International Journal of Computational Intelligence Systems, Vol.3, No. 2 (June, 2010), 237-248

Soft Computing: A Continuously Evolving Concept

Piero P. Bonissone
General Electric Global Research, One Research Circle K1-4A10A
Niskayuna, NY 12309, USA
E-mail: bonissone@ge.com

Received: 03-11-2009
Accepted: 28-05-2010

Abstract

Abstract. Soft Computing (SC) is a concept with constantly evolving semantics, as researchers have adopted its main philosophy while adding various interpretations and facets to this concept. Originally defined as a loose association or partnership of components, SC has gone through several transformational phases. This paper will trace some of the phases experienced by the author as part of his understanding of the evolution of SC and its role in constructing decision-making models. The first phase is the hybridization phase, driven by the inherit ease of integration of SC components. The second phase is a two-level model characterization, based on the split between object-level and meta-level reasoning. This phase, inspired by traditional AI problem formulation, led to a third phase, in which we addressed the knowledge and meta-knowledge representation required by each of these reasoning levels using a linguistics analogy. The fourth phase is the extension of the heuristics used at the meta-level, e.g. Metaheuristics (MH's) from evolutionary algorithms to other search methods. The fifth and last phase, further described in this paper, is the proposal for a strong separation between offline MH's (used for design and tuning) and online MH's (used for models selection or aggregation.) This last view suggests a broader use of SC components, since it enables us to use hybrid SC techniques at each of the MH's levels as well as at the object level. Furthermore, this separation facilitates the model lifecycle management, which is required to maintain the models vitality and prevent their obsolescence over time.

Keywords: Fuzzy sets, Neural networks, Evolutionary Algorithms, Metaheuristics, Meta-reasoning.

1. Historic Background

1.1. *The Origins*

Soft Computing (SC) is a concept with constantly evolving meaning, which has benefitted from the initial vision of Prof. Zadeh and the contributions of many researchers in the field, who have provided additional facets and semantic variations.

The coining of this term goes back to 1994, when Prof. Lotfi Zadeh at IIZUKA'94 introduced this concept as "an association of computing methodologies that includes as its principal members fuzzy logic (FL), neuro-computing (NC), evolutionary computing (EC) and probabilistic computing (PC)".[1] Furthermore, Zadeh contrasted this new concept with Hard Computing by highlighting SC ability to "exploit the tolerance for imprecision, uncertainty, and partial truth to achieve tractability, robustness, low solution cost, and better rapport with reality"- although these properties are more typical of fuzzy systems than of the other SC components.

The concept "*association of components*" has rather loose semantics, and as such it allowed many researchers to provide their own interpretations of this idea, refining this term in various ways, not always coherent among themselves. At the same time, we witnessed the advent of a similar concept, *Computational Intelligence*, which came of age in 1994 with the launch of the *First IEEE World Congress on Computational Intelligence (WCCI),* in Orlando Florida. This congress was allegedly the first organized attempt to create a common forum for three of SC basic technologies (fuzzy, neural, and evolutionary computation). *Computational Intelligence* (CI) has a broad overlapping with Soft Computing. Based on the definition provided by the IEEE Computational Intelligence, CI covers *biologically and linguistically motivated computational paradigms*.[2] Its scope seems to exclude probabilistic reasoning systems, while including other nature-inspired methodologies, such as swarm computing, ant colony optimization, etc. Readers interested in the origins of the CI concept should consult references.[3-5] Rather than trying to differentiate between the two terms, we will follow the evolution of Soft Computing and trace this evolution within the context of creating SC based decision-making models.

2. A Personal, Retrospective View of Soft Computing

2.1. *Phase 1: Hybridization*

In 1997, the author noted that the various SC components not only were coming of age but they were slowly converging to create *Hybrid Soft Computing systems.*[6] The SC components were labeled as *reasoning* and *search* techniques. We considered the reasoning techniques as knowledge-driven tools to translate domain knowledge into models, while we considered the search techniques as data-driven tools to extract models from the data, rather than starting from the domain expert. Figure 1 (adapted from reference 6)

