Hybrid Soft Computing Systems: Where Are We Going?

Piero P. Bonissone[1]

Abstract.

Soft computing is an association of computing methodologies that includes fuzzy logic, neuro-computing, evolutionary computing, and probabilistic computing. After a brief overview of Soft Computing components, we will analyze some of its most synergistic combinations. We will emphasize the development of smart algorithm-controllers, such as the use of fuzzy logic to control the parameters of evolutionary computing and, conversely, the application of evolutionary algorithms to tune fuzzy controllers. We will focus on three real-world applications of soft computing that leverage the synergism created by hybrid systems.

1 SOFT COMPUTING OVERVIEW

Soft computing (SC) is a term originally coined by Zadeh to denote systems that "… exploit the tolerance for imprecision, uncertainty, and partial truth to achieve tractability, robustness, low solution cost, and better rapport with reality" [1]. Traditionally SC has been comprised by four technical disciplines. The first two, probabilistic reasoning (PR) and fuzzy logic (FL) reasoning systems, are based on ***knowledge-driven*** reasoning. The other two technical disciplines, neuro computing (NC) and evolutionary computing (EC), are ***data-driven*** search and optimization approaches [2]. Although we have not reached a consensus regarding the scope of SC or the nature of this association [3], the emergence of this new discipline is undeniable [4].

This paper is the reduced version of a much more extensive coverage of this topic, which can be found in [5].

2 SC COMPONENTS AND TAXONOMY

2.1 Fuzzy Computing

The treatment of imprecision and vagueness can be traced back to the work of Post, Kleene, and Lukasiewicz, multiple-valued logicians who in the early 1930's proposed the use of three-valued logic systems (later followed by infinite-valued logic) to represent *undetermined*, *unknown*, or other possible intermediate truth-values between the classical Boolean *true* and *false* values [6]. In 1937, the philosopher Max Black suggested the use of a *consistency profile* to represent vague concepts [7]. While vagueness relates to ambiguity, fuzziness addresses the lack of sharp set-boundaries. It was not until 1965, when Zadeh proposed a complete theory of fuzzy sets (and its isomorphic fuzzy logic), that we were able to represent and manipulate ill-defined concepts [8].

[1] GE Corporate Research and Development, One Research Circle, Niskayuna, NY 12309, USA. email: bonissone@crd.ge.com

In a narrow sense, fuzzy logic could be considered a fuzzification of Lukasiewicz Aleph-1 multiple-valued logic [9]. In the broader sense, however, this narrow interpretation represents only one of FL's four facets [10]. More specifically, FL has a *logical* facet, derived from its multiple-valued logic genealogy; a *set-theoretic* facet, stemming from the representation of sets with ill-defined boundaries; a *relational* facet, focused on the representation and use of fuzzy relations; and an *epistemic* facet, covering the use of FL to fuzzy knowledge based systems and data bases. A comprehensive review of fuzzy logic and fuzzy computing can be found in [11].

Fuzzy logic gives us a language, with syntax and local semantics, in which we can translate qualitative knowledge about the problem to be solved. In particular, FL allows us to use linguistic variables to model dynamic systems. These variables take fuzzy values that are characterized by a label (a sentence generated from the syntax) and a meaning (a membership function determined by a local semantic procedure). The meaning of a linguistic variable may be interpreted as an elastic constraint on its value. These constraints are propagated by fuzzy inference operations, based on the *generalized modus-ponens*. This reasoning mechanism, with its interpolation properties, gives FL a robustness with respect to variations in the system's parameters, disturbances, etc., which is one of FL's main characteristics [12].

2.2 Probabilistic Computing

Rather than retracing the history of probability, we will focus on the development of probabilistic computing (PC) and illustrate the way it complements fuzzy computing. As depicted in Figure 1, we can divide probabilistic computing into two classes: single-valued and interval-valued systems.

Bayesian belief networks (BBNs), based on the original work of Bayes [13], are a typical example of single-valued probabilistic reasoning systems. They started with approximate methods used in first-generation expert systems, such as MYCIN's confirmation theory [14] and PROSPECTOR's modified Bayesian rule [15], and evolved into formal methods for propagating probability values over networks [16-17]. In general, probabilistic reasoning systems have exponential complexity, when we need to compute the joint probability distributions for *all* the variables used in a model. Before the advent of BBNs, it was customary to avoid such computational problems by making unrealistic, global assumptions of conditional independence. By using BBNs we can decrease this complexity by encoding domain knowledge as structural information: the presence or lack of conditional dependency between two variables is indicated by the presence or lack of a link connecting the nodes representing such variables in the network topology. For specialized topologies (trees, poly-trees, directed acyclic graphs), efficient propagation algorithms have been proposed by Kim and Pearl [18]. However, the complexity of

multiple–connected BBNs is still exponential in the number of nodes of the largest sub-graph. When a graph decomposition is not possible, we resort to approximate methods, such as clustering and bounding conditioning, and simulation techniques, such as logic samplings and Markov simulations.

Dempster-Shafer (DS) systems are a typical example of interval-valued probabilistic reasoning systems. They provide lower and upper probability bounds instead of a single value as in most BBN cases. The DS theory was developed independently by Dempster [19] and Shafer [20]. Dempster proposed a calculus for dealing with interval-valued probabilities induced by multiple-valued mappings. Shafer, on the other hand, started from an axiomatic approach and defined a calculus of belief functions. His purpose was to compute the credibility (degree of belief) of statements made by different sources, taking into account the sources' reliability. Although they started from different semantics, both calculi were identical.

