EDITOR'S INTRODUCTION TO CHAPTER 1

Fusion of soft-computing (SC) and hard-computing (HC) methodologies is a well-known approach for practicing engineers. Since the utilization of evolutionary computation, fuzzy logic, and neural networks began to emerge in various applications, also complementary unions of SC and HC have been in common use. Nevertheless, the actual term "Fusion of Soft Computing and Hard Computing" was invented as recently as 1998. But why do we need such a term *now* if the basic ideas have been in applications use already for more than a decade? We will return to this relevant question at the end of the following discussion.

Soft computing is sometimes called—particularly in Japan—"human-like information processing." This is a somewhat misleading description, because the human brain has two specialized hemispheres, the left one (logic, mathematics, analysis, serial processing, etc.) and the right one (creativity, recognition, synthesis, parallel processing, etc.), and they are connected closely to one another through the *corpus callosum*, which consists of over 200 million nerves [1]. Each hemisphere is continually supporting and complementing the activity of the other. Roger W. Sperry shared the 1981 Nobel Prize in Medicine for his discoveries concerning "the functional specialization of the cerebral hemispheres" [2]. Based on his brilliant work that enhanced our comprehension of the higher functions of human brain, obvious analogies have been established between the left brain and HC, as well as between the right brain and SC. Furthermore, the massive bundle of connecting nerves corresponds to the fusion interface between the two computing paradigms; this crucial communications extension to the above brain analogy is hardly ever mentioned in the literature.

Computationally Intelligent Hybrid Systems. Edited by Seppo J. Ovaska
ISBN 0-471-47668-4

The aims of human-like *hybrid* information processing are directed toward developing either computationally intelligent (CI) or artificially intelligent (AI) systems—that is, computer-based implementations with high machine IQ. The key distinction between these two common terms was defined by Bezdek in 1994 [3]. In his practical definition, CI is a subset of AI; and the term "artificially intelligent" is reserved for systems where computational intelligence is augmented by "incorporating knowledge (tidbits) in a non-numerical way."

Hayashi and Umano published a seminal article on fusing fuzzy logic and neural networks in 1993 [4]. Their pragmatic discussion on different fusion categories was an activator for the fuzzy-neuro boom of the late 1990s. By explicitly emphasizing the complementary fusion approach, they stimulated the (Japanese) engineering community to develop *hybrid intelligent systems*. Besides, they wisely pointed out that to combine individual methodologies effectively, considerable attention should be paid to the interface section; it is generally not effective to force two algorithms together.

Chapter 1 of the present book, "Introduction to Fusion of Soft Computing and Hard Computing," is authored by Seppo J. Ovaska. It forms a solid basis for the complementary fusion of soft computing and hard computing methodologies by introducing a representative set of fusion architectures and defining their functional mappings. In addition, it gives a concise definition of "Fusion of Soft Computing and Hard Computing" and offers a rationale for the term. Most of the reviewed real-world examples contain *loose* connections of SC and HC algorithms, while the more *intimate* connections offer future research and development (R&D) potential. As an interpretation of Chapter 1, the principal R&D challenges of this fusion field can be summarized as follows:

- Interface sophistication beyond the current means
- True symbiosis of soft computing and hard computing techniques
- Fusion implementations at different levels of hierarchy (algorithm, subsystem, system, and entity)

Let us now return to the question, *Why* do we need the term "Fusion of Soft Computing and Hard Computing"? This concept needs to be recognized explicitly, because in that way we can highlight that there is a lot of unused R&D potential in the *interface section* between SC and HC methodologies. In the highly successful human brain, the left and right hemispheres are linked closely together. By following such an analogy, it could be anticipated that a high degree of interaction between soft and hard computing would lead to better implementation economy of CI and AI systems, improved tolerance against incomplete or uncertain sensor data, enhanced learning and adaptation capabilities, and increased machine IQ. All those characteristics are needed for developing autonomously intelligent systems. Furthermore, the fuzzy-neuro era of the 1990s started after Hayashi and Umano summarized the principal fusion structures with an insightful discussion on the entire fusion approach [4].

A similar boost in the R&D of computationally or artificially intelligent hybrid systems could follow the present book.

REFERENCES

1. P. Russell, *The Brain Book*, Routledge & Kegan Paul, London, UK, 1979, pp. 48–63.
2. The Nobel Assembly at the Karolinska Institute, "Press Release: The 1981 Nobel Prize in Physiology or Medicine," Oct. 1981, Stockholm, Sweden. WWW page. Available from <http://www.nobel.se/medicine/laureates/1981/press.html>.
3. J. C. Bezdek, "What is Computational Intelligence?" in J. M. Zurada, R. J. Marks II, and C. J. Robinson, eds., *Computational Intelligence: Imitating Life*, IEEE Press, Piscataway, NJ, 1994, pp. 1–11.
4. I. Hayashi and M. Umano, "Perspectives and Trends of Fuzzy-Neural Networks," *Journal of the Japan Society of Fuzzy Theory and Systems* **5**, 178–190 (1993), in Japanese.

CHAPTER 1

INTRODUCTION TO FUSION OF SOFT COMPUTING AND HARD COMPUTING

SEPPO J. OVASKA
Helsinki University of Technology, Espoo, Finland

1.1 INTRODUCTION

The roots of this book on the fusion of soft computing and hard computing methodologies can be traced back to the IEEE International Conference on Systems, Man, and Cybernetics that was held in San Diego, California, in 1998. One of the highlights of that conference was the panel discussion on "New Frontiers in Information/Intelligent Systems." Lotfi A. Zadeh, the father of fuzzy logic and soft computing, was the moderator of the panel, and the panelists were all world-class scholars in the field of soft computing. While the discussion was surely stimulating and provided inspiring visions, something was missing—the dominating and continuing role of conventional hard computing in developing successful products and profitable services was not recognized at all—and that caused us to focus our research interests on the complementary *fusion* of these two principal methodologies.

1.1.1 Soft Computing

From the practicing engineer's point of view, a large amount of real-world problems can be solved competitively by hard computing. The term "hard computing" is not nearly as widely accepted or known as "soft computing," but Zadeh used it already in his original definition of soft computing (quoted from a memo [1] written in 1991 and updated in 1994; references 2–4 added by the author of the present chapter):

Computationally Intelligent Hybrid Systems. Edited by Seppo J. Ovaska
ISBN 0-471-47668-4

> Soft computing differs from conventional (hard) computing in that, unlike hard computing, it is tolerant of imprecision, uncertainty, partial truth, and approximation. In effect, the role model for soft computing is the human mind. The guiding principle of soft computing is: Exploit the tolerance for imprecision, uncertainty, partial truth, and approximation to achieve tractability, robustness and low solution cost. The basic ideas underlying soft computing in its current incarnation have links to many earlier influences, among them Zadeh's 1965 paper on fuzzy sets [2]; the 1973 paper on the analysis of complex systems and decision processes [3]; and the 1981 paper on possibility theory and soft data analysis [4]. The inclusion of neural computing and genetic computing in soft computing came at a later point.
>
> At this juncture, the principal constituents of soft computing (SC) are fuzzy logic (FL), neural computing (NC), genetic computing (GC) and probabilistic reasoning (PR), with the latter subsuming belief networks, chaos theory and parts of learning theory. What is important to note is that soft computing is not a mélange. Rather, it is a partnership in which each of the partners contributes a distinct methodology for addressing problems in its domain. In this perspective, the principal constituent methodologies in SC are complementary rather than competitive. Furthermore, soft computing may be viewed as a foundation component for the emerging field of conceptual intelligence.

In the present chapter, however, we use consistently the term "neural networks" (NN) instead of "neural computing," and we use the more general "evolutionary computation" (EC) as a replacement for "genetic computing." Otherwise, we follow strictly Zadeh's original definition of soft computing (SC). Furthermore, it should be pointed out that many people actually use the word "computing" when they mean hard computing (HC) and that they use "computational intelligence" as a substitute of soft computing. Soft computing is a low-level cognition in the style of the human mind as well as evolution of species in the nature, and it is in contrast to symbolic artificial intelligence (AI) that consists of expert systems, machine learning, and case-based reasoning. The mainstream AI is firmly committed to hard computing instead of soft computing.

SC is already a significant field of research and development; nearly 64,000 soft computing-related journal and conference papers were published between 1989 and 2002 by the IEEE and IEE together. Figure 1.1 illustrates the growth of published journal and conference articles on soft computing during those 14 years (based on the IEEE *Xplore*™ database; data collected on December 10, 2003). From 1989 to 1994, the number of conference papers was growing rather linearly, while the five-year period 1994–1998 was surprisingly flat. The year 2002 set a new record of conference publishing—nearly 6000 publications in one year. On the other hand, the number of journal articles has remained practically constant (around 1100) during the past few years. Therefore, it can be stated that soft computing has an established position in the research community, as well as a stabilizing publishing volume. It should be remembered, however, that in parallel with the recognized IEEE and IEE journals, there is a growing number of new journals on soft computing and its constituents.