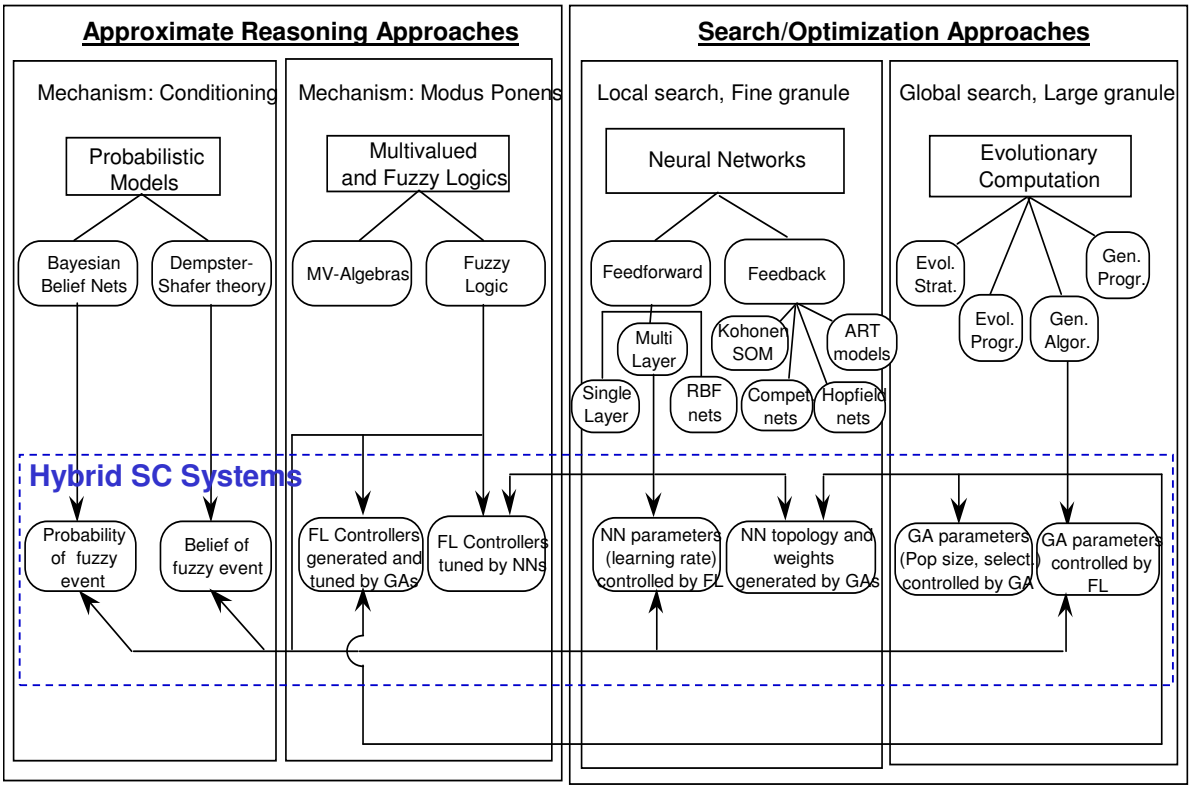

Fig. 1. Soft Computing Overview (adapted from reference 6).

illustrates this concept. The hybridization was a natural consequence of trying to integrate domain knowledge with field data. Other researchers[7] considered other facets of the concept of SC, which was also revisited by Zadeh in 1998.[8]

In 1999, the author focused on the ease with which SC components could be integrated to form hybrid SC systems.[9] Specifically, we stated that "*we have seen an increasing number of hybrid algorithms, in which two or more SC technologies have been integrated to leverage the advantages of individual approaches. By combining smoothness and embedded empirical qualitative knowledge with adaptability and general learning ability, these hybrid systems improve the overall algorithm performance.*" In the same paper, we also focused on the synergy generated by the use of *search* components to generate or tune *reasoning* components and illustrated this synergy in four real-world applications.

The concept of hybrid SC was embraced and further developed by many other researchers, who explored the use of global search methods, such as evolutionary algorithms for generating probabilistic systems,[10] fuzzy systems,[11] and neural networks,[12] to mention a few.

2.2. *Phase 2: Two-level Modeling (Object- and Meta-Reasoning)*

In 2003, the author proposed to view the modeling problem as a two-level problem.[13] This view was influenced by traditional AI problem formulation approaches. The first level was the *object-level*, in which SC techniques were used to implement run-time models to solve domain-specific problems. The second one was the *meta-level*, in which SC techniques were used to generate, improve, update, and control the object-level models. This two-level decomposition also suggested symmetry between reasoning and search methodologies, so that we could use knowledge and reasoning to control search and vice-versa. In the same reference 13, and later in reference 14, the author proposed a distinction between *offline* and *online Metaheuristics* (*MH's*). *Offline MH's* deal with the batch design of object-level models. Once the design is complete, run-time object-level models are generated and used to solve the problem without further modifications. This relationship is analogous to the compiler/run-time model relationship that we find in Machine Learning, where ML algorithms extract relevant information from the training set and generate run-time models. *Online MH's* on the other hand, are used to monitor, guide, and control the resources of the run-time model. Figures 2 and 3 (adapted from reference 14) illustrate the use of offline MH's for model parameter design and on-line MH's for model parameter run-time control.

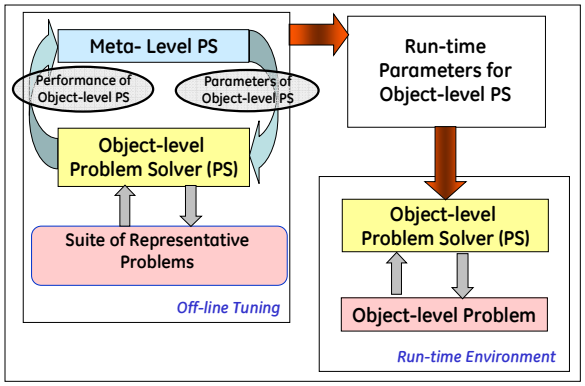

Fig. 2. Schematic of Offline Meta-Heuristics (adapted from reference 14).

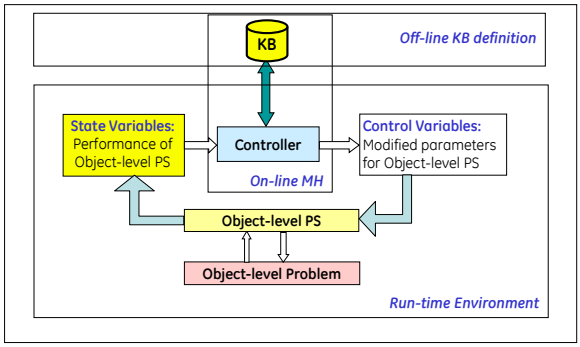

Fig. 3. Schematic of Online Meta-Heuristics (adapted from reference 14).

2.3. *Phase 3: Domain Knowledge Representation*

Having established this two-level structure, we decided to focus on the knowledge representation required by each structure. To measure the depth of such knowledge, the author proposed a *linguistics analogy*, in which the knowledge's depth ranges from lexical (e.g., event codes), to morphological (e.g., *event code*

taxonomy), syntactic (e.g., *signatures* derived from event code ordering), semantic, (e.g., *first-principle based meaning*) and pragmatic (e.g., context-dependent *model selection*).[15]

This domain knowledge ordering was used to establish a decision framework defined as the cross-product of the time-horizon over which a decision was needed (from single decision to short, medium, long and life-long term) and the knowledge depth required by the decision making model, as illustrated in Figure 4. By observing this figure, we observe that only for short-term (tactical) horizon applications we can develop models that are based on relatively shallow knowledge (from lexical to syntactic). In these cases, it is common to construct an ensemble of such models, ensuring their diversity (in the sense of errors' uncorrelation) and performing a fusion to increase the output's reliability.

us to decompose the meaning of a communication (model) into the meanings of its components and their relationships.

For instance, by incorporating engineering knowledge, we can identify key system variables, leverage their functional dependency to verify the correctness of other variables, extract the most informative features to create more compact representations, etc. Furthermore, the performance metrics associated with these tasks become less precise and more qualitative in nature. This characteristic is very suitable for the use of fuzzy system as possible fitness evaluator for the evolutionary algorithms that might be used to explore the models.

