Original paper

Soft Computing 1 (1997) 6–18

Soft computing: the convergence of emerging reasoning technologies

Piero P. Bonissone

Abstract The term Soft Computing (SC) represents the combination of emerging problem-solving technologies such as Fuzzy Logic (FL), Probabilistic Reasoning (PR), Neural Networks (NNs), and Genetic Algorithms (GAs). Each of these technologies provide us with complementary reasoning and searching methods to solve complex, real-world problems. After a brief description of each of these technologies, we will analyze some of their most useful combinations, such as the use of FL to control GAs and NNs parameters; the application of GAs to evolve NNs (topologies or weights) or to tune FL controllers; and the implementation of FL controllers as NNs tuned by backpropagation-type algorithms.

Keywords Soft computing, hybrid systems, fuzzy systems, neural networks, evolutionary systems

1 Soft computing and approximate reasoning systems

Soft Computing (SC) is a recently coined term describing the symbiotic use of many emerging computing disciplines. According to Zadeh (1994): "*... in contrast to traditional, hard computing, soft computing is tolerant of imprecision, uncertainty, and partial truth.*" In the context of our discussion we will consider Fuzzy Logic (FL), Probabilistic Reasoning (PR), Neural Networks (NNs), and Genetic Algorithms (GAs) as Soft Computing main components.

Fuzzy Logic, 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.

Probabilistic Reasoning 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, first explored by Rosenbaltt (1959), Widrow and Hoff (1960), 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 back-propagation type of algorithm [introduced by Werbos (1974)], NNs give us a supervised learning algorithm that performs fine-granule local optimization.

Genetic Algorithms, proposed by Holland (1975), give us a way to perform randomised 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 characteristics to their *offsprings*.

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. This is illustrated in Figure 1.

As we attempt to solve real-world problems, however, 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. The relevant available information is usually in the form of empirical prior knowledge and input-output data representing instances of the system's behavior. Therefore, we need approximate reasoning systems capable of handling such imperfect information. Soft Computing technologies provide us with a set of flexible computing tools to perform these approximate reasoning and search tasks.

In the remaining of this paper we will describe and contrast Soft Computing technology components: fuzzy and probabilistic reasoning, neural networks, and genetic algorithms. Then we will illustrate some examples of hybrid systems developed by leveraging combinations of these components, such as the control of GAs and NNs parameters by FL; the evolution of NNs topologies and weights by GAs or its application to tune FL controllers; and the realization of FL controllers as NNs tuned by backpropagation-type algorithms. Figure 2 provides a graphical summary of these hybrid algorithms and their components. The interest reader should

Piero P. Bonissone
General Electric Corporate Research and Development 1 Research Circle, K1 5C32A Schenectady, NY 12309, USA
e-mail: bonissone@crd.ge.com

This work was developed during my sabbatical stay at the Instituto de Investigaciones en Inteligencia Artificial (IIIA-CSIC), in Bellaterra, Barcelona) Spain. This paper is dedicated to all my IIIA friends who made my stay pleasant and unforgettable.

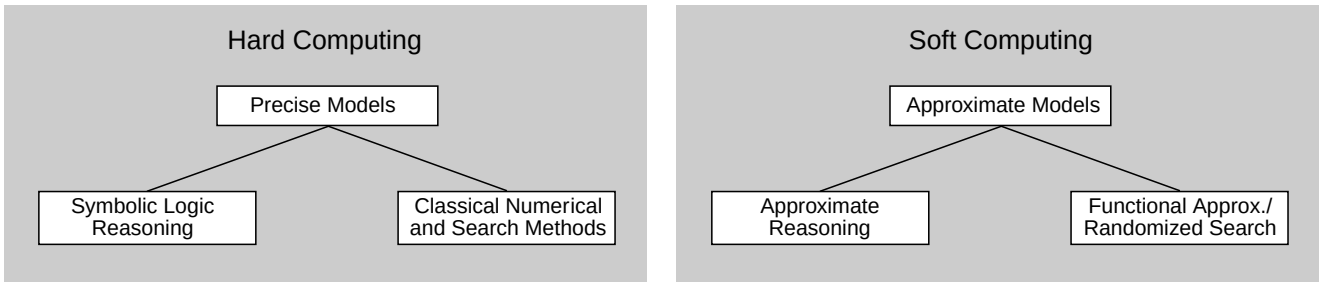

Fig. 1. Hard and Soft Computing

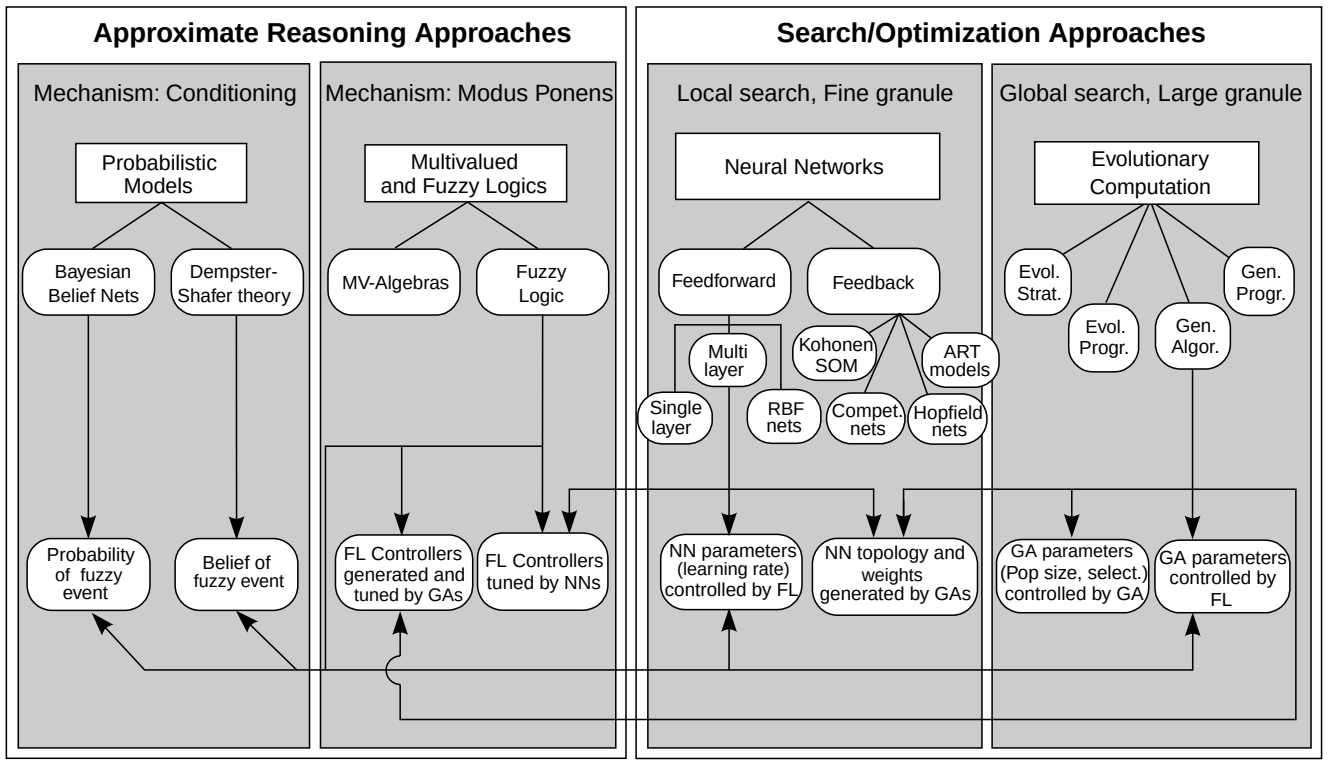

Fig. 2. Soft Computing Overview

consult Bouchon-Meunier *et al.* (1995) for an extensive coverage of this topic.

2
Probability and fuzziness

2.1
Distinctions

Randomness and fuzziness capture two rather different types of uncertainty and imprecision. In *randomness*, the uncertainty is derived by the nondeterministic membership of a point from the sample space in a well-defined region of that space. The sample space describes the set of possible values for the random variable. The point is the outcome of the system. The well-defined region represents the event whose probability we want to predict. The characteristic function of the region dichotomizes the sample space: either the point falls within the boundary of the region, in which case its membership value in the region is one and the event is true, or it falls outside the region, in which case its membership value in the region is zero and the event is false. A probability value describes the tendency or frequency with which the random variable takes values inside the region.

On the other hand, in *fuzziness* the uncertainty is derived from the partial membership of a point from the universe of discourse in an imprecisely defined region of that space. The region represents a fuzzy set. The characteristic function of the fuzzy set does not create a dichotomy in the universe of discourse. It defines a mapping from such universe into the real-valued interval $[0, 1]$ instead of the set $\{0, 1\}$. A partial membership value does not represent any frequency. Rather, it describes the degree to which that particular element of the universe of discourse satisfies the property that characterizes the fuzzy set [see Zadeh (1965)].

