Heterogeneous Fuzzy Logic Networks: Fundamentals and Development Studies

Witold Pedrycz, *Fellow, IEEE*

***Abstract*—The recent trend in the development of neurofuzzy systems has profoundly emphasized the importance of synergy between the fundamentals of fuzzy sets and neural networks. The resulting frameworks of the neurofuzzy systems took advantage of an array of learning mechanisms primarily originating within the theory of neurocomputing and the use of fuzzy models (predominantly rule-based systems) being well established in the realm of fuzzy sets. Ideally, one can anticipate that neurofuzzy systems should fully exploit the linkages between these two technologies while strongly preserving their evident identities (plasticity or learning abilities to be shared by the transparency and full interpretability of the resulting neurofuzzy constructs). Interestingly, this synergy still becomes a target yet to be satisfied. This study is an attempt to address the fundamental interpretability challenge of neurofuzzy systems. Our underlying conjecture is that the transparency of any neurofuzzy system links directly with the logic fabric of the system so the logic fundamentals of the underlying architecture become of primordial relevance. Having this in mind the development of neurofuzzy models hinges on a collection of logic driven processing units named here fuzzy (logic) neurons. These are conceptually simple logic-oriented elements that come with a well-defined semantics and plasticity. Owing to their diversity, such neurons form essential building blocks of the networks. The study revisits the existing categories of logic neurons, provides with their taxonomy, helps understand their functional features and sheds light on their behavior when being treated as computational components of any neurofuzzy architecture. The two main categories of aggregative and reference neurons are deeply rooted in the fundamental operations encountered in the technology of fuzzy sets (including logic operations, linguistic modifiers, and logic reference operations). The developed heterogeneous networks come with a well-defined semantics and high interpretability (which directly translates into the rule-based representation of the networks). As the network takes advantage of various logic neurons, this imposes an immediate requirement of structural optimization, which in this study is addressed by utilizing various mechanisms of genetic optimization (genetic algorithms). We discuss the development of the networks, elaborate on the interpretation aspects and include a number of illustrative numeric examples.**

***Index Terms*—Aggregative and referential fuzzy neurons, fuzzy neurocomputing, fuzzy neurons, genetic algorithm, interfaces of fuzzy models (decoding and encoding), logic approximation, network transparency and interpretability, pruning transformations of logic neurons and networks.**

Manuscript received August 16, 2003; revised November 13, 2003. This work was supported in part by the Canada Research Chair (CRC) Program, in part by the Natural Sciences and Engineering Research Council (NSERC), and in part by the Alberta Software Engineering Research Consortium (ASERC).

The author is with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton AB T6R 2G7, Canada, and also with the Systems Research Institute, Polish Academy of Sciences, Warsaw 01-447, Poland (e-mail: pedrycz@ee.ualberta.ca).

Digital Object Identifier 10.1109/TNN.2004.837785

I. Introduction

LOGIC AND fuzzy logic occupy a dominant role in what could be called transparent modeling—a trend strongly supported by granular computing [2]. The constructs of transparent modeling come with a well-defined semantics [3], [4], [32], [33]. The models can be easily interpreted as a collection of rules, analogies, associations or other basic entities describing experimental data. In the synergistic collaboration with fuzzy logic, neural networks deliver a vast array of learning abilities ranging from unsupervised to fully supervised schemes. The attractiveness of fuzzy neurocomputing directly relates to the design of effective and highly symbiotic links that are established between fuzzy sets and neural networks. In the array of approaches, architectures and detailed models [15], [19]–[21] we encounter different schemes of interaction between fuzzy sets and neurocomputing and other related adaptation schemes. The essence of the successful synergy (as also clearly demonstrated in [14] and [16]) lies in the retention of the well-defined identity of the two contributing technologies. In most of the synergistic frameworks, one of them becomes predominant and this results either in more profound learning abilities (and accuracy of approximation) or higher transparency and interpretability of the model. The accuracy-interpretability tradeoff is commonly visible in the constructs of fuzzy neurocomputing. Ideally, we would like to see these two modeling requirements being met to the highest extent. The evident requirement leading to the satisfaction of such objective calls for the elementary constructs exhibiting high-learning abilities along with the sound logic underpinnings. Interestingly, the constructs currently available in the literature position themselves on one or another side of the fuzzy neurocomputing. Some of them lean quite visible toward fuzzy rule based systems with very limited learning capabilities that could exhibit in a quite limited way (say, as adjustable confidence factors of individual rules). There could be a substantial learning slant and high-adaptive capabilities that tend to compromise the interpretability of the resulting construct. For instance, we may see systems with a significant number of layers where each layer is pretended to carry out some logic processing but sometimes those seem to be pretentious claims permeating the literature rather than strongly substantiated statements. For instance, the product operation used in some input layer of some "standard" constructs may refer to the and operation. This is fully legitimate as the and operation is one among t-norms. There is, however, some uneasiness to accept the standard sum as the model of the or operation (unfortunately this is a well rooted position one can easily encounter in the literature). Overall, there is

a tendency toward far more attention being placed on the neural side of neurofuzzy systems with the approximation capabilities being highly glorified and focused upon and the interpretation abilities being left out and quietly reduced. This tendency may not be surprising at all: the approximation abilities are easier to quantify and eventually easier to realize.