To build meta-level models, we need to use *pragmatics*, which uses external, contextual knowledge to fully understand the meaning of our communication. While

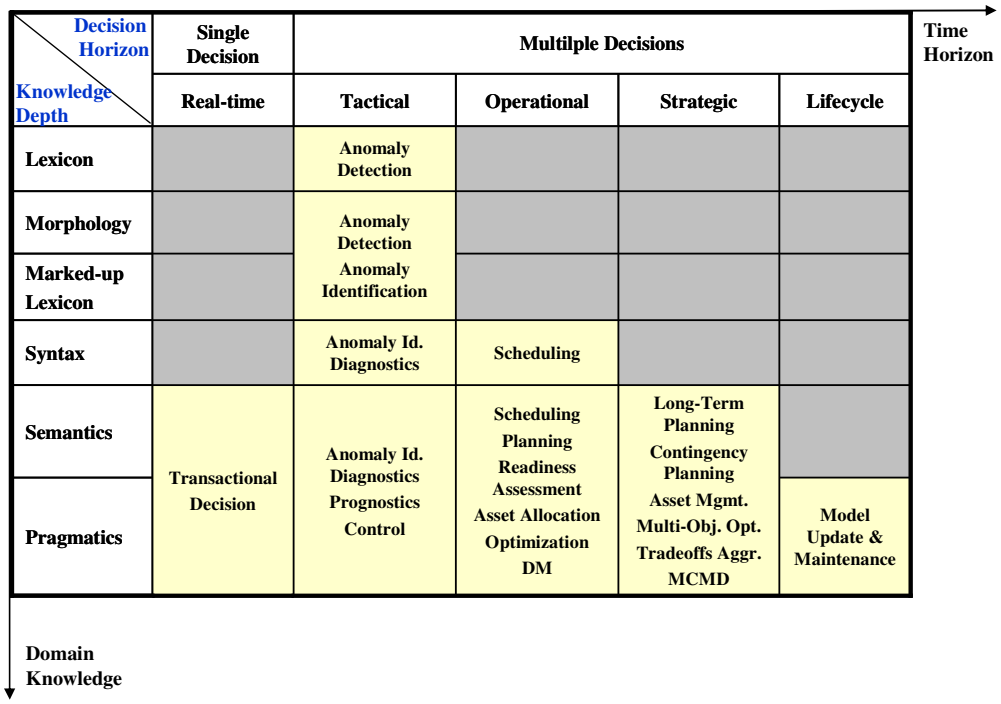

Fig. 4. Framework for SC applications (adapted from reference 15).

As the time horizon increases, deeper domain knowledge is required to create the object-level models, the models outputs are usually complex (schedules, plans) and become less suitable for fusion. Such sophisticated models require the use of semantics, since - as it in the case of system analysis – semantics allows all prior levels dealt with information contained in the message itself (object-level), pragmatics requires higher-level knowledge (meta-level) to provide the contextual information needed to disambiguate a sentence, correctly interpret its meaning, etc. For model building, this consists in leveraging contextual

information (such as operational regimes, environmental conditions, health deterioration) to determine the degree of applicability of local models and to select the best (or the best mixture).

The selection of the most appropriate SC techniques, in conjunction with "sibling" disciplines, such as Al, Statistics, and Information Theory, depends on the type of available domain knowledge. Table 1 depicts the most useful SC approaches for different knowledge types (labeled according to our linguistics analogy).

Examples of two-level models and their required knowledge types are illustrated in Figure 5 and 6, which have been adapted from reference 13 to incorporate the view presented in reference 15.

In Figure 6 we can observe three object-level models based fuzzy logic control (FLC), a neural fuzzy system (ANFIS), and an evolutionary algorithm (EA). Each of these models is supervised at run-time by a fuzzy controller (FLC) or a combination of fuzzy sets (FS) and statistics based models to determine the best mixture of object-level models, or to modify their resources or parameters, such as population size and probability of mutation for the EA. More specific information about these applications can be found in references 22-26.

2.4. *Phase 4: Extending Offline Metaheuristics*

The two-level approach was further refined in reference 27. When dealing at the meta-level, we extended global search techniques from evolutionary algorithms to a variety of meta-heuristics, such as *relaxation* and *search MH's*. With this extension we emphasized the fact that the meta-heuristics were used to perform a search in the object-model design space. As such, we should be able to use a variety of search methods, such as taboo search, scatter search, hill climbing, greedy like, multi-start, variable neighborhood, simulated annealing, evolutionary search, etc. This is illustrated in Figure 7, adapted from reference 27.

Table 1. SC Techniques & Domain Knowledge.

SC/Stat/AI Techniques	**Domain Knowledge**
Self-Organizing Maps (SOM) Kolmogorov Complexity, One-class Support Vector Machine, Neural Networks, Unsupervised Machine Learning techniques, fuzzy clustering, non-parametric statistics	Lexicon and Morphology
Supervised Machine Learning techniques, NN, Fuzzy Classifiers, CART, Random Forest, MARS	Marked-up Lexicon
Automated Kernel Splitting, Grammatical Inference, Evolutionary Algorithms (EA)	Syntax
Feature extraction/selection, fuzzy models, 1st Principle based simulations, temporal reasoners, Case-based Reasoners, planners, Evolutionary Algorithms	Semantics
Model Selection/Mixing, EA, MOEA, Fuzzy models for preference aggregation and tradeoffs	Pragmatics

In Figure 5 we can observe a variety of object-level models or problem solvers (PS) based on fuzzy sets, neural networks, traditional controllers, Bayesian networks, etc. Each model needs specific parametric information that was instantiated and tuned offline by a meta-level PS (either a neural network (NN) using local search, or an evolutionary algorithm (EA) using global search.) More specific information about these applications can be found in references 12 and 16-21.

