Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2001), *15*, 267–279. Printed in the USA.
Copyright © 2001 Cambridge University Press 0890-0604/01 $12.50

Soft Computing for diagnostics in equipment service

PIERO BONISSONE AND KAI GOEBEL
GE Corporate Research & Development, Information Systems Lab, Niskayuna, NY 12309, USA

(RECEIVED October 27, 2000; ACCEPTED December 12, 2000)

Abstract

We present methods and tools from the Soft Computing (SC) domain, which is used within the diagnostics and prognostics framework to accommodate imprecision of real systems. SC is an association of computing methodologies that includes as its principal members fuzzy, neural, evolutionary, and probabilistic computing. These methodologies enable us to deal with imprecise, uncertain data and incomplete domain knowledge typically encountered in real-world applications. We outline the advantages and disadvantages of these methodologies and show how they can be combined to create synergistic hybrid SC systems. We conclude the paper with a description of successful SC case study applications to equipment diagnostics.

Keywords: Diagnosis; Diagnostics; Hybrid Systems; Soft Computing

1. INTRODUCTION

In the industrial world, we encounter a wide range of modeling problems that require the analysis of uncertain and imprecise information. Usually, an incomplete understanding of the problem domain further compounds this modeling problem. For example, to support the service of equipment, we need to generate models that can analyze the equipment data, interpreting their past behavior and predicting their future behavior. These problems pose a challenge to traditional modeling techniques and represent a great opportunity for the application of soft computing methodologies.

In an effort to yield higher margins, many manufacturing companies are shifting their operation to the service field. Therefore, diagnostics and prognostics play a significant role in this paradigm shift. A typical example of this service focus is the use of long-term service agreement (LTSA) contracts with guaranteed uptime. These contracts strongly motivate the service provider to keep equipment in working order as opposed to performing a maintenance action once a failure has occurred. As a consequence, service should be optimized to prevent failures and to maximize uptime while avoiding superfluous maintenance. Some of these objectives can be accomplished by using tools that measure the system state and indicate incipient failures. Such tools must have a high level of sophistication and must be able to incorporate monitoring, fault detection, decision making about possible preventive or corrective action, and execution monitoring. Because of the complexity of the task, AI and in particular Soft Computing have been leveraged in the implementation of these tools. Some applications of Soft Computing techniques in support of service tasks, such as anomaly detection and identification, diagnostics, prognostics, estimation and control, have been reported in Bonissone et al. (1995), Chen and Bonissone (1998), and Bonissone et al. (1999*a*). In this paper, we will briefly describe the components of Soft Computing and illustrate some of their most successful applications to equipment service.

Reprint requests to: Dr. Kai Goebel, GE CR&D, Information Systems Lab, K1-5C4A, One Research Circle, Niskayuna, NY 12309, USA. E-mail: goebelk@crd.ge.com

1.1. Soft computing

The term *Soft Computing* (SC), a subfield of Artificial Intelligence, was originally coined by Zadeh (1994) as an association of computing methodologies that "... exploit the tolerance for imprecision, uncertainty, and partial truth to achieve tractability, robustness, low solution cost, and better rapport with reality. (p. 1)" According to Zadeh (1998), Soft Computing "includes as its principal members fuzzy logics (FL), neuro-computing (NC), evolutionary computing (EC) and probabilistic computing (PC). (p. 23)" It should be noted, however, that we have not reached a consensus yet as to the exact scope or definition of SC (see, e.g., Dubois & Prade, 1998).

A reason for SC's success is the *synergy* derived from its components. SC's main characteristics are its intrinsic capability to create *hybrid systems* that are based on a loose (or tight) integration of these technologies. This integration provides complementary reasoning and searching methods that allow the combination of domain knowledge and empirical data to develop flexible computing tools and solve complex problems. Figure 1 portrays a categorization of these SC algorithms and their hybrid combinations. Extensive coverage of this topic can be found in Bouchon-Meunier et al. (1995), Bonissone (1997), and Bonissone et al. (1999*a*).

1.2. SC components and taxonomy

Fuzzy logic (FL), introduced by Zadeh (1965), gives us a language, with syntax and local semantics, in which we can translate 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 Reasoning (PR) such as Bayesian Belief Networks (BBN), based on the original work of Bayes (1763) and refined by Pearl (1982), 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, can be traced back to the work of McCulloch and Pitts (1943), who showed that a network of binary decision units (BDNs) could implement any logical function. Neural networks were further explored by Rosenblatt (1959) and Widrow and Hoff (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 algorithm that performs fine-granule local optimization. A comprehensive current review of neuro-computing can be found in Fiesler and Beale (1997).

Evolutionary Computing covers many important families of stochastic algorithms, including *evolutionary strategies* (ES), proposed by Rechenberg (1965) and Schwefel (1965), *evolutionary programming* (EP), introduced by Fogel (1962), and *genetic algorithms* (GAs), based on the work of Fraser (1957) and Holland (1962, 1975), which contain as a subset *genetic programming* (GP), introduced