The underlying conjecture of this study is that neurofuzzy systems should be constructed on a basis of simple processing units—fuzzy (logic) neurons whose transparency and learning abilities are accentuated to the highest possible extent [12], [25], [27]. This would assure us that the resulting constructs will directly benefit from these features that will manifest in the overall network. The notion of logic (two-valued logic) has been broadly exploited as the sound basis for Boolean networks. On the other hand, the concept of fuzzy logic and logic processing seems to be far less exploited in this setting.

The primordial objectives of this study are fourfold.

- The revisit and systematize the array of the existing fuzzy logic neurons (fuzzy neurons, for short) with respect to their functionality, underlying logic, interpretation aspects and learning abilities. These are well documented in the existing literature but still require some systematization and their "readability."
- To develop architectures of fuzzy neural networks based on different fuzzy neurons. As the systems of such character are inherently heterogeneous, their functionality could be quite diversified and a suitable arrangement of the neurons in successive layers could result in a surprisingly rich collection of logic expressions and nonlinear characteristics of the neural mappings.
- To discuss various schemes of the development of the networks with a special emphasis put toward the structural aspects of learning and its realization in terms of genetic optimization.
- To discuss interpretability of fuzzy networks and introduce means of their effective readability through pertinent pruning mechanisms.

The organization of the material is structured in a way it reflects the research agenda outlined above. First, in Section II, we introduce basic processing modules of fuzzy neurons and elaborate on their underlying taxonomy that is pertinent when building heterogeneous network and allocating the neurons to successive layers. The understanding of the functionality of the neurons is essential to the resulting characteristics of the networks and their further interpretation. Section III highlights the relationships between the fuzzy neurons and fuzzy relational equations. This is of particular interest as they help us reveal links in terms of existing analytical, semianalytical and optimization mechanisms of solving this category of equations. A general topology of the network is outlined in Section IV, which is followed by a comprehensive discussion on the evolutionary development framework of such networks (Section V). Section VI links the logic-driven network with the modeling environment and this helps us emphasize the links with the existing experimental data. Interpretation issues of the networks that lead us to a systematic way of pruning connections and eliminating reference points of the neurons are covered in Section VII. Experimental studies are included in Section VIII.

The terminology used here adheres to the standards used in two-valued logic, many-valued logic, and fuzzy logic. The logic operators are modeled via t- and s-norms. If not stated otherwise, in this study we use two standard realizations of t- and s-norms in the form of a product and probabilistic sum. An overbar denotes a complement treated in a usual way encountered in logic (that is $\bar{x} = 1 - x$).

II. Basic Types of Logic Neurons

In this section, we discuss the main categories of the logic neurons as they were introduced and discussed in [25], [27], [28]. The underlying taxonomy involves aggregative and referential neurons and very much ties up to their logic functionality. Their names reflect the underlying processing realized by the neurons. The aggregative neurons concentrate on the logic type of aggregation of the inputs (truth values) while the referential neurons are aimed at logic processing of results of referential transformations of the corresponding truth values.

A. Aggregative Neurons

Formally, these neurons realize a logic mapping from $[0,1]^n$ to $[0,1]$. Two main classes of the processing units exist in this category.

OR neuron realizes an *and logic aggregation* of inputs $\mathbf{x} = [x_1, x_2, \ldots, x_n]$ with the corresponding connections (weights) $\mathbf{w} = [w_1, w_2, \ldots, w_n]$ and then summarizes the partial results in an or-wise manner (hence, the name of the neuron). The concise notation underlines this flow of computing, $y = OR(\mathbf{x}; \mathbf{w})$ while the realization of the logic operations gives rise to the expression (referred to as an s-t combination)

$$y = S_{i=1}^{n}(x_i t w_i). \tag{1}$$

The two essential operators used in the composition are the t- and s-norms. Let us recall that by t-norms we mean *an and* type of logic connective used to aggregate two fuzzy sets. The commonly used examples of t-norms include minimum, product, and Lukasiewicz and connective. The typical examples of s-norms (that are realizations of or logic connectives) involve maximum, probabilistic sum $(a + b - ab)$, and Lukasiewicz *or connective*. Bearing in mind the interpretation of the logic connectives (t- and s-norms), the OR neuron realizes the following logic expression being viewed as an underlying logic description of the processing of the input signals

$$(x_1 \text{ and } w_1) \text{ or } (x_2 \text{ and } w_2) \text{ or} \ldots \text{or } (x_n \text{ and } w_n). \tag{2}$$

Apparently the inputs are logically "weighted" by the values of the connections before producing the final result (see also [8]). In other words, we can treat "y" as a truth value of the above statement where the truth values of the inputs are affected by the corresponding weights. Noticeably, lower values of w_i discount the impact of the corresponding inputs; higher values (especially those being positioned close to 1) do not affect the original truth values of the inputs resulting in the logic formula. In limit, if all connections $w_i, i = 12, \ldots, n$ are set to 1 then the neuron produces a plain *or*-combination of the inputs, $y = x_1$ or x_2 or $\ldots$ or x_n. The values of the connections set to zero eliminate the corresponding inputs. Computationally,