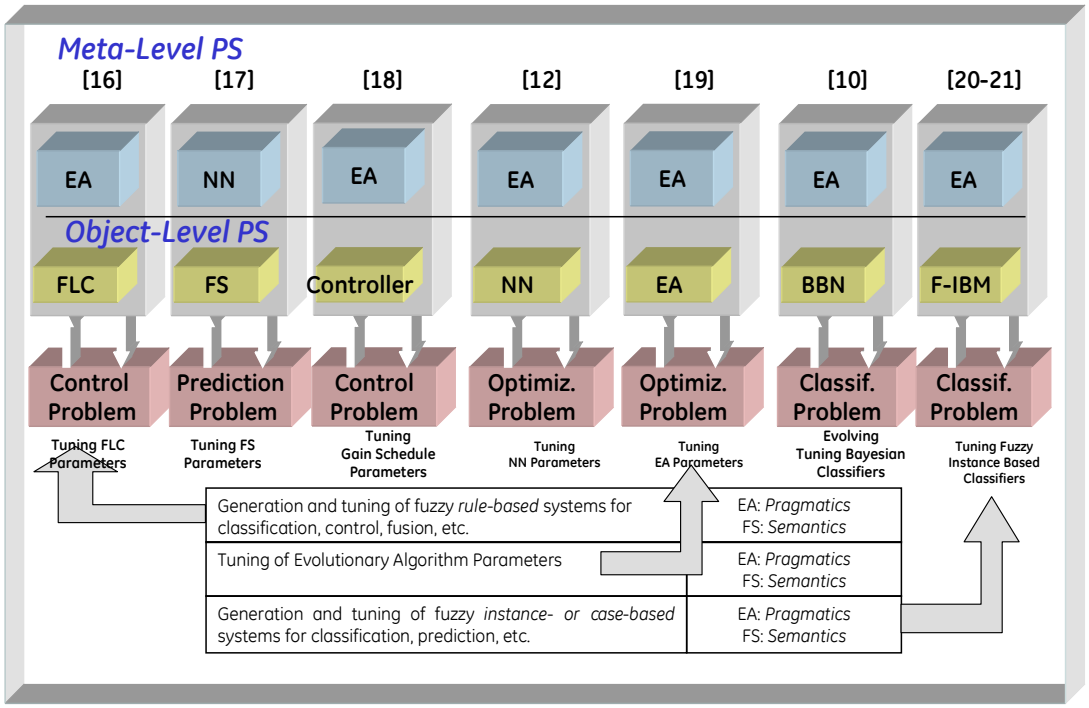

Fig. 5. Example of Offline Meta-Heuristics (adapted from reference 15).

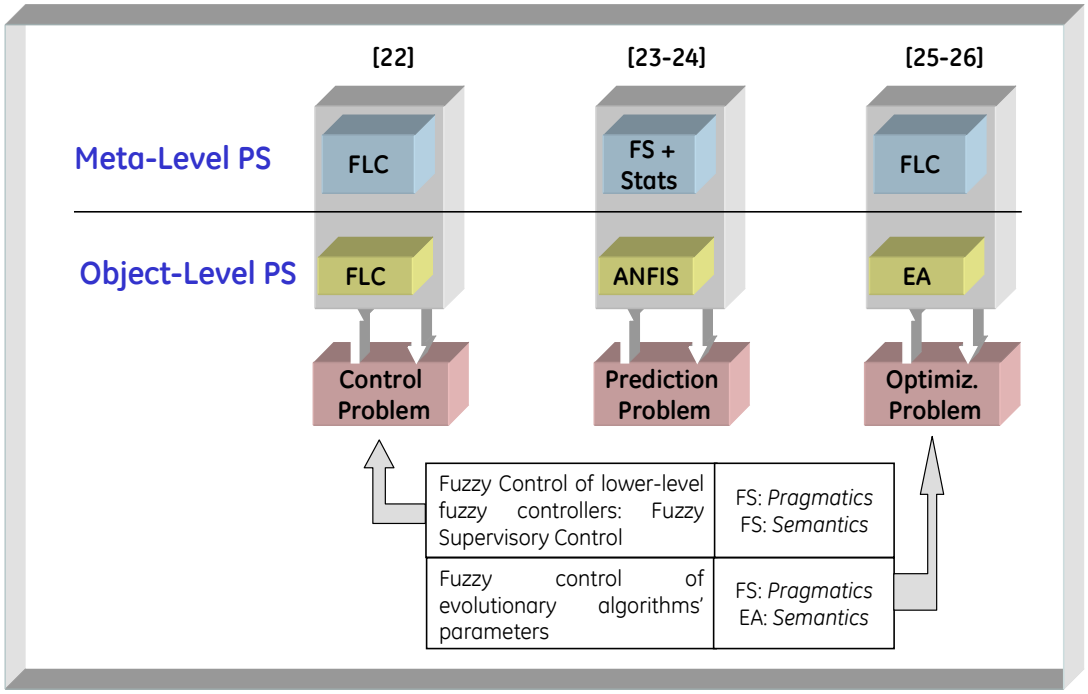

Fig. 6. Example of Online Meta-Heuristics (adapted from reference 15).

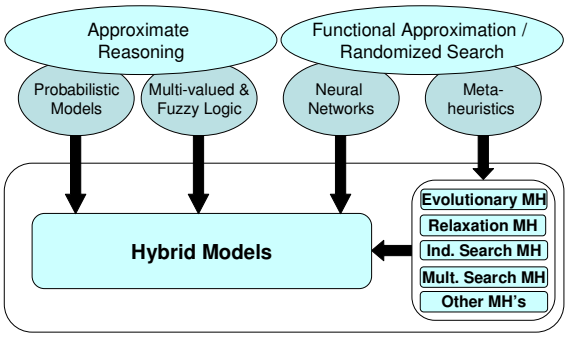

Fig. 7. Meta-Heuristics Extension (adapted from reference 27).

3. A Prospective View of Soft Computing

Guided by the retrospective view described in the previous section, we will extend our definition of building SC models by suggesting the following steps:

- Create a strong separation between *offline MH's*, which are applied in batch mode during the model(s) design phase, and *online MH's*, which - if needed - will be part of the run-time model architecture and will be designed by the offline MH's.
- Use SC technologies to build any of the MH's and object-level models.

Specifically, when building a model we will distinguish among these stages:

(i) *Model Design: Offline-MH's* to design, tune, optimize, adapt to changes, and maintain the run-time models over time.