2.2
Interpretations

Not all probabilities have frequentistic interpretations. For example, subjective probabilities, introduced by DeFinetti (1937), can be defined in terms of the willingness of a rational agent to accept a bet, in which the ratio of its associated cost and prize reflects the probability of the event. Similarly, fuzzy membership values may have more than one interpretation. We could treat them as *possibility* values, i.e. fuzzy restrictions that act as elastic constraints on the value that may be assigned to a variable [see Zadeh (1978)], or as *similarity* values, i.e. the complement of the distances among possible worlds [see Ruspini (1989), Ruspini (1990)], or as *desirability or preference* values, i.e. the partial order induced by the membership function on the universe of discourse [see Dubois and Prade (1992)].

2.3
Probabilistic reasoning systems

The earliest probabilistic techniques are based on single-valued representations. These techniques started from approximate methods, such as the modified Bayesian rule, introduced by Duda *et al.* (1976) and confirmation theory, proposed by Shortliffe and Buchanan (1975), and evolved into formal methods for propagating probability values over Bayesian Belief Networks [see Pearl (1982), Pearl (1988a)]. Another trend among the probabilistic approaches is represented by interval-valued representations such as Dempster-Shafer theory [see Dempster (1967), Shafer (1976), Smets (1991)]. In all approaches, the basic inferential mechanism is the *conditioning* or updating operation. We will briefly review two main currents within probabilistic reasoning: Bayesian Belief Networks, and Dempster-Shafer's theory of belief.

2.3.1
Bayesian belief networks

Over the last ten years, considerable efforts have been devoted to improve the computational efficiency of Bayesian Belief Networks for trees, poly-trees, and Directed Acyclic Graphs (DAGs), such as influence diagrams [see Howard and Matheson (1984), Schachter (1986), Agogino and Rege (1987)].

An efficient propagation of belief on Bayesian Networks has been originally proposed by Pearl (1982), Pearl (1986a). In his work, Pearl describes an efficient updating scheme for trees and, to a lesser extent, for poly-trees [see Kim and Pearl (1983), Pearl (1986b), Pearl (1988b)]. However, as the graph complexity increases from trees to poly-trees to general graphs, so does the computational complexity. The complexity for trees is $O(n^2)$, where n is the number of values per node in the tree. The complexity for poly-trees is $O(K^m)$, where K is the number of values per parent node and m is the number of parents per child. This number is the size of the table attached to each node. Since the table must be constructed manually and updated automatically, it is reasonable to assume that the value of m will be small and so will the table. The complexity for multiconnected graphs is $O(K^n)$, where K is the number of values per node and n is the size of the largest nondecomposable subgraph. To handle such complexity, techniques such as moralization and propagation in a tree of cliques [see Lauritzen and Spiegelhalter (1988)] and loop cutset conditioning [see Suermondt *et al.* (1991), Stillman (1991)] are typically used to decrease the value of n, decomposing the original problem represented by the graph into a set of smaller problems or subgraphs.

When this problem decomposition process is not possible, exact methods are abandoned in favor of approximate methods. Among these methods the most common are clustering, bounding conditioning, discussed by Horvitz *et al.* (1989), and simulation techniques, such as logic samplings and Markov simulations described by Henrion (1989).

2.3.2
Dempster-Shafer theory of belief

Belief functions have been introduced in an axiomatic manner by Shafer (1976). Their original purpose was to compute the degree of belief of statements made by different sources or witnesses from a subjective probability of the sources reliability.

Many other interpretations of belief functions have been presented, ranging from functions induced from a probability measure by multivalued mappings [see Dempster (1967)] or by compatibility relations [see Lowrance *et al.* (1986)], to probability of provability [see Pearl (1988b)], to inner measures [see Ruspini (1987), Fagin and Halpern (1989)], to a nonprobabilistic model of transferable belief [see Smets (1991)].

All interpretations share the same static component of the theory: the Möbius Transform, which defines a mapping from basic probability assignments, masses assigned to subsets of the frame of discernment, to the computation of the lower bound (belief) of a proposition, a region defined in the same frame of discernment. An inverse Möbius transform can be used to recover the masses from the belief. All these interpretations also share the same definition of the upper bound, usually referred to as plausibility.

More specifically, this formalism defines a function that maps subsets of a space of propositions Θ on the [0,1] scale. The sets of partial beliefs are represented by mass distributions of a unit of belief across the propositions in Θ. This distribution is called basic probability assignment (bpa). The total certainty over the space is 1. A non-zero *bpa* can be given to the entire space Θ to represent the degree of ignorance, which models the source lack of complete reliability. Given a space of propositions Θ, referred to as frame of discernment, a function m: $2^{\Theta} \rightarrow [0, 1]$ is called a basic probability assignment if it satisfies the following three conditions:

$$m(\phi) = 0 \qquad \text{where } \phi \text{ is the empty set} \tag{1}$$

$$0 < m(A) < 1 \tag{2}$$

$$\sum_{A \subseteq \Theta} m(A) = 1 \tag{3}$$

The certainty of any proposition A is then represented by the interval $[Bel(A), P^{*}(A)]$, where $Bel(A)$ *and* $P^{*}(A)$ are defined as:

$$Bel(A) = \sum_{\phi \neq x \subseteq A} m(x) \tag{4}$$

$$P^{*}(A) = \sum_{x \cap A \neq \phi} m(x) \tag{5}$$

where $x \subseteq \Theta$. From the above definitions the following relation can be derived:

$$Bel(A) = 1 - P^*(\neg A) \tag{6}$$

(4) and (5) represent the *static* component of the theory, which is common to all interpretations. However, these interpretations do not share the same *dynamic* component of the theory: the process of updating (i.e., conditioning or evidence combination). This issue has been recently addressed by various researchers, such as Halpern and Fagin (1990), Smets (1991).

Inference mechanism: Conditioning Given the beliefs (or masses) for two propositions A and B, Dempster's rule of combination can be used, under assumptions of independence, to derive their combined belief (or mass).

If m_1 and m_2 are two *bpas* induced from two independent sources, a third *bpa*, $m(C)$, expressing the pooling of the evidence from the two sources, can be computed by using Dempster's rule of combination:

$$m(C) = \frac{\sum_{A_i \cap B_j = C} m_1(A_i) \cdot m_2(B_j)}{1 - \sum_{A_i \cap B_j \neq \phi} m_1(A_i) \cdot m_2(B_j)} \tag{7}$$

Dempster's rule allows us to consider and pool discounted pieces of evidence, i.e. evidence whose belief can be less than one. On the other hand, conditioning can only be done with certain evidence. If proposition B is true (i.e., event B has occurred), then $Bel(B) = 1$ and from Demspster rule of combination, we can derive a formula for conditioning A given B, $Bel(A|B)$:

$$Bel(A|B) = \frac{Bel(A \cup \neg B) - Bel(\neg B)}{1 - Bel(\neg B)} \tag{8}$$

This expression is compatible with the interpretation of Belief as evidence, and as inner measure. However, this expression is not compatible with the interpretation of belief as the lower envelope of a family of probability distributions. Under such interpretation, the correct expression for conditioning is

$$Bel(A \| B) = \frac{Bel(A \cap B)}{Bel(A \cap B) + Pl(\neg A \cap B)} \tag{9}$$

The interested reader is referred to Shafer (1990) for a clear explanation and an updated bibliography on belief functions.

As for the case of belief networks, a variety of exact and approximate methods have been proposed to perform inferences using belief functions. Typically, the exact methods require additional constraints on the structure of the evidence.

All above probabilistic methods use the operation of *conditioning* to update the probability values and perform a probabilistic inference. We will now switch our focus to fuzzy logic based systems, a class of approximate reasoning systems whose inference mechanism is not conditioning, but an extension of *modusponens*.

2.4 Fuzzy logic based reasoning systems

Fuzzy logic approaches are based on a fuzzy-valued representation of uncertainty and imprecision. Typically they use Linguistic Variables, as proposed by Zadeh (1978), Zadeh (1979), to represent different information granularities, and Triangular-norms to propagate the fuzzy boundaries of such granules [see Schweizer and Sklar (1963), Schweizer and Sklar (1983), Dubois and Prade (1984), Bonissone (1987), Bonissone and Decker (1986), Bonissone *et al.* (1987)].

The basic inferential mechanism used in fuzzy reasoning systems is the *generalized modus-ponens*, introduced by Zadeh (1979), which makes use of inferential chains (syllogisms).