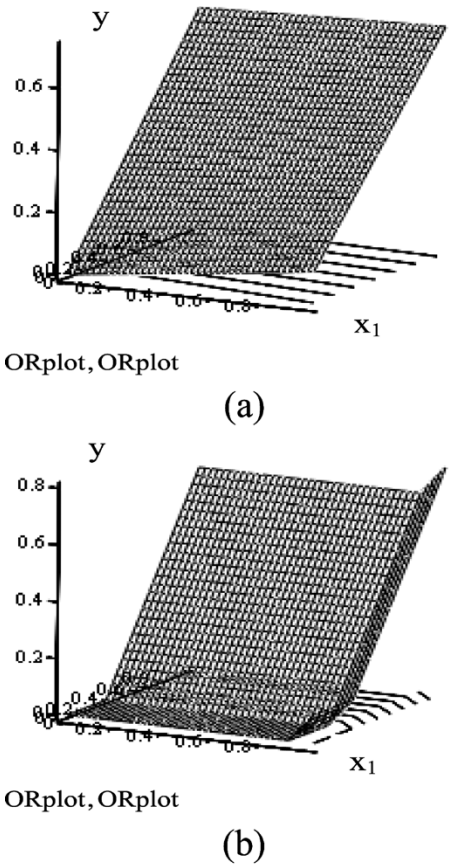

Fig. 1. Characteristics of the OR neuron for selected pairs of t- and s-norms. In all cases, the corresponding connections are set to 0.1 and 0.7 with intent to visualize their effect on the input–output characteristics of the neuron. (a) Product and probabilistic sum. (b) Lukasiewicz and and or connectives.

the OR neuron exhibits nonlinear characteristics (that is inherently implied by the use of the t- and s-norms that are evidently nonlinear mappings). The plots of the characteristics of the OR neuron shown in Fig. 1 visualize this effect (note that the characteristics are affected by the use of some norms). The connections of the neuron contribute to its adaptive character; the changes in their values form the crux of the parametric learning.

AND neuron The neurons in the category, denoted by $y = \mathrm{AND}(\mathbf{x}; \mathbf{w})$ with $\mathbf{x}$ and $\mathbf{w}$ being defined as in case of the OR neuron, are governed by the expression

$$y = \mathrm{T}_{i=1}^{n}(x_i \mathrm{s} w_i). \tag{3}$$

Here the *or and and connectives* are used in a reversed order. First, the inputs are combined with the use of the s-norm and the partial results are aggregated and-wise. Higher values of the connections reduce impact of the corresponding inputs. In limit $w_i = 1$ eliminates the relevance of x_i. With all w_i set to 0, the output of the AND neuron is just an and aggregation of the inputs

$$y = x_1 \text{ and } x_2 \text{ and} \ldots \text{and } x_n. \tag{4}$$

One can also consider the complements of w_i, (that is $u_i = 1 - w_i$) and in this way come up with the same interpretation of the connections as done for the OR neurons. Obviously this could be a matter of convenience and some tradition inherited from neural networks where higher value of the connections reflects its more evident impact. The characteristics of the AND neuron

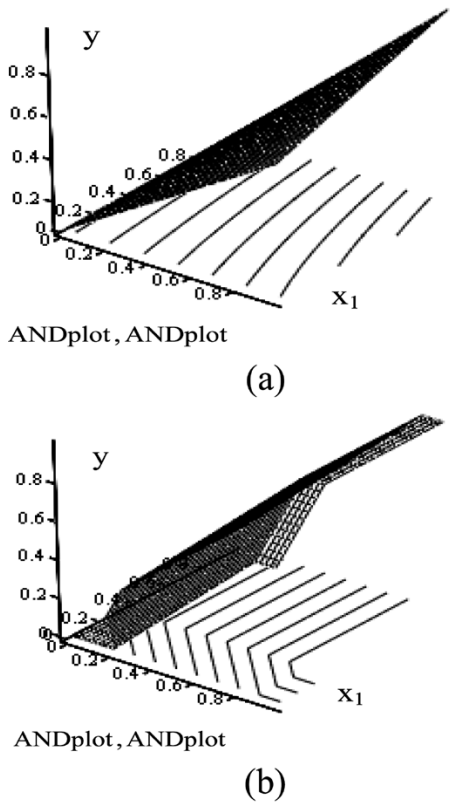

Fig. 2. Characteristics of AND neurons for selected pairs of t- and s-norms. In all cases, the connections are set to 0.1 and 0.7 with intent to visualize their effect on the characteristics of the neuron. (a) Product and probabilistic sum. (b) Lukasiewicz logic connectives.

are shown in Fig. 2; note the influence of the connections and the specific realization of the triangular norms on the mapping completed by the neuron.

Let us conclude that the neurons are highly nonlinear processing units depending upon the specific realizations of the logic connectives. They also come with potential plasticity whose usage becomes critical when learning the networks involving these neurons.