(ii) *Model Architecture:*
  (a) *Online-MH's* to integrate/interpolate among multiple (local) object-models, manage their complexity, and improve their overall performance;
  (b) *Multiple object-models*, either in parallel configuration (ensemble) or sequential configuration (cascade, loop), to integrate functional approximation with optimization and reasoning with imperfect data (imprecise and uncertain); or for simpler problems:
  (c) *Single object-model* to provide an individual SC functionality (functional approximation, optimization, or reasoning with imperfect data).

(iii) *Model Representation:* structure and parameters for **each** (*MH's* and object) model.

This strategy allows us to leverage SC capabilities at every level. We can manage complexity by finding the best model architecture to support problem decomposition, create high-performance local models with limited competence regions and allow for smooth interpolations among them, and promote robustness to imperfect data by aggregating diverse models. In the next section we will examine a case study to further describe this concept.

4. Examples of SC to Develop Offline MH's, Online MH's, and Object-level Models

Let's take a brief look at some case studies where we employed SC techniques for model design (using *offline MH's*), for model control (using *online MH's*), and for object-level models generation. Table 2 illustrates various instances of this use of SC techniques.

4.1. *Anomaly Detection Model*

For illustration purpose we will analyze the third case study of Table 2, which deals with a classification problem (anomaly detection) for a fleet of physical assets (such as an aircraft engines or a gas turbines). A critical component of the Prognostics and Health Management (PHM) of these assets is the timely and correct detection and identification of any abnormal behavior. Anomaly detection leverages unsupervised learning techniques, such as clustering. Its goal is to extract the underlying structural information from the data, define *normal* structures and regions, and identify departures from such regions. After detecting an abnormal change, (e.g. a departure from a normal region), we need to identify its cause. There are many factors that could cause such change:

- A **system fault**, which could eventually lead to a failure;
- A **sensor fault**, which is creating an incorrect measurement;
- An **inadequate** anomaly detection **model** that is creating false alarms due to bad model design, inadequate model update, execution outside the model's region of competence, etc;
- A sudden, unexpected **operational transient**, which is stressing the system by creating an abrupt load change. In turn, this transient could be originated by an operator error, who is requesting such sudden change; by an incorrect reference (set-up) vector – in case of operation automation - which is also requesting such abrupt change; or by

Table 2. SC Techniques & Domain Knowledge.

Problem Instance	**Problem Type**	**Model Design (*Offline MH's*)**	**Model Controller (*Online MH's*)**	**Object-level models**	**References**
Anomaly Detection (System)	Classification	Model T-norm tuning	Fuzzy Aggregation	*Multiple Models:* SVM, NN, Case-Based, MARS	[28]
Anomaly Detection (System)	Classification	Manual design	Fusion	*Multiple Models:* Kolmogorov Complexity, SOM. Random Forest, Hotteling T2, AANN	[29, 30]
Anomaly Detection (Model)	Classification & Prediction	EA tuning of fuzzy supervisory termset	Fuzzy Supervisory	*Multiple Models:* Ensemble of AANN's	[31, 32]
Insurance Underwriting: Risk management	Classification	EA	Fusion	*Multiple Models:* NN, Fuzzy, MARS,	[33, 34]
Load, HR, NOx Forecast	Prediction	Multiple CART trees	Fusion	*Multiple Models:* Ensemble of NN's	[35, 36]
Aircraft Engine Fault Recovery	Control/Fault Accommodation	EA tuning of linear control gains	Crisp supervisory	*Multiple Models (Loop):* SVM + linear control	[18]
Power Plant Optimization	Optimization	Manual design	Fusion	*Multiple Models (Loop):* MOEA + NN's	[36, 37, 38]
Flexible Mfg. Optimization	Optimization	Manual design	Fuzzy supervisory	Evolutionary Algorithms	[14, 39]

a bad controller, which is over- or under-compensating for some perceived state change.

In this case study, further described in references 31-32, we focused on the prevention of false alarms caused by anomaly detection models whose accuracy was inadequate when compared with the signal level of the anomaly.

4.1.1. *Object-level models (AANN's)*

One of the best techniques to detect anomalies is the auto-associative neural networks (AANN's). They are feedforward neural networks with network structure satisfying requirements for performing restricted auto-association. The inputs to the AANN's go through a dimensionality-reduction, as their information is combined and compressed in intermediate layers. In the ideal case, the AANN's outputs should be identical to the inputs. Their differences (residuals) and their gradient information are used to train the AANN's to minimize such differences. This network computes the largest Non-Linear Principal components (NLPCA's) – the nodes in the intermediate layer– to identify and remove correlations among variables. When an anomaly occurs, the AANN's will detect a departure from the covariance information obtained during training and captured by the NLPCA's. This disruption will generate larger residuals that will trigger the anomaly notification.[40-41]

4.1.2. *Online MH's (fuzzy supervisory system)*

Training a global AANN on the entire operating space of the asset usually does not produce the required accuracy. Global models are designed to achieve a compromise among completeness (for coverage) and high-fidelity (for accuracy). As a result, we typically end up with models that have small biases but large variability. This variability is usually too large to allow us to distinguish between model error and anomalous asset behavior. To achieve the required accuracy we used an ensemble of local models (AANN's) with limited, overlapping regions of applicability aggregated by a fuzzy supervisory system. This allowed us to leverage the performance of customized local models and combine their outputs, using a smooth interpolation mechanism as we moved across adjacent operating regions.

4.1.3. *Offline MH's (evolutionary algorithms)*

Having established a structure for the run-time AD model we needed to design the AD model. We trained

each AANN separately, within its region of competence, and we defined a set of fuzzy transition rules for the supervisory controller. Then we defined an *Offline MH*, using an Evolutionary Algorithm to tune the parameters of the membership functions of the fuzzy supervisory to minimize a figure of merit that aggregated all the residuals during normal conditions. This process is illustrated in Figure 8.