2.4.1 Triangular norms: a review

Since Triangular norms play such an important role in the definition of the generalized modus ponens, we will provide the reader with a brief overview of these operators. Triangular norms (T-norms) and their dual T-conorms are two-place functions from $[0, 1] \times [0, 1]$ to $[0, 1]$ that are monotonic, commutative and associative. They are the most general families of binary functions that satisfy the requirements of the conjunction and disjunction operators, respectively. Their corresponding boundary conditions satisfy the truth tables of the Boolean AND and OR operators.

Any triangular norm $T(A, B)$ falls in the interval $T_w(A, B) \leq T(A, B) \leq Min(A, B)$, where

$$T_w(A, B) = \begin{cases} \min(A, B) & \text{if } \max(A, B) = 1, \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

The corresponding DeMorgan dual T-conorm, denoted by $S(A, B)$, is defined as

$$S(A, B) = 1 - T(1 - A, 1 - B) \tag{11}$$

$T_w(A, B)$ is referred to as the drastic T-norm (to reflect its extreme behavior) and is clearly non-continuous. By changing one of the axioms of the T-norms [see Schweizer and Sklar (1963)] we can derive a subset of T-norms, referred to as *copulas*, such that any copula $T(A, B)$ falls in the interval $Max(0, A + B - 1) \leq T(A, B) \leq Min(A, B)$.

In the original version of fuzzy logic proposed by Zadeh (1965), the conjunction and disjunction operators are the *minimum* and *maximum*, i.e. the upper and lower bounds of the T-norm and T-conorm ranges, respectively. These operators are the only ones satisfying distributivity and idempotency [see Bellman and Giertz (1973)]. Other selection of T-norms and T-conorms provide different logics with different properties [see Klement (1981), Bonissone and Decker (1986)].

Perhaps the most notable selection is the one based on the lower bound of the T-norms (Lukasiewicz T-norm) [see Lukasiewicz (1967)] and its dual T-conorm. This logic satisfies the *law of the excluded-middle*, at the expense of distributivity, and is the basis of MV-Algebras [see Di Nola and Gerla (1986)]. In the fuzzy logic community this algebra was originally referred to as *bold* algebra, a concept first introduced by Giles (1981). Mundici (1995) has provided us with interesting semantics for this algebra, presenting it as a decision

making or voting paradigm in a context in which the information source can have up to a maximum number of k lies (errors). This paradigm is also known as Ulam's game with k lies and has direct applications to on-line error correcting code.

Fuzzy reasoning systems can be used in many applications, from advice providing expert systems, to soft constraint propagation, to decision making systems, etc. Within this paper we will limit our scope to Fuzzy Controllers (FCs), reasoning systems composed of a Knowledge Base (KB), an inference engine, and a defuzzification stage. The KB is comprised by a rule base, describing the relationship between state vector and output, and by the semantics of the linguistic terms used in the rule base. The semantics are established by scaling factors delimiting the regions of saturation and by termsets defining a fuzzy partition in the state and output spaces [see Bonissone and Chiang (1993)].

The concept of a fuzzy controller was initially outlined by Zadeh (1973) and first explored by Mamdani and Assilian (1975), Kickert and Mamdani (1978) in the early seventies. Currently it represents one of the most successful applications of fuzzy logic based systems [see Bonissone *et al.* (1995)].

2.4.2
Inference mechanism: generalized modus ponens

Rule base The most common definition of a fuzzy rule base R is the disjunctive interpretation initially proposed by Mamdani and Assillian (1975) and found in the majority of Fuzzy Controller applications:

$$R = \bigcup_{i=1}^{m} r_i = \bigcup_{i=1}^{m} (\bar{X}_i \rightarrow Y_i) \tag{12}$$

R is composed of a disjunction of m rules. Each rule r_i defines a mapping between a fuzzy state vector $\overline{X_i}$ and a corresponding fuzzy action Y_i. Each rule r_i is represented by the Cartesian product operator. There are, however, other representations for a fuzzy rule, which are based on the material implication operator and the conjunctive interpretation of the rule base [see Trillas and Valverde (1985)]. For a definition of different rule types, the interested reader should consult Misumoto and Zimmerman (1982), Lee (1990). A particularly useful type of FC is the Takagi-Sugeno (TS) controller [see Takagi and Sugeno (1985)]. In this type of controller the output Y_i of each rule r_i: $(\overline{X_i} \rightarrow Y_i)$ is no longer a fuzzy subset of the output space but rather a first order polynomial in the state space $\overline{X_i}$, i.e.:

$$Y_i = F_i(\overline{X_i}) = c_0 + \sum_{j=1}^{n} c_j X_{i,j} \tag{13}$$

where $X_{i,j}$ is the jth element of the n-dimensional vector $\overline{X_i}$.

Modus ponens The inference engine of a FC can be defined as a parallel forward chainer operating on fuzzy production rules. An input vector $\overline{I}$ is matched with each n-dimensional state vector $\overline{X_i}$, i.e., the Left Hand Side (LHS) of rule $r_i = \overline{X_i} \rightarrow Y_i$. The degree of matching λ_i indicates the degree to which the rule output can be applied to the overall FC output. The main inferences issues for the FC are; the definition of the fuzzy predicate evaluation, which is usually a possibility measure [see Zadeh (1978)]; the LHS evaluation, which is typically a triangular norm [see Schweizer and Sklar (1963), Schweizer and Sklar (1983), Bonissone (1987)]; the conclusion detachment, which is normally a triangular norm or a material implication operator; and the rule output aggregation, which is usually a triangular conorm for the disjunctive interretation of the rule base, or a triangular norm for the conjuctive case.

Under commonly used assumptions we can describe the output of the Fuzzy System as

$$\mu_Y(y) = \bigvee_{i}^{m} min[\lambda_i, \mu_{Y_i}(y)]) \tag{14}$$

where λ_i is the degree of applicability of rule r_i

$$\lambda_i = \bigwedge_{j}^{n} \Pi(X_{i,j} | I_j) \tag{15}$$

and $\Pi(X_{i,j}|I_j)$ is the possibility measure representing the matching between the reference state variable and the input element[1]

$$\Pi(X_{i,j} I_j) = \bigvee_{x_j} (min[\mu_{x_{i,j}}(x_j), \mu_I(x_j)]) \tag{17}$$

(14), (15), and (17) describe the generalized modus ponens [see Zadeh (1979)], which is the basis for interpreting a fuzzy-rule set.

2.5
Defuzzification

The most important FC design choice is the selection of the defuzzification mode. The output of the rule aggregation stage is a composite membership distribution defined on the space of control actions. This distribution must be summarized into a scalar value before it is passed to an actuator for execution. This summarization can be performed by a variety of defuzzifiers: the Mean of Maxima (MOM), the Center of Gravity (COG), the Height Method (HM).

The selection of the defuzzifier is a tradeoff between storage requirements (MOM lends itself to easy compilation), performance (COG typically exhibits the smoothest performance), and computational time (HM is faster to compute than COG) [see Mizumoto (1989)].

2.6
Complementarity

The distinction between probability and fuzziness has been presented and analyzed in many different publications, such as Berdek (1994), Dubois and Prade (1993), Klir and Folger (1988) to mention a few. Most researches in probabilistic reasoning and fuzzy logic have reached the same conclusion about the complementarity of the two theories [see Bonissone (1991)]. This complementarity was first noted by Zadeh (1968), who introduced the concept of the

[1] When the input is crisp, the degree of matching is the evaluation of the reference membership distribution at the point representing the value of the input:

$$\Pi(X_{i,j}|I_j) = \min[\mu X_{i,j}(x_0), \mu I_j(x_0)] = \mu x_{i,j}(x_0) \tag{16}$$

probability measure of a fuzzy event. Let A be a fuzzy event, i.e, a subset of a finite sample space X. Let also x_i represent the ith singleton in such a space and $p(x_i)$ be its probability. Then $P(A)$, the probability of the fuzzy event A, is the weighted sum of the probability of each singleton in the sample space multiplied by $\mu_A(x_i)$, the partial degree to which the singleton x_i belongs to the fuzzy subset A, i.e.:

$$P(A) = \sum_{x_i \in X} p(x_i) \times \mu_A(x_i) \tag{18}$$

In 1981 Smets, extended the theory of belief functions to fuzzy sets by defining the belief of a fuzzy event [see Smets (1981), Smets (1988)]. Let A be a fuzzy event of a finite sample space X. Let also S_i represent the ith available piece of evidence, i.e. a non-empty subset of the frame of discernment with an assigned probability mass $m(S_i)$. Then $Bel(A)$, the belief of the fuzzy event A, is the weighted sum of the probability masses of each piece of evidence multiplied by the minimum degree to which the evidence supports the event, i.e. it is included in the fuzzy region defining the event:

$$Bel(A) = \sum_{\phi \neq S_i \subseteq X} m(S_i) \times \bigwedge_{x_j \in S_i} \mu_A(x_j) \tag{19}$$

We have established the orthogonality and complementarity between probabilistic and possibilistic methods. Given their duality of purpose and characteristics, it is clear that these technologies ought to be regarded as being complementary rather than competitive.