B. Referential (Reference) Neurons

The essence of referential computing deals with processing logic predicates. The two-argument (or generally multivariable) predicates such as *similar, included in, and dominates* are essential components of any logic description of a system. In general, the truth value of the predicate is a degree of satisfaction of the expression $P(x, a)$ where "a" is a certain reference value (reference point). Depending upon the meaning of the predicate (P), the expression $P(x, a)$ reads as "x is similar to a," "x is included in a," "x dominates a," etc. This terminology makes sense as we are concerned with truth values in [0, 1] and therefore the term ":point" should be interpreted in this context. In case of many variables, the compound predicate comes in the form $P(x_1, x_2, \ldots, x_n, a_1, a_2, '', a_n)$ or more concisely $P(\mathbf{x}; \mathbf{a})$ where $\mathbf{x}$ and $\mathbf{a}$ are vectors in the n-dimensional unit hypercube. We envision the following realization of $P(\mathbf{x}; \mathbf{a})$:

$$P(\mathbf{x}; \mathbf{a}) = P(x_1, a_1) \text{ and } P(x_2, a_2) \text{ and} \ldots \text{and } P(x_n, a_n) \tag{5}$$

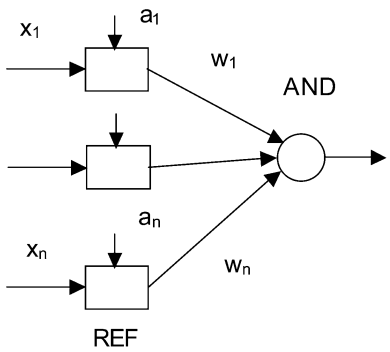

Fig. 3. Schematic view of computing realized by a reference neuron an involving two processing phases (referential computing and aggregation).

meaning that the satisfaction of the multivariable predicate relies on the satisfaction realized for each variable separately. As the variables could come with different levels of relevance as to the overall satisfaction of the predicates, we represent this effect by some weights (connections) $w_1, w_2, \ldots, w_n$ so that (5) rewrites in the form

$$P(\mathbf{x}; \mathbf{a}, \mathbf{w}) = [P(x_1, a_1) \text{ or } w_1] \text{ and } [P(x_2, a_2) \text{ or } w_2] \text{ and } \ldots \text{ and } [P(x_n, a_n) \text{or } w_n]. \quad (6)$$

Taking another look at the above expression and using a notation $z_i = P(x_i, a_i)$, it converts to a certain AND neuron $y = \text{AND}(\mathbf{z}; \mathbf{w})$ with the vector of inputs $\mathbf{z}$ being the result of the computations done for the logic predicate. Then the general notation to be used reads as $\text{REF}(\mathbf{x}; \mathbf{w}, \mathbf{a})$ and using the explicit notation we have

$$y = T_{i=1}^{n}(\text{REF}(x_i, a_i) \text{sw}_i). \quad (7)$$

In essence, as visualized in Fig. 3, we may conclude that the reference neuron is a realized in a two-stage construct where first we determine the truth values of the predicate (with a treated as a reference point) and then treat these results as the inputs to the AND neuron.

So far we have used the general term of predicate computing not confining ourselves to any specific nature of the predicate. Among a number of possibilities, we discuss the three of them, which tend to occupy an important role. Those are inclusion, dominance, and match (similarity) predicates. As the names stipulate, the predicates return truth values of satisfaction of the relationship of inclusion, dominance, and similarity of a certain argument "x" with respect to the given reference "a." The essence of all these calculations is in the determination of the given truth values and this is done in the carefully developed logic framework so that the operations retain their semantics and interpretability. What makes our discussion coherent is the fact that the proposed operations originate from triangular norms. The inclusion operation, denoted by $\subset$ is modeled by an implication $\rightarrow$ that is induced by a certain left continuous t-norm [31]

$$a \rightarrow b = \sup\{c \in [0,1] \,|\, atc \leq b\}, a, b \in [0,1]. \quad (8)$$

For instance, for the product the inclusion takes on the form $a \rightarrow b = \min(1, b/a)$. The intuitive form of this predicate is self-evident: the statement "x is included in a" and modeled as $\text{INCL}(x, a) = x \rightarrow a$ comes with the truth value equal to 1 if x is less or equal to a (which in other words means that x is included in a) and produces lower truth values once x starts exceeding the truth values of "a." Higher values of "x" (those above the reference point "a") start generating lower truth values of the predicate. It is worth mentioning that (8) generates a family of implication operators induced by specific t-norms. While the detailed form could vary between different t-norms, all of them preserve the general format of dependency as described above.

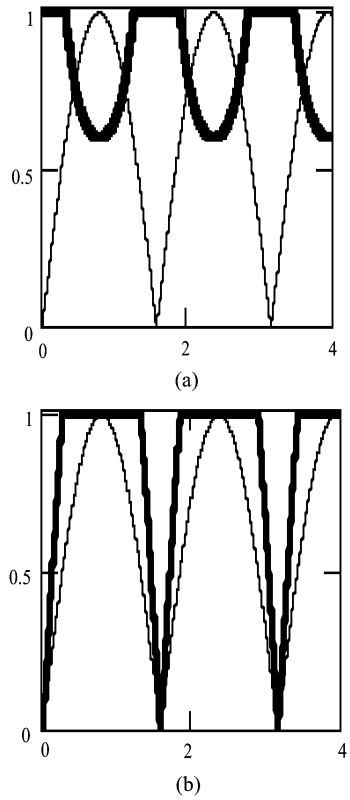

Fig. 4. Temporal signal $x(t)$ and its acceptance signals (levels of the signals—thick lines) formed with respect to its (a) lower and (b) upper threshold. The complements of the acceptance are then treated as warning signals.