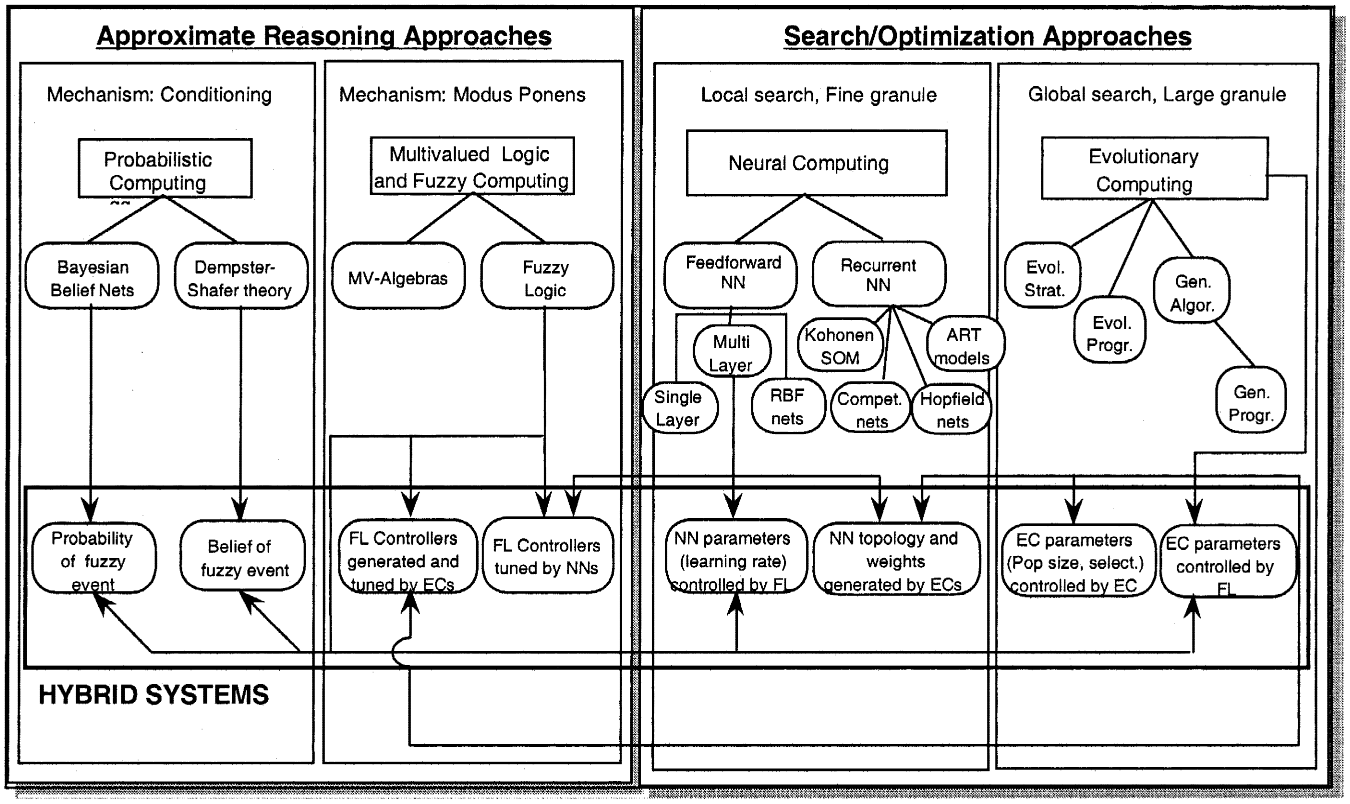
Approximate Reasoning Approaches
Search/Optimization Approaches
Mechanism: Conditioning
Mechanism: Modus Ponens
Local search, Fine granule
Global search, Large granule
Probabilistic Computing
Multivalued Logic and Fuzzy Computing
Neural Computing
Evolutionary Computing
Bayesian Belief Nets
Dempster-Shafer theory
MV-Algebras
Fuzzy Logic
Feedforward NN
Recurrent NN
Evol. Strat.
Evol. Progr.
Gen. Algor.
Gen. Progr.
Multi Layer
Single Layer
RBF nets
Kohonen SOM
ART models
Compet. nets
Hopfield nets
Probability of fuzzy event
Belief of fuzzy event
FL Controllers generated and tuned by ECs
FL Controllers tuned by NNs
NN parameters (learning rate) controlled by FL
NN topology and weights generated by ECs
EC parameters (Pop size, select.) controlled by EC
EC parameters controlled by FL
HYBRID SYSTEMS

Fig. 1. Soft Computing components and hybrid systems.

by Koza (1992). Genetic Algorithms (GA), a major component of evolutionary computing, give us a method to perform a randomized global search in a solution space. In such a 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 nonexistent. Our solution must be generated by leveraging two kinds 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. These two kinds of resources determine the two main approaches found in Soft Computing: *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 more than one field, as we will see in a subsequent section.

1.3. 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 is exemplified by first-principle derived models (based on differential or difference equations), by first-principle qualitative models (based on symbolic, qualitative calculi—see Forbus (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 is represented by 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.

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), and so forth. Some techniques, such as feature extraction, filtering and validation gates, imputation models, and virtual sensors (which model the recorded data as a function of others variables) have been developed to address these problems.

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 estimation. 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 search for the best parameters, and to iterate in this process. For example, the knowledge base (KB) in a Mamdani type fuzzy system (Mamdani & Assilian, 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 & Sugeno, 1985) increases 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 that is equivalent to Radial Basis Function (Bersini et al., 1995). The same model can be translated into a structured network, such as Adaptive Neural Fuzzy Inference Systems (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 tuning or to postulate new structures. An extensive coverage of these approaches can be found in Bonissone (1997).

2. HYBRID ALGORITHMS: THE SYMBIOSIS

Over the past few years we have seen an increasing number of hybrid algorithms, in which two or more 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 such combinations, as depicted in the lower box of Figure 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.

2.1. FL tuned by NN

Within the limited scope of using NNs to tune FL controllers, we want to mention the seminal work on ANFIS, proposed by Jang (1993). ANFIS is a representative hybrid system in which NNs are used to tune a FLC. ANFIS consists of a six-layer 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 generates 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.