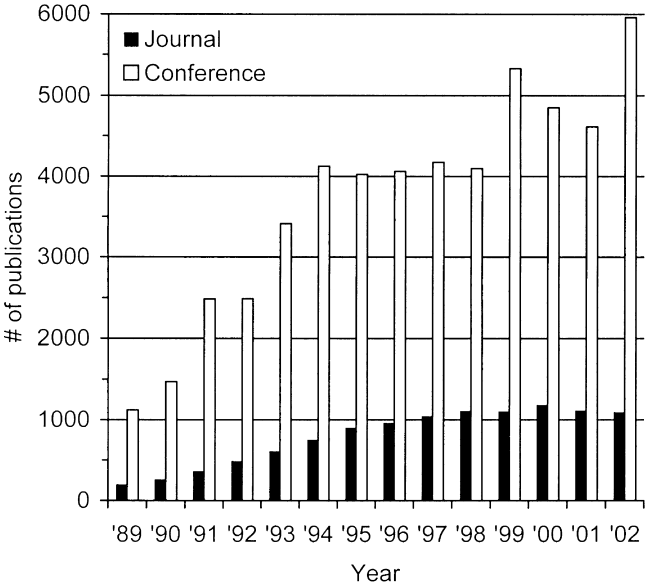


Figure 1.1. Growth of the number of IEEE/IEE journal and conference articles on soft computing and its applications.

Soft computing methods are penetrating gradually to industrial applications: They are no longer just in Japan and South Korea, but also in the United States and Europe. In the majority of such applications, SC is hidden inside systems or subsystems, and the end user does not necessarily know that soft computing methods are used in control, fault diagnosis, pattern recognition, signal processing, and so on. This is the case when SC is mainly utilized for improving the performance of conventional HC algorithms or even replacing them. Another class of applications uses soft computing for implementing computationally intelligent and user-friendly features that could not be realized competitively by HC. One important reason behind the growing industrial acceptance of SC is the existence of advantageous combinations of individual SC methods, such as fuzzy-neuro, genetic algorithm (GA) like fuzzy, and neuro-GA [5,6]. From the applications point of view, those combined constituents of SC are often more efficient than pure evolutionary computation, fuzzy logic, or neural networks.

1.1.2 Fusion of Soft-Computing and Hard-Computing Methodologies

Soft-computing methods are also commonly fused to more conventional hard-computing techniques in industrial products, instead of using SC unaccompanied. While academic researchers typically prefer *either* SC *or* HC methodologies, the complementary fusion of SC and HC is, increasingly, a natural way of thinking

for practicing engineers, who are facing demanding real-world problems to be solved competitively. Recently, Goldberg presented three principal requirements that are essential in making SC widespread among different applications [7]:

- Scalable results
- Practicality
- Little models or theory

Besides, he emphasized the importance of more complex integration of individual SC methods in developing autonomously intelligent systems. We would definitely include the complementary fusion of SC and HC in that requirements list.

There exist several vague ideas and informal definitions of the new concept "Fusion of Soft Computing and Hard Computing" in the researchers' and engineers' minds, but none of the used ones has yet obtained a dominating position. Our specific definition of this emerging concept can be stated as follows:

> An algorithm- or system-level union of particular soft computing and hard computing methodologies, in which either methodology is signal-, parameter-, or structure-wise dependent upon and receives functional reinforcement from the other.

The noteworthy role of the fusion of SC and HC was highlighted in a recent special issue, edited by Ovaska and VanLandingham, on "Fusion of Soft Computing and Hard Computing in Industrial Applications" that was published in the *IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews* [8]. A summarizing overview article [9] and eight original contributions [10–17], discussing advanced applications, were presented in that pioneering special issue. The feedback from the research and development community that the guest editors received during and after the editing process was mostly positive but also somewhat negative. This negative feedback was consistently related to the *vagueness* of the entire fusion concept, because, at that time, there was not available any systematic treatment on different fusion categories or their characteristic features in the context of integrated SC and HC. The qualitative work of Agarwal on combining neural and conventional paradigms for modeling, prediction, and control is a partial attempt toward that direction [18]. On the other hand, in the context of pure SC, a landmark discussion on various fusion categories was presented by Hayashi and Umano in 1993 [19] (in Japanese), and a discussion on the characteristic features for knowledge acquisition was presented by Furuhashi in 2001 [20]. An extended discussion of Hayashi's and Umano's fusion categories, including also genetic algorithms, is available in reference 5 (in English).

To establish a theoretical framework for the fusion of SC and HC, we will analyze below both the structural categories and characteristic features of the core fusion topologies, and thus provide a similar basis for the fusion of SC and HC that already exists for the fusion or hybridization of individual SC methods [5,20]. It is crucial for the successful development of computationally intelligent

hybrid systems to pay particular attention to the integration and interaction of SC and HC constituents. Our aim is to remove the criticized vagueness aspect that was mentioned above. In this way, the fusion of SC and HC may obtain a similar solid position in the research and development community as the fusion of individual SC methods attained already in the middle of the 1990s. The ultimate motivation behind our fusion discussion is naturally to advance the global use of SC in real-world applications.

This chapter is organized as follows. In Section 1.2, we introduce the structural categories of the fusion of SC and HC, as well as provide a brief discussion on the procedure how these categories were identified. Next, in Section 1.3, we discuss the characteristic features of SC and HC constituents in a few representative fusion applications. Section 1.4 gives a demonstrative categorization of several published fusions of SC and HC in industrial applications. Finally, Section 1.5 closes this chapter with a conclusion and a brief discussion on the acceptance of SC methods to industrial use.

1.2 STRUCTURAL CATEGORIES

By following the original and pragmatic thinking of Hayashi and Umano [19] and after analyzing a substantial number of application articles, we will next introduce 12 core categories for the complementary fusion of SC and HC. They belong to the class of "little models or theory" that was emphasized by Goldberg in the keynote address of reference 7. In his practical view, only little but sufficient models or theory are advancing the spread of SC, because research and development engineers prefer plain and straightforward solutions. Our core fusion categories are first summarized in Table 1.1. The adopted qualitative measure, *fusion grade* [5], describes the strength of a particular connection between SC and HC constituents. Table 1.2 gives brief definitions of the employed fusion grades: low, moderate, high, and very

TABLE 1.1. Structural Fusion Categories

#	Acronym	Description	Fusion Grade
1	SC&HC	SC and HC are isolated from each other	Low
2	SC/HC	SC and HC are connected in parallel	Moderate
3	SC\HC	SC with HC feedback	Moderate
4	HC\SC	HC with SC feedback	Moderate
5	SC-HC	SC is cascaded with HC	Moderate
6	HC-SC	HC is cascaded with SC	Moderate
7	HC=SC	HC-designed SC	High/very high
8	SC=HC	SC-designed HC	High/very high
9	HC+SC	HC-augmented SC	Very high
10	SC+HC	SC-augmented HC	Very high
11	HC//SC	HC-assisted SC	Very high
12	SC//HC	SC-assisted HC	Very high

TABLE 1.2. Definitions of Fusion Grades

Fusion Grade	Qualitative Definition
Low	SC and HC in simultaneous or alternative use without explicit connections
Moderate	Connections on input/output signal level only
High	SC or HC used for determining values of internal parameters
Very high	HC or SC used for determining internal structure or generating/manipulating internal data

high. Below is a description of each core fusion category, as well as an introductory discussion on six supplementary categories.

1.2.1 Soft Computing and Hard Computing Are Isolated from Each Other

In SC&HC-type systems (Fig. 1.2), both the SC and HC constituents have their independent roles, and there is no explicit connection between them. A descriptive example of such a loose union is an advanced elevator group dispatcher, where the primary call allocation algorithm is usually based on SC methods (NN, FL, or EC), while the backup algorithm, which becomes effective should the hall call interface have a serious failure, is typically a simple HC procedure (finite-state machine). In such cases, the fusion grade is naturally low, because there is no direct information exchange between the individual SC and HC blocks. The mathematical mappings realized by SC&HC are formulated below:

$$\begin{aligned} \mathbf{y}_1(n) &= f_{\mathrm{SC}}(\boldsymbol{\theta}_{\mathrm{SC}};\mathbf{x}_1(n)) \\ \mathbf{y}_2(n) &= f_{\mathrm{HC}}(\boldsymbol{\theta}_{\mathrm{HC}};\mathbf{x}_2(n)) \end{aligned} \tag{1.1}$$

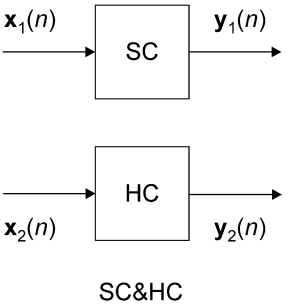


Figure 1.2. Partnership of soft computing and hard computing without explicit connection. All the arrow lines may contain several individual signals or data/parameter elements.