The dominance predicate acts in a dual manner. It returns 1 once "x" dominates "a" (so that its values exceeds "a") and values below 1 for x lower than the given threshold. The formal model can be realized as $\text{DOM}(x, a) = a \rightarrow x$. With regard to the reference neuron, the notation is equivalent to the one being used in the previous case (7), that is $\text{DOM}(\mathbf{x}; \mathbf{w}, \mathbf{a})$ with the same meaning and role played by $\mathbf{a}$ and $\mathbf{w}$.

The similarity (match) operation is an aggregate of these two, $\text{SIM}(x, a) = \text{INCL}(x, a) t \text{DOM}(x, a)$ which is appealing from the intuitive standpoint: we say that x is similar to a if x is included in a and x dominates a. Noticeably, if $x = a$ the predicate returns 1; if x moves apart from "a" the truth value of the predicate becomes reduced. The resulting similarity neuron is denoted by $\text{SIM}(\mathbf{x}; \mathbf{w}, \mathbf{a})$ and reads as

$$y = T_{i=1}^{n}(\text{SIM}(x_i, a_i) \text{sw}_i). \quad (9)$$

The reference operations form an interesting generalization of the threshold operations. Consider that we are viewing "x" as a signal of time whose behavior needs to be monitored with respect to some bounds (α and β). If the signal does not exceed some threshold α then the acceptance signal should go off. As shown in Fig. 4(a), if the signal does not exceeded the value of