In the left part of figure 8 we can observe the run-time anomaly detection (AD) model. In the center part of figure 8 we can see an instance of the term set used by the fuzzy supervisory system (the scale of the operational state variables was normalized as a percentage of the range of values to preserve proprietary information). In the right part of figure 8, we can see the offline MH's, based on an evolutionary algorithm (EA)

We have established that Soft Computing is an evolving concept, a multifaceted idea that gives the model-builder a gamut of techniques for integrating domain knowledge with data-extracted information at any model-building level.

5.1. *Designing and Structuring SC models*

Rather than creating amorphous hybrid SC systems, we are advocating a design methodology based on the use of offline MH's to search for the most appropriate models. The design of these models usually needs to follow a problem decomposition strategy to manage problem complexity and create manageable components that can be adapted to changes, and maintained over time. To address this complexity, we also advocate the use of a hierarchical architecture, controlled by an

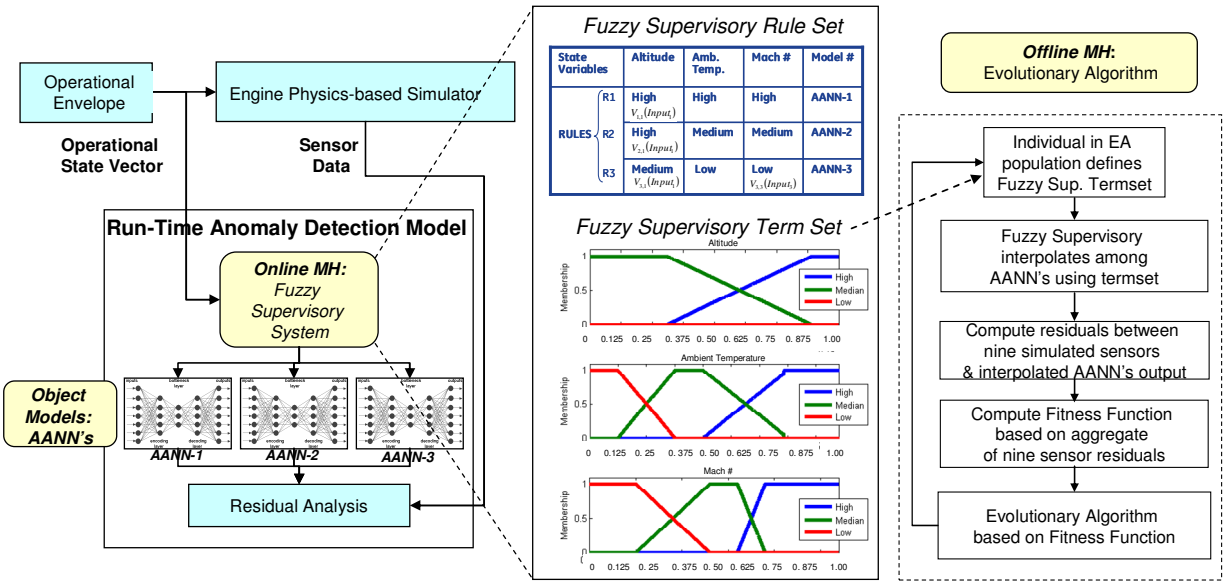

Fig. 8. Offline MH (EA) to tune a run-time AD model composed of an Online MH (Fuzzy Supervisory System) and an ensemble of local object-level models (AANN's) - (adapted from reference 32).

in a wrapper configuration, used to tune the term set. More information about this application can be found in references 31-32.

5. Conclusions and Remarks on Model Lifecycle Management

online MH's, which usually acts as a supervisor, a fusion mechanism, or a resource controller to integrate multiple (local) object-level models, improving performance (e.g., accuracy) and robustness when dealing with imperfect data. Finally the object-level models could be individual (for simple problems) or multiple models (in parallel or serial configuration) to address the performance/complexity tradeoff. The

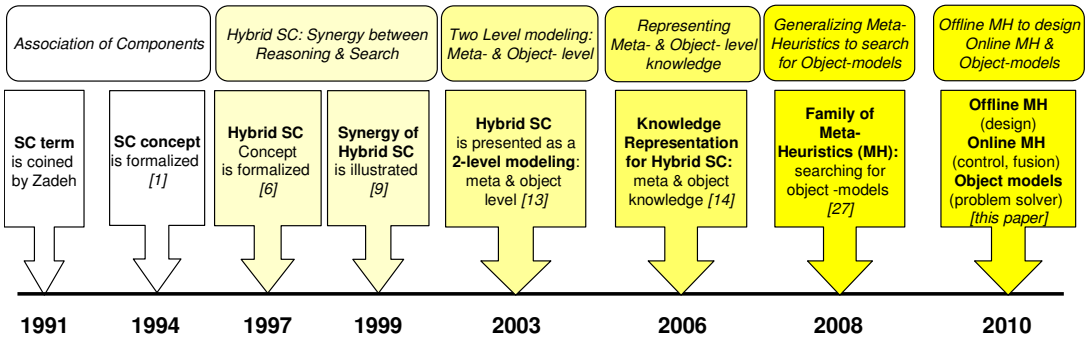

Fig. 9. Timeline depicting the author's perception of SC evolution.

intrinsic ability of SC techniques to be easily integrated with other sibling techniques (such as Statistics or AI) allows us to leverage SC at all three levels of modeling. The timeline depicting the author's perception of the evolution of the concept of Soft Computing is illustrated in Figure 9.

5.2. *Model Lifecycle for SC models*

The strong separation between *offline* and *online MH's* advocated in this paper allows us to address another challenge: the automation of SC models lifecycle.[21,42-45]

In reference 21, we noted that "*in real-world applications, before we can use a model in a production environment we must address the model's entire lifecycle, from its design and implementation, to its validation, tuning, production testing, use, monitoring, and maintenance. By maintenance we mean all the steps required to keep the model vital (e.g. non obsolete) and able to adapt to changes*".

What causes models to become obsolete? Models are built under contextual, domain-knowledge, and data assumptions. When any of these assumptions becomes invalid, the conditions for the models applicability no longer hold and the models must be updated. In the case of manually-designed models, their updating must also follow a manual process, creating potential bottlenecks and scalability risks. On the other hand, in the case of automatically-designed models, their updating will benefit from the re-use of the same process (e.g., *Offline MH*'s) to create the models new versions.