3
Neural networks

Fuzzy logic enables us to translate and embed empirical, qualitative knowledge about the problem to be solved into reasoning systems capable of performing approximate pattern matching and interpolation. Fuzzy logic however does not have adaptation or learning features, since it lacks the mechanism to extract knowledge fron existing data. Of course, it could be argued that it is possible to use fuzzy clustering methods, such as Fuzzy C-means [see Bezdek and Harris (1978), Bezdek (1981)] to provide more accurate definitions of the membership functions of the state and output variables, in a typical unsupervised mode. However, FL systems are not able to *learn* from examples of input-output pairs, in a typical supervised mode.

On the other hand, this is the typical characteristic of Neural Networks, another Soft Computing technology. NNs and Perceptorns started in the early 60s as algorithms to train adaptive elements. Their origins can be traced to the works of Rosenbaltt (1959) on *spontaneous learning*, Stark *et al.* (1962) on competitive learning, and Widrow and Hoff (1960) on the development of ADALINE and MADALINE algorithms.

Typically NNs are divided into *Feed-Forward* and *Recurrent/Feedback* networks. The Feed-Forward networks include single-layer perceptrons, multilayer perceptrons, and Radial Basis function nets (RBFs) [see Moody and Darken (1989)], while the Recurrent nets cover Competitive networks, Kohonen (1982) Self Organizing Maps, Hopfield, J. (1982) nets, and ART models [see Carpenter and Grossberg 1983), Carpenter and Grossberg (1987), Carpenter and Grossberg (1990)]. While feedforward NNs are used in supervised mode, recurrent NNs are typically geared toward unsupervised learning, associative memory, and self-organization. In the context of our paper we will only consider feedforward NNs. Given the functional equivalence already proven between RBF and fuzzy systems [see Jang *et al.* (1993)] we will further limit our discussion to multilayer feedforward nets.

A feedforward multilayer NN is composed of a network of processing units or neurons. Each neuron performs the weighted sum of its input, using the resulting sum as the argument of a non-linear activation function. Originally the activation functions were sharp thresholds (or Heavyside) functions, which evolved to piecewise linear saturation functions, to differentiable saturation functions (or sigmoids), and to gaussian functions (for RBFs). For a given interconnection topology, NNs train their weight vector to minimize a quadratic error function.

Prior to backpropagation, proposed by Werbos (1974), there was no sound theoretical way to train multilayers, feedforward networks with nonlinear neurons. On the other hand single-layer NNs (perceptrons) were too limited, as they could only provide linear partitions of the decision space. While their limitations were evidenced by Minsky and Papert (1969), Hornick *et al.* (1989) proved that a three-layers NN was an universal functional approximator. Therefore, with the advent of BP, multilayers feedforward NNs became extremely popular. Since then, most researchers on NNs have focused their efforts on improving BP's converge speed: by using estimates of the second derivatives, under simplifying assumptions of a quadratic error surface, as in Quickprop [see Fahlman (1988)]; by changing the size of the step size in a self-adapting fashion such as SuperSAB [see Tollenaere (1990)]; or by using second order information, as in the Conjugate Gradient Descent method described by Moller (1990). An excellent history of Adaptive NNs is provided by Widrow (1990).

3.1
Learning

In the context of this paper, we will consider learning only in the context of Soft Computing. Therefore, we will limit our discussion to *structural* and *parametric* learning, which are the counterpart of system identification and parameter estimation in classical system theory. For a fuzzy controller learning (or tuning) entails defining (or refining) the knowledge base (KB), which is composed of the a parameter set (state and output scaling factors, state and output termsets) and a structure (the rule base). The parameter set describe the local semantics of the language and the rule set describe the syntactic mapping. For Neural Networks, structural learning means the synthesis of the network topology (i.e., the number of hidden layers and nodes), while parametric learning implies determining the weight vectors that are associated to each link in a given topology.

Learning can be facilitated by the availability of complete or partial feedback. In the case of total feedback (a teacher providing an evaluation at every iteration or a training set describing the correct output for a given input vector) we have supervised learning. When only partial feedback is available (every so often we are told if we succeed or failed) we have

reinforcement learning. When no feedback is available we have the case of unsupervised learning.

3.1.1
Supervised learning

In the context of supervised learning, ANFIS (Adaptive Neural Fuzzy Inference Systems) [see Jang (1993)] is a great example of an architecture for tuning fuzzy system parameters from input-output pairs of data. The fuzzy inference process is implemented as a generalized neural network, which is then tuned by gradient descent techniques. It is capable of tuning antecedent parameters as well as consequent parameters of TSK-rules which use a softened trapezoidal membership function. It has been applied to a variety of problems, including chaotic timeseries prediction and the IRIS cluster learning problem. As a fuzzy system, it does not require a large data set and it provides transparency, smoothness, and representation of prior knowledge. As a neural system, it provides parametric adaptability.

3.1.2
Steepest descent

Backpropagation neural-net based techniques usually depend on a differentiable input-output map, which restricts their applicability. For controllers, it is practical to evaluate their performance over the whole trajectory rather than individual states, and a few such evaluations provide a crude local snapshot of the performance surface as a function over parameter space. This snapshot can then guide a steepest descent algorithm to determine the two scaling factors (K_p and K_i) in the design of a fuzzy logic PI controller.

The evaluation function is a metric based on the total deviation of the actual trajectory from an ideal trajectory, which is crafted based on the specifications of the controller, such as rise time, settling time, steady-state error band and steady-state oscillation. The metric can be quite flexible if desired.

The method uses a logarithmic search of the parameter space followed by a linear one to identify the appropriate scaling factors. The same method can be applied to other critical parameters such as the centers of the output membership functions. For low-dimensional searches, this method can be applied easily to any kind of system and remains reasonably efficient, since it is easy to parallelize.

3.1.3
Reinforcement learning

Reinforcement learning exploits the availability of expert knowledge in the area of exerting control actions as well as evaluating system state. Approximate linguistic rules can be used to initialize the two knowledge bases which deal with action selection and action evaluation. The resulting system is capable of learning to control a complex, dynamic system in the absence of desired output, with only a delayed, somewhat uninformative reinforcement signal from the environment. This system has been used to control systems from the inverted pendulum to the space shuttles attitude control, [see Berenji and Khedkar (1992)].

3.1.4
Structural learning: rule clustering

The previously mentioned systems deal mainly with parameter identification once the structure has been fixed. However, identifying the number of rules in a fuzzy system or fixing the granularity of the fuzzy partition of the input space is a structure identification problem which also needs to be solved. If expert rules are not available, then other known properties of the unknown function may be available and could be exploited. For instance, in industrial settings, many mappings are implemented as approximations using look-up tables or analytic interpolation functions. If a fuzzy system can be reverse engineered from such information, then knowledge extraction can help to refine or upgrade the system.

If analytical information about the mapping is available, then various algorithms can be used to extract a near-optimal fuzzy rulebase which is equivalent to the mapping. On the other hand, the function can be sampled for data points or the look-up table can be used to generate the data points according to a desired distribution. If there is sufficient data, it is possible to adapt neural network clustering methods to extract clusters in product space which correspond to fuzzy rules. The method uses the joint criteria of incompleteness as well as accuracy in prediction to add rules to the database and then conducts a deletion phase to prune redundant knowledge. This system extracts a set of rules from a single online pass over a reasonably small dataset. It can then be tuned by using the same gradient optimization techniques to tune the parameters as have been discussed above. The reader is referred to Berenji and Khedkar (1993), Khedkar (1993) for a detailed discussion of reinforcement learning and rule clustering.

4
Evolutionary computing

In the previous section we discussed supervised learning of fuzzy system parameters. Since gradient descent techniques may become mired in local minima, global search techniques have also been explored. We will focus our attention on a randomized global search paradigm, which is commonly referred to as Evolutionary Computation (EC). This paradigm covers several variations, such as *Evolutionary Strategies* (ES), addressing continuous function optimization [see Rechenberg (1965), Schwefel (1965)]; *Evolutionary Programs* (EP), generating finite state automata that describe strategies or behaviors [see Fogel (1962), Fogel *et al.* (1966)]; *Genetic Algorithms* (GAs), providing continuous and discrete function optimization, system synthesis, tuning, testing, etc. [see Holland (1975)]; and *Genetic Programming* (GP), evolving computer programs to approximately solve problems, such as generating executable expressions to predict timeseries, etc. [see Koza (1992)].