Probabilistic computing provides a way to evaluate the outcome of systems affected by randomness (or other types of probabilistic uncertainty). PC's basic inferential mechanism - conditioning - allows us to modify previous estimates of the system's outcome based on new evidence.

2.2.1 Comparing Probabilistic and Fuzzy Computing.

In this brief review of fuzzy and probabilistic computing, we would like to emphasize that randomness and fuzziness capture two different types of uncertainty. In randomness, the uncertainty is derived from the non-deterministic membership of a point from a sample space (describing the set of possible values for the random variable), into a well-defined region of that space (describing the event). A probability value describes the tendency or frequency with which the random variable takes values inside the region. In fuzziness, the uncertainty is derived from the deterministic but *partial* membership of a point (from a reference space) into an imprecisely defined region of that space. The region is represented by a fuzzy set. The characteristic function of the fuzzy set maps every point from such space into the real-valued interval [0,1], instead of the set {0,1}. A partial membership value does not represent a frequency. Rather, it describes the degree to which that particular element of the universe of discourse satisfies the property that characterizes the fuzzy set. In 1968, Zadeh noted the complementary nature of these two concepts, when he introduced the probability measure of a fuzzy event [21]. In 1981, Smets extended the theory of belief functions to fuzzy sets by defining the belief of a fuzzy event [22]. These are the first two cases of hybrid systems illustrated in Figure 1.

2.3 Neural Computing

The genealogy of neural networks (NN) could be traced back to 1943, when McCulloch and Pitts showed that a network of binary decision units (BDNs) could implement any logical function [23]. Building upon this concept, Rosenblatt proposed a one-layer feedforward network, called a *perceptron*, and demonstrated that it could be trained to classify patterns [24-26]. Minsky and Papert [27] proved that single-layer perceptrons could only provide linear partitions of the decision space. As such they were not capable of separating nonlinear or non-convex regions. This caused the NN community to focus its efforts on the development of multilayer NNs that could overcome these limitations. The training of these networks, however, was still problematic. Finally, the introduction of backpropagation (BP), independently developed by Werbos [28], Parker [29], and LeCun [30], provided a sound theoretical way to train multi-layered, feed-forward networks with nonlinear activation functions. In 1989, Hornik et al. proved that a three-layer NN (with one input layer, one hidden layer of squashing units, and one output layer of linear units) was a universal functional approximator [31].

Topologically, NNs are divided into *feedforward* and *recurrent* networks. The feedforward networks include single- and multiple-layer *perceptrons*, as well as radial basis functions (RBF) networks [32]. The recurrent networks cover competitive networks, self-organizing maps (SOMs) [33], Hopfield nets [34], and adaptive resonance theory (ART) models [35]. While feed-forward NNs are used in supervised mode, recurrent NNs are typically geared toward unsupervised learning, associative memory, and self-organization. In the context of this paper, we will only consider feed-forward NNs. Given the functional equivalence already proven between RBF and fuzzy systems [36] we will further limit our discussion to multi-layer feed-forward networks. A comprehensive current review of neuro-computing can be found in [37].

Feedforward multilayer NNs are computational structures that can be trained to learn patterns from examples. They are composed of a network of processing units or neurons. Each neuron performs a 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). By using a training set that samples the relation between inputs and outputs, and a learning method that trains their weight vector to minimize a quadratic error function, neural networks offer the capabilities of a supervised learning algorithm that performs fine-granule local optimization.

2.4 Evolutionary Computing

Evolutionary computing (EC) algorithms exhibit an *adaptive* behavior that allows them to handle non-linear, high dimensional problems without requiring differentiability or explicit knowledge of the problem structure. As a result, these algorithms are very robust to time-varying behavior, even though they may exhibit low speed of convergence. EC covers many important families of stochastic algorithms, including *evolutionary strategies* (ES), proposed by Rechenberg [38] and Schwefel [39], *evolutionary programming* (EP), introduced by Fogel [40-41], and *genetic algorithms* (GAs), based on the work of Fraser [42], Bremermann [43], Reed et al. [44], and Holland [45-47], which contain as a subset *genetic programming* (GP), introduced by Koza [48].

The history of EC is too complex to be completely summarized in a few paragraphs. It could be traced back to Friedberg [49], who studied the evolution of a learning machine capable of computing a given input-output function; Fraser [42] and Bremermann [43], who investigated some concepts of genetic algorithms using a binary encoding of the genotype; Barricelli [50], who performed some numerical simulation of evolutionary processes; and Reed et al. [44], who explored similar concepts in a simplified poker game simulation. The interested reader is referred to [51] for a comprehensive overview of evolutionary computing and to [52] for an encyclopedic treatment of the same subject. A collection of selected papers illustrating the history of EC can be found in [53].

As noted by Fogel [51], ES, EP, and GAs share many common traits: "…Each maintains a population of trial solutions, imposes random changes to those solutions, and incorporates 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."

Finally, we would like to remark that EC components have increasingly shared their typical traits: ES have added recombination operators similar to GAs, while GAs have been extended by the use of real-number-encoded chromosomes, adaptive mutation rates, and additive mutation operators.

2.5 Soft Computing Taxonomy

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 that describe the state or behavior of the physical system.

