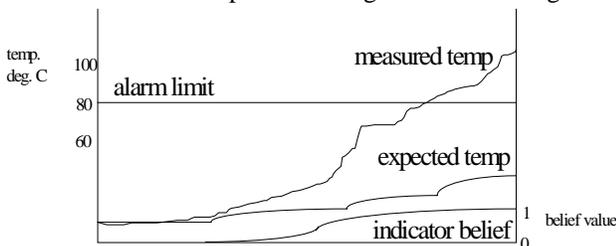


Fig. 6: Stator parallel temperature (Longtin et al., 1994)

problem is aggravated by the fact that, say, a shaft fault may also cause an increase in vibration and that the pressure is medium. A decision has to be made whether a blade fault or a shaft fault is present. Belief values assigned to system states increase and indicate an abnormal condition before hard alarm thresholds are surpassed (Fig. 6)

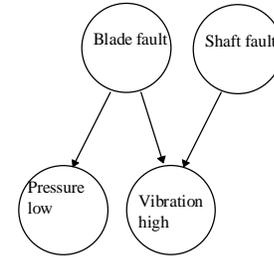


Fig. 7: Example BBN used for Generators

BBNs use probabilistic information to support this reasoning. Both prior probabilities – probabilities in the absence of any additional information – and conditional probabilities – explaining relationships between pairs of nodes – can be used via some mathematical manipulations to reverse the arcs and reason about the fault given symptoms vibration and pressure. Arc reversal is the diagnostician's problem who is only exposed to the symptoms and must find their cause.

A network with several thousand nodes creates problems of its own in that the solution is NP-hard. Furthermore, finding the conditional probabilities is not always straight forward. However, it is desirable to find good approximate solutions rather than insisting on exact solutions (Morjaria, 1996) and even a network set up with gross simplifications where probabilities are assigned values based on heuristics (high/low) appears to perform in a reasonable manner. Some design rules have to be observed when setting up a BBN. For example, no cycles may be introduced into the network. An extension to the BBN includes the introduction of decision nodes which allow the incorporation of potential action to be taken based on a current situation. The decision node would then perform a local optimization based on the data at hand. A BBN with decision nodes is called an Influence Diagram.

Today, monitoring BBNs have been deployed to many generators around the world (Morjaria, 1996) and are observed remotely to deliver diagnostic information in a timely and cost efficient fashion. Remote operation can reduce the cost of on-site availability of computational power and expertise. Remote operation does create some problems which range from bandwidth limitations to the question of what to do with all the data. Besides generators, BBNs have also been successfully used in several other large scale systems such as locomotives (Morjaria et al., 1998).

## Adaptive Classification for Gas Turbines Anomalies

**Problem.** Gas Turbines are used in many applications. They can range from stationary power plants to the use in airplanes or helicopters. Depending on the particular use, the design and size of the gas turbine will vary. On a coarse level, however, gas turbines use the same operating principles. It is also desirable to be able to detect incipient failures to recognize faults before they cause costly secondary damage or result in equally undesired shutdowns. Service providers have long tracked the behavior of the engine by measuring many system parameters and by using trend analysis to detect changes which might be indicative of developing failures (Doel, 1990). Challenges of these schemes are that faults may not be recognized distinctively due to large amounts of noise as well as changing operating conditions which constantly move the estimate for “normal” operation. The former is caused in part by changing environmental conditions for which corrections with first principle models or regression models work only to some extent. The latter is due to changes of schedules, maintenance, etc. which is not necessarily known to the analyst.

**Solution.** The solution used evaluates data in multi-variate feature space, uses fuzzy clusters, a digital filter, and allows the cluster to adapt to changing environments. The evaluation in multi-variate feature space helps distinguishing some faults because system variables are interconnected and changes in one variable may result in changes of other variables as well. For example, a bleed problem, where air leaks from the compressor, will result in a less efficient engine. However, in case of an aircraft engine, the controller demands a certain thrust level and will cause more fuel to be injected into the combustors. This in turn will raise both the turbine speed as well as the exhaust gas temperature. Any particular change may be too small to be picked up alone. However, in a three-dimensional space, the change is more pronounced and can be detected more easily. However, there is still the potential for high misclassification and a crisp classifier will not work very well. The approach chosen uses fuzzy clusters which have a lower degree of membership in the region of overlap, but the classification error becomes smaller. Fig. 8 shows fuzzy clusters in the exhaust gas vs. turbine speed space.