As noted by Fogel (1995) in his historical perspective and a comparison of these paradigms: . . . *the three main lines of investigation – genetic algorithms, evolution strategies, and evolutionary programming – share many similarities. Each maintains a population of trial solutions, imposes random changes to those solutions, and incorporate selection to determine which solutions to maintain in future generations* . . . . Fogel also notes that GAs *emphasize models of*

genetic operators as observed in nature, such as crossing-over, inversion, and point mutation, and apply these to abstracted chromosomes, while ES and EP *emphasize mutational transformations that maintain behavioral linkage between each parent and its offspring*. In this paper, we will limit our analysis to Genetic Algorithms.

4.1
Genetic algorithms

Genetic Algorithms (GAs) are perhaps the most widely known of the above paradigms. In the context of designing fuzzy controllers, it is relatively easy to specify an evaluation of the trajectory or the controller as a whole, but it is difficult to specify desired step-by-step actions, as would be required by supervised learning methods. Thus Genetic Algorithms can use such an evaluation function to design a fuzzy controller.

GAs are a new programming paradigm that has been applied to much more difficult (NP-hard) optimization problems such as scheduling with very promising results. GAs encode the solution to a given scheduling problem in a binary- (or real-valued) string [see Holland (1975), Goldberg (1978), Michalewics (1994)]. Each string's element represents a particular feature in the solution. The string (solution) is evaluated by a fitness function to determine the solution's quality: good solutions survive and have off-springs, while bad solutions are discontinued. Solution's constraints are modeled by penalties in the fitness function or encoded directly in the solution data structures. To improve current solutions, the string is modified by two basic type of operators: cross-over and mutations. Cross-over are deterministic operators that capture the best features of two parents and pass it to a new off-spring string. Mutations are probabilistic operators that try to introduce needed solutions features in populations of solutions that lack such feature.

Some GAs have exhibited exceptional performances in large scale scheduling problems. However, many unanswered questions still remain. Design questions range from the type of solution and constraints encoding to probability of mutation, definition of fitness function, desired type of cross-over operations (to encode context dependent heuristics), etc. More fundamental questions include the applicability conditions of GAs, comparative analyses with other scheduling techniques, and, in general, a deeper understanding of the way GAs explore the solution space.

4.2
Simulated annealing

A special case of the genetic algorithm approach is the method known as Simulated Annealing (SA), which is considered a probabilistic hill-climbing technique [see Romeo and Sangiovanni-Vincentelli (1985)]. SA is a more restricted version of GAs, with well understood convergence properties. Simulated Annealing can be seen as a GA in which crossovers are disabled and only mutations implemented by the probability of jumping the energy barrier are allowed. Furthermore, the population size is typically one. SA is also a global search strategy and can work in very high-dimensional searches, given enough computational resources. An interesting hybrid algorithm that spans the space from GAs to SAs has been proposed by Adler (1993). In his algorithm the GAs operators use Simulated Annealing to determine if the newly generated solution is better than the best of its parents (in the case of the crossover operator) or better than the original solution (in the case of the mutation operator).

5
Hybrid algorithm: 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 (FL, NN, GA) have been integrated to improve the overall algorithm performance. In the sequel we will analyze a few of such combinations.

5.1
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. 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. More specifically, the typical equation for the weight changes in a NN is:

$$\Delta W_n = -\eta \nabla E(W_n) + \alpha \Delta W_{n-1} \tag{20}$$

in which ΔW_n represents the changes to the weight vector W_n, $E(W_n)$ is the error function at the nth iteration, η is the learning rate and α is the momentum. The learning rate η is a function of the step size k and determines how fast the algorithm will move along the error surface, following its gradient. Therefore the choice of η has an impact on the accuracy of the final approximation and on the speed of convergence. The smaller the value of η the better the approximation but the slower the convergence. Jacobs (1988) established a heuristic rule, known as the *Delta-bar-delta rule* to increase the size of η if the sign of ∇E was the same over several consecutive steps. Arabshahi *et al.* (1992) developed a simple Fuzzy Logic controller to modify η as a function of the error and its derivative, considerably improving Jacobs' heuristics.

5.2
GAs controlled by FL

The use of Fuzzy Logic to translate and improve heuristic rules has also been applied to manage the resource of GAs (population size, selection pressure) 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), Herrera *et al.* (1995a), 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.

In the same reference we can see two examples of this adaptability used to avoid the premature convergency of the GA to an inferior solution. This problem occurs when, due to selection pressure, disruption caused by the crossover operators, and inadequate parameter settings, the GA exhibits a lack of diversity in its population.

The first approach is based on dynamic crossover operators applied to real-coded chromosomes. These operators use different type of aggregators: *t-norms* and *t-conorms* to emphasize exploration properties, and *averaging operators* to show exploitation properties.

The second approach (in the same reference) uses two FL controllers to control the use of the exploitative crossover and the selection pressure. For this purpose two diversity measures are defined: the *genotypic diversity*, which measures the (normalized) average distance of the population from the best chromosome; and the *phenotypic diversity*, which measures the ratio between the best fitness and the average fitness. These diversity measures are the inputs to the FLCs. Every five generations the FLCs evaluate these measures to adjust the probability of using an exploitative crossover (based on averaging aggregators) and the selection pressure (keeping or eliminating diversity in the next generation).

It should be noted that there are other ways of controlling the GAs parameters setting. Specifically, GAs have also been applied at the meta-level to control the resource parameters of object-level GAs [see Grefenstette (1986)].

5.3 FL controller tuned by GAs

Many researchers have explored the use of genetic algorithms to tune fuzzy logic controllers. Cordon *et al.* (1995) contains 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).

C. Karr, one of the precursors in this quest [see Karr (1991b), Karr (1991a), Karr (1993)], used GAs to modify the membership functions in the termsets of the variables used by the FCs. 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.

Herrera *et al.* (1995b) directly tuned each rule used by the FC. They used a real encoding for a four-parameter characterization of a trapezoidal membership value in each termset. Each rule was represented by the concatenation of the membership values used in the rule antecedent (state vector) and consequent (control action). The population was the concatenation of all rules so represented. A customized (max-min arithmetical) crossover operator was also proposed. The fitness function was a sum of quadratic errors.

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 [see 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 Takagi-Sugeno rule [see Takagi and Sugeno (1985)], concatenating the membership distributions in the rule antecedent with the polynomial coefficients of the consequent.

Also interesting is the approach taken by Surmann *et al.* (1993), who modify the usual quadratic fitness function by addition of an entropy term describing the number of activated rules.

In Bonissone *et al.* (1996), we followed the tuning order suggested by Zheng (1992) for manual tuning. We 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, we proceeded to tune the *termsets*, causing medium-size effects. Finally, if additional improvements were needed, we 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.

5.4 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 then Back-Propagation (BP) 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 [see Maniezzo (1994), Patel and Maniezzo (1994)] 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, Back-Propagation would perform the final parameter tuning.

McInerney and Dhawan improved Kitano's algorithm by using the GA to escape from the local minima found by the backpropagation 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 [see McInerney and Dhawan (1993)].

For an extensive review of the use of GAs in NNs, the reader is encouraged to consult Schaffer *et al.* (1992), Yao (1992).

5.5
FL controller tuned by NNs

Among the first to propose the combined use of FL and NNs was Lee and Lee (1974), who proposed a multi-input/multi-output neuron model, in contrast with the binary-step output function advocated in the mid seventies. Since then we have witnessed many FL-NNs combinations – see Takagi (1990) for a more exhaustive coverage.

Within the limited scope of using NNs to tune FL Controllers, we already mentioned the seminal work on ANFIS (Adaptive Neural Fuzzy Inference Systems) by Jang (1993). 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 λ_i – see (15) – will add up to one. The fifth layer computes the polynomial coefficients in the right-hand-side of each Takagi-Sugeno rule, as described in (13). 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 and 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.

Many other variations of FLC tuning by NN have been developed, such as the ones described in Kawamura *et al.* (1992), Bersini *et al.* (1993), Bersini *et al.* (1995).

5.6
Hybrid GAs

Since GAs are quite robust with respect to being trapped in local minima (due to the global nature of their search) but rather inaccurate and inefficient in finding the global minimum, several modifications have been proposed to exploit their advantage and compensate for their shortcoming. Of special interest is the work by Renders and Bersini (1994), who proposed two type of hybrid GAs. The fist type consists in interweaving GAs with Hill Climbing techniques (GA + HC): the solution selection no longer depends on the *instantaneous* evaluation of the fitness function applied to the solution but rather applied to a refinement of the solution obtained via Hill Climbing techniques. The second type of hybrid consists in embedding optimization techniques in the crossover operator used by the GAs. The population size is $\lambda(n+1)$ individuals, of which only λ individuals are replaced by offsprings. Each offspring is obtained from a group of $(n+1)$ parents via a *simplex crossover*. His results show by combining the two hybrid methods, (GA + HC) and (Simplex crossover) the resulting hybrid algorithm outperforms each of its components in achieving maximum fitness, reliably, accurately, and minimizing computing time. An analysis of the tradeoff between accuracy, reliability and computing time for hybrid GAs can be found in Renders and Flasse (1976).