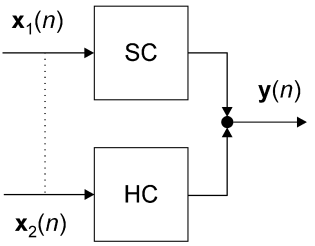
where $f_{SC}(\cdot;\cdot)$ and $f_{HC}(\cdot;\cdot)$ are arbitrary SC and HC algorithms (mapping functions), respectively; $\mathbf{x}(n)$ is the input and $\mathbf{y}(n)$ the output of such an algorithm; and $\boldsymbol{\theta}_{SC}$ and $\boldsymbol{\theta}_{HC}$ are the corresponding system parameters (coefficients, weights, etc.). All the boldface symbols represent vectors in this discussion; multiple inputs and outputs are identified by a subscript; and n is a discrete-time index. Although we are assuming that the SC and HC blocks are discrete-time systems, the presented fusion structures could straightforwardly be modified for continuous-time systems as well.

1.2.2 Soft Computing and Hard Computing Are Connected in Parallel

The second category, SC/HC, is presently of moderate practical interest. It offers possibilities for creating versatile combinations of SC and HC. In this parallel-connected topology, SC is complementing the behavior and capabilities of a primary HC system, or vice versa (Fig. 1.3). A typical example of such a configuration is an augmented linear controller (HC-type), where the nonlinearities and possible uncertainties of a plant are compensated by a parallel-connected SC-type controller. Such a closed-loop system could often be stabilized even without the supplementary SC controller, and the role of SC is merely to fine-tune the control performance and provide additional robustness. Besides, such parallel SC and HC systems are often considerably easier to design and more efficient to implement than pure HC systems with comparable performance. In this important category, the fusion grade is moderate. The characteristic function of SC/HC is defined as

$$\mathbf{y}(n) = f_{SC}(\boldsymbol{\theta}_{SC};\mathbf{x}_1(n)) \oplus f_{HC}(\boldsymbol{\theta}_{HC};\mathbf{x}_2(n)) \tag{1.2}$$

where the merging operator, $\oplus$, denotes either addition or combining of data/signal vectors, that is, $\mathbf{a} \oplus \mathbf{b}$ is either $\mathbf{a} + \mathbf{b}$ or $[\mathbf{a}\,\mathbf{b}]$, where $\mathbf{a}$ and $\mathbf{b}$ are arbitrary row/column vectors.



SC/HC

Figure 1.3. Soft computing and hard computing in parallel. Dotted line shows a typical connection.

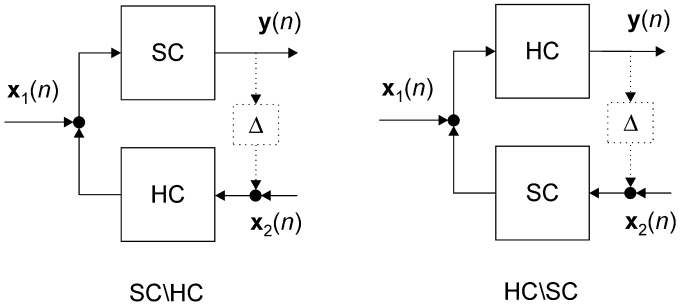


Figure 1.4. Soft computing with hard computing feedback and hard computing with soft computing feedback. Dotted line shows a typical connection.

1.2.3 Soft Computing with Hard-Computing Feedback and Hard Computing with Soft-Computing Feedback

Also the feedback parallel connections, SC\HC and HC\SC, are currently of some applications use. In these cases, the output of HC feedback or SC feedback is added to the input of the primary SC and HC blocks, respectively (Fig. 1.4). Notice that all fusion structures containing feedback must have a delay element Δ on the feedback loop to make the system realizable. The reversed parallel connections are used in a variety of control applications—for example, for improving the transient performance of a closed control loop. As with the forward parallel connection, SC/HC, the feedback parallel connections have also moderate fusion grades. The input–output mappings of SC\HC and HC\SC can be expressed in the following compact form:

$$\mathbf{y}(n) = f_{\mathrm{SC}}(\boldsymbol{\theta}_{\mathrm{SC}};[\mathbf{x}_1(n) \oplus f_{\mathrm{HC}}(\boldsymbol{\theta}_{\mathrm{HC}};\mathbf{x}_2(n))]) \tag{1.3}$$

$$\mathbf{y}(n) = f_{\mathrm{HC}}(\boldsymbol{\theta}_{\mathrm{HC}};[\mathbf{x}_1(n) \oplus f_{\mathrm{SC}}(\boldsymbol{\theta}_{\mathrm{SC}};\mathbf{x}_2(n))]) \tag{1.4}$$

1.2.4 Soft Computing Is Cascaded with Hard Computing or Hard Computing Is Cascaded with Soft Computing

There are two obvious alternatives to form hybrid cascades of SC and HC algorithms, SC-HC or HC-SC, depending on the sequential order of those functional blocks (Fig. 1.5). In these widespread configurations, the first block is usually a kind of pre-processor and the second one acts as a post-processor. A typical example of the SC-HC configuration is a dual-stage optimization procedure, where the initial optimization phase is carried out by evolutionary computation (global search), and the intermediate result is then refined by some gradient-based algorithm (local search) to yield the final optimum. This kind of complementary fusion leads to computationally efficient "global" optimization. It should be pointed out, however, that such a hybrid method cannot be applied if the local gradient of the criterion function is incalculable or difficult to obtain. Besides, when the

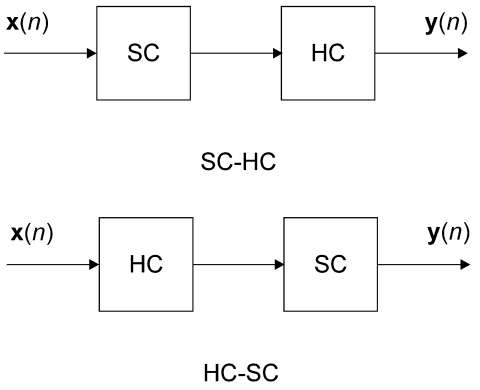


Figure 1.5. Cascades of soft computing and hard computing.

surface of the criterion function is very complicated, local gradient plays a smaller role even in the local search. Therefore, the dual-stage optimization procedure may offer clear advantages in specific applications, but is not as general-purpose a technique as EC alone. Moreover, instead of using a gradient-based algorithm in the second optimization stage, also other local-information-based algorithms (e.g., the simplex algorithm [21]) are applicable. On the other hand, a typical HC-SC configuration is an SC-based pattern recognition system, where the feature vectors are first constructed by HC-based linear/nonlinear filters or interdomain transforms. Soft computing techniques are then used for automatic pattern recognition or classification tasks. Also these cascade configurations have moderate fusion grades. Equations (1.5) and (1.6) describe the cascade structures SC-HC and HC-SC, respectively:

$$\mathbf{y}(n) = f_{\mathrm{HC}}(\boldsymbol{\theta}_{\mathrm{HC}}; f_{\mathrm{SC}}(\boldsymbol{\theta}_{\mathrm{SC}}; \mathbf{x}(n))) \tag{1.5}$$

$$\mathbf{y}(n) = f_{\mathrm{SC}}(\boldsymbol{\theta}_{\mathrm{SC}}; f_{\mathrm{HC}}(\boldsymbol{\theta}_{\mathrm{HC}}; \mathbf{x}(n))) \tag{1.6}$$

1.2.5 Soft-Computing-Designed Hard Computing and Hard-Computing-Designed Soft Computing

The next two combinations, SC=HC and HC=SC, have high or very high fusion grades. This is because SC is explicitly assisting the design or configuring of a HC system, or vice versa (Fig. 1.6). A usual example of SC=HC is a simple linear controller with SC-based gain scheduling for improved handling of varying dynamical characteristics. Such a hybrid controller could be tuned coarsely by experienced human operators; and the gain scheduling extension is further adjusting the control parameters within the predefined stability and performance range. In contrast, the conventional back-propagation training of layered neural networks is actually an HC=SC-type fusion. Another example of the HC=SC-type fusion is a dynamic neural network model with the computational structure and possibly

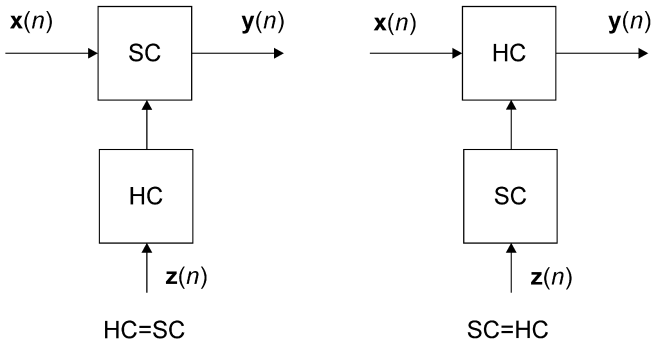


Figure 1.6. Hard-computing-designed soft computing and soft-computing-designed hard computing.