If in addition a digital filter is added to the scheme, a small lag is introduced, but the misclassification is further reduced because noise in the overlapping region is decreased as well. The adaptive properties of the clusters allow the centroids to move and their shape to change. To achieve this goal, the centroid was superimposed with an exponential filter with small parameter which forces the centroid to be updated according to the current data. Data was used for updating only when it was classified as part

of a particular cluster. That is, data of type “normal” was used to update the cluster “normal” but not the cluster “bleed valve fault” and vice versa.

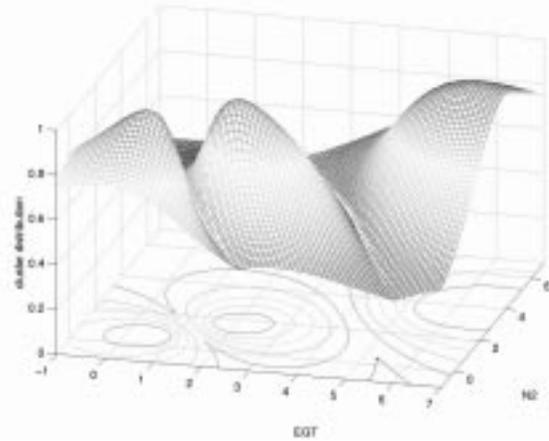


Fig. 8: Fuzzy clusters in multi-dimensional space

**Results.** The system was tested with historical gas turbine data. It showed superior behavior compared to traditional trending approaches which was seen in a low false negative rate (none observed with the data available) and a much reduced false positive rate which improved by several orders of magnitude. Fig. 9 shows the adaptation of a cluster during operation. The “x” denote the location of the centroid of cluster “normal” and the squares denote the location of a fault cluster. During a software change, the normal region changes considerably in one dimension as seen by the vertical jump of the operating point in the graph. Because there were no changes in the other two dimensions, the algorithm adapts correctly to the new location while retaining the ability to detect faults.

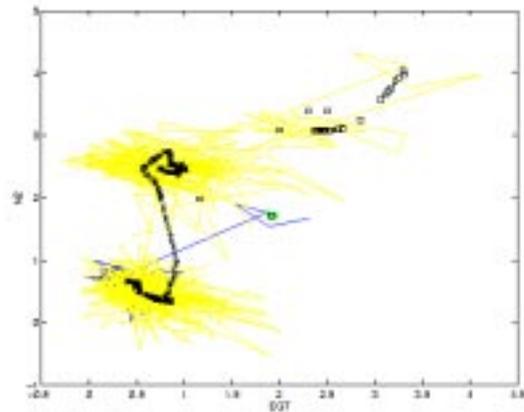


Fig. 9: Adaptation of clusters during operation

## Fuzzy Classification of MRI Signatures For Incipient Failure Detection

**Problem.** A magnetic resonance (MR) machine is a medical device which operates by taking a multitude of measurements of an object to create images of that object. Sensors used are arranged in a circular fashion and the object is moved through the ring of sensors which results in a series of sliced images. An image is created by assembling the different sensor readings. If a component of the MR machine fails, the fault may show up as deterioration of the image such as blurs or streaks. To evaluate MR status, one can perform system tests whereby a phantom – for example a plastic head – is inserted and the image and other data sets evaluated against a benchmark image and data sets. It is then crucial to discern from the data which component of the machine has failed. However, there is no straightforward one-to-one match between data and faults.

A diagnostic system must be able to correctly classify faults and to reject false positives, thus exhibiting robust behavior in noisy environments. From an engineering perspective, the user is interested in an approach which can be understood and which is transparent. This will also help to incorporate newly found fault patterns and other relationships. These new relationships should ideally be recognized even if they are outside the training set. The diagnostic system should be adaptive enough to identify unknown faults and label them as such.

**Solution** To extract and analyze information from signatures of system tests and to identify any potentially present faults a system was designed which consists of a feature extraction module using features in time, frequency and wavelet spaces, several thresholding filters, a classification module using rule-based reasoning, and a database relating the fault type to possible recommended fixes (Fig. 10).

During feature extraction, data is used from particular slice readouts which contain information from the sensors but also about hardware, software version, site information, etc. Statistical, frequency, and wavelet features such as mean, standard deviation, slope, power, time of maxima, spike and outlier information etc. is obtained and stored. As a first processing step, a thresholding filter categorizes data into normal and abnormal based on a few critical features. A fuzzy rule-based system determines particular fault types by establishing the degree of belonging to a fuzzy partition and the associated fault. In case of ambiguous results, for example when several faults are indicated, the most likely one is established using closeness measures to prototype faults. The closeness measures are then aggregated to form a confidence value. Finally, a recommended

correction of the fault situation is presented by mapping the fault to the most likely cause and by linking the outcome to electronic documents concerned with the repair of the faulty condition.