As an illustrative example of this concept – further documented in references 45-46 - we describe the automation of the underwriting of insurance policy applications, based on an assessment of the applicants' risk.

5.2.1. *Run-time model (fuzzy classifier)*

The run-time model for this application was a fuzzy classifier, whose boundaries were computed to minimize the cost of misclassification. The fuzzy boundaries were used to capture a tradeoff between risk reduction (leading to stringent restrictions) and price competitiveness (leading to more tolerant restrictions.)

5.2.2. *Offline MH's for model generation (evolutionary algorithms)*

Since we wanted to minimize the cost of misclassification, it was necessary to establish a baseline of correct decisions or standard reference decision (SRD) set. First we collected about 3,000 cases of insurance policies that were underwritten in the past under currently valid assumptions. After scrubbing about 10% of these cases to remove questionable decisions, we refined the original set and created the SRD, which represented the behavior of the model (classifier) which we wanted to build. Then, we built a fitness function based on the cost of misclassification using the SRD as our target and used evolutionary search in the space of fuzzy classifiers to instantiate and evolve populations of competing models. After a sufficient number of generations, we selected the best individual of the final population to become the run-time model (classifier) to be placed in production.

5.2.3. *Offline MH's for model updating (evolutionary algorithms)*

As noted earlier, during the life of the classifier it might be necessary to change some of the underwriting rules embedded in the classifier. These modifications could be caused by new government regulations, changes among data suppliers, new medical findings, different competitive market pressures, etc. We identified the subset of SRD cases whose decisions were affected by the changes and we requested a panel of expert underwriters to assign new decisions (if needed) to the selected cases. The edited and updated SRD represented the new target that we wanted our classifier to approximate. At this point, we used the same EA-based optimization tool, employed during the initial tuning, to find a parametric configuration that defined the new classifier that better approximated the new SRD.

The proposed methodology for using SC components to build decision-making models provides a clean separation between design and run-time issues, and furthermore supports the models lifecycle maintenance, a necessary step for their deployment in real-world applications.

References

1. L. A. Zadeh, "Fuzzy logic and soft computing: Issues, contentions and perspectives", in *Proc. IIZUKA'94: 3rd Int. Conf. Fuzzy Logic, Neural Nets and Soft Computing*, (1994), pp. 1–2.
2. http://ieee-cis.org/about_cis/scope/
3. J. Bezdek, On the relationship between Neural Networks, Pattern Recognition, and Intelligence, *International Journal Approximate Reasoning*, vol. 6, (Elsevier, Amsterdam,1992), pp. 85-107.
4. R. Marks II, Computational versus Artificial, *IEEE Transactions on Neural Networks*, 4(5) (1993) 737-739.
5. J. Bezdek, What is Computational Intelligence?, in *Computational Intelligence Imitating Life*, J. Zurada, R. Mark II, C. Robinson eds., (IEEE Press, New York, NY, 1994).
6. P. P. Bonissone, Soft computing: The convergence of emerging reasoning technologies, *Soft Computing Fusion of Foundations, Methodologies Applications,* 1(1) (1997) 6–18.
7. D. Dubois and H. Prade, Soft computing, fuzzy logic, and Artificial Intelligence, *Soft Computing Fusion of Foundations, Methodologies Applications*, 2(1) (1998) 7–11.
8. 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 Fusion of Foundations, Methodologies Applications*, 2(1) (1998) 23–25.
9. P. P. Bonissone, Y-T Chen, K. Goebel, & P. Khedkar, Hybrid Soft Computing Systems: Industrial and Commercial Applications, *Proceedings of the IEEE*, 87(9) (1999), pp. 1641-1667.
10. P. Larrañaga, J.A. Lozano, Synergies between evolutionary computation and probabilistic graphical models, *International J. Approximate Reasoning,* 31(3) (Elsevier, Amsterdam, 2002), pp. 155–156.
11. O. Cordón, F. Gomide, F. Herrera, F. Hoffmann, L. Magdalena, Ten years of genetic fuzzy systems: current framework and new trends, *Fuzzy Sets and Systems* 141(1) (Elsevier, Amsterdam, 2004), pp. 5–31.
12. X. Yao, Evolving Artificial Neural Networks, *Proceedings of the IEEE*, (87)9 (1999), pp. 1423-1447.
13. P. Bonissone, Soft computing and meta-heuristics: using knowledge and reasoning to control search and vice-versa, *Proc. of the SPIE, Vol. 5200, Applications and Science of Neural Networks, Fuzzy Systems and Evolutionary Computation V*, (2003), pp. 133–149.
14. P. Bonissone, R. Subbu, N. Eklund, and T. Kiehl, Evolutionary Algorithms + Domain Knowledge = Real-World Evolutionary Computation, *IEEE Transactions on Evolutionary Computation,* 10(3) (2006), pp. 256-280.
15. P. Bonissone, Domain Knowledge and Decision Time: A Framework for Soft Computing Applications, *2006 International Symposium on Evolving Fuzzy Systems (EFS 2006)*, (2006), pp 19-24.
16. P. Bonissone, P. Khedkar, Y-T Chen, Genetic Algorithms for Automated Tuning of Fuzzy Controllers: A Transportation Application, *Proceedings of the 1996 IEEE Conference on Fuzzy Systems (FUZZ-IEEE'96)*, (1996), pp. 674-680.
17. J.S.R. Jang, ANFIS: Adaptive-network-based-fuzzy-inference-system, *IEEE Trans. on Systems, Man, and Cybernetics*, 23(3) (1993), pp. 665-685.
18. K. Goebel, R. Subbu, P. Bonissone, "Controller Adaptation to Compensate Deterioration Effects", *GE GR Technical Report*, 2006GRC298, May 4, 2006.
19. J. J. Grefenstette, Optimization of control parameters for genetic algorithms, *IEEE Trans. Systems, Man, Cybern.*, 16(1) (1986), pp. 122–128.
20. P. Bonissone, A. Varma, Predicting the Best Units within a Fleet: Prognostic Capabilities Enabled by Peer Learning, Fuzzy Similarity, and Evolutionary Design Process", in *Proc. FUZZ-IEEE 2005*, (2005), pp. 312-318.
21. P. Bonissone, A. Varma, K. Aggour, and Feng Xue, Design of local fuzzy models using evolutionary algorithms, *Computational Statistics and Data Analysis*, 51, (2006), pp. 398-416.
22. P. Bonissone and K. Chiang, Fuzzy Logic Hierarchical Controller for A Recuperative Turboshaft Engine: from Mode Selection to Mode Melding, in *Industrial*