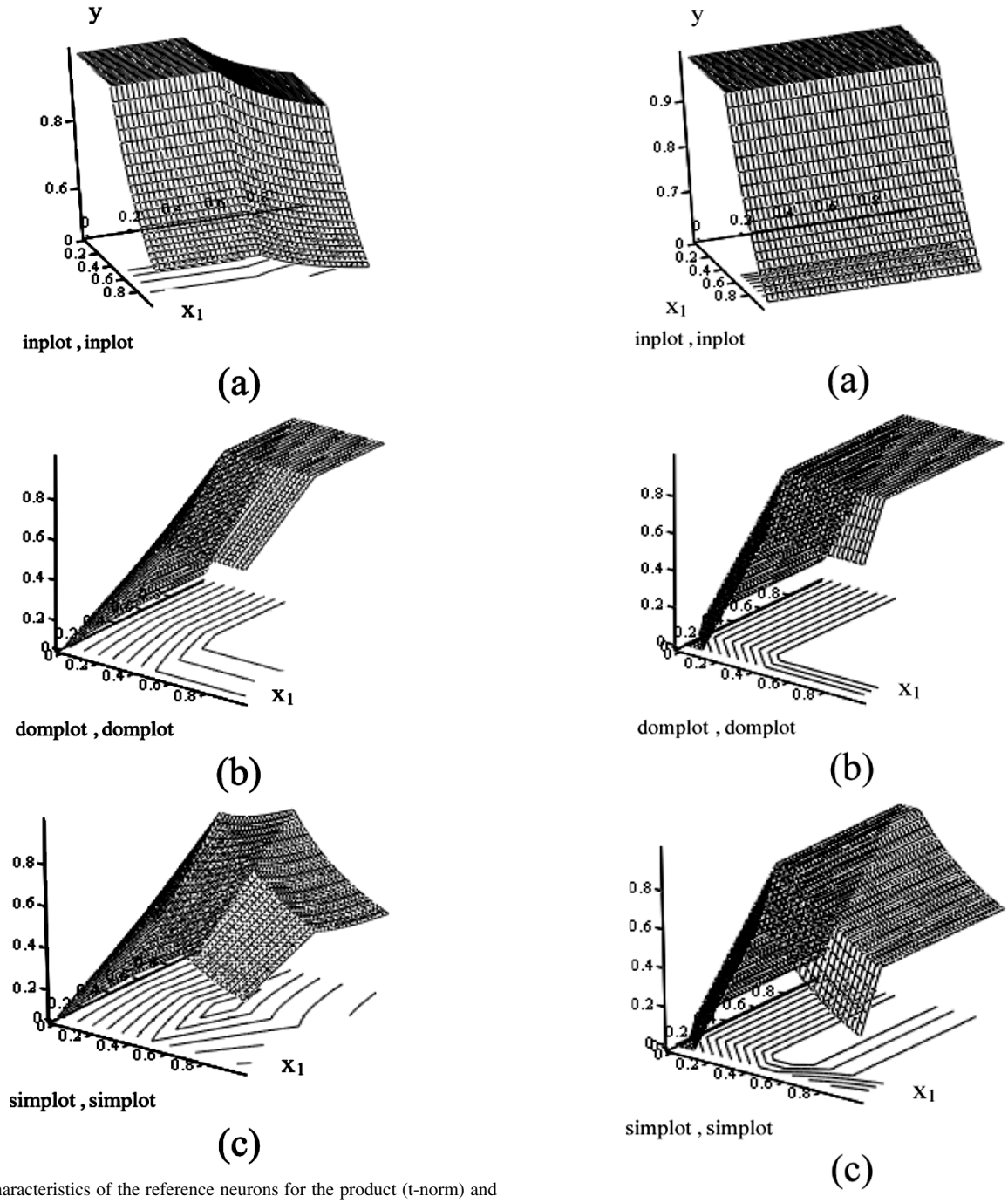

Fig. 5. Characteristics of the reference neurons for the product (t-norm) and probabilistic sum (s-norm). In all cases, the connections are set to 0.1 and 0.7 with intent to visualize the effect of the weights on the relationships produced by the neuron. The point of reference is set to (0.5, 0.5). (a) Inclusion neuron. (b) Dominance neuron. (c) Similarity neuron.

Fig. 6. Characteristics of the reference neurons for the Lukasiewicz t-norm and s-norm (that is $a\ t\ b = \max(0, a+b-1)$ and $a\ s\ b = \min(1, a+b)$). In all cases, the connections are set to 0.1 and 0.7 with intent to visualize the effect of the weights. The point of reference is set to (0.5, 0.5). (a) Inclusion neuron. (b) Dominance neuron. (c) Similarity neuron.

0.6, its acceptance is equal to 1, otherwise it becomes reduced and attains lower values. Likewise we require another acceptance mechanism indicating a situation where the signal does not go below another threshold value of β. In case of fuzzy predicates, the level of acceptance assumes values in the unit interval rather than being a Boolean variable. The strength of acceptance reflects how much the signal adheres to the assumed thresholds. An example illustrating this behavior is shown in Fig. 4. Here the values of α and β are set up to 0.6 and 0.5, respectively.

The plots of the referential neurons with two input variables are shown in Figs. 5 and 6. Here we have included two realizations of the t-norms to illustrate their effect on the nonlinear characteristics of the processing units.

It is worth noting that by moving the reference point to the origin of the unit hypercube, the dominance neuron starts resembling the aggregative neuron. More specifically, for $\mathbf{a} = \mathbf{0} = [0\,0\,0\ldots0]$ the dominance neuron reduces to the AND neuron.

One can draw a loose analogy between some types of the referential neurons and the two categories of processing units encountered in neurocomputing. The analogy is based upon the local versus global character of processing realized therein. Per-

ceptrons come with the global character of processing. Radial basis functions realize a local character of processing as focused on receptive fields. In the same vein, the inclusion and dominance neurons are after the global nature of processing while the similarity neuron carries more confined, local processing.

III. Relationships of Fuzzy Neurons With Fuzzy Relational Equations

One can look at the fuzzy neurons discussed in the previous section from a slightly different perspective. The s-t and t-s calculations are in essence examples of so-called s-t and t-s composition operators used in fuzzy sets. The inputs and connections can be treated as discrete n-dimensional fuzzy sets whose convolution is computed by means of the s-t composition (OR neuron) and t-s composition (AND neuron). This simple observation links fuzzy neurons with the theory of fuzzy relational equations—a well established area of fundamental and applied pursuits in fuzzy sets. The early results in this domain go back as early as the mid seventies as proposed by Sanchez with a number of significant results obtained afterwards (see [5], [22], [24], and [31]). We can treat a fuzzy neuron as a realization of some fuzzy relational equation $y = \mathrm{OR}(\mathbf{x};\mathbf{w})$ or $y = \mathrm{AND}\ (\mathbf{x};\mathbf{w})$. Two fundamental problems are sought: (a) solving the equation with respect to $\mathbf{w}$ for $\mathbf{x}$ and y given (usually referred in the theory of relational equations to as an estimation problem), and (b) solving the equation with respect to $\mathbf{x}$ assuming that $\mathbf{w}$ and y are provided (which refers to as an inverse problem). In both cases, the theory provides us with interesting and general results. The first problem concerns the estimation of the connections of the neuron and is inherently tied to the learning of the network composed of logic neurons. Its generalized version involving solving a system of relational equations with a finite set of input–output pairs $(\mathbf{x}(k),\mathbf{y}(k))$ given is the standard version of the estimation problem. The inverse problem is focused on constructing inputs $(\mathbf{x})$ leading to the required output. The theory of fuzzy relational equations shows that in general there could be families (rather than unique solutions) to such equations, states how to effectively construct extreme (viz. maximal or minimal) solutions. These findings hold under a strong assumption that there is a nonempty family of solutions. Furthermore the generality of the results is assured for some subset of t- and s-norms used in the design of the composition operator; these solutions are obtained for max (sup)—t composition and min (inf)—s convolution of fuzzy sets [31]. Interestingly, the generality of the solutions is not guaranteed for the general s-t or t-s composition for any t- and s-norm. The relevance of the theoretical framework should be cast in a certain setting; in essence we can envision that the solutions can only be approximate and we can use them as a starting (initial) configuration of the connections the learning could start off. This is, in particular, quite relevant in case of gradient-based methods for which the choice of the starting point plays an essential role. The point worth mentioning here is that while relational equations form an important theoretical frameworks, they are not extremely well positioned to deal with problems of high dimensionality (in virtue of their logic framework they never intended to assume a leading role there). We relate to them in our discussion yet exploiting a different optimization environment of genetic algorithms, their role becomes more profound at the interpretation end along with some specialized systems obtained after pruning.