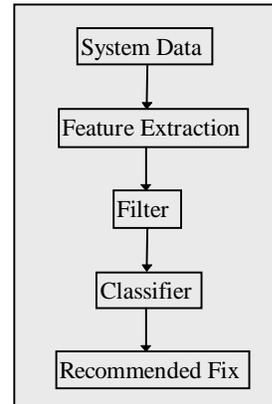


Fig. 10: System Architecture for MRI Fault Detection

**Results.** Cases (90 plots each for 4 cases per fault) were equally divided into training and validation cases. Features were clustered using fuzzy partitions into “high”, “medium” or “low”. Fig. 11 (Ramani, 1999) shows the feature space with faults “Shim”, “Gradient”, RF Receive”, and RF Transmit”. Each data point is assigned based on closeness to prototype faults. Cases are evaluated based on their closeness (Euclidean) to the prototype cases and assigned a confidence value based on the closeness. Results were visualized by showing the individual features (Fig. 12) and a “typical” feature representing the fault. The system was implemented with automatic email feedback which allowed to track the use and also to evaluate the success rate of the tool. Based on that feedback, the accuracy is about 98%

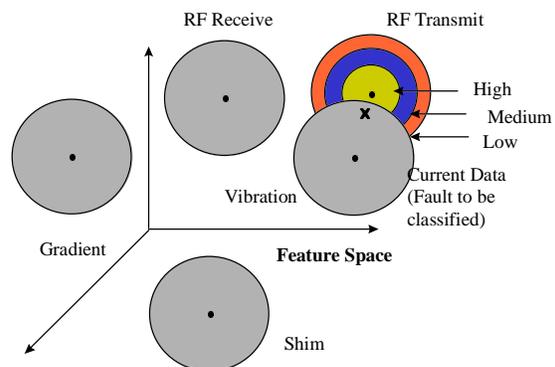


Fig. 11: Fuzzy Partitioning of Feature Space (Ramani, 1999)

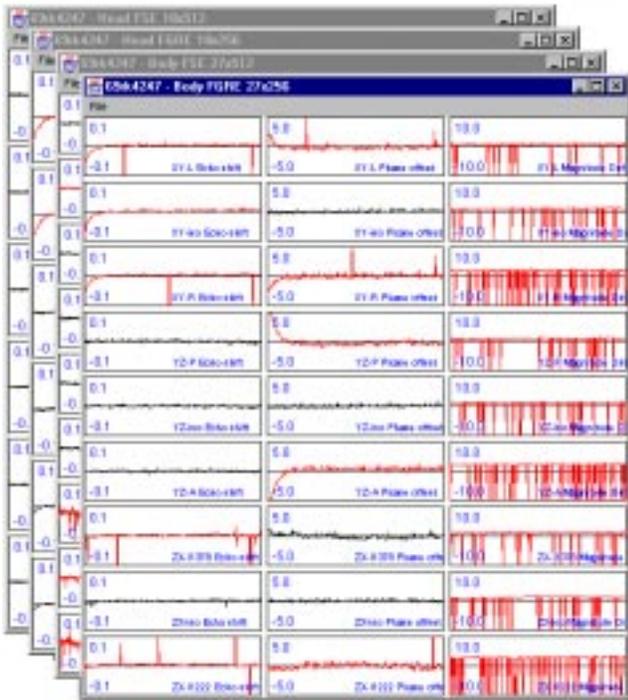


Fig. 12: Visualization of Output (Ramani, 1999)