some physical parameters taken from a traditional differential or difference equations-based model. The SC=HC and HC=SC mappings can be expressed conveniently by the following formulae:

$$\mathbf{y}(n) = f_{\mathrm{SC}}\left(\overbrace{\tilde{\boldsymbol{\theta}}_{\mathrm{SC}}, f_{\mathrm{HC}}(\boldsymbol{\theta}_{\mathrm{HC}}; \mathbf{z}(n))}^{\text{Parameters}}; \mathbf{x}(n)\right) \tag{1.7}$$

$$\mathbf{y}(n) = f_{\mathrm{HC}}\left(\overbrace{\tilde{\boldsymbol{\theta}}_{\mathrm{HC}}, f_{\mathrm{SC}}(\boldsymbol{\theta}_{\mathrm{SC}}; \mathbf{z}(n))}^{\text{Parameters}}; \mathbf{x}(n)\right) \tag{1.8}$$

Here, the principal SC (or HC) algorithm has two classes of parameters: internal parameters, $\tilde{\boldsymbol{\theta}}_{\bullet \mathrm{C}}$, and externally supplied parameters, $f_{\bullet \mathrm{C}}(\boldsymbol{\theta}_{\bullet \mathrm{C}}, \mathbf{z}(n))$. They both are needed for providing the final output. In some applications, however, the principal SC (or HC) algorithm may not have any internal parameters, but all the necessary parameters are provided by an external HC (or SC) algorithm.

1.2.6 Hard-Computing-Augmented Soft Computing and Soft-Computing-Augmented Hard Computing

The next intimate structures, HC+SC and SC+HC, have very high fusion grades (Fig. 1.7). In the former case, the HC block is first extracting some internal data, $\mathbf{z}(n)$, from the SC algorithm and, after some further processing, the resulting HC output is added or combined to the principal SC output. Moreover, in the latter case, the SC block is processing certain internal data from the HC algorithm, and the final output is the sum or combination of the individual HC and SC outputs. These two categories are not widely used in real-world applications, but they offer great potential for developing advanced fusions of SC and HC methodologies.

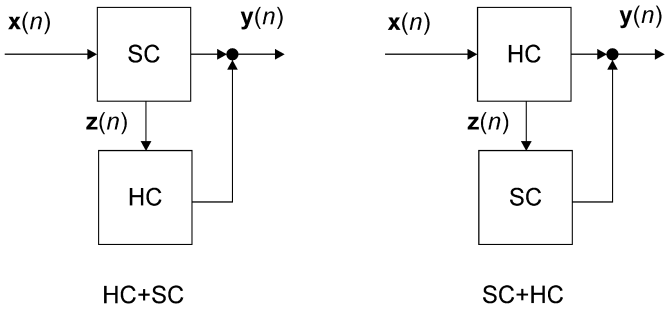


Figure 1.7. Hard-computing-augmented soft computing and soft-computing-augmented hard computing.

In Section 1.3, we will identify one promising HC+SC-type fusion with emerging application interest. HC+SC and SC+HC structures are formulated as

$$\mathbf{y}(n) = f_{\mathrm{SC}}(\boldsymbol{\theta}_{\mathrm{SC}};\mathbf{x}(n)) \oplus f_{\mathrm{HC}}(\boldsymbol{\theta}_{\mathrm{HC}};\mathbf{z}(n)) \tag{1.9}$$

$$\mathbf{y}(n) = f_{\mathrm{HC}}(\boldsymbol{\theta}_{\mathrm{HC}};\mathbf{x}(n)) \oplus f_{\mathrm{SC}}(\boldsymbol{\theta}_{\mathrm{SC}};\mathbf{z}(n)) \tag{1.10}$$

1.2.7 Hard-Computing-Assisted Soft Computing and Soft-Computing-Assisted Hard Computing

HC-assisted SC and SC-assisted HC are master–slave-type structures that are used moderately/sparsely in real-world applications. In HC/SC, some internal data are first extracted from the SC algorithm (master), processed by the specific HC algorithm (slave), and fed back into the SC algorithm, which completes the primary task. Thus, the HC constituent is an integral part of the computational procedure. An example of this kind of fusion is a hybrid genetic algorithm with Lamarckian or Baldwinian local search strategies [22], where a local search is following the selection–crossover–mutation chain in the repeated evolution loop; and the role of local search is to refine the chromosome population yielding to a new population for fitness evaluation and further genetic manipulations. SC//HC, on the other hand, is an analogous symbiosis between a principal HC algorithm and an assisting SC algorithm. Both of these structures (Fig. 1.8) have very high fusion grades, and they carry out the following input–output mappings:

$$\mathbf{y}(n) = f_{\mathrm{SC}}\left(\tilde{\boldsymbol{\theta}}_{\mathrm{SC}};\overbrace{\mathbf{x}(n), f_{\mathrm{HC}}(\boldsymbol{\theta}_{\mathrm{HC}};\mathbf{z}(n))}^{\text{Inputs}}\right) \tag{1.11}$$

$$\mathbf{y}(n) = f_{\mathrm{HC}}\left(\tilde{\boldsymbol{\theta}}_{\mathrm{HC}};\overbrace{\mathbf{x}(n), f_{\mathrm{SC}}(\boldsymbol{\theta}_{\mathrm{SC}};\mathbf{z}(n))}^{\text{Inputs}}\right) \tag{1.12}$$

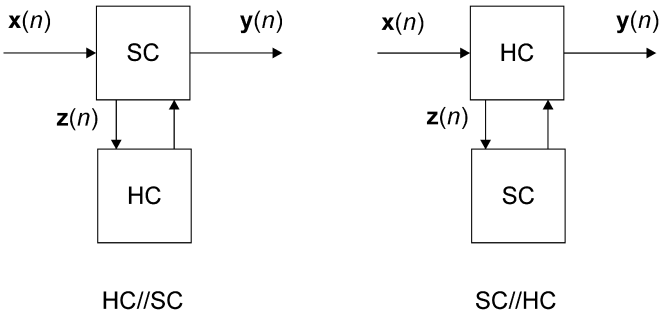


Figure 1.8. Hard-computing-assisted soft-computing and soft-computing-assisted hard computing.

1.2.8 Supplementary Categories

In classifying the different kind of fusions of neural networks and fuzzy logic, Hayashi et al. proposed 11 structural categories [5]. This is about the same as our 12 core categories for the fusion of SC and HC. On the other hand, Agarwal identified only the three most obvious fusion structures of neural networks and conventional paradigms (i.e., NN/HC, NN-HC, and HC-NN) in his article that was published in 1995 [18]. Another relevant discussion on integration architectures, related to symbolic artificial intelligence (SA), FL, GA, and NN, was presented by Fu in his 1996 conference tutorial [23]. He identified five integration architectures with three coupling strengths. Those hybrid architectures share obvious similarities with our core categories.

Because the field of HC is much wider and more heterogeneous than the field of SC, we decided to classify only the universal categories. If the very details of specific HC algorithms were taken into consideration, the number of fusion categories could be increased considerably. Since the fusion structures of Figs. 1.2–1.8 have the minimum number of SC and HC blocks—only one of each type—they are said to be canonic. It should be pointed out that more complex, large-scale systems are naturally modeled as interconnections of these canonic building blocks.

Most of the fusion categories discussed above were created following the synergy between the fusion of SC and HC and the fusion of individual SC methods [5,19]. In addition, numerous journal and conference articles were thoroughly analyzed to verify the coverage of intermediate category sets. After identifying a relevant structural category, like HC\SC, its reverse category, SC\HC, was also carefully considered for the final category set. This particular reverse category, although not in wide use among practicing engineers, offers apparent potential for developing advantageous fusions of SC and HC. Moreover, Fig. 1.9 illustrates the reverse categories of HC+SC and SC+HC. Due to the small number of existing applications, these two categories were not included in the core category set of Table 1.1. Equations (1.13) and (1.14) give the mappings of the reversed SC+HC and

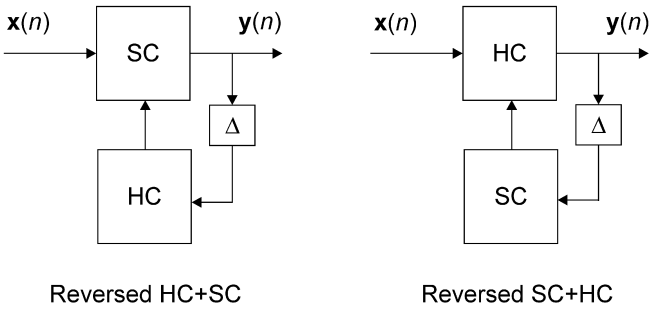


Figure 1.9. Reversed HC+SC and SC+HC structures.