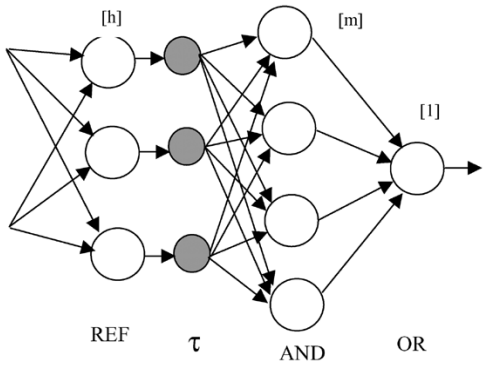

Fig. 7. General architecture of the network constructed with logic-based neurons; see a detailed description in text. The dimensions of the layers is marked by numbers in bracket (upper part of the figure).

IV. General Topology of the Network

As we have developed a host of logic processing units, we can use then in the developing a general architecture of the network. In this design, we are guided by several requirements. First, we would like to achieve a substantial level of flexibility so that the structure could be easily and effectively adjusted to the experimental data. Second, we would like to assure a high level of interpretability: evidently each neuron comes with a well-defined semantics and our intent is to retain it so at the very end the network can be easily mapped (translated) into a well-structured and transparent logic expression. This quest for interpretability and transparency has been clearly identified in the most recent literature [3]. In the logic description, we will dwell upon the well-established components of fuzzy logic: logic operators and linguistic modifiers. Having these requirements in mind, a general architecture of the network is shown in Fig. 7.

The network comes with several layers where each of them has a clear functional role to play and is highly interpretable. The first layer (referred to as a referential processing) is composed of "h" referential neurons (inclusion, dominance, similarity). The results of this processing are taken to some power (indicated by some small shadowed circle) and then combined and-wise in the second layer of the network. The elements there are AND neurons with all connections hardwired to zero. The width of this layer is equal to "m." In the sequel, the results are combined by the layer of OR neurons. Let us now move on to the computational details by the same time concentrating on the interpretation of the underlying processing. The truth values generated by the referential neurons reflect the level of satisfaction of the logic predicates

$$z_1 = P_1(\mathbf{x};\mathbf{a}_1) \quad z_2 = P_2(\mathbf{x};\mathbf{a}_2), \ldots, z_h = P_h(\mathbf{x};\mathbf{a}_h). \quad (10)$$

The powers of z_i, denoted as $\tau_i(z_i)$ where τ_i assumes a few discrete values (say 1/2, 0, 1, 2, and 4) are interpreted as lin-

guistic modifiers operating upon z_i and producing some concentration or dilution effect [31]. More specifically, the collection of the modifiers maps on the corresponding powers in a usual way we encounter in the literature where

1/2 more or less (dilution effect);
0 unknown;
1 true (neutral);
2 very (concentration effect);
4 very (very) = very^2 (strong concentration effect).

The result of this processing (coming as the output of the shadowed circle) is a logically modified referential logic predicate with a straightforward interpretation. For instance, the expression $\text{INCL}([x_1, x_2], [0.2, 0.7], [0.6, 0.9])^{0.5}$ translates into the following linguistic statement:

$$y = \text{more or less } [[x_1 \text{ included in } 0.6] \text{ or } 0.2 \text{ and } [x_2 \text{ included in } 0.9] \text{ or } 0.7]$$

(noticeably, the core part of this expression could be extracted by carrying out some additional pruning; we elaborate on this matter in Section VII). The AND layer of the network combines and-wise referential expressions constructed in the previous layer. Finally, the output layer includes a single OR neuron whose connections represent the relevance of information originating at the input nodes. By looking at the two layers combined (the ones formed by the AND and OR neurons), we note that this forms a realization of the generalized version of the Shannon theorem which shows this type of network realization of any Boolean function. Obviously, here we are concerned with the continuous version (so the realization need to be viewed as approximation) and the network operates on the results of referential computing rather than direct inputs.

While the AND and OR neurons are the two categories of the standard processing units being encountered in a number of neurofuzzy constructs, the layer of linguistic hedges requires more attention. They are useful in the linguistic structuralization of the network (in the sense they affects linguistically the outputs of the referential neurons). From the computational standpoints, the linguistic hedges help develop the required nonlinear characteristics between input and output variables; noticeably the relationships can be easily formed by choosing a suitable hedge.

In what follows, we visualize the diversity of the characteristics of the networks leading to nonlinear and multimodal relationships we can easily construct by putting together various neurons, see Fig. 8. Note that the linguistic hedges play an important role in shaping the logic mapping completed by the network.