6
Conclusions

We should note that Soft Computing technologies are relatively young: Neural Networks originated in 1959, Fuzzy Logic in 1965, Genetic Algorithms in 1975, and probabilistic reasoning (beside the original Bayes' rule) started in 1967 with Dempster's and the in early 80s with Pearl's work. Originally, each algorithm had well-defined labels and could usually be identified with specific scientific communities, e.g. fuzzy, probabilistic, neural, or genetic. Lately, as we improved our understanding of these algorithms' strengths and weaknesses, we began to leverage their best features and developed hybrid algorithms. Their compound labels indicate a new trend of co-existence and integration that reflects the current high degree of interaction among these scientific communities. These interactions have given birth to Soft Computing, a new field that combines the versatility of Fuzzy Logic to represent qualitative knowledge, with the data-driven efficiency of Neural Networks to provide fine-tuned adjustments via local search, with the ability of Genetic Algorithms to perform efficient coarse-granule global search. The result is the development of hybrid algorithms that are superior to each of their underlying SC components and that provide us with the better real-world problem solving tools.

References

1. **Adler, D.:** Genetic Algorithm and Simulated Annealing: A Marriage Proposal. In: IEEE International Conference on Neural Networks, pages 1104–1109, San Francisco, 1993
2. **Agogino, A. M.; Rege, A.:** IDES: Influence Diagram Based Expert System. Mathematical Modelling 8 (1987) 227–233
3. **Arabshahi, P.; Choi, J. J.; Marks, R. J.; Caudell, T. P.:** Fuzzy Control of Backpropagation. In: First IEEE International Conference on Fuzzy Systems, pages 967–972, San Diego, CA 1992
4. **Bayes, T.:** An essay towards solving a problem in the doctrine of chances. Philosophical Transactions of the Royal Society of London, 53:370–418 (1763) (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.)
5. **Bellman, R.; Giertz, M.:** On the Analytic Formalism of the Theory of Fuzzy Sets. Information Science 5, (1) (1973) 149–156
6. **Berenji, H. R.; Khedkar, P.:** Learning and tuning fuzzy logic controllers through reinforcements. IEEE Transactions on Neural Networks, 3(5) 1992
7. **Berenji, H. R.; Khedkar, P.:** Clustering in product space for fuzzy inference. In: Second IEEE International Conference on Fuzzy Systems, San Francisco, CA 1993
8. **Bersini, H.; Nordvik, J. P.; Bonarini, A.:** A simple Direct Adaptive Fuzzy Controller Derived from its Neutral Equivalent. In: IEEE International Conference on Neural Networks, pages 345–350, San Francisco, CA 1993

9. **Bersini, H.; Nordvik, J. P.; Bonarini, A.:** Comparing RBF and Fuzzy Inference Systems on Theoretical and Practical Basis. In: International Conference on Artificial Neural Networks (ICANN95), pages 169–174, Paris, France 1995
10. **Bezdek, J. C.; Harris, J. O.:** Fuzzy partitions and relations: An axiomatic basis for clustering. Fuzzy Sets and Systems 1 (1978) 112–127
11. **Bezdek, J. C.:** Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum Press, New York 1981
12. **Bezdek, J. C.:** Editorial: Fuzzines vs Probability - Again (! ?). IEEE Transactions on Fuzzy Systems, 2(1): 1–3, 1994
13. **Bonissone, P. P.: Chiang, K. H.:** Fuzzy logic controllers: From development to deployment. In: Proceedings of IEEE Conference on Neural Networks, pages 610–619. IEEE 1993
14. **Bonissone, P. P.; Decker, K.:** Selecting Uncertainty Calculi and Granularity: An Experiment in Trading-off Precision and Complexity. In: Kanal, L. N.; Lemmer, J.F. (eds.); Uncertainty in Artificial Intelligence, pages 217–247. North Holland 1986
15. **Bonissone, P. P.; Gans, S. S.; Decker, K. S.:** RUM: A Layered Arichitecture for Reasoning with Uncertainty. In: Proceedings 10th Intern. Joint Conf. Artificial Intelligence, pages 891–898. AAAL 1987
16. **Bonissone, P. P.; Badami, V.; Chiang, K.; Khedkar, P.; Marcelle, K.; Schutten, M.:** Industrial Applications of Fuzzy Logic at General Electric. Proceedings of the IEEE, 83(3) (1995) 450–465
17. **Bonissone, P. P.; Khedkar, P.; Chen, Y.:** Genetic algorithms for automated tuning of fuzzy controllers: A transportation application. In: Fifth IEEE International Conference on Fuzzy Systems, New Orleans 1996
18. **Bonissone, P. P.:** Summarizing and Propagating Uncertain Information with Triangular Norms. Int. J. Approx. Reas 1(1) (1987) 71–101
19. **Bonissone, P. P.:** Approximate Reasoning Systems: A Personal Perspective. In: Proceedings of the American Association for Artificial Intelligence (AAAI-91), pages 923–929, Anaheim, CA. AAAI 1991
20. **Bouchon-Meunier, B.; Yager, R.; Zadeh, L.:** Fuzzy Logic and Soft Computing. World Scientific, Singapore 1995
21. **Burkhardt, D.; Bonissone, P. P.:** Automated Fuzzy Knowledge Base Generation and Tuning. In: Proc. First IEEE Int. Conf. on Fuzzy Systems (FUZZ-IEEE92), pages 179–188, San Diego, CA, USA IEEE 1992
22. **Carpenter, A.; Grossberg, S.:** A Massively parallel architecture for a self-organizing neural pattern recognition machine. Comp. Vision Graph. Image Proc. 37 (1983) 54–115
23. **Carpenter, A.; Grossberg, S.:** ART 2: Self-organization of stable category recognition codes for analog output patterns. Appl. Optics 26(23) (1987) 4919–4930
24. **Carpenter, A.; Grossberg, S.:** ART 3 Hierarchical Search: Chemical Transmitter in Self-organising Pattern Recognition Architectures. In: Int. Joint Conf. on Neural Networks (IJCNN'90), pages 30–33, Washington, DC 1990
25. **Cordon, O.; Herrera, H.; Lozano, M.:** A classified review on the combination fuzzy logic-genetic algorithms bibliography. Technical report 95129, url:http:/ /decsai.ugr.s/herrera/figa.html, Department of Computer Science and AI, Universidad de Granada, Granada, Spain 1995
26. **DeFinetti, B.:** La prévision: ses lois logiques, see sources subjectives. Annales de l'Institut H. Poincaré, 7 (1937) 1–68
27. **Dempster, A. P.:** Upper and lower probabilities induced by a multivalued mapping. Ann. Math. Stat. 38 (1967) 325–339
28. **Di Nola, A.; Gerla, G.:** Non-standard fuzzy sets. Fuzzy Sets and Systems 18 (1986) 173–181
29. **Dubois, D.; Prade, H.:** Criteria Aggregation and Ranking of Alternatives in the Framework of Fuzzy Set Theory. In: Zimmerman, H. J.; Zadeh, L. A.; Gaines, B. R. (eds.). TIMS/Studies in the Management Science, Vol. 20, pages 209–240. Elsevier Science Publishers 1984
30. **Dubois, D.; Prade, H.:** Possibility theory as a basis for preference propagation in automated reasoning. In: First IEEE International Conference on Fuzzy Systems, pages 821–832, San Diego, CA 1992
31. **Dubois, D.; Prade, H.:** Fuzzy Sets and Probability: Misunderstandings, Bridges and Gaps. In: Proceedings of the Second IEEE International Conference on Fuzzy Systems, pages 1059–1068, San Francisco, CA. IEEE 1993
32. **Duda, R. O.; Hart, P.E.; Nilsson, N. J.:** Subjective Bayesian methods for rule-based inference systems. In: Proc. AFIPS 46, pages 1075–1082, New York. AFIPS Press 1976
33. **Fagin, R.; Halpern, J. H.:** Uncertainty, belief, and probability. In: Proc. 11th Intern. Joint Conf. on Artificial Intelligence, pages 1161–1167, Detroit, MI 1989
34. **Fahlman, S.:** Faster-learning variations on backpropagation: An empirical study. In: Touretzky, D.; Hinton, G.; Sejnowski, T. (eds.): Proc. of the Connectionist Model Summer School. Morgan Kaufmann, San Mateo, CA 1988
35. **Fogel, L. J.; Owens, A. J.; Walsh, M. J.:** Artificial Intelligence through Simulated Evolution. John Wiley, New York, NY 1988
36. **Fogel, L. J.:** Autonomous Automata. Industrial Research 4 (1962) 14–19
37. **Fogel, D. B.: Evolutionary Computation.** IEEE Press, New York, NY 1995
38. **Giles, R.:** Lukssiewicz logic and fuzzy set theory. In: Mamdani, E. H. and Gaines, B. (eds.): Fuzzy Reasoning and its applications, pages 117–131. Academic Press, London 1981
39. **Goldberg, D. E.:** Genetic Algorithms. Addison Wesly, Reading, MA 1978
40. **Grefenstette, J.:** Optimization of Control Parameters for Genetics Algorithms. IEEE Trans. Syst. Man Cyber. 16 (1986) 122–128
41. **Halpern, J. Y.; Fagin, R.:** Two views of belief: Belief as generalized probability and belief as evidence. In: Proc. Eight National Conference on Artificial Intelligence, pages 112–119, Boston, MA 1990
42. **Henrion, M.:** Practical Issues in Constructing a Bayes' Belief Network. In: Kanal, L.; Levitt, T. S.; Lemmer, J. F. (eds.): Uncertainty in Artificial Intelligence 3, pages 161–173. North-Holland 1989
43. **Herrera, F.; Lozano, M.:** Adaptive Genetic Algorithms Based on Fuzzy Techniques. In: Proc. of IPMU'96, pages 775–780, Granada, Spain 1996
44. **Herrera, F.; Lozano, M.; Verdegay, J. L.:** Tackling fuzzy genetic algorithms. In: Winter, G.; Periaux, J.; Galan, M.; Cuesta, P. (eds.): Genetic algorithms in Engineering and Computer Science, pages 167–189. Wyley and Sons 1995
45. **Herrera, F.; Lozano, M.; Verdegay, L.:** Tuning fuzzy logic control by genetic algorithms. Int. Journal Approximate Reasoning (IJAR) 12(3/4) (1995) 299–315
46. **Holland, J. H.:** Adaptation in Natural and Artificial Systems. MIT Press, Cambridge, MA 1975
47. **Hopfield, J.:** Neural Networks and Physical Systems with Emergent Collective Computational Abilities. Proc. Natl. Acad. Sci. 79, (1982) 2554–2558
48. **Hornick, K.; Stinchcombe, M.; White, H.:** Multilayer feedforward networks are universal approximators. Neural Networks 2 (1989) 359–366
49. **Horvitz, E. J.; Suermondt, H. J.; Cooper, G.F.:** Bounded conditioning flexible inference for decisions under scarce resources. In: Proceedings of the Fifth Workshop on Uncertainty in Artificial Intelligence, pages 182–193 1989
50. **Howard, R. A.; Matheson, J. E.:** Influence diagrams. In: Howard, R. A.; Matheson, J. E. (eds.): The Principles and Applications of Decision Analysis, volume 2, pages 719–762. Strategic Decisions Group, Menlo Park, California 1984
51. **Jacobs, R. A.:** Increased rates of convergence through learning rate adaptation. Neural Networks 1 (1988) 295–307
52. **Jang, R.; Sun, J.; C.-T; Darken, C.:** Functional Equivalence Between Radial Basis Function Networks and Fuzzy Inference Systems. IEEE Transactions on Neural Networks 4(1) (1993) 156–159
53. **Jang, J. S. R.:** ANFIS: Adaptive-network-based-fuzzy-inference-system. IEEE Trans. Syst. Man Cyber. 23(3) (1993) 665–685
54. **Karr, C. L.:** Design of an adaptive fuzzy logic controller using genetic algorithms. In: Proc. Int. Conf. on Genectic Algorithms (ICGA'91), pages 450–456, San Diego, CA., USA 1991