2.2. 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 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 a 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 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. This shortcoming was addressed by using a genetic algorithm to tune the model's scaling factors and membership functions.

2.3. 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. This problem has been partially resolved by using variable weight granularity or by switching from direct binary encoding to parametric or grammar encoding (see Bonissone et al., 1999*b*, for a more complete review).

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 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 and Dhawan (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) and Yao (1999).

2.4. 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 trade-off: 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).

2.5. GAs controlled by FL

The use of Fuzzy Logic to translate and improve heuristic rules has also been applied to manage GA's 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 et al., 1995, and Lee and Tagaki, 1993).

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

3. DIAGNOSTICS 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, and so forth.

The outcome of diagnosis and prognosis drives planning and execution. Possible planning includes planning of corrective action that can be either reactive or proactive. Corrective action includes reconfiguration of the current system or subsystem, 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 (besides imprecise data and incomplete understanding of the problem domain) are the ability to adjust 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 that can reason about the information gathered and come to an optimal judgment within the constraints of the system.

The following section shows some case studies of SC applications to diagnostics and maintenance. In particular, we show the use of neuro-fuzzy solutions for voltage breakdown prediction, a joint neural network and decision tree solution for paper mill web breakage prediction, automated tuning of fuzzy controller via genetic algorithms, generator diagnostics using Bayesian Belief Nets, and adaptive fuzzy classification for aircraft engine diagnostics.

4. CASE STUDIES OF SC APPLICATIONS IN DIAGNOSTICS AND MAINTENANCE

4.1. Distributed ANFIS models to predict voltage breakdown in power distribution network

4.1.1. Problem

Power systems consist typically of a complex web of interconnected components such as bulk transmission and local power generation. Power blackouts are often times started by the spread of initially localized events. These events are caused by voltage instabilities, which then extend to voltage collapses (Yabe et al., 1995). For example, problems 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 and customer behavior. Initial voltage collapse symptoms are 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.

4.1.2. Solution

Most techniques for voltage stability calculation compute 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.

To start, each surface was represented via ANFIS (Yabe et al., 1995). Training was performed using a range of power

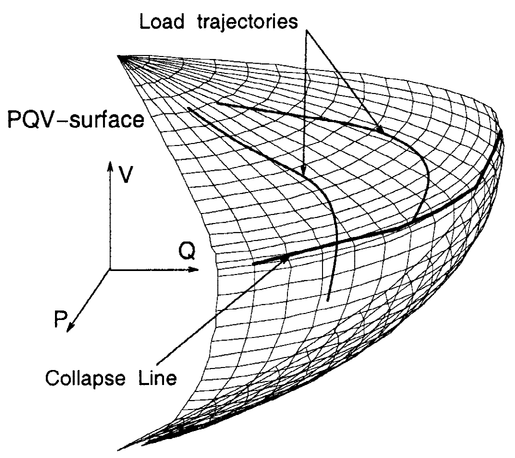

Fig. 2. Load-voltage surface with trajectories and collapse line.

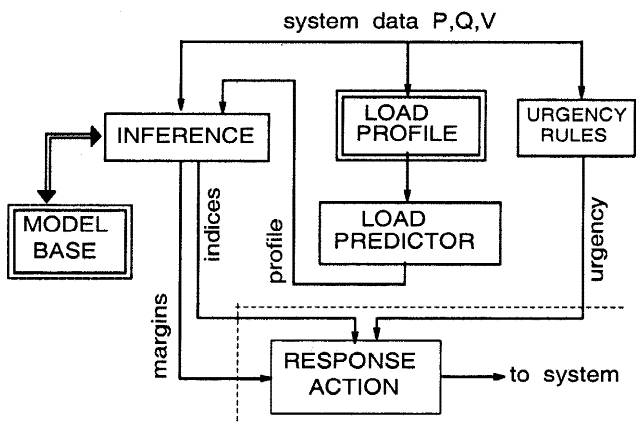

Fig. 4. Architecture of the system.

system configurations and recording local observations while the system was driven to collapse. An inference module was used to infer 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, and constant power factor (PF) as displayed in Figure 3 in the reactive power versus active power space, which describes the relationship between current point and collapse point.

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

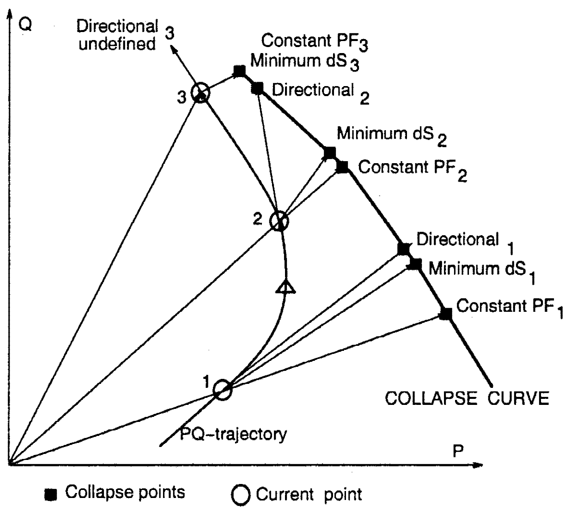

Fig. 3. Collapse points.

4.1.3. *Results*

The system was tested on a variety of different settings ranging from small to large systems. 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 and the ability to interpolate between rules. Figure 4 shows the interaction of the different modules employed in this scheme where a small, two-power-plants system test was modeled each with a generator, an exciter, a power system stabilizer, a turbine, and a boiler. The network configuration was used to test the predictive capability of the system, which performed very well. A different setup with a different amount of shunt compensation was used as 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, and so forth.