HC+SC structures, respectively:

$$\mathbf{y}(n) = f_{\mathrm{SC}}\left(\boldsymbol{\theta}_{\mathrm{SC}}; \overbrace{\mathbf{x}(n), f_{\mathrm{HC}}(\boldsymbol{\theta}_{\mathrm{HC}}; \mathbf{y}(n-\Delta))}^{\text{Inputs}}\right) \tag{1.13}$$

$$\mathbf{y}(n) = f_{\mathrm{HC}}\left(\boldsymbol{\theta}_{\mathrm{HC}}; \overbrace{\mathbf{x}(n), f_{\mathrm{SC}}(\boldsymbol{\theta}_{\mathrm{SC}}; \mathbf{y}(n-\Delta))}^{\text{Inputs}}\right) \tag{1.14}$$

It should be noted that here the fusion structures contain a feedback loop and, therefore, the value of $\mathbf{y}(n)$ must be delayed at least by one sampling period—that is, $\Delta \geq 1$—to make the systems realizable. These reversed structures have two kinds of inputs: the primary input, $\mathbf{x}(n)$, and the auxiliary input, $f_{\bullet \mathrm{C}}(\boldsymbol{\theta}_{\bullet \mathrm{C}}; \mathbf{y}(n-\Delta))$.

Structural transposition or flow-graph reversal of the defined core categories leads to two additional structures (Fig. 1.7 → Fig. 1.10) that may offer innovation potential. Their mathematical mappings are formulated in Eqs. (1.15) and (1.16). However, to our best knowledge, they are not commonly in applications use and,

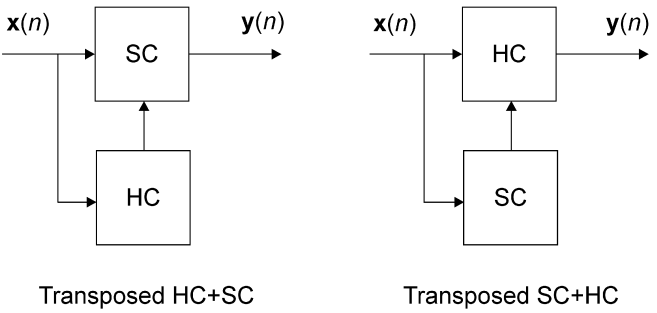


Figure 1.10. Transposed HC+SC and SC+HC structures.

therefore, we decided to omit them from Table 1.1. On the other hand, the fusion categories of Fig. 1.10 could be used at least for initializing the internal states or memory of SC or HC models.

$$\mathbf{y}(n) = f_{\mathrm{SC}}\left(\boldsymbol{\theta}_{\mathrm{SC}}; \overbrace{\mathbf{x}(n), f_{\mathrm{HC}}(\boldsymbol{\theta}_{\mathrm{HC}}; \mathbf{x}(n))}^{\text{Inputs}}\right) \tag{1.15}$$

$$\mathbf{y}(n) = f_{\mathrm{HC}}\left(\boldsymbol{\theta}_{\mathrm{HC}}; \overbrace{\mathbf{x}(n), f_{\mathrm{SC}}(\boldsymbol{\theta}_{\mathrm{SC}}; \mathbf{x}(n))}^{\text{Inputs}}\right) \tag{1.16}$$

For completeness of this category presentation, reversed versions of the transposed HC+SC and SC+HC are depicted in Fig. 1.11. Both of these structures have also a feedback loop that must contain a delay element Δ. Moreover, they can be expressed mathematically as

$$\mathbf{y}(n) = f_{\mathrm{SC}}(\boldsymbol{\theta}_{\mathrm{SC}}; [\mathbf{x}(n) \oplus f_{\mathrm{HC}}^{\Delta}(\boldsymbol{\theta}_{\mathrm{HC}}; \mathbf{z}(n))]) \tag{1.17}$$

$$\mathbf{y}(n) = f_{\mathrm{HC}}(\boldsymbol{\theta}_{\mathrm{HC}}; [\mathbf{x}(n) \oplus f_{\mathrm{SC}}^{\Delta}(\boldsymbol{\theta}_{\mathrm{SC}}; \mathbf{z}(n))]) \tag{1.18}$$

In Eqs. (1.17) and (1.18), the necessary loop delay, Δ, is shown as a superscript of the HC and SC algorithms, respectively.

It should be pointed out that similar to the core HC+SC and SC+HC structures of Fig. 1.7, the six supplementary fusion structures of Figs. 1.9–1.11 are also intimate fusions of SC and HC and, therefore, have very high fusion grades. In Section 1.4, we will classify a representative collection of application papers on the fusion of SC and HC according to the structural categories of Table 1.1.

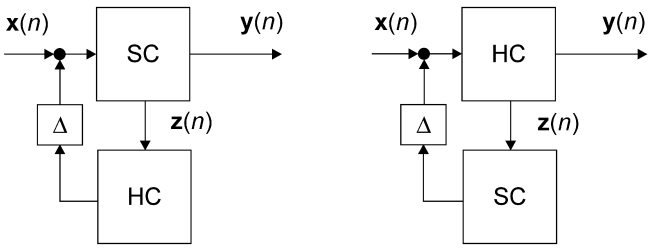


Figure 1.11. Transposed-reversed HC+SC and SC+HC structures.

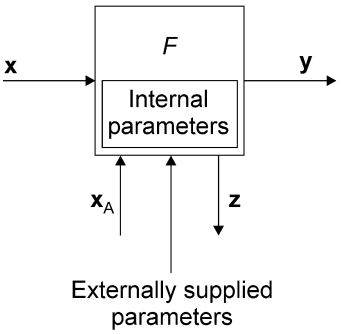


Figure 1.12. Generic SC or HC system with signal/data inputs and outputs, as well as externally supplied system parameters.

1.2.9 General Soft-Computing and Hard-Computing Mapping Functions

We will next define mathematically the general mapping functions (algorithms), $f_{SC}(\cdot;\cdot)$ and $f_{HC}(\cdot;\cdot)$, used in Eqs. (1.1)–(1.18). Our compact definition follows the operational structure and notations of Fig. 1.12. It should be noticed, however, that only SC=HC- and HC=SC-type fusions use the externally supplied parameters that are shown in Fig. 1.12. The nonlinear or linear mapping function, $F \in \{f_{SC}, f_{HC}\}$, from the primary input vector, $\mathbf{x} \in \Re^N$, and the auxiliary input vector (optional), $\mathbf{x}_A \in \Re^K$, to the internal output vector (optional), $\mathbf{z} \in \Re^L$, and the primary output vector, $\mathbf{y} \in \Re^M$, can be expressed as

$$F{:}\langle \mathbf{x} \in \Re^N,\ \mathbf{x}_A \in \Re^K \rangle \longrightarrow \langle \mathbf{z} \in \Re^L,\ \mathbf{y} \in \Re^M \rangle \tag{1.19}$$

where $\mathbf{x} = [x_1, x_2, \ldots, x_N]^T$, $\mathbf{x}_A = [x_{A,1}, x_{A,2}, \ldots, x_{A,K}]^T$, $\mathbf{z} = [z_1, z_2, \ldots, z_L]^T$, and $\mathbf{y} = [y_1, y_2, \ldots, y_M]^T$. Here, the auxiliary input vector, $\mathbf{x}_A$, and the internal output vector, $\mathbf{z}$, can be considered as partial- or full-state data of the dynamical mapping function, F. All the input and output spaces may naturally have different dimensions; therefore, this general definition covers both single-input single-output (SISO)-type and multiple-input multiple-output (MIMO)-type systems.

1.3 CHARACTERISTIC FEATURES

The central motivation behind fusing elemental SC and HC methods is the aim to overcome the limitations of individual methods. This applies to all fusion categories; the resulting hybrid systems are expected to have such valuable characteristics that would not be either possible or practical to obtain by using a single methodological class only. Table 1.3 contains the most important characteristic features of evolutionary computation [24], fuzzy logic [25], and neural networks [26]. All these constituents of SC have useful functional characteristics that are potentially needed in

TABLE 1.3. Characteristic Features of EC, FL, and NN Methods

SC Method	Characteristic Features
EC	• Search methods inspired by the Neo-Darwinian paradigm • No gradient of the error surface needed • Avoidance of local optima through random perturbations • Computationally intensive
FL	• Rule-based approach • Tolerance for imprecision, partial truth, and uncertainty • Capability to approximate any continuous function or system • Possibility to utilize human intuition
NN	• Black box approach • Massively parallel distributed structure • Supervised or unsupervised learning from data • Generalization ability • Capability to approximate any continuous function or system

real-world applications, because the conceptual structure of HC is overly precise in relation to the imprecision of the world around us, and there is a growing demand for computationally intelligent hybrid systems.