55. **Karr, C. L.:** Genetic algorithms for fuzzy controllers. AI Expert, 6(2) (1991) 27–33
56. **Karr, C. L.:** Fuzzy control of ph using genetic algorithms. IEEE Transactions on Fuzzy Systems, 1 (1993) 46–53
57. **Kawamura, A.; Watanabe, N.; Okada, H.: Asakawa, K.:** A Prototype of Neuro-Fuzzy Cooperation Systems. In: First IEEE International Conference on Fuzzy Systems, pages 1275–1282, San Diego, CA 1992
58. **Khedkar, P. S.:** Learning as Adaptive Interpolation in Neural Fuzzy Systems. PhD thesis, University of California, Berkeley 1993
59. **Kickert, W. J. M.; Mamdani, E. H.:** Analysis of a fuzzy logic controller. Fuzzy Set and Systems, 12 (1978) 29–44
60. **Kim, J. H.; Pearl, J.:** A computational model for causal and diagnostic reasoning in inference engines. In: Proc. 8th. Intern. Joint Conf. on Artificial Intelligence, pages 190–193, Karlsruhe, Germany 1983
61. **Kinzel, J.; Klawoon, F.; Kruse, R.** Modifications of genetic algorithms for designing and optimizing fuzzy controllers. In: Proc. First IEEE Conf. on Evolutionary Computing (ICEC'94), pages 28–33, Orlando, FL., USA 1994
62. **Kitano, H.:** Empirical Studies on the Speed of Convergence of Neural Networks Training Using Genetic Algorithms. In: Eigth National Conference in Artificial Intelligence (AAAI'90), pages 789–796, Boston, MA 1990
63. **Klement, P.:** Operations on Fuzzy Sets and Fuzzy Numbers Related to Triangular Norms. In: Proceedings of the 11th International Symposium on Multiple Valued Logic, pages 218–225, Oklahoma City, OK. IEEE Computer Society Press 1981
64. **Klir, G.; Folger, T. A.:** Fuzzy Sets, Uncertainty, and Information. Prentice Hall, Englewood Cliffs, New Jersey 1988
65. **Kohonen, T.:** Self-Organized Formation of Topologically Correct Feature Maps. Biolog. Cybernetics 43 (1982) 59–69
66. **Koza, J.:** Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, MA 1992
67. **Lauritzen, S. L.; Spiegelhalter, D.:** Local computations with probabilities on graphical structures and their application to expert systems. J. Roy. Stat. Soc. Ser B 50 (1988)
68. **Lee, S. C.; Lee, E. T.:** Fuzzy Sets and Neural Networks. J. Cybernetics 4(2) (1974) 83–103
69. **Lee, M. A.; Tagaki, H.:** Dynamic control of genetic algorithm using fuzzy logic techniques. In: Forrest, S. (ed.). Proceedings of the fifth International Conference on Genetic Algorithms, pages 76–83. Morgan Kaufmann 1993
70. **Lee, M.; Takagi, H.:** Integrating design stages of fuzzy systems using genetic algorithms. In: Proc. Second IEEE Conf. on Fuzzy Systems (FUZZ-IEEE'93), pages 612–617, San Francisco, CA., USA 1993
71. **Lee, C. C.:** Fuzzy Logic in Control Systems: Fuzzy Logic Controller - Part I, II, IEEE Trans. Syst. Man Cyber. 20(2) (1990) 404–435
72. **Lowrance, J. D.; Garvey, T. D.; Strat, T. M.:** A Framework for Evidential-Reasoning Systems. In: Proc. 5th National Conference on Artificial Intelligence, pages 896–903, Menlo Park, CA. AAAI 1986
73. **Lukasiewicz, H.:** Many-valued systems for propositional logic. In: McCall, S. (ed.): Polish Logic: 1920–1939. Oxford University Press 1967
74. **Mamdani, E. H.; Assilian, S.:** An experiment in linguistic synthesis with a fuzzy logic controller. Int. J. Man Machine Studies 7(1) (1975) 1–13
75. **Maniezzo, V.:** Genetic Evolution of the Topology and Weight Distribution of Neural Networks. IEEE Transactions on Neural Networks 1994
76. **McInerney, M.; Dhawan, A. P.:** Use of Genetic Algorithms with Backpropagation in Traning of Feedforward Neural Networks. In: IEEE International Conference on Neural Networks, pages 203–208, San Francisco, CA 1993
77. **Michslewicz, Z.:** Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag, New York, NY 1994
78. **Minsky, M. L.; Papert, S.:** Perceptrons. MIT Press, Cambridge, MA 1969
79. **Mizumoto, M.; Zimmerman, H.:** Comparison of Various Fuzzy Reasoning Methods. Fuzzy Sets and Systems 8 (1982) 253–283
80. **Mizumoto, M.:** Improvements Methods of Fuzzy Controls. In Proceedings of the Third International Fuzzy Systems Association, pages 60–62. IFSA 1989
81. **Moller, M.:** A scaled conjugate gradient algorithm for fast supervised learning. PhD thesis, Comp. Science Dept., University of Aarhus, Denmark 1990
82. **Montana, D. J.; Davis, L.:** Traning Feedforward Neural Networks Using Genetic Algorithms. In: Eleventh International Joint Conference on Artificial Intelligence, pages 762–767, Detroit, ML 1989
83. **Moody, J.; Darken, C.:** Fast learning in networks of locally tuned processing units. Neural Comput. 1 (1989) 281–294
84. **Mundici, D.:** Averaging the Truth-Value in Lukaasiewicz Logic. Studia Logica 55 (1995) 113–127
85. **Patel, M. J.; Maniezzo, V.:** NN's and GA's Evolving co-operative Behavior in adaptive learning agents. In: First IEEE Conference on Evolutionary Computation, pages 290–295, 1994
86. **Pearl, J.:** Reverend Bayes on Inference Engines: a Distributed Hierarchical Approach. In: Proceedings Second National Conference on Artificial Intelligence, pages 133–136. AAAI 1982
87. **Pearl, J.:** Fusion, propagation, and structuring in belief networks. Art. Intelligence 29 (1986) 241–288
88. **Pearl, J.:** On evidential reasoning on a hierarchy of hypothesis. Art. Intelligence 28 (1988) 9–15
89. **Pearl, J.:** Evidential reasoning under uncertainty. In: Shrobe, Howard E. (ed.) Exploring Artificial Intelligence, pages 381–418. Morgan Kaufmann, , San Mateo, CA 1988
90. **Pearl, J.:** Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan-Kaufmann, San Mateo, California 1988
91. **Rechenberg, I.:** Cybernetic Solution Path of an Experimental Problem. Royal Aircraft Establishment, Library Translation No. 1122, 1965
92. **Renders, Y. M.; Bersini, H.:** Hybridizing Genetic Algorithms with Hill Climbing Methods for Global Optimization: Two Possible Ways. In: First IEEE Conference on Evolutionary Computation, pages 312–317, 1994
93. **Renders, J. M.; Flasse, S. P.:** Hybrid Methods Using Genetic Algorithms for Global Optimization. IEEE Trans. Syst. Man Cyber. 26(2) (1976) 243–258
94. **Romeo, F.; Sangiovanni-Vincentelli, A.:** Probabilistic Hill-Climbing Algorithms: Properties and Applications. Computer Science Press, Chapel Hill, NC 1985
95. **Rosenbaltt, F.:** Two theorems of statistical separability in the perceptron. In: Mechanization of Thought Processes, pages 421–456, Symposium held at the National Physical Laboratory, HM Stationary Office, London 1959
96. **Ruspini, E. H.:** Epistemic logic, probability, and the calculus of evidence. In: Proc. Tenth Intern. Joint Conf. on Artificial Intelligence, Milan, Italy 1987
97. **Ruspini, E. H.:** On the Semantics of Fuzzy Logic. Technical Note 475, Artificial Intelligence Center, SRI International, Menlo Park, California 1989
98. **Ruspini, E. H.:** Possibility as similarity: The semantics of fussy logic. In: Proc. of the Sixth Conference on Uncertainty in Artificial Intelligence, pages 281–289, Cambridge, MA 1990
99. **Schachter, R. D.:** Evaluating influence diagrams. Op. Res. 34 (1986) 871–882
100. **Schaffer, J. D.; Whitley, D.; Eshelman, L. J.:** Combinations of Genetic Algorithms and Neural Networks: A Survey of the State of the Art. In: International Workshop on Combinations of Genetic Algorithms and Neural Networks (COGANN-92), pages 1–37, 1992
101. **Schwefel, H.-P.:** Kybernetische Evolution als Strategie der Experimentellen Forschung in der Strömungstechnik. PhD thesis, Diploma Thesis Technical University of Berlin, Germany 1965
102. **Schweizer, B.; Sklar, A.:** Associative functions and abstract semigroups. Publ. Math Debreceno 10 (1963) 69–81
103. **Schweizer, B.; Sklar, A.:** Probabilistic Metric Spaces. North Holland, New York 1983