4.2. Prediction of paper web breakage in paper mill using neural nets and induction trees

4.2.1. *Problem*

Paper mills have enormous size, oftentimes 100 meters in length or more. Making paper involves running miles of wet and dry webs over and between large cylinders that convert pulp into the final product. The speed of the web can reach 60 mph. Under those conditions, it is not surprising that the web breaks about once a day. Breakage results in a machinery standstill of about 90 minutes, which in turn causes plant-wide revenue loss of several million dollars per year. It is therefore desirable to prevent breakage by predicting imminent breakage and being able to take preventive action.

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

4.2.2. Solution

A number of approaches were considered, such as fuzzy causal modeling, principal component analysis, and learning vector quantization. After initial down select, two approaches were chosen because of their complementary nature: 1) neural nets with fuzzy accelerators and 2) 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.

4.2.3. Results

The neural net was trained via back-propagation algorithm using normalized input sensor data. Output was a quantized measure before breakage. This measure was designed such that the last 10 readings before breakage attained a value of 0.9, the next to the last 20 readings a value of 0.5 and the 10 readings before that a value of 0.1. The interpretation could be little, medium, and long time to breakage. Training was carried out using fuzzy accelerators that adaptively adjust momentum, learning rate, and steepness parameters after each iteration. These accelerators react to the rate at which learning commences. For example, if the error derivative was the same over several learning epochs and the decrease of the error was small, the learning rate was increased. The other parameters were adjusted in a similar fashion whereby the condition was evaluated using fuzzy reasoning. During training, a correct classification of 99.1% was achieved for the neural network while the validation set achieved 86.8% correct classification.

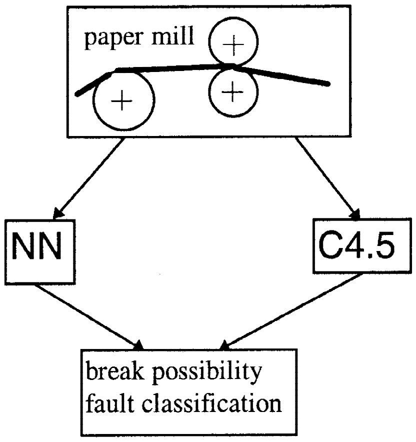

Fig. 5. Hybrid system for breakage detection.

For the induction tree, C4.5 was used. It was trained and tested with the same data as the neural net. Classes were assigned the linguistic labels "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 was:

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

This method achieved a 95.3% and 87.2% correct classification for training and testing, respectively.

The supplemental nature of neural nets and decision trees allows a high classification rate with simultaneous explanation of the outcome by aggregating the result. Further details on this application can be found in Bonissone et al. (1999*b*).

4.3. Method for automated tuning of a raw mix proportioning controller in cement plants

4.3.1. Problem

A cement plant basically grinds limestone, sandstone, and sweetener in a mill, and processes the resulting powder in a kiln. To ensure the correct mix and proportions of chemical elements for making cement, it is critical to control the Raw Mix Proportioning (RMP) by keeping the output of the raw mill to be close to specified set points of physical properties (i.e., Lime Saturation Factor (LSF), Alumina Modulus (ALM), and Silica Modulus (SIM)). These properties are all functions of the fractions of four metallic oxides present in the material and can be picked up by sensors (IMACON and Quarcon) with a time delay of a few minutes. The control objective is then to regulate the values of LSF, ALM, and SIM, as they appear after the raw mill, so that they are close to the set points and fluctuate as little as possible. This can be done by continually changing the proportion in which the three raw materials are mixed before they enter the raw mill.

4.3.2. Solution

The proposed controller employs two fuzzy logic supervisory controls to minimize set-point tracking error, while providing a smooth control action and insuring that there is no sudden control change that imposes shock to the system. The fuzzy logic supervisory controller (FLSC) uses the tracking error and its change-in-error to recommend a change in the control output that modifies the current control settings. Specifically, the FLSC consists of three pairs of fuzzy proportional integral (FPI) controllers. In the method described here, the FPI is tuned off-line using genetic algorithms to modify FPI's most sensitive parameters: scaling factors (SF) and membership functions (MF).

The overall scheme is shown in Figure 6. It consists of FLSC closing the loop around the plant simulation (PSIM), using only the current set points as its state input. The actual set points are compared with the desired set points to generate an error as input to the FLSC, which in turn outputs control actions back to the simulator. Off-line, a GA uses the setup for evaluating various FPI parameters.

The design of the control algorithm requires a hierarchical system for handling multiple, possibly conflicting goals. The latter arise because the three properties (LSF, SIM, and ALM) have to be controlled simultaneously, and each may require a different proportion to be accurately controlled. The hierarchical controller module is illustrated in Figure 7, where the low-level controllers control the output to minimize the error in each of the three set points of LSF, SIM, and ALM, while the supervisory controller determines the best compromise among these control actions. This hierarchical control scheme permits the decomposition of complex problems into a series of smaller and simpler ones. As these simpler problems are solved, typically by using low-level controllers, they are recombined to address the larger problem. The fuzzy logic supervisory controller governs this recombination by performing soft switching between different modes of operation. The soft switching allows more than one mode of operation (with its corresponding controller) to be active at any one time. Through the assignment of a linear combination of low level controllers to a given mode, the engineer can trade off safety and efficiency against performance.