The HC field, on the other hand, is more diverse than the SC field, and thus it is not feasible to create a comprehensive table that would contain the characteristic features of all specific classes of hard computing methods. Therefore, we built Table 1.4, which lists only five representative HC methods with their primary characteristic features. The fusion needs and potential of these HC methods are discussed below.

1.3.1 Proportional Integral Derivative Controllers

Proportional integral derivative (PID) controllers are certainly the most widely used control algorithms in industrial applications. Thus, practicing engineers favor them also in developing future products and systems. However, the requirements of control performance are continuously increasing, and they cannot always be met by using a fixed linear approximation unaccompanied. This problem has been solved successfully, for example, by complementary EC=HC-, FL=HC-, and NN=HC-type fusions of SC and HC [27–29]. In such hybrid systems, the SC constituent is tuning the PI(D) control parameters either incrementally or continuously. To ensure the stability of auto-tuning PI(D) systems, the range and tuning interval of control parameters must be constrained appropriately.

1.3.2 Physical Models

A variety of process models for estimation, prediction, control, and simulation is developed by modeling the entire physical system using its characteristic differential

TABLE 1.4. Characteristic Features of Specific HC Methods

HC Method	Characteristic Features
PID controller	• Simplified linear approximation • Manual tuning possible • Computationally efficient
s-domain models with physical origin	• Solid physical basis (differential equations) • Sensitivity to parameter variation • Tedious derivation of high-order models
Fletcher–Powell optimization algorithm	• Makes use of derivatives of the error surface • Variable metric method • Efficient unconstrained optimization
General parameter adaptation method	• Reduced-rank adaptation • Computationally efficient • Applicable with many computational structures
Stochastic system simulation	• Suitable for various kind of complex systems • Demanding phase of model building • Time-consuming brute force technique

or difference equations. This requires a thorough understanding of the entire system to be modeled. On the other hand, such core knowledge is usually available in matured research and development organizations. Nevertheless, differential equation models and their *s*-domain counterparts become cumbersome to deal with if the system to be modeled is considerably nonlinear or time-variant, or a high-order model, including also critical secondary effects, is needed to obtain the required accuracy. In these demanding cases, the physics-originated low- or moderate-order linear models have been complemented by neural networks or fuzzy logic-based nonlinear models [30]. The linear approximation remains the "backbone," and its behavior is enhanced by NN/HC- or FL/HC-type fusions, where the SC constituent compensates for system nonlinearities, parameter uncertainty, or other inadequacies of the primary HC model.

1.3.3 Optimization Utilizing Local Information

The Fletcher–Powell optimization algorithm [31] is a widely used unconstrained optimization method that makes use of derivatives of the error surface. This kind of variable-metric algorithm works usually well when the error surface is moderately difficult. However, with highly peaky error landscapes that are typical for complex nonlinear optimization problems, all the derivatives-based methods have a notable possibility of getting stuck on some (poor) local optimum. This undesired possibility has been reduced substantially in advanced HC//EC- and EC-HC-type hybrid optimization procedures [17,32].

1.3.4 General Parameter Adaptation Algorithm

Various neural network models are used frequently for time-series prediction in embedded real-time systems—for example, in automatic fault detection and systems' state estimation. In many applications, however, the predicted time series has specific time-varying characteristics that would actually need an adaptive model for maximizing the prediction accuracy. Unfortunately, fully adaptive neural network implementations have high computational burden. Thus, they are rarely used in real-time applications. With radial basis function networks (RBFNs), such a disturbing time-variance problem was relieved considerably by introducing an intimate HC+NN-type fusion structure [33]. In that hybrid structure, the HC constituent is a simple reduced-rank adaptation scheme, the general parameter (GP) method, which tunes the behavior of the RBFN model around a nominal operation point. The desired input–output mapping at the nominal operation point is represented accurately by the fixed RBFN part, and the simple general parameter extension is able to compensate for reasonable time-varying characteristics in the time series. Similar intimate fusion schemes could also be applied with other layered neural networks that are used in predictive configurations. For example, the fixed output layer of multilayer perceptron (MLP) networks could be complemented by a least-mean-square (LMS)-type adaptive linear combiner [34].

1.3.5 Stochastic System Simulators

Stochastic system simulation is a widely applied technique for analyzing and optimizing complex systems that do not have tractable analytical models. There are several commercial simulation packages available with efficient model building functions and versatile visualization capabilities of simulation results. Monte Carlo-method-based stochastic simulation is considered as a "brute force" technique for optimizing parameters of complex systems. From the computation time point of view, it would be desired to keep the number of simulation cycles at a minimum. On the other hand, that would affect directly the statistical reliability of the simulation results. Such a problem could be reduced by combining evolutionary computation with stochastic system simulation. In an HC//EC-type fusion, the stochastic system simulator is used to compute the values of the fitness function needed in evolutionary optimization [17], and the EC constituent provides an intelligent search of the enormous solution base.

1.3.6 Discussion and Extended Fusion Schemes

Table 1.5 summarizes the principal characteristic features of the fusions of SC and HC that were discussed above. The constructive fusion approaches have provided improved results over nonhybrid techniques. These improvements are chiefly in such areas as computational efficiency, design cost, robustness, and tractability. In addition, the SC-enhanced HC algorithms form a natural intermediate step on the methodological evolution path from pure HC toward dominating SC. Such an

TABLE 1.5. Characteristic Features of Representative Fusions

HC Constituent	Characteristic Features of the Fusion
PID controller	*General* • Automatic tuning of control parameters • Manual tuning still possible • Stability predictable *EC=HC* • Global search of control parameters *FL=HC* • Possibility to utilize human intuition *NN=HC* • Learning from data
s-domain models with physical origin	*General* • Solid physical basis • Ability to handle nonlinearities *FL/HC* • Possibility to utilize human intuition *NN/HC* • Learning from data
Fletcher–Powell optimization algorithm	*EC-HC* • Global search • Efficient unconstrained optimization
General parameter adaptation method	*HC+NN* • Partially adaptive neural network model • Computationally efficient
Stochastic system simulation	*HC//EC* • Optimization method for complex systems • Global search • Demanding phase of model building

evolutionary multistep path is highly preferred in industrial R&D organizations and among practicing engineers.

As mentioned earlier in this chapter, our structural categorization is based on a little model approach. In this approach, each SC or HC constituent is represented by a compact model, and the categorization is based on the types of *interconnections* of these models. However, the fusion of SC and HC could further be extended into "Soft Computing in a Hard Computing Framework" or "Hard Computing in a Soft Computing Framework." In such extended fusion schemes, the resulting system contains only SC constituents that are being implemented in an HC framework, or only HC constituents that are being embedded in an SC framework. Representative examples of such fusions are: genetic algorithms formulated in an object-oriented framework [35]; fuzzy rules coded in a Petri net [36]; neural networks embedded in inverse-model-based control techniques [37]; and various mathematical programming methods combined in a genetic algorithm architecture [38].

1.4 CHARACTERIZATION OF HYBRID APPLICATIONS

In Table 1.6, we list a demonstrative collection of reference articles that contain some fusion structure(s) belonging to one of the core categories of Table 1.1 [10,17,27,32,33,39–54]. To make the reference table more instructive, the SC constituent was divided into three subconstituents: evolutionary computation, fuzzy logic, and neural networks. Because the first category, SC&HC, was not included in this consideration, Table 1.6 contains all together 33 possible fusions of SC and HC methodologies. The carefully selected reference articles, as well as the

TABLE 1.6. Fusion Examples from the Literature

Category	SC	HC	Reference
SC/HC	EC	Mixed-integer linear programming	39
	FL	P-type controller	40
	NN	Two PI controllers with cascaded notch filters	41
SC\HC	EC	Linear combination of visual perception	42
	FL	First-order linear filter	43
	NN	Noise filter	44
HC\SC	EC	Spacecraft navigation and control	45
	FL	Feedback linearization controller	46
	NN	Linearizing control	47
SC-HC	EC	Powell's unconstrained optimization method	32
	FL	Linear quadratic Gaussian-type controller	48
	NN	Narrow-band lowpass filters (integrators)	49
HC-SC	EC	Linear programming for initial population	50
	FL	Extended Kalman filter	48
	NN	Feature extraction algorithm	10
HC=SC	EC	Constraint satisfaction techniques	51
	FL	Gradient-descent method	40
	NN	Newton–Gauss optimization algorithm	52
SC=HC	EC	Two PI controllers with cross-coupling	27
	FL	Dynamic programming algorithm	53
	NN	Optimizing long-range predictive controller	54
HC+SC	EC	—	—
	FL	—	—
	NN	General parameter adaptation algorithm	33
SC+HC	EC	—	—
	FL	—	—
	NN	—	—
HC//SC	EC	Stochastic system simulator	17
	FL	—	—
	NN	—	—
SC//HC	EC	—	—
	FL	—	—
	NN	—	—

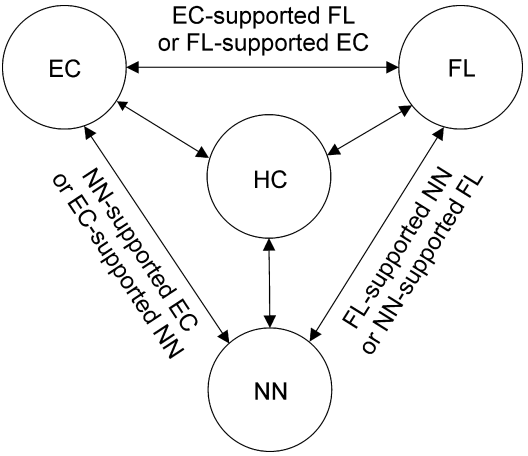


Figure 1.13. Fusion of hard-computing and individual/fused soft-computing methods.

recent overview article [9], guide the reader to the realm of specific fusion implementations. Besides, advanced implementation aspects are discussed systematically in the subsequent chapters of the present book.