When we solve real-world problems, we realize that such systems are typically ill-defined, difficult to model, and possess large solution spaces. In these cases, precise models are impractical, too expensive, or non-existent. Our solution must be generated by leveraging two kinds of resources: *problem domain knowledge* of the process or product and *field data* that characterize the behavior of the system. The relevant available domain knowledge is typically a combination of first principles and 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 may be incomplete and noisy.

We can observe from Figure 1 that the two main approaches in soft computing are *knowledge-driven* reasoning systems (such as probabilistic and fuzzy computing) and *data-driven* search and optimization approaches (such as neuro and evolutionary computing). This taxonomy, however, is soft in nature, given the existence of many hybrid systems that span across more than one field.

3 SOFT COMPUTING SOLUTIONS

3.1 Alternative Approaches to SC

The alternative approaches to SC are the traditional knowledge-driven reasoning systems and the data-driven systems. The first class of approaches are exemplified by first-principle-derived models (based on differential or difference equations), by first-principle-qualitative models (based on symbolic, qualitative calculi [54-55], by classical Boolean systems, such as theorem provers (based on unification and resolution mechanisms), or by expert systems embodying empirical or experiential knowledge. All these approaches are characterized by the encoding of problem domain knowledge into a model that tries to replicate the system's behavior. The second class of approaches are the regression models and crisp clustering techniques that attempt to derive models from any information available from (or usually buried in) the data.

Knowledge-driven systems, however, have limitations, as their underlying knowledge is usually incomplete. Sometimes, these

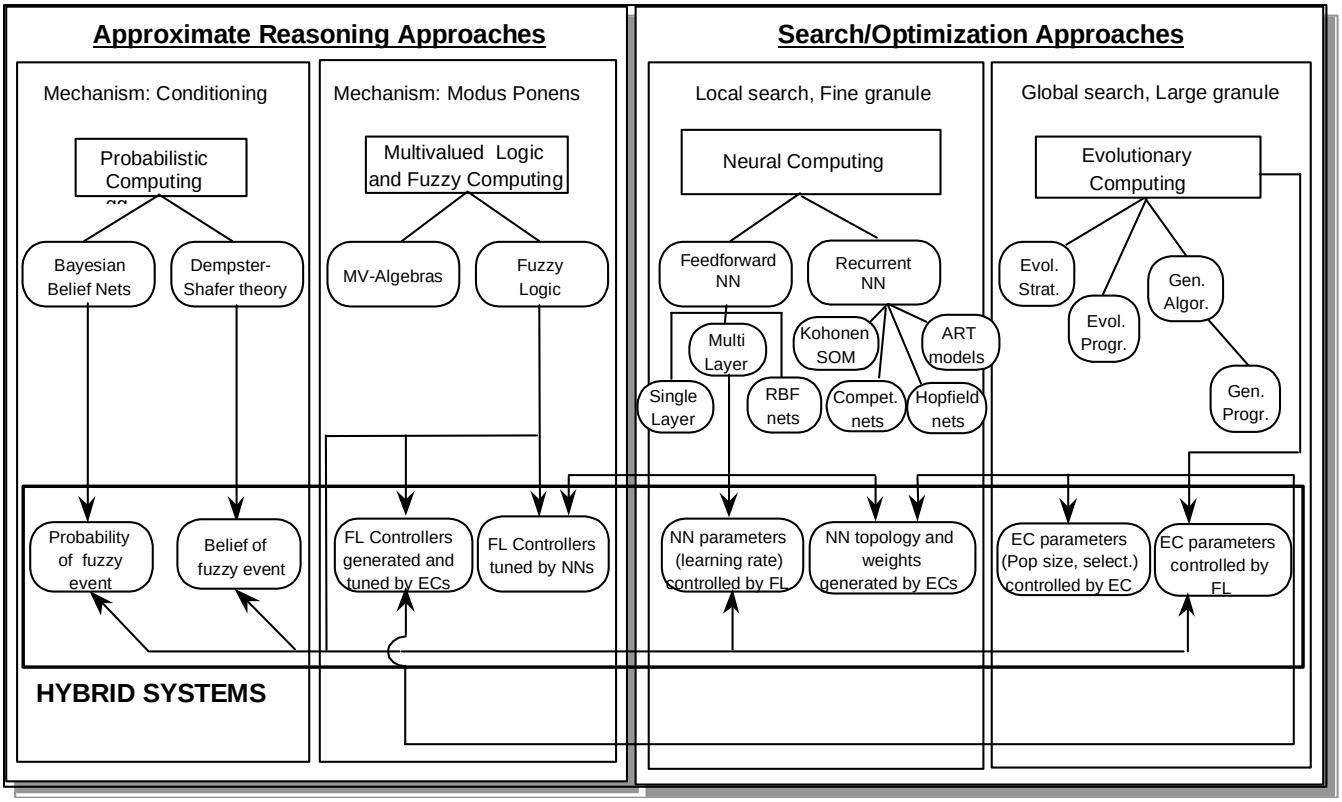

Figure 1: Soft Computing Components and Hybrid Systems

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

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

The fundamental problem of these classical approaches lies in representing and integrating uncertain, imprecise knowledge in data-driven methods or in making use of somewhat unreliable data in a knowledge-driven approach.

3.2 Soft Computing Solutions

Although it would be presumptuous to claim that soft computing *solves* this problem, it is reasonable to affirm 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 former is used to determine the order of the differential equations and the latter determines its coefficients. Hence, in this traditional approach we have ***model = structure + parameters (+ search)***. This equation does not change with the advent of soft computing. However, we now have a much richer repertoire to represent the structure, to tune the parameters, and to iterate this process. It is understood that the search method used to find the parameter values is an important and implicit part of the above equation, which needs to be chosen carefully for efficient model construction.