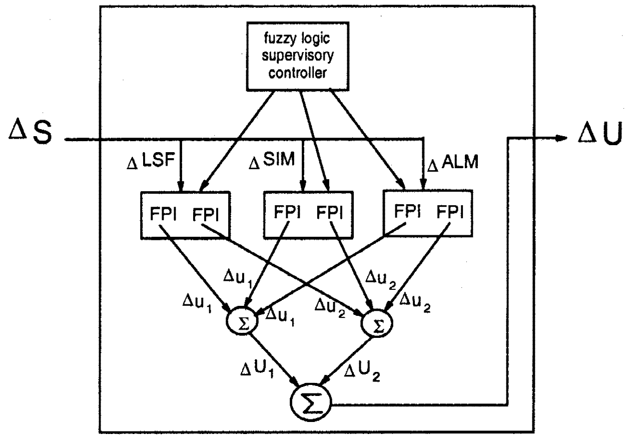

Fig. 7. Fuzzy logic supervisory controller module.

We adopted GAlib, a C++ library of genetic algorithm objects, as the software development tool (developed at MIT to promote the study of genetic algorithms for general purpose optimization). To study the effects of different objectives, we minimized four fitness functions, considering tracking accuracy, throttle jockeying, raw material cost, and combining a weighted sum of the first three. For comparison purposes, 16 tests were conducted by taking a cross-product of the scaling factor values before and after GA tuning, the membership function parameter values before and after GA tuning, and the four fitness functions.

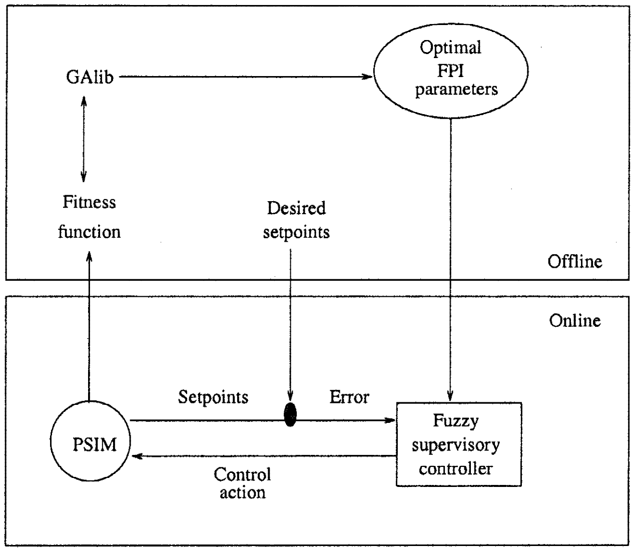

Fig. 6. System schematic for automatic tuning of a fuzzy supervisory controller in cement plants.

4.3.3. Results

The GA parameters were: population size = 50, number of generations = 25, crossover rate = 0.6, and mutation rate = 0.001. All structures in each generation were evaluated and an elitist strategy was used to guarantee monotonic convergence. It was shown that this approach resulted in a controller that was superior to the manually designed one, and with only modest computational effort. This makes it possible to customize automated tuning to a variety of different cement plants. The tests demonstrated that GAs are powerful search methods and are very suitable for automated tuning of FPI controllers. The Genetic Algorithms were able to come up with near-optimal FPI controllers within a reasonable amount of time according to different search criteria. In addition, the tests were also designed to demonstrate that parameters tuning should be performed in the order of their significance. In particular, tuning of scaling factors should come first, since they have global effects on all the control rules in a rule base. Tuning membership functions will only give marginal improvements for an FPI with tuned scaling factors.

The tuning of the supervisory fuzzy controller, in conjunction with a plant simulator, had the objective of verifying that the system could exhibit stable operation for more than a week under severe external disturbances in the oxide content of the raw material as well as under parameter noise. The control system was tested in the Gujarat Ambuja Cement Limited (GACL) plant in Kodinar, India and is now commercially available from IMA. The interested reader can obtain further details about the control system in Bonissone and Chen (2000*a*, 2000*b*, 2000*c*).

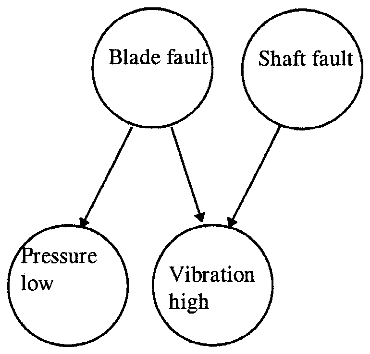

Fig. 8. Example BBN used for generators.

4.4. Generator diagnosis using BBN

4.4.1. 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 that 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 maximum power. Generators can deliver power in a wide range. Often they are bundled in a series of generators that are switched on and off according to demand while the process of the individual generator is ideally run at a steady state.

Due to a large number of moving parts and large forces and speeds at work, there is always the potential for failure such as shaft cracks, bearing failures, blade failures, nozzle failures, and so forth. As in many diagnostic systems, the individual component cannot be observed directly. For example, it is not possible 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 are a large number of causal interconnections between system components, which makes approaches like rule-based expert systems hard to set up and to manage. Thousands of those relations exist and correspondingly many rules. The problem is worsened because the magnitude of the relationship is often times only poorly understood and because different types of relations (denoting flow of information or an actual physical influence) are not easy to distinguish.

4.4.2. Solution

A modeling tool that addresses the needs outlined above is the Bayesian Belief Net. BBNs have been successfully used in diagnostic tasks for generators (Morjaria & Santosa, 1996). For the generator considered, interdependencies were initially modeled using causal nets. Later, the network was amended with probabilistic information. Cause-and-effect relations (such as shown simplified in Fig. 8) 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. Because a shaft fault may also cause an increase in vibration, a decision has to be made whether a blade fault or a shaft fault is present.

BBNs use probabilistic information to solve this problem. Both prior probabilities—probabilities in the absence of any additional information—and conditional probabilities—explaining relationships between pairs of nodes—can be used via mathematical manipulations to reverse the arcs and reason about the fault given symptoms.