As shown in Table 1.6, the number of reported applications with respect to HC+SC, SC+HC, HC//SC, and SC//HC is still very limited. However, as research and development on various applications with respect to the fusion of SC and HC further evolve, these intimate structures will likely become a promising area to be explored.

One noticeable trend in the fusion of SC and HC is the simultaneous partnership of HC and *multiple* SC methodologies as illustrated in Fig. 1.13 [6]. Here, the fused SC methodologies are targeted to provide intelligent behavior or high machine IQ (MIQ) for a variety of advanced systems and products. This attractive goal of intelligent behavior was defined by Fogel et al. in 1965 [55]:

> ... intelligent behavior can result from an ability to predict the environment coupled with the selection of an algorithm which permits the translation of each prediction into a suitable response.

Already in that early definition, intelligent behavior was seen as a coupling of some reliable predictor and a robust translator; and these individual units could be realized efficiently and competitively using the constructive fusion of SC and HC.

1.5 CONCLUSIONS AND DISCUSSION

We introduced 12 structural core categories for the fusion of SC and HC. Figure 1.14 shows a qualitative diagram, where each of these categories is positioned according

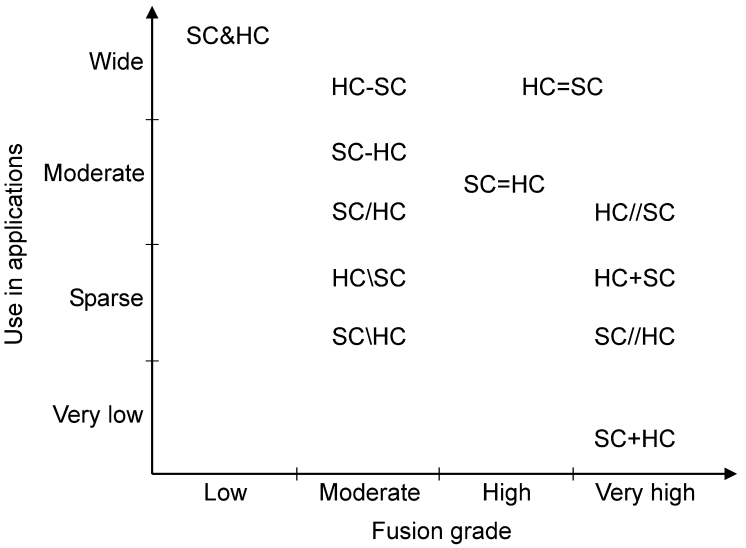


Figure 1.14. Qualitative classification of core fusion categories.

to its fusion grade and current use in engineering applications. By comparing our core categories to those related to the fusion of individual SC methods and after reviewing a large amount of application articles, it can be concluded that the fusion of SC and HC is already a well-defined concept for developing computationally intelligent hybrid systems that largely overcome the shortages and limitations of individual SC and HC methods. In addition to those 12 core categories, we also identified six supplementary categories that offer considerable innovation potential for future applications.

Although the HC and SC research communities are quite separated, it is good to observe that many SC-oriented researchers and engineers are openly favoring the various fusion opportunities. However, the HC community generally seems to be more reluctant in applying the complementing SC techniques. This situation may change progressively following positive experiences, along with the associated evolving emphasis in various engineering curricula. Currently, there is usually an unfortunate division between HC- and SC-oriented courses related to major application topics like electric power systems, industrial electronics, systems engineering, and telecommunications. This separation is clearly hindering the industrial acceptance of SC, because those separated courses do not typically consider the methodological fusion approach at all, but mostly neglect totally the other class of available methodologies.

As a final thought, this fusion discussion is closely related to Zadeh's original aims of fuzzy logic providing an advanced interface between imprecise human reasoning and the highly precise calculations of digital computers.

ACKNOWLEDGMENTS

This work was supported by the Academy of Finland under Grants 80100 and 203436. The author wishes to thank Drs. Randy L. Haupt and Tamal Bose (Electrical and Computer Engineering, Utah State University) for providing the visiting scientist position that made it possible to complete this manuscript in the summer of 2003. Many discussions with Drs. Akimoto Kamiya (Information Engineering, Kushiro National College of Technology) and Hugh F. VanLandingham (Electrical and Computer Engineering, Virginia Polytechnic Institute and State University) were extremely useful.

REFERENCES

1. L. A. Zadeh, "A Definition of Soft Computing in Greater Detail," WWW page, 1991/1994. Available from <http://www.cs.berkeley.edu/~mazlack/BISC/BISC-DBM-soft.html>.
2. L. A. Zadeh, "Fuzzy Sets," *Information and Control* **8**, 338–353 (1965).
3. L. A. Zadeh, "Outline of a New Approach to the Analysis of Complex Systems and Decision Processes," *IEEE Transactions on Systems, Man, and Cybernetics* **3**, 28–44 (1973).
4. L. A. Zadeh, "Possibility Theory and Soft Data Analysis," in L. Cobb and R. M. Thrall, eds., *Mathematical Frontiers of the Social and Policy Sciences*, Westview Press, Boulder, CO, 1981, pp. 69–129.
5. I. Hayashi, M. Umano, T. Maeda, A. Bastian, and L. C. Jain, "Acquisition of Fuzzy Knowledge by NN and GA—a Survey of the Fusion and Union Methods Proposed in Japan," *Proceedings of the 2nd International Conference on Knowledge-Based Intelligent Electronic Systems*, Adelaide, Australia, Apr. 1998, pp. 69–78.
6. H. Takagi, "R&D in Intelligent Technologies: Fusion of NN, FS, GA, Chaos, and Human," *Half-Day Tutorial/Workshop, IEEE International Conference on Systems, Man, and Cybernetics*, Orlando, FL, Oct. 11, 1997.
7. D. E. Goldberg, "A Meditation on the Application of Soft Computing and Its Future," in R. Roy, M. Köppen, S. Ovaska, T. Furuhashi, and F. Hoffmann, eds., *Soft Computing and Industry: Recent Applications*, Springer-Verlag, London, UK, 2002, pp. XV–XVIII.
8. S. J. Ovaska and H. F. VanLandingham, "Guest Editorial: Special Issue on Fusion of Soft Computing and Hard Computing in Industrial Applications," *IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews* **32**, 69–71 (2002).
9. S. J. Ovaska, H. F. VanLandingham, and A. Kamiya, "Fusion of Soft Computing and Hard Computing in Industrial Applications: An Overview," *IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews* **32**, 72–79 (2002).
10. B. Sick, "Fusion of Hard and Soft Computing Techniques in Indirect, Online Tool Wear Monitoring," *IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews* **32**, 80—91 (2002).
11. C. Shi and A. D. Cheok, "Performance Comparison of Fused Soft Control/Hard Observer Type Controller with Hard Control/Hard Observer Type Controller for Switched

Reluctance Motors," *IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews* **32**, 99–112 (2002).