3.3 Example of SC Models

For example, the knowledge base (KB) in a Mamdani- type fuzzy system [56] 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 function 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 a system is the result of interpolating among the outputs of all relevant rules. The inference's outcome is a membership function defined on the output space, which is then aggregated (defuzzified) to produce a crisp output. With this inference mechanism we can define a deterministic mapping between each point in the state space and its 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 system [57] 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 [58], results in a model that is equivalent to radial basis functions [59]. The same model can be translated into a structured network, such as the adaptive neural fuzzy inference systems (ANFIS) proposed by Jang [60]. In ANFIS the ruleset determines the topology of the net (model structure), while dedicated nodes in the corresponding layers of the net (model parameters) define the termsets and the polynomial coefficients. Similarly, in the traditional neural networks the topology represents the model structure and the links' weights represent the model parameters.

While NNs and structured nets use local search methods, such as backpropagation, to tune their parameters, it is possible to use evolutionary computation based global search methods to achieve the same parametric tuning or to postulate new structures. An extensive coverage of these approaches can be found in [11, 61-62].

The main reason for soft computing popularity is the *synergy* derived from its components. SC's main characteristic is its intrinsic capability to create *hybrid systems* that are based on a (loose or tight) integration of these technologies. This integration provide us with complementary reasoning and searching methods that allows us to combine domain knowledge and empirical data to develop flexible computing tools and solve complex problems.

We will briefly analyze some of the most synergistic combinations of soft computing technologies, with an emphasis on the development of smart algorithm-controllers, such as the use of FL to control EC parameters. We will also discuss the application of EC to tune FL controllers; and the implementation of FL controllers as NNs tuned by backpropagation-type algorithms. We will focus on three real-world applications of SC that leverage the synergism created by hybrid systems.

4 HYBRID SOFT COMPUTING APPLICATIONS

4.1 EC controlled by FL: An Agile Manufacturing Application.

Fuzzy logic enables us to easily translate 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 [16]. 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 can been used to monitor the performance of NNs or GAs and modify their control parameters.

In the past, the selection of GA parameters was often left to the intuition and experience of the GA user. Although several techniques for the selection of GA parameters have been proposed in the literature [17-18], these parameters were computed off-line and kept static during the algorithm's evolution. With fuzzy logic we can translate and improve heuristic rules to provide a dynamic control of EC resources (population size, selection pressure, and probabilities of crossover and mutation). By managing these resources during their transition from exploration (global search in

the solution space) to exploitation (localized search in the discovered regions of that space that appear to be promising) we can improve the algorithm's its efficiency and convergence speed [19-21].

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 generic enough to be applicable to other object-level problem domains and if its run-time overhead is offset by the run-time performance improvement of the algorithm.

We illustrate this approach with an example in an agile manufacturing problem [22]. The object-level problem is the specification of a system for optimal design, manufacturing, and supplier planning for printed circuit assembly manufacturing. This problem formulation poses the optimal selection of designs that realize a given functional specification, the selection of parts to realize a design, the selection of suppliers to supply these parts, and the selection of production facilities to manufacture the chosen design, as a global optimization problem where each selection has the potential to affect other selections. The goal is to optimize an aggregate non-linear evaluation function of total cost and total time. The total cost and total time for realizing a printed circuit assembly are highly coupled, non-linear functions dependent on characteristics of a chosen design, characteristics of a chosen manufacturing facility, and parts supply chain characteristics. The nature of the problem and the evaluation function do not lend themselves easily to optimization using traditional techniques such as linear programming. A GA-based optimization is more easily applied to this problem domain, is robust, and simultaneously searches multiple solutions. In this application we used a FC to monitor population diversity and evolution time. Depending on the various evolution stages, the FC modifies the GA's population size and the probability of mutation to improve the solution quality (measured as the standard deviation of the population of best solutions.)

4.2 FL tuned by GAs: A Transportation Application

The second application describes a dual role: the use of GAs to tune a fuzzy controller in situation where the evaluation of the fitness function is computationally expensive. Many researchers have explored the use of genetic algorithms to tune fuzzy logic controllers. Karr, one of the pioneers in this quest, used GAs to modify the membership functions in the termsets of the variables used by the FCs [23]. In the transportation application chosen to illustrate this approach, we have followed the tuning order suggested by Zheng for manual tuning [24]. We first 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 [25]. 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.

Specifically we will describe the design and tuning of a fuzzy controller for tracking a fuel-optimal velocity profile for a rail-based transportation system, while enforcing compliance with a prescribed velocity profile, and providing a smooth ride.

This approach exemplifies the synergy of SC technologies. 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 we were able to translate locomotive engineers training procedures into an executable model that exhibited a reasonable performance. However, this performance, was far from optimal, even after manual tuning of the model. By using a genetic algorithm to tune the model's scaling factors and membership functions, we demonstrated that this approach was able to find much better solutions within a reasonable amount of time, under different train handling criteria. In addition, we showed that not all parameters needed to be treated equally, and that sequential optimization could greatly reduce computational effort by tuning the scaling factors first. The scalability of this approach enabled us to tune the controller for each track profile, producing a library of offline-customized controllers. For a more detailed description of this application, see references [5, 72].