Applications of Fuzzy Control and Intelligent Systems, J. Yen, R. Langari, and L. Zadeh (eds.), (IEEE Press, 1995).
23. P. Bonissone and K. Goebel, When will it break? A Hybrid Soft Computing Model to Predict Time-to-break Margins in Paper Machines, *Proc. SPIE 2002*, (2002), pp. 53-64.
24. P. Bonissone, K. Goebel, Y-T Chen, Predicting Wet-End Web Breakage in Paper Mills, *Proc. AAAI Spring Symposium*, (2002).
25. F. Xue, A.C. Sanderson, P. Bonissone, R.J. Graves, Fuzzy Logic Controlled Multi-Objective Differential Evolution, in *Proc. FUZZ-IEEE 2005,* (2005), pp. 720-725.
26. R. Subbu and P. Bonissone, A Retrospective View of Fuzzy Control of Evolutionary Algorithm Resources, in *Proc. FUZZ-IEEE 2003*, (2003), pp. 143-148.
27. J. L. Verdegay, R. Yager, and P. Bonissone, On Heuristics as a Fundamental Constituent of Soft Computing, *Fuzzy Sets and Systems*, 159(7) (Elsevier, Amsterdam, 2008), pp. 846-855.
28. P. Bonissone, K. Goebel, and W. Yan, Classifier Fusion using Triangular Norms, in *Proc. Multiple Classifier Systems (MCS) 2004*, (2004), pp. 154-163.
29. P. Bonissone. N. Iyer, Soft Computing Applications to Prognostics and Health Management (PHM): Leveraging field data and domain knowledge, in *Proc. 9th International Work-Conference on Artificial Neural Networks (IWANN 2007)*, (2007), pp. 928-939.
30. A. Varma, P. Bonissone, W. Yan, N. Eklund, K. Goebel, N. Iyer, S. Bonissone, Anomaly Detection using Non-Parametric information, in *Proc. ASME Turbo Expo 2007: Power for Land, Sea and Air,* (2007).
31. X Hu, P. Bonissone, R. Subbu, Robust Model Selection Decision-making using a Fuzzy Supervisory Approach, in *Proc. IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making 2009*, (2009).
32. P. Bonissone, X Hu, R. Subbu, A Systematic PHM Approach for Anomaly Resolution: A Hybrid Neural Fuzzy System for Model Construction, in *Proc. PHM 2009*, (2009).
33. P. Bonissone, "Automating the Quality Assurance of an on-line Knowledge-Based classifier by fusing multiple off-line classifiers", in *Modern Information Processing: From Theory to Applications*, Bouchon-Meunier, Coletti, Yager (Eds.), (Elsevier, 2005), pp. 147-158.
34. P. Bonissone, R. Subbu, and K. Aggour, Evolutionary Optimization of Fuzzy Decision Systems for Automated Insurance Underwriting, in *Proc. FUZZ-IEEE 2002*, (2002), pp. 1003-1008.
35. F. Xue, R. Subbu, P. Bonissone, Locally Weighted Fusion of Multiple Predictive Models, in *Proc. IEEE International Joint Conference on Neural Networks (IJCNN'06),* (2006), pp. 2137-2143.
36. P. Bonissone, F. Xue, and R. Subbu, "Fast Meta-models for Local Fusion of Multiple Predictive Models" *Applied Soft Computing Journal,* 2008, doi:10.1016/j.asoc.2008.03.006
37. R. Subbu, P. Bonissone, S. Bollapragada, K. Chalermkraivuth, N. Eklund, N. Iyer, R. Shah, F. Xue and W. Yan, A review of two industrial deployments of multi-criteria decision-making systems at General Electric, in *Proc. IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making (MCDM 2007),* (2007).
38. R. Subbu, P. Bonissone, N. Eklund, W. Yan, N. Iyer, F. Xue, R. Shah, Management of Complex Dynamic Systems based on Model-Predictive Multi-objective Optimization, in *Proc. CIMSA 2006*, (2006), pp. 64-69.
39. R. Subbu and P. Bonissone, A Retrospective View of Fuzzy Control of Evolutionary Algorithm Resources, in *Proc. FUZZ-IEEE 2003*, (2003), pp. 143-148.
40. M.A. Kramer, Autoassociative neural networks, *Computers & Chemical Engineering*, 16(4) (1992), pp. 313-328.
41. J.W. Hines, I E. Uhrig, Use of Autoassociative Neural Networks for Signal Validation, *Journal of Intelligent and Robotic Systems,* 21(2) (1998), pp. 143-154.
42. P. Bonissone, A. Varma, K. Aggour, An Evolutionary Process for Designing and Maintaining a Fuzzy Instance-based Model (FIM), in *Proc. First Workshop of Genetic Fuzzy Systems (GFS 2005)*, (2005).
43. P. Bonissone, The life cycle of a fuzzy knowledge-based classifier, in *Proc. North American Fuzzy Information Processing Society (NAFIPS 2003)*, (2003), pp. 488-494.
44. P. Bonissone, Development and Maintenance of Fuzzy Models in Financial Applications, in *Soft Methodology and Random Information Systems*, Lopez-Diaz, Gil, Grzegorzewski, Hyrniewicz, Lawry (eds.), (Springer, 2004).
45. A. Patterson, P. Bonissone, and M. Pavese, Six Sigma Quality Applied Throughout the Lifecycle of and Automated Decision System, *Journal of Quality and Reliability Engineering International*, 21(3) (2005), pp. 275-292.
46. K. Aggour, P. Bonissone, W. Cheetham, R. Messmer, Automating the Underwriting of Insurance Applications, *AI Magazine,* 27(3) (2006), pp. 36-50.