12. G. D. Buckner, "Intelligent Bounds on Modeling Uncertainties: Applications to Sliding Mode Control," *IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews* **32**, 113–124 (2002).
13. R. Sterritt and D. W. Bustard, "Fusing Hard and Soft Computing for Fault Management in Telecommunications Systems," *IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews* **32**, 92–98 (2002).
14. M. Oosterom, R. Babuska, and H. B. Verbruggen, "Soft Computing Applications in Aircraft Sensor Management and Flight Control Law Reconfiguration," *IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews* **32**, 125–139 (2002).
15. J. J. Murray, C. J. Cox, G. G. Lendaris, and R. Saeks, "Adaptive Dynamic Programming," *IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews* **32**, 140–153 (2002).
16. S.-B. Cho, "Incorporating Soft Computing Techniques into a Probabilistic Intrusion Detection System," *IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews* **32**, 154–160 (2002).
17. R. H. Kewley and M. J. Embrechts, "Computational Military Tactical Planning System," *IEEE Transactions on Systems, Man, and Cybernetics—Part C: Applications and Reviews* **32**, 161–171 (2002).
18. M. Agarwal, "Combining Neural and Conventional Paradigms for Modeling, Prediction, and Control," *Proceedings of the 4th IEEE Conference on Control Applications*, Albany, NY, Sept. 1995, pp. 566–571.
19. I. Hayashi and M. Umano, "Perspectives and Trends of Fuzzy-Neural Networks," *Journal of the Japan Society of Fuzzy Theory and Systems* **5**, 178–190 (1993), in Japanese.
20. T. Furuhashi, "Fusion of Fuzzy/Neuro/Evolutionary Computing for Knowledge Acquisition," *Proceedings of the IEEE* **89**, 1266–1274 (2001).
21. L. Yang, J. Yen, A. Rajesh, and K. D. Kihm, "A supervisory Architecture and Hybrid GA for the Identifications of Complex Systems," *Proceedings of the 1999 Congress on Evolutionary Computation*, Washington, DC, July 1999, pp. 862–869.
22. D. Newth, M. Kirley, and D. G. Green, "An Investigation of the Use of Local Search in NP-Hard Problems," *Proceedings of the 26th Annual Conference of the IEEE Industrial Electronics Society*, Nagoya, Japan, Oct. 2000, **4**, pp. 2710–2715.
23. L. M. Fu, "Tutorial 5: Knowledge & Neural Heuristics," *Half-Day Tutorial/Workshop, IEEE International Conference on Neural Networks*, Washington, DC, June 2, 1996.
24. D. B. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, IEEE Press, Piscataway, NJ, 2000.
25. B. Kosko, *Fuzzy Engineering*, Prentice Hall, Upper Saddle River, NJ, 1996.
26. S. Haykin, *Neural Networks: A Comprehensive Foundation*, Prentice Hall, Upper Saddle River, NJ, 1999.
27. Y. Zhao, R. M. Edwards, and K. Y. Lee, "Hybrid Feedforward and Feedback Controller Design for Nuclear Steam Generators over Wide Range Operation Using Genetic Algorithm," *IEEE Transactions on Energy Conversion* **12**, 100–105 (1997).

28. K. C. Jeong, S. H. Kwon, D. H. Lee, M. W. Lee, and J. Y. Choi, "A Fuzzy Logic-Based Gain Tuner for PID Controllers," *Proceedings of the FUZZ-IEEE*, Anchorage, AK, May 1998, pp. 551–554.

29. S. Omatu, T. Iwasa, and M. Yoshioka, "Skill-Based PID Control by Using Neural Networks," *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, San Diego, CA, Oct. 1998, pp. 1972–1977.

30. Y. Tipsuwan and M.-Y. Chow, "Analysis of Training Neural Compensation Model for System Dynamics Modeling," *Proceedings of the International Joint Conference on Neural Networks*, Washington, DC, July 2001, pp. 1250–1255.

31. R. Fletcher and M. J. D. Powell, "A Rapidly Convergent Descent Method for Minimization," *Computer Journal* **6**, 163–168 (1963).

32. T. S. Kim and G. S. May, "Optimization of Via Formation in Photosensitive Dielectric Layers Using Neural Networks and Genetic Algorithms," *IEEE Transactions on Electronics Packaging and Manufacturing* **22**, 128–136 (1999).

33. D. F. Akhmetov, Y. Dote, and S. J. Ovaska, "Fuzzy Neural Network with General Parameter Adaptation for Modeling of Nonlinear Time-Series," *IEEE Transactions on Neural Networks* **12**, 148–152 (2001), errata published in *IEEE Transactions on Neural Networks* **12**, 443 (2001).

34. B. Widrow and S. D. Stearns, *Adaptive Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1985.

35. C. S. Krishnamoorthy, P. P. Venkatesh, and R. Sudarshan, "Object-Oriented Framework for Genetic Algorithms with Application to Space Truss Optimization," *Journal of Computing in Civil Engineering* **16**, 66–75 (2002).

36. W. Chiang, K. F. R. Liu, and J. Lee, "Bridge Damage Assessment Through Fuzzy Petri Net Based Expert System," *Journal of Computing in Civil Engineering* **14**, 141–149 (2000).

37. M. A. Hussain, "Review of the Applications of Neural Networks in Chemical Process Control—Simulation and Online Implementation," *Artificial Intelligence in Engineering* **13**, 55–68 (1999).

38. J. Rachlin, R. Goodwin, S. Murthy, R. Akkiraju, F. Wu, S. Kumaran, and R. Das, "A-Teams: An Agent Architecture for Optimization and Decision Support," in J. Mueller, M. Singh, and A. Rao, eds., *Lecture Notes in Artificial Intelligence: Intelligent Agents V*, Springer-Verlag, Heidelberg, Germany, 1999, **1555**, pp. 261–276.

39. T. Nishi, T. Inoue, and Y. Hattori, "Development of a Decentralized Supply Chain Optimization System," *Proceedings of the International Symposium on Design, Operation and Control of Next Generation Chemical Plants*, Kyoto, Japan, Dec. 2000, pp. 141–151.

40. J. Abonyi, L. Nagy, and F. Szeifert, "Adaptive Sugeno Fuzzy Control: A Case Study," in R. Roy, T. Furuhashi, and P. K. Chawdhry, eds., *Advances in Soft Computing: Engineering Design and Manufacturing*, Springer-Verlag, London, UK, 1999, pp. 135–146.

41. D. M. McDowell, G. W. Irwin, G. Lightbody, and G. McConnell, "Hybrid Neural Adaptive Control for Bank-to-Turn Missiles," *IEEE Transactions on Control Systems Technology* **5**, 297–308 (1997).

42. S. Schaal and D. Sternad, "Learning of Passive Motor Control Strategies with Genetic Algorithms," in L. Nadel and D. Stein, eds., *Lectures in Complex Systems*, Addison-Wesley, Boston, MA, 1992, pp. 631–643.

43. J. Abonyi, R. Babuska, M. Ayala Botto, F. Szeifert, and L. Nagy, "Identification and Control of Nonlinear Systems Using Fuzzy Hammerstein Models," *Industrial & Engineering Chemistry Research* **39**, 4302–4314 (2000).

44. Q. Zhang and S. J. Stanley, "Real-Time Water Treatment Process Control with Artificial Neural Networks," *Journal of Environmental Engineering* **125**, 153–160 (1999).

45. S. McClintock, T. Lunney, and A. Hashim, "A Genetic Algorithm Environment for Star Pattern Recognition," *Journal of Intelligent & Fuzzy Systems* **6**, 3–16 (1998).

46. L.-C. Lin and T.-B. Gau, "Feedback Linearization and Fuzzy Control for Conical Magnetic Bearings," *IEEE Transactions on Control Systems Technology* **5**, 417–426 (1997).

47. K. Fregene and D. Kennedy, "Control of a High-Order Power System by Neural Adaptive Feedback Linearization," *Proceedings of the IEEE International Symposium on Intelligent Control/Intelligent Systems and Semiotics*, Cambridge, MA, Sept. 1999, pp. 34–39.

48. A. Ben-Abdennour and K. Y. Lee, "An Autonomous Control System for Boiler Turbine Units," *IEEE Transactions on Energy Conversion* **11**, 401–406 (1996).

49. M. G. Simões and B. K. Bose, "Neural Network Based Estimation of Feedback Signals for a Vector Controlled Induction Motor Drive," *IEEE Transactions on Industry Applications* **31**, 620–629 (1995).

50. G. R. Raidl, "An Improved Genetic Algorithm for the Multiconstrained 0–1 Knapsack Problem," *Proceedings of the IEEE International Conference on Evolutionary Computation*, Anchorage, AK, May 1998, pp. 207–211.

51. N. Barnier and P. Brisset, "Optimization by Hybridization of a Genetic Algorithm with Constraint Satisfaction Techniques," *Proceedings of the IEEE International Conference on Evolutionary Computation*, Anchorage, AK, May 1998, pp. 645–649.

52. D. Gorinevsky, A. Kapitanovsky, and A. Goldenberg, "Neural Network Architecture for Trajectory Generation and Control of Automated Car Parking," *IEEE Transactions on Control Systems Technology* **4**, 50–56 (1996).

53. F.-C. Lu and Y.-Y. Hsu, "Fuzzy Dynamic Programming Approach to Reactive Power/Voltage Control in a Distribution Substation," *IEEE Transactions on Power Systems* **12**, 681–688 (1997).

54. G. Prasad, E. Swidenbank, and B. W. Hogg, "A Neural Net Model-Based Multivariable Long-Range Predictive Control Strategy Applied in Thermal Power Plant Control," *IEEE Transactions on Energy Conversion* **13**, 176–182 (1998).

55. L. J. Fogel, A. J. Owens, and M. J. Walsh, "Artificial Intelligence through a Simulation of Evolution," in D. B. Fogel, ed., *Evolutionary Computation: The Fossil Record*, IEEE Press, Piscataway, NJ, 1998, pp. 230–254.