4.3 Fusion of Fuzzy Cased-based Reasoning with ANFIS models: A Financial Application

The third application illustrates the use of a fusion model to estimate residential property values for real estate transactions. In the two previous applications we described hybrid systems which were derived from the tight integration of two or more SC technologies. However, in this application we built multiple estimators using different technologies on the same database. The estimators were based on location (LOCVAL), comparable properties (AICOMP), and a generative model (AIGEN). Then we used a loosely integrated hybrid system, based on a fuzzy-rule set, to combine the results of each estimator, rather than the techniques themselves.

The first model was based solely on the location and living area of the properties, as shown in Figure 2. A dollar per square foot measure was constructed for each point in the county, by suitably averaging the observed, filtered historical market values in the vicinity of that point. This locational value estimator (LOCVAL) produces two output values: *Locational_Value* (a $/sq.ft. estimate) and *Deviation_from_prevailing_value.* The local averaging is done by an exponentially decreasing radial basis function with a “space constant” of 0.15-0.2 miles. It can be described as the weighted sum of radial basis functions (all of the same width), each deviation for houses within the area covered and is derived using a situated at the site of a sale within the past 1 year and having an amplitude equal to the sales price. Deviation from prevailing value is the standard similar approach.

In order to use the LOCVAL estimator correctly, the input values (a valid, geocoded address and a living area in squared feet) must be present and accurate for the locational estimator to work correctly. If either is missing, or clearly out-of-range, the estimator will not to make any prediction.

The second model, AICOMP, relied on a case based reasoning (CBR) process similar to the sales comparison approach [73] used by certified appraisers to estimate a residential property's value. The CBR process consists of selecting relevant cases (which would be nearby house sales), adapting them, and aggregating those adapted cases into a single estimate of the property value. Our approach consists of the following four steps [74]:

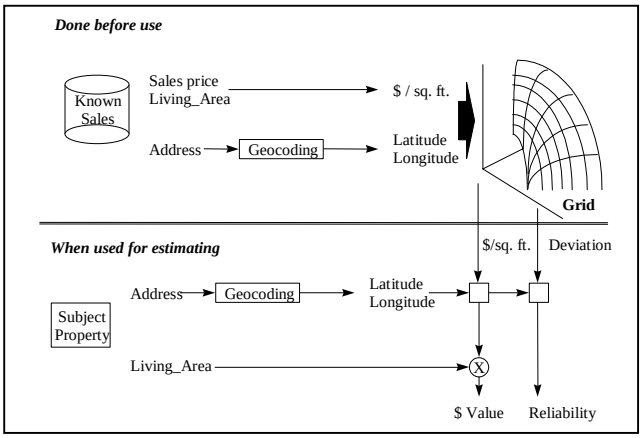

Figure 2: Locational Value method (LOCVAL)

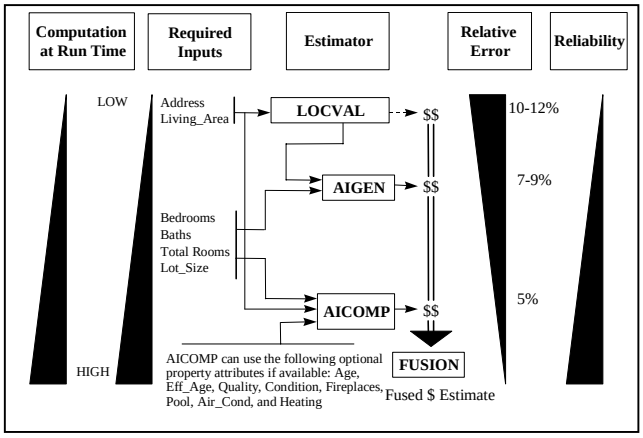

Figure 3: Data comparison of multiple approaches

1. *Retrieving recent sales from a case-base.* Upon entering the subject property attributes, AICOMP retrieves potentially similar comparables from the case-base. This initial selection uses six attributes: address, date of sale, living area, lot area, number of bathrooms, and bedrooms.

2. *Comparing the subject property with the retrieved cases.* The comparables are rated and ranked on a similarity scale to identify the most similar ones to the subject property. This rating is obtained from a weighted aggregation of the decision maker preferences, expressed as fuzzy membership distributions and relations.

3. *Adjusting the sales price of the retrieved cases.* Each property's sales price is adjusted to reflect their differences from the subject property. These adjustments are performed by a rule set that uses additional property attributes, such as construction quality, conditions, pools, fireplaces, etc.

4. *Aggregating the adjusted sales prices of the retrieved cases.* The best four to eight comparables are selected. The adjusted sales price and similarity of the selected properties are combined to produce an estimate of the subject value with an associated reliability value.

The third method, called AIGEN, is a generative AI method in which a fuzzy-neural net, a modified version of ANFIS, is trained by using a subset of cases from the case-base. The resulting run-time system provides an estimate of the subject's value.

Each model produced a property value and an associated reliability value. The latter was a function of the "averageness" or "typicality" of the subject property based on its physical characteristics (such as lot size, living area, total room). These typical values were represented by possibilistic distributions (fuzzy sets). We computed the degree to which each property satisfied each criterion. The overall property value reliability was obtained by considering the conjunction of these constraint satisfactions (i.e., the minimum of the individual reliability values). A more detailed description of this process can be found in [75].

The computation time, required inputs, errors and reliability values for these three methods are shown in Figure 3. The locational value method took the least time and information, but produced the largest error. The CBR approach took the largest time and number of inputs, but produced the lowest error, among the models. However, the fusion of the three models produced the most accurate and reliable results.