## References

- Arabshahi, P., Choi, J.J., Marks, R.J., and Caudell, T.P. (1992). Fuzzy Control of Backpropagation. *First IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'92)*, pages 967-972, San Diego, CA., USA.
- Bayes, T. 1763. An essay towards solving a problem in the doctrine of chances, *Philosophical Transactions of the Royal Society of London*, 53:370-418. Facsimile reproduction with commentary by E.C. Molina in Facsimiles of Two Papers by Bayes E. Deming, Washington, D.C., 1940, New York, 1963. Also reprinted with commentary by G.A. Barnard in *Biometrika*, 25, 293-215, 1970.
- Back, T., Fogel, D. & Michalewicz, Z. 1997. *Handbook of Evolutionary Computation*, Institute of Physics, Bristol, UK and Oxford University Press, NY.
- Bersini, H., Bontempi, G., & Decaestecker, C. 1995. Comparing RBF and fuzzy inference systems on theoretical and practical basis. *Conference Proceedings of International Conference on Artificial Neural Networks. ICANN '95, Paris, France*, vol.1., pp. 169-74.
- Bonissone, P. 1997. Soft Computing: the Convergence of Emerging Reasoning Technologies', *Soft Computing A Fusion of Foundations, Methodologies and Applications*, Springer-Verlag, Vol. 1, no. 1, pages 6-18..
- Bonissone, P., Badami, V., Chiang, K. H., Khedkar, P. S., Marcelle, K., Schutten, M. J. 1995. Industrial Applications of Fuzzy Logic at General Electric', *Proceedings of the IEEE*, Vol. 83, no. 3, pages 450-465.
- Bonissone, P., Khedkar, P., and Chen, Y. 1996. Genetic Algorithms for automated tuning of fuzzy controllers, A transportation Application, *Fifth IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'96)*, pages 674-680, New Orleans, LA., USA.
- Bouchon-Meunier, B., Yager, R., and Zadeh, L. 1995. *Fuzzy Logic and Soft Computing*. World Scientific, Singapore.
- Burkhardt, D. and Bonissone, P. 1992. Automated Fuzzy Knowledge Base Generation and Tuning, *First IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'92)*, pp. 179-188, San Diego, CA., USA.
- Chen, Y.T. & Bonissone, P. 1998. Industrial Applications of Neural Networks at General Electric, Technical Information Series, 98CRD79, General Electric CRD, October 98.
- Cordon, O., Herrera, H. and Lozano, M. 1995. A classified review on the combination fuzzy logic-genetic algorithms bibliography. Tech. Report 95129, URL: <http://decsai.ugr.s/~herrera/flga.html>, Department of Computer Science and AI, Universidad de Granada, Granada, Spain.
- Dempster, A. P. 1967. Upper and lower probabilities induced by a multivalued mapping, *Annals of Mathematical Statistics*, 38:325-339.
- Doel, D. 1990. The Role for Expert Systems in Commercial Gas Turbine Engine Monitoring, *Proceedings of the Gas Turbine and Aeroengine Congress and Exposition*, Brussels, Belgium.
- Dubois, D & Prade, H. 1998. Soft computing, fuzzy logic, and Artificial Intelligence, *Soft Computing A Fusion of Foundations, Methodologies and Applications*, Springer-Verlag, Vol. 2, no. 1, pages 7-11.
- Fiesler, E. & Beale, R. 1997. *Handbook of Neural Computation*, Institute of Physics, Bristol, UK and Oxford University Press, NY.
- Forbus, K. 1981. Qualitative Reasoning about Physical Processes, *Proceedings Seventh International Joint Conference on Artificial Intelligence*, Karlsruhe, Germany.