The solution of a network with several thousand nodes—as is the case of a generator—is NP-hard. Another issue is finding the conditional probabilities, which is not always straightforward. However, finding good approximate solutions rather than insisting on exact solutions (Morjaria & Santosa, 1996) appears to give acceptable results, and even a network setup with gross simplifications where probabilities are assigned values based on heuristics (high/low) performs oftentimes 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 & Santosa, 1996) and are observed remotely to deliver diagnostic information in a timely and cost-efficient fashion. Besides generators, BBNs have also been successfully used in several other large-scale systems such as locomotives (Morjaria et al., 1998).

4.5. Adaptive classification for gas turbines' anomalies

4.5.1. Problem

Gas turbines are used in many applications. They can range from stationary power plants to 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. To detect incipient failures, service providers have long tracked the behavior of the engine by measuring many system parameters and by using trend analysis to detect changes that 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 that constantly move the estimate for "normal" operation. The former are caused in part by changing environmental conditions for which corrections with first-principle models or regression models work only to some extent. The latter are due to changes of schedules, maintenance, and so on, which are not necessarily known to the analyst.

4.5.2. Solution

The solution used employed fuzzy clusters, a digital filter, and allowed the cluster to adapt to changing environments. Evaluation in multivariate feature space helped distinguish some faults because system variables are correlated. For example, a bleed problem, where air leaks from the compressor, will result in a less efficient engine. However, in the 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 that have a lower degree of membership in the region of overlap. Figure 9 shows fuzzy clusters in the exhaust gas temperature (EGT) versus turbine speed (N2) space. A digital filter was added to the scheme, thus introducing a small lag. However, the misclassification was further reduced because noise in the overlapping region was decreased as well. The adaptive properties of the clusters allowed 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 forced the centroid to be updated according to the current data. Data were used for updating only when they are classified as part of a particular cluster. Specifically, data of type "normal" were used to update the cluster "normal" but not the cluster "bleed valve fault" and vice versa.

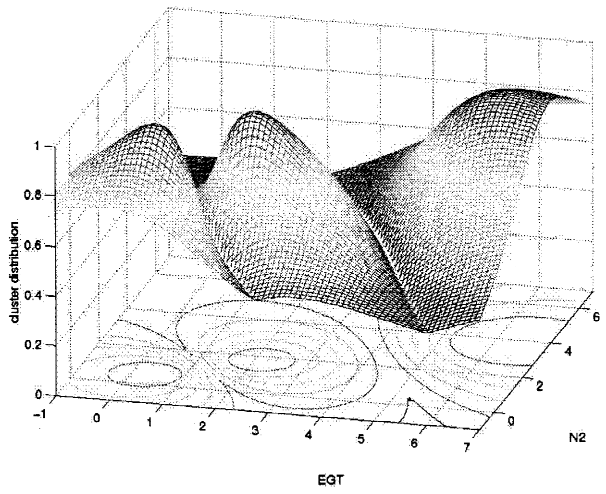

Fig. 9. Fuzzy clusters in multidimensional space.

4.5.3. Results

The system was tested with historical gas turbine data. It showed superior behavior compared to traditional trending approaches, which was manifested 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. Figure 10 shows the adaptation of a cluster during operation. The "x" denotes 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.

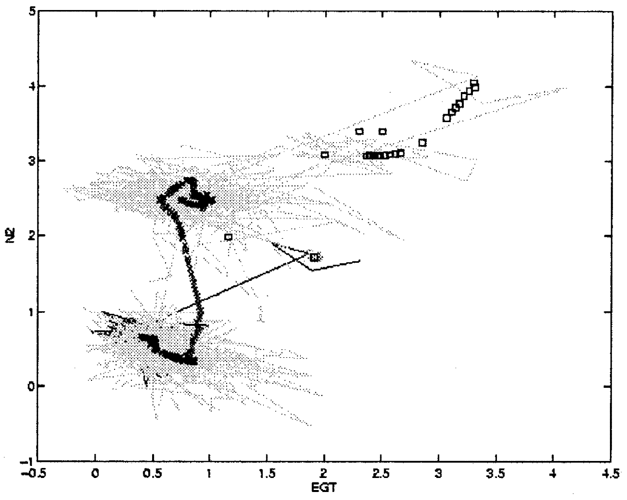

Fig. 10. Adaptation of clusters during operation.

5. FINAL REMARKS

Soft Computing is having an impact on many diagnostic industrial and commercial operations. It provides alternative approaches to traditional knowledge-driven reasoning systems or pure data-driven systems and it overcomes their shortcomings by synthesizing a number of complementary reasoning and searching methods over a large spectrum of problem domains. We have reviewed Soft Computing's main components (Fuzzy Logic, Probabilistic Reasoning, Neural Networks, and Evolutionary Computing) and surveyed some of their successful applications for equipment monitoring, diagnostics, and control, where SC components were used either alone or in a hybrid fashion. These studies stem from real-world, high-impact business problems, such as gas turbine service and diagnosis, voltage breakdown prediction, generator diagnostics, cement kiln controller tuning, and paper web breakage prediction.

The SC components leverage the tolerance for imprecision, uncertainty, and incompleteness, which is intrinsic to the problems to be solved, and generate tractable, low-cost, robust solutions to such problems. The synergy derived from hybrid systems stems from the relative ease with which we can translate problem domain knowledge into initial model structures whose parameters are further tuned by local or global search methods. The payoff for a conjunctive use of techniques is a more accurate and robust solution than a solution derived from the use of any single technique alone. This synergy comes at comparatively little expense because typically the methods do not try to solve the same problem in parallel but they do it in a mutually complementary fashion. In other words, no single technique should be expected to be the best for finding every model structure and tuning all system parameters.