5 CONCLUSIONS

Soft computing is having an impact on many industrial and commercial operations, from scheduling to predictive modeling and control. It provides us with 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.

These systems 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 these 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. This is a form of complementary or *tight* hybridization. Apart from this type of hybridization, we also discussed the fusion of estimators – this type of model fusion or *loose* hybridization does not combine features of the methodologies themselves, but only their results. The primary motivation here is to increase reliability rather than to make model construction easier.

We have illustrated this synergy by describing three applications in configuration management, control, and valuation. These applications exemplify 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. This review illustrates the interaction of knowledge and data in SC. To tune *knowledge-derived models* we first translate domain knowledge into an initial structure and parameters and then use global or local data search to tune the parameters. To control or limit search by using prior knowledge we first use global or local search to derive the models (structure + parameters), we embed knowledge in operators to improve global search, and we translate domain knowledge into a controller to manage the solution convergence and quality of the search algorithm.

The payoff of this 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. Another way to say this is that the model needs a structure and parameters, and a search method to discover them, and no single technique should be expected to be the best for all problems.

For example, in our control application, a hierarchical fuzzy controller was used to embody the qualitative knowledge used by locomotive engineers to manually perform the task, but manual tuning of this controller would have been an inferior solution. Fast tuning of this controller was obtained by a global search approach, a genetic algorithm that yielded a robust controller whose parameters did not have to be specialized for each particular initial condition.

Another advantage to the hybridization of techniques is that it is easier to think of alternative solutions to the same problem, as was evidenced by the property estimation application. If there are several possibilities for the structure and the search methods, many more pairings of technologies are possible, and problem solving becomes easier. For instance, AIGEN used a fuzzy model with local gradient search, but one could also use a pure neural network trained by an evolutionary algorithm. Of course, computation time, cost, business needs, and data requirements will then influence the choice of the combination of technologies. 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 non-overlapping 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, as seen in our last application.

The future appears to hold a lot of 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.

REFERENCES

[1] L.A. Zadeh, 'Fuzzy Logic and Soft Computing: Issues, Contentions and Perspectives', in *Proc. of IIZUKA'94: Third Int. Conf. on Fuzzy Logic, Neural Nets and Soft Computing*, 1-2, Iizuka, Japan, 1994.

[2] L.A. Zadeh, '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*, **2**(1), 23-25, (1998).

[3] D. Dubois and H. Prade, 'Soft computing, fuzzy logic, and Artificial Intelligence', *Soft Computing: A Fusion of Foundations, Methodologies and Applications*, **2**(1), 7-11, (1998).

[4] P.P. Bonissone, 'Soft Computing: the Convergence of Emerging Reasoning Technologies', *Soft Computing A Fusion of Foundations, Methodologies and Applications*, **1**(1), 6-18, (1997).

[5] P.P. Bonissone, Y.-T. Chen, K. Goebel, and P.S. Khedkar, 'Hybrid Soft Computing Systems: Industrial and Commercial Applications', *Proceedings of the IEEE*, **87**(9), 1641-1667, (1999).

[6] N. Rescher, *Many-valued Logic*, McGraw-Hill, New York, NY, 1969.

[7] M. Black, 'Vaguenes: an Exercise in Logical Analysis', *Phil.Sci*. **4**, 427-455, (1937).

[8] L.A. Zadeh, 'Fuzzy sets', *Information and Control*, **8**, 338-353, (1965).

[9] J. Lukasiewicz, *Elementy Logiki Matematycznej [Elements of Mathematical Logic]*, Warsaw, Poland: Panstowowe Wydawinctow Naukowe,1929.

[10] L.A. Zadeh, 'Foreword', in *Handbook of Fuzzy Computation*, E.H. Ruspini, P.P. Bonissone, and W. Pedycz, Eds., Bristol, UK: Institute of Physics, 1998.

[11] E.H. Ruspini, P.P. Bonissone, and W. Pedycz, *Handbook of Fuzzy Computation*, Bristol, UK: Institute of Physics, 1998.

[12] Y-M. Pok and J-X. Xu, 'Why is Fuzzy Control Robust', *Proc. Third IEEE Intl. Conf. on Fuzzy Systems (FUZZ-IEEE'94)*, 1018-1022, Orlando, FL, 1994.

[13] T. Bayes, 'An essay towards solving a problem in the doctrine of chances', *Philosophical Trans. 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).

[14] E. Shortliffe and B. Buchanan, 'A Model of Inexact Reasoning in Medicine', *Mathematical Biosciences,*. **23**, 351-379, (1975).

[15] R. Duda, P. Hart, and N. Nilsson, 'Subjective Bayesian Methods for Rule-Based Inference Systems', *Proc. AFIPS*, **45**,1075-1082, New York, NY: AFIPS Press, (1976).

[16] J. Pearl, 'Reverend Bayes on Inference Engines: a Distributed Hierarchical Approach', *Proc. 2nd Natl. Conf. on Artificial Intelligence*, 133-136, Menlo Park, CA: AAAI, 1982.

[17] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, San Mateo, CA: Morgan-Kaufmann, 1988.

[18] J. Kim and J. Pearl, 'A Computational Model for Causal and Diagnostic Reasoning in Inference Engines', in *Proc. Eighth Int. Joint Conf. on Artificial Intelligence*, 190-193, Karlsruhe, Germany, 1983.