- Herrera & Lozano 1996. Adaptive Genetic Algorithms Based on Fuzzy Techniques. *Proc. of IPMU'96*, pages 775-780, Granada, Spain.
- Holland, J.H 1975 *Adaptation in Natural and Artificial Systems*, MIT Press, Cambridge, MA
- Jacobs, R.A. 1988. Increased rates of convergence through learning rate adaptation. *Neural Networks*, 1:295-307.
- Jang, J.S.R. 1993. ANFIS: Adaptive-network-based-fuzzy-inference-system. *IEEE Transactions on Systems, Man, and Cybernetics*, 23:665-685.
- Karr C.L., 1991. Design of an adaptive fuzzy logic controller using genetic algorithms, *Proc. Int. Conf. on Genetic Algorithms (ICGA'91)*, pages 450-456, San Diego, CA., USA.
- Khedkar, P.S. & Keshav, S. 1992. *Fuzzy Prediction of Timeseries*. Proceedings of the 1<sup>st</sup> IEEE International Conference on Fuzzy Systems.
- Kinzel J., Klawoon, F., and Kruse, R.. 1994. Modifications of genetic algorithms for designing and optimizing fuzzy controllers, *Proc. First IEEE Conf. on Evolutionary Computing (ICEC'94)*, pages 28-33, Orlando, FL., USA
- Kitano, H. 1990. Empirical studies on the speed of convergence of neural network training using genetic algorithms. *AAAI-90 Proceedings. Eighth National Conference on Artificial Intelligence*, vol.2., pp. 789-95.
- Kuipers, B. 1985. Commonsense Reasoning about Causality: Deriving Behavior from Structure, *Qualitative Reasoning about Physical Systems*, pp 169-203, ed. D. Bobrow, MIT Press, Cambridge, MA.
- Lee, M.A. and Tagaki, H. 1993. Dynamic control of genetic algorithm using fuzzy logic techniques. In Forrest, S. , ed., *Proceedings of the fifth International Conference on Genetic Algorithms*, pp. 76-83. Morgan Kaufmann.
- Longtin, K., Morjaria, M., Rink, F., Stanton, D., Klemperer, G., Burns, C., Stein, J. 1994. Generator Expert Monitoring System as an Enabler for a Condition based Maintenance Program for Utility Generators, *Proceedings of the American Power Conference*, pp. 1078-1084.
- Mamdani E.H and Assilian, S. 1975. An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller, *Int. J. Man Machine Studies*, 7(1): 1-13.
- McInerney M., Dhawan, A.P. 1993. Use of genetic algorithms with backpropagation in training of feedforward neural networks. *Proceedings of 1993 IEEE International Conference on Neural Networks (ICNN '93)*, San Francisco, CA, vol.1, pp. 203-8.
- Montana D.J. & Davis, L. 1989. Training feedforward neural networks using genetic algorithms. *IJCAI-89 Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, vol.1, pp. 762-7.
- Morjaria, M. & Santosa, F. 1996. Monitoring Complex Systems with Causal Networks. *IEEE Computational Sciences and Engineering*, Qinter 1996, vol.3, (no.4), pp.9-10.
- Morjaria, M., Azzaro, S., Bush, J., Nash, J., Smith, M., Smith, W. 1998. System and Method for Isolating Failures in a Locomotive. *US Patent 5,845,272*.
- Ramani, V. 1999. Unsupervised Rule Generation for Maintenance of a Diagnostic System. *Working Notes of the 1999 Spring Symposium "AI in Equipment Service Maintenance and Support"*, Palo Alto.
- Rosenblatt, F. 1959. Two theorems of statistical separability in the perceptron, *Mechanization of Thought Processes*, pages 421-456, Symposium held at the National Physical Laboratory, HM Stationary Office, London.
- Ruspini, E., Bonissone, P. & Pedycz, W. 1998. *Handbook of Fuzzy Computation*, Institute of Physics, Bristol, UK.
- Shafer, G. 1976. *A Mathematical Theory of Evidence*. Princeton University Press, Princeton, NJ.
- Takagi, T. and Sugeno, M. 1985. Fuzzy Identification of Systems and Its Applications to Modeling and Control, *IEEE Transactions on Systems, Man, and Cybernetics*, 15:116-132.
- Vonk, E. Jain, L.C and Johnson, R.P. 1997. *Automatic Generation of Neural Network Architecture Using Evolutionary Computation*. World Scientific Publ. Co., Singapore.
- Wasserman P.D. 1989. *Neural Computing: Theory and Practice*. Van Nostrand Reinhold.
- Werbos, P. 1974. *Beyond Regression: New Tools for Predictions and Analysis in the Behavioral Science*. PhD thesis, Harvard University, Cambridge, MA.

Widrow, B. and Hoff, M.E. 1960. Adaptive switching circuits, *IRE Western Electric Show and Convention Record*, Part 4, pages 96-104.

Yabe, K., Koda, J., Yoshida, K., Chiang, K.H., Khedkar, P., Leonard, D. & Miller, N.W. 1995. Conceptual Designs of AI-Based Systems for Local Prediction of Voltage Collapse. *IEEE Transactions on Power Systems*, Feb. 1996, vol.11, (no.1):137-45.

Zadeh, L. A. 1965. Fuzzy sets. *Information and Control*, 8:338-353.

Zadeh, L. A. 1994. Fuzzy Logic and Soft Computing: Issues, Contentions and Perspectives, in *IIZUKA'94: 3rd International Conference on Fuzzy Logic, Neural Nets and Soft Computing*, pages 1-2, Iizuka, Japan.

Zadeh, L. A. 1998. Some reflection on soft computing, granular computing and their roles in the conception, design and utilization of information/intelligent systems, *Soft Computing A Fusion of Foundations, Methodologies and Applications*, Springer-Verlag, Vol. 2, no. 1, pages 23-25.

Zheng L., 1992. A Practical Guide to Tune Proportional and Integral (PI) Like Fuzzy Controllers. *First IEEE International Conference on Fuzzy Systems, (FUZZ-IEEE'92)*, pages 633-640, S.Diego, CA.,USA.