A step in further improving system performance is the exploitation of parallel systems. These systems may be designed to rely to the maximum amount on nonoverlapping data and use different techniques to arrive at their conclusions. In *information fusion*, the outputs of these heterogeneous models will be compared, contrasted, and aggregated.

The future appears to hold much promise for the novel use and combinations of SC applications. The circle of SC's related technologies will probably widen beyond its current constituents. The push for low-cost solutions combined with the need for intelligent tools will result in the deployment of hybrid systems that efficiently integrate reasoning and search techniques.

On the application front, we will likely see a drive towards prognostic and autonomous capabilities. With an increase of service-related operations, it will be increasingly attractive to be able to forecast anomalous trends and conditions, and correct them before their effects are fully developed. In addition, remotely monitored systems will bear the need to operate autonomously, thus requiring intelligent agents to regulate their operations.

In the future, we expect that the combination of Soft Computing with advances in the areas of computer vision, voice recognition, natural language processing, and so on, will further improve and expand our problem-solving capability to a large spectrum of industrial and commercial problems.

REFERENCES

Arabshahi, P., Choi, J.J., Marks, R.J., & Caudell, T.P (1992). Fuzzy control of backpropagation. *First IEEE Int. Conf. Fuzzy Systems (FUZZ-IEEE'92)*, San Diego, CA, pp. 967–972.

Back, T., Fogel, D., & Michalewicz, Z. (1997). *Handbook of evolutionary computation*, Bristol, UK: Institute of Physics, and New York: Oxford University Press.

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* (1970) *25*, pp. 293–215.

Bersini, H., Bontempi, G., & Decaestecker, C. (1995). Comparing RBF and fuzzy inference systems on theoretical and practical basis. *Proc. Int. Conf. Artificial Neural Networks. ICANN '95*, Paris, France, Vol.1, pp. 169–174.

Bonissone, P. (1997). Soft Computing: The convergence of emerging reasoning technologies. *Soft Computing: A Fusion of Foundations, Methodologies and Applications*, *1(1)*, 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*, *83(3)*, 450–465.

Bonissone, P., & Chen, Y.-T. (2000*a*), *System and method for providing raw mix proportioning control in a cement plant with a fuzzy logic supervisory control*. US Patent #6,113,256.

Bonissone, P., & Chen Y.-T. (2000*b*). *System and method for providing raw mix proportioning control in a cement plant with gradient-based predictive controller*. US Patent #6,120,173.

Bonissone, P., & Chen, Y.-T. (2000*c*). *System and method for providing raw mix proportioning control in a cement plant*. US Patent #6,120,172.

Bonissone, P., Chen, Y.-T., Goebel K., & Khedkar, P. (1999*a*). Hybrid soft computing systems: Industrial and commercial applications. *Proceedings of the IEEE 87*(9), 1641–1667.

Bonissone, P., Chen, Y.-T., & Khedkar, P. (1999*b*) *System and method for predicting a web break in a paper machine*, US Patent #5,942,689.

Bonissone, P., Khedkar, P., & Chen, Y.-T. (1996). Genetic algorithms for automated tuning of fuzzy controllers, A transportation application. *Fifth IEEE Int. Conf. Fuzzy Systems (FUZZ-IEEE'96)*, New Orleans, LA. pp. 674–680.

Bouchon-Meunier, B., Yager, R., & Zadeh, L. (1995). *Fuzzy logic and soft computing*. Singapore: World Scientific.

Burkhardt, D., & Bonissone, P. (1992). Automated fuzzy knowledge base generation and tuning, *First IEEE Int. Conf. Fuzzy Systems (FUZZ-IEEE'92)*, San Diego, CA, pp. 179–188.

Chen, Y.-T., & Bonissone, P. (1998). Industrial applications of neural networks at General Electric, *Technical Information Series*, 98CRD79, General Electric CRD, Schenectady, NY.

Cordon, O., Herrera, H., & 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. *Proc. 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*, *2(1)*, 7–11.

Fiesler, E., & Beale, R. (1997). *Handbook of neural computation*, Bristol, UK: Institute of Physics, and New York: Oxford University Press.

Fogel, L.J. (1962). Autonomous automata, *Industrial Research 4*, 14–19.

Forbus, K. (1981). Qualitative reasoning about physical processes. *Proc. Seventh Int. Joint Conf. Artificial Intelligence*, Karlsruhe, Germany.

Fraser, A.S. (1957). Simulation of genetic systems by automatic digital computers. I. Introduction. *Australian Journal of Biological Sciences 10*, 484–491.

Herrera, F., & Lozano, M. (1996). Adaptive genetic algorithms based on fuzzy techniques. *Proc. IPMU'96*, Granada, Spain, pp. 775–780.

Holland, J.H. (1962). Outline of a logical theory of adaptive systems, *Journal of the ACM 9*, 297–314.

Holland, J.H. (1975) *Adaptation in natural and artificial systems*, Cambridge, MA: MIT Press.

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(3)*, 665–685.

Karr C.L. (1991). Design of an adaptive fuzzy logic controller using genetic algorithms, *Proc. Int. Conf. on Genetic Algorithms (ICGA'91)*, San Diego, CA, pp. 450–456.

Khedkar, P.S., & Keshav, S. (1992). Fuzzy prediction of timeseries. *Proc. First IEEE Int. Conf Fuzzy Systems*, pp. 281–288.

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)*, Orlando, FL, pp. 28–33.