[19] A. P. Dempster, 'Upper and lower probabilities induced by a multivalued mapping', *Annals of Mathematical Statistics*, **38**, 325-339, (1967).

[20] G. Shafer, *A Mathematical Theory of Evidence*. Princeton, NJ: Princeton University Press, 1976.

[21] L.A. Zadeh, 'Probability Measures of Fuzzy Events'. *J. Math. Analysis and Appl.*, **10**, 421-427, (1968).

[22] Ph. Smets, 'The Degree of Belief in a Fuzzy Set', *Information Science*, **25**, 1-19, 1981.

[23] W.S. McCulloch and W. Pitts, 'A Logical Calculus of the Ideas Immanent in Nervous Activity', *Bull Math Biophysics*, **5**, 115-133, (1943).

[24] F. Rosenblatt, 'The perceptron, a Perceiving and Recognizing Automaton', Project PARA, Cornell Aeronautical Lab. Rep., no. 85-640-1, Buffalo, NY, 1957.

[25] F. Rosenblatt, 'Two theorems of statistical separability in the perceptron', in *Proc. Mechanization of Thought Processes*, 421-456, Symposium held at the National Physical Laboratory, HM Stationary Office, London, 1959.

[26] F. Rosenblatt, *Principle of Neurodynamics: Perceptron and the theory of Brain Mechanisms*, Washington, DC: Spartan Books, 1962.

[27] M. Minsky and S. Papert, *Perceptrons*, Boston, MA: MIT Press, 1969.

[28] P. Werbos, *Beyond Regression: New Tools for Predictions and Analysis in the Behavioral Science*. Ph.D. thesis, Harvard University, Cambridge, MA, 1974.

[29] D. Parker, 'Learning Logic', Tech. Report TR-47, Center for Computational Research in Economics and Management Science, MIT, Cambridge, MA, 1985.

[30] Y. LeCun, 'Une procedure d'apprentissage pour reseau a seuil symetrique', *Cognitiva*, **85**, 599-604, CESTA, Paris, France, (1985).

[31] K. Hornick, M. Stinchcombe, and H. White, 'Multilayer feedforward networks are universal approximators', *Neural Networks*, **2**, 359-366, (1989).

[32] J. Moody and C. Darken, 'Fast learning in networks of locally tuned processing units', *Neural Computation*, **1**, 281-294, (1989).

[33] T. Kohonen, 'Self-Organized Formation of Topologically Correct Feature Maps', *Biological Cybernetics*, **43**, 59-69, (1982).

[34] J. Hopfield, 'Neural Networks and Physical Systems with Emergent Collective Computational Abilities', *Proc. Acad. Sci.*, **79**, 2554-2558, (1982).

[35] A. Carpenter and S. Grossberg, 'A Massively parallel architecture for a self-organizing neural pattern recognition machine', *Computer, Vision, Graphics, and Image Processing*, **37**, 54-115, (1983).

[36] R. Jang, C-T. J. Sun and C. Darken, 'Functional Equivalence Between Radial Basis Function Networks and Fuzzy Inference Systems', *IEEE Trans. on Neural Networks*, **4**(1), 156-159, (1993).

[37] E. Fiesler, and R. Beale, *Handbook of Neural Computation*, Bristol, UK: Institute of Physics, and New York, NY: Oxford University Press, 1997.

[38] I. Rechenberg, "Cybernetic Solution Path of an Experimental Problem," *Royal Aircraft Establishment*, Library Translation no. 1122, 1965.

[39] H-P. Schwefel, *Kybernetische Evolution als Strategie der Experimentellen Forschung in der Stromungstechnik*, Diploma Thesis Technical University of Berlin, Germany, 1965.