104. Shafer, G.: A Mathematical Theory of Evidence. Princeton University Press Princeton, NJ 1976

105. Shafer, G.: Perspectives on the theory and practice of belief functions. Int. J. Approx. Reason. 4(5/6) (1990) 323–362

106. Shortliffe, E. H.; Buchanan, B.: A model of inexact reasoning in medicine. Math Biosci. 23 (1975) 351–379

107. Smets, Ph.: The Degree of Belief in a Fuzzy Set. Inf. Sci. 25 (1981) 1–19

108. Smets, Ph.: Belief functions. In: Smets, Ph.; Mamdani, A.; Dubois, D.; Prade, H. (eds.) Non-Standard Logics for Automated Reasoning. Academic Press, New York 1988

109. Smets, Ph.: The transferable belief model and other interpretations of dempster-shafer's model. In: Bonissone, P. P.; Henrion, M.; Kanal, L. N.; Lermmer, J. F. (eds.) Uncertainty in Artificial Intelligence 6, pages 375–383, North-Holland, Amsterdam 1991

110. Stark, L.; Okajima, M.; Whipple, G.: Computer Pattern Recognition Techniques: Electrocardiographic Diagnosis. Comm. of the ACM 5 (1962) 527–532

111. Stillman, J.: On heuristics for finding loop cutsets in multiply-connected belief networks. In: Bonissone, P. P.; Henrion, M.; Kanal, L. N.; Lemmer, J. F. (eds.): Uncertainty in Artificial Intelligence 6, pages 233–243. North-Holland, Amsterdam 1991

112. Suermondt, J.; Copper, G.; Heckerman, D.: A combination of cutset conditioning with clique-tree propagation in the pathfinder system. In: Bonissone, P. P.; Henrion, M.; Kanal, L. N.; Lemmer, J. F. (eds.): Uncertainty in Artificial Intelligence 6, pages 245–253. North-Holland, Amsterdam 1991

113. Surmann, H.; Kanstein, A.; Goser, K.: Self-organising and genetic algorithms for an automatic design of fuzzy control and decision systems. In: Proc. EUFIT'93, pages 1097–1104, Aachen, Germany 1993

114. Takagi, T.; Sugeno, M.: Fuzzy Identification of Systems and Its Applications to Modeling and Control. IEEE Trans. Syst. Man Cyber. 15(1) (1985) 116–132

115. Takagi, H.; Fusion Technology of Fuzzy Theory and Neural Networks - Survey and Future Directions. In: IIZUKA'90: International Conference on Fuzzy Logic and Neural Networks, pages 13–26, Iizuka, Japan

116. Tollensere, T.: SuperSAB: fast adaptive backpropagation with good scaling properties. Neural Networks 3 (1990) 561–573

117. Trillas, E.; Valverde, L.: On Mode and Implication in Approximate Reasoning. In: Gupta, M.; Kandel, A.; Bandler, W.; Kiszka, J. (eds.): Approximate Reasoning in Expert Systems, pages 157–166. Elsevier Amsterdam, The Netherlands 1985

118. Werbos, P.: Beyond Regression: New Tools for Predictions and Analysis in the Behavioral Science. PhD thesis, Harvard University, Cambridge, MA 1974

119. Widrow, B.; Hoff, M. E.: Adaptive switching circuits. In: IRE Western Electric Show and Convention Record, Part 4, pages 96–104, 1960

120. Widrow, B.: ADALINE: An Engineering Model for Neurocomputing. In: IIZUKA'90: International Conference on Fuzzy Logic and Neural Networks, pages 5–9, Iizuka, Japan, 1990

121. Yao, X.: A review of evolutionary artificial neural networks. Technical report, Common Wealth Scientific and Industrial Research Organization, Victoria, Australia, 1992

122. Zadeh, L. A.: Fuzzy sets. Inf. Control 8 (1965) 338–353

123. Zadeh, L. A.: Probability Measures of Fuzzy Events. J. Math. Anal. Appl. 10 (1968) 421–427

124. Zadeh, L. A.: Outline of a new approach to the analysis of complex systems and decision processes. IEEE Trans. Syst. Man Cyber. SMC-3 (1973) 28–44

125. Zadeh, L. A.: Fuzzy sets as a basis for a theory of possibility. Fuzzy Sets and Systems 1 (1978) 3–28

126. Zadeh, L. A.: A theory of approximate reasoning. In: Hayes, P.; Michie, D.; Mikulich, L. I. (eds.): Machine Intelligence, pages 149–194. Halstead Press, New York 1979

127. Zadeh, L. A.: 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, Iisuka, Japan 1994

128. Zheng, Li: A Practical Guide to Tune Proportional and Integral (PI) Like Fuzzy Controllers. In: Proceedings of the IEEE International Conference on Fuzzy Systems, pages 633–640. IEEE 1992