Kitano, H. (1990). Empirical studies on the speed of convergence of neural network training using genetic algorithms. *AAAI-90 Proc. Eighth Natl. Conf. Artificial Intelligence*, vol. 2, pp. 789–795.

Koza, J. (1992). *Genetic programming: On the programming of computers by means of natural selection*, Cambridge, MA: MIT Press.

Kuipers, B. (1985). Commonsense reasoning about causality: Deriving behavior from structure. In *Qualitative reasoning about physical systems*, (Bobrow, D., Ed.) pp. 169–203. Cambridge, MA: MIT Press.

Lee, M.A., & Tagaki, H. (1993). Dynamic control of genetic algorithm using fuzzy logic techniques. In *Proc. Fifth Int. Conf. Genetic Algorithms*, (Forrest, S., Ed.), pp. 76–83.

Mamdani E.H., & Assilian, S. (1975). An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man Machine Studies 7(1)*, 1–13.

McCulloch W.S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity *Bulletin of Mathematical Biophysics 5*, 115–133.

McInerney, M., Dhawan, A.P. (1993). Use of genetic algorithms with backpropagation in training of feedforward neural networks. *Proc. 1993 IEEE Int. Conf. Neural Networks (ICNN '93)*, San Francisco, CA, vol.1, pp. 203–208.

Montana, D.J., & Davis, L. (1989). Training feedforward neural networks using genetic algorithms. *IJCAI-89 Proc. Eleventh Int. Joint Conf. Artificial Intelligence*, vol. 1, pp. 762–767. San Francisco: Morgan Kaufmann.

Morjaria, M., Azzaro, S., Bush, J., Nash, J., Smith, M., & Smith, W. (1998). System and method for isolating failures in a locomotive. US Patent 5,845272.

Morjaria, M., & Santosa, F. (1996). Monitoring complex systems with causal networks. *IEEE Computer Science and Engineering 3(4)*, 9–10.

Pearl, J. (1982). Reverend Bayes on inference engines: A distributed hierarchical approach, *Proc. 2nd Natl. Conf. on Artificial Intelligence*, pp. 133–136, Menlo Park, CA: AAAI.

Rechenberg, I. (1965). Cybernetic solution path of an experimental problem. *Royal Aircraft Establishment*, Library Translation no. 1122.

Rosenblatt, F. (1959). Two theorems of statistical separability in the perceptron. In *Mechanization of thought processes*, Symposium held at the National Physical Laboratory, pp. 421–456. London: HM Stationary Office.

Ruspini, E., Bonissone, P., & Pedycz, W. (1998). *Handbook of fuzzy computation*, Bristol, UK: Institute of Physics.

Schwefel, H.-P. (1965). *Kybernetische Evolution als Strategie der experimentellen Forschung in der Stromungstechnik*. Diploma Thesis, Technical University of Berlin, Germany.

Shafer, G. (1976). *A mathematical theory of evidence*. Princeton, NJ: Princeton University Press.

Takagi, T., & Sugeno, M. (1985). Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man, and Cybernetics 151*, 116–132.

Vonk, E., Jain, L.C., & Johnson, R.P. (1997). *Automatic generation of neural network architecture using evolutionary computation*. Singapore: World Scientific Publ. Co.

Wasserman, P.D. (1989). *Neural computing: Theory and practice*. Van Nostrand Reinhold, New York.

Werbos, P. (1974). *Beyond regression: New tools for predictions and analysis in the behavioral sciences*. PhD Thesis, Cambridge, MA: Harvard University.

Widrow, B., & Hoff, M.E. (1960). Adaptive switching circuits, *IRE Western Electric Show and Convention Record*, Part 4, pp. 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 11(1)*, pp. 137–145.

Yao, X. (1999). Evolving artificial neural networks, *Proceedings of the IEEE 87(9)*, 1423–1447.

Zadeh, L.A. (1965). Fuzzy sets. *Inf. Cont 8*, 338–353.

Zadeh, L.A. (1994). Fuzzy logic and soft computing: Issues, contentions and perspectives, *IIZUKA'94: 3rd Int. Conf. Fuzzy Logic, Neural Nets and Soft Computing*, Iizuka, Japan, pp. 1–2.

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*, Vol. 2, pp. 23–25. Springer-Verlag.

Zheng L. (1992). A practical guide to tune proportional and integral (PI) like fuzzy controllers. *First IEEE Int. Conf. Fuzzy Systems, (FUZZ-IEEE'92)*, San Diego, CA, pp. 633–640.

Piero P. Bonissone, Ph.D. in EECS, University of California, Berkeley, 1979, has been a computer scientist at GE Corporate Research and Development since 1979. Dr. Bonissone has carried out research in AI, expert systems, simulation, fuzzy sets, and Soft Computing. In 1993, he received the Coolidge Fellowship from GE for overall technical accomplishments. In 1996, he became a Fellow of the American Association for Artificial Intelligence. He is the President Elect of the IEEE Neural Network Council and an Adjunct Professor of ECSE at the Rensselaer Polytechnic Institute, in Troy, NY. Dr. Bonissone has coedited four books and published more than 100 articles. He received 19 patents from the U.S. Patent Office for his work on reasoning with uncertainty and fuzzy control.

Kai Goebel received the degree of Diplom-Ingenieur from the Technische Universität München, Germany in 1990. He received the M.S. and Ph.D. from the University of California at Berkeley in 1993 and 1996, respectively. He is currently with General Electric's Corporate Research and Development. His research interests include classification, information fusion, and Soft Computing. Dr. Goebel has been an adjunct professor of the CS Department at Rensselaer Polytechnic Institute (RPI), Troy, NY, since 1998, where he teaches classes in Soft Computing. He is a member of ASME, VDI, and AAAI.