[40] L.J. Fogel, 'Autonomous Automata', *Industrial Research*, **4**, 14-19, (1962).
[41] L.J. Fogel, A.J. Owens, and M.J. Walsh, *Artificial Intelligence through Simulated Evolution*, New York, NY: John Wiley, 1966.
[42] A.S. Fraser, 'Simulation of Genetic Systems by Automatic Digital Computers. I . Introduction', *Australian J. of Biological Sci*., **10**, 484-491, (1957).
[43] H. Bremermann, 'The Evolution of Intelligence. The Nervous System as a Model of its Environment', Technical Report no. 1 Contract no. 477(17), Dept. of Mathematics, University of Washington, Seattle, 1958.
[44] J. Reed, R. Tooms, and N. Baricelli, 'Simulation of Biological Evolution and Machine Learning', *J. Theo. Biol.*, **17**, 319-342, (1967).
[45] J.H. Holland, 'Outline of a Logical Theory of Adaptive Systems', *J. ACM*, **9**, 297-314, (1962).
[46] J.H. Holland, 'Nonlinear Environments Permitting Efficient Adaptation', *Computer and Information Science IIs*, New York, NY: Academic Press, 1967.
[47] J.H. Holland, *Adaptation in Natural and Artificial Systems*, Cambridge, MA: MIT Press, 1975.
[48] J. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, Cambridge, MA: MIT Press, 1992.
[49] R.M. Friedberg, 'A Learning Machine: Part I.', *IBM Journal of Research and Development*, **3**, 282-287, (1958).
[50] N.A. Barricelli, 'Esempi Numerici di Processi di Evoluzione', *Methodos*, 45-68, (1954).
[51] D.B. Fogel, *Evolutionary Computation*. New York, NY: IEEE Press, 1995.
[52] T. Back, D.B. Fogel, and Z. Michalewicz, *Handbook of Evolutionary Computation*, Bristol, UK: Institute of Physics, and New York, NY: Oxford University Press, 1997.
[53] D.B. Fogel, *The Fossil Record*, New York, NY: IEEE Press, 1998.
[54] K. Forbus, 'Qualitative Reasoning about Physical Processes', *Proc. 7th Int. Joint Conf. on Artificial Intelligence*, Karlsruhe, Germany, 1981.
[55] B. Kuipers, 'Commonsense Reasoning about Causality: Deriving Behavior from Structure', *Qualitative Reasoning about Physical Systems*, D. Bobrow, Ed., 169-203, Cambridge, MA: MIT Press, 1985.
[56] E.H. Mamdani and S. Assilian, 'An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller', *Int. J. Man Machine Studies*, **7**(1), 1-13, (1975).
[57] T. Takagi and M. Sugeno, 'Fuzzy Identification of Systems and Its Applications to Modeling and Control', *IEEE Trans. on Systems, Man, and Cybernetics*, **15**, 116-132, (1985).
[58] R. Babuska, R. Jager, and H.B. Verbruggen, 'Interpolation Issues in Sugeno-Takagi Reasoning', *Proc. Third IEEE Int. Conf. on Fuzzy Systems (FUZZ-IEEE'94)*, 859-863, Orlando, FL, 1994.
[59] H. Bersini, G. Bontempi, and C. Decaestecker, 'Comparing RBF and fuzzy inference systems on theoretical and practical basis', *Proc. of Int. Conf. on Artificial Neural Networks. ICANN '95, Paris, France*, **1**, 169-74, 1995.
[60] J.S.R. Jang, 'ANFIS: Adaptive-network-based-fuzzy-inference-system', *IEEE Trans. on Systems, Man, and Cybernetics*, **23**, 665-685, (1993).
[61] T. Back, D.B. Fogel, and Z. Michalewicz, *Handbook of Evolutionary Computation*, Bristol, UK: Institute of Physics, and New York, NY: Oxford University Press, 1997.
[62] E. Fiesler, and R. Beale, *Handbook of Neural Computation*, Bristol, UK: Institute of Physics, and New York, NY: Oxford University Press, 1997.
[63] P.P. Bonissone, V. Badami, K.H. Chiang,, P.S. Khedkar, K. Marcelle, M.J. Schutten, 'Industrial Applications of Fuzzy Logic at General Electric', *Proc. of the IEEE*, **83**(3), 450-465, (1995).
[64] K.A. De Jong, *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. Ph.D. Thesis, University of Michigan, Ann Arbor, MI. Also available as Dissertation Abstract International, 36(10), 5140B, University Microfilms no. 76-9381, 1975.
[65] J. Grefenstette, 'Optimization of control parameters for genetic algorithms', *IEEE Transactions on Systems, Man, and Cybernetics*, **16**(1), (1986).
[66] O. Cordon, H. Herrera, and M. Lozano, '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, 1995.
[67] M.A. Lee, and H. Tagaki, 'Dynamic control of genetic algorithm using fuzzy logic techniques', *Proc.Fifth Int. Conf. on Genetic Algorithms*, pp. 76-83. Morgan Kaufmann, CA. 1993.
[68] F. Herrera and M. Lozano, 'Adaptive Genetic Algorithms Based on Fuzzy Techniques', *Proc. of IPMU'96*, 775-780, Granada, Spain, 1996
[69] R. Subbu, A. Anderson, and P.P. Bonissone, 'Fuzzy Logic Controlled Genetic Algorithms versus Tuned Genetic Algorithms: An Agile Manufacturing Application', *Proc. IEEE Int. Symposium on Intelligent Control*, NIST, Gaithersburg, Maryland, 1998.
[70] C.L. Karr, 'Design of an adaptive fuzzy logic controller using genetic algorithms'. *Proc. Int. Conf. on Genetic Algorithms (ICGA'91)*, 450-456, San Diego, CA., 1991.
[71] L. Zheng, 'A Practical Guide to Tune Proportional and Integral (PI) Like Fuzzy Controllers', *Proc First IEEE Int. Conf. on Fuzzy Systems, (FUZZ-IEEE'92)*, 633-640, S. Diego, CA, 1992.
[72] P.P. Bonissone, P.S. Khedkar, and Y-T Chen, 'Genetic Algorithms for automated tuning of fuzzy controllers, A transportation Application', *Proc. Fifth IEEE Int. Conf. on Fuzzy Systems (FUZZ-IEEE'96)*, 674-680, New Orleans, LA., 1996.
[73] Appraisal Institute, *Appraising Residential Properties, Part VI*, Chicago IL, 1994.
[74] P.P. Bonissone and W. Cheetham. 'Financial Applications of Fuzzy Case-Based Reasoning to Residential Property Valuation', *Proc. Sixth Int. Conf. On Fuzzy Systems (FUZZ-IEEE'97)*, 37-44, Barcelona, Spain, 1997.
[75] P.P. Bonissone, W. Cheetham, D. Golibersuch, and P.S. Khedkar, 'Automated Residential Property Valuation: An Accurate and Reliable Based on Soft Computing', in *Soft Computing in Financial Engineering*, R. Ribeiro, H. Zimmermann, R.R. Yager, and J. Kacprzyk, Eds., Heidelberg, Germany: Physica-Verlag (Springer-Verlag), 1998.