Owing to the inherent roots of the logic backbone of the neurons, we can directly exploit the transparency of the construct when capturing the essence of the data. If the network is large enough, we can conclude that such network can capture the data with any required accuracy. While the theorem of universal approximation is not of our concern, it becomes advantageous to note that a suitable topology is at immediate reach. As an example, consider a finite data set of input–output pairs $\{\mathbf{x}(k), y(k)\} k = 1, 2, \ldots, N$. Then the network with "N" referential neurons of the matching nature, linguistic modifiers assuming high values and a single OR neuron with the connections

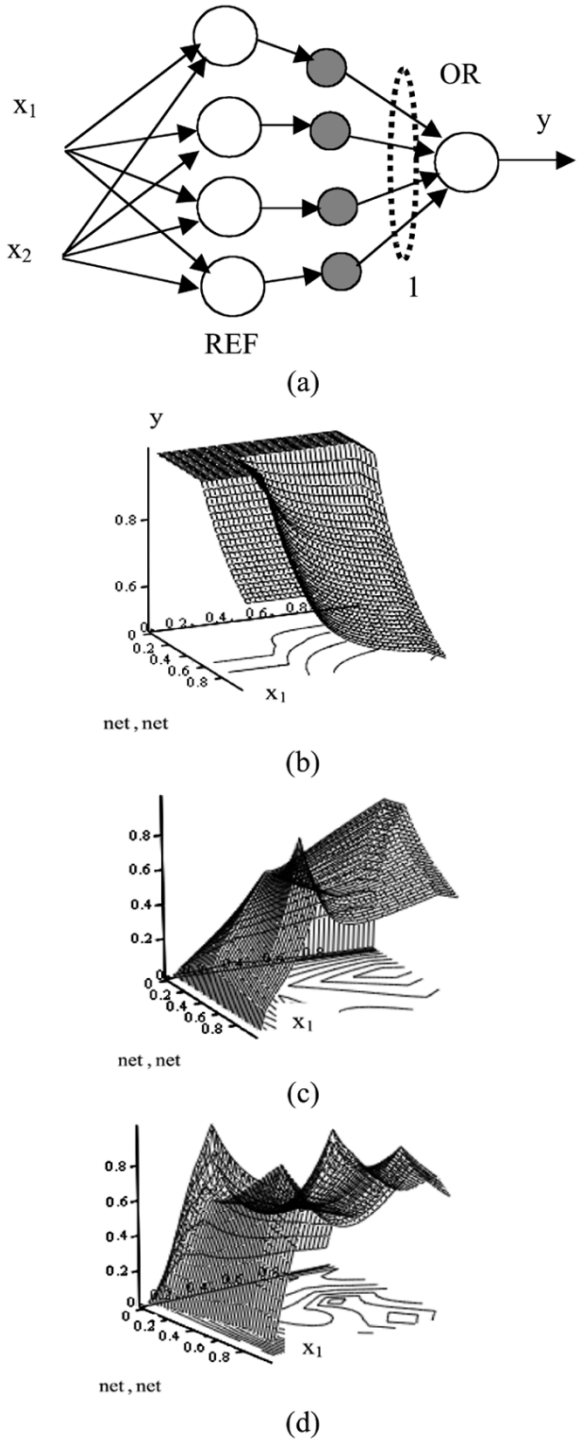

Fig. 8. Input–output characteristics of the network with the topology shown as in (a); here all connections of the OR neuron are set up to 1.0 and the number of the referential neurons is equal to 2 (b)–(c) or 4 (d), (b) two inclusion neurons $\text{INCL}(x_1, x_2, \mathbf{w}, \mathbf{a})\mathbf{w} = [0.4\ 0.2], \mathbf{a} = [0.5\ 0.3]$ (1st neuron) and $\mathbf{w} = [0.0\ 0.0]\mathbf{a} = [0.2\ 0.9]$ (2nd neuron) $\tau_1 = 2\ \tau_2 = 0.5$; (c) two SIM neurons $\text{SIM}(x_1, x_2, \mathbf{w}, \mathbf{a})$ with $\mathbf{w} = [0.4\ 0.2]$ and $\mathbf{a} = [0.8\ 0.3]$ (1st neuron) and $\mathbf{w} = [0.0\ 0.0], \mathbf{a} = [0.4\ 0.9]$ (2nd neuron), $\tau_1 = 4\ \tau_2 = 0.5$; (d) four SIM neurons with the weights and connections equal to $\mathbf{w} = [0.1\ 0.0]$ and $\mathbf{a} = [0.1\ 0.3]$ (1st neuron) $\mathbf{w} = [0.2\ 0.1], \mathbf{a} = [0.8\ 0.2]$ (2nd neuron), $\mathbf{w} = [0.2\ 0.1]\mathbf{a} = [0.5\ 0.7]$ (3rd neuron) $\mathbf{w} = [0.2\ 0.1], \mathbf{a} = [0.9\ 0.8]$ (4th neuron), $\tau_1 = 1\ \tau_2 = 1, \tau_3 = 4\ \tau_4 = 2$. Note a multimodal character of the characteristics caused by the existence of several referential neurons.

equal to the required target values $y(1), y(2), \ldots y(N)$ becomes a suitable network. Interestingly, we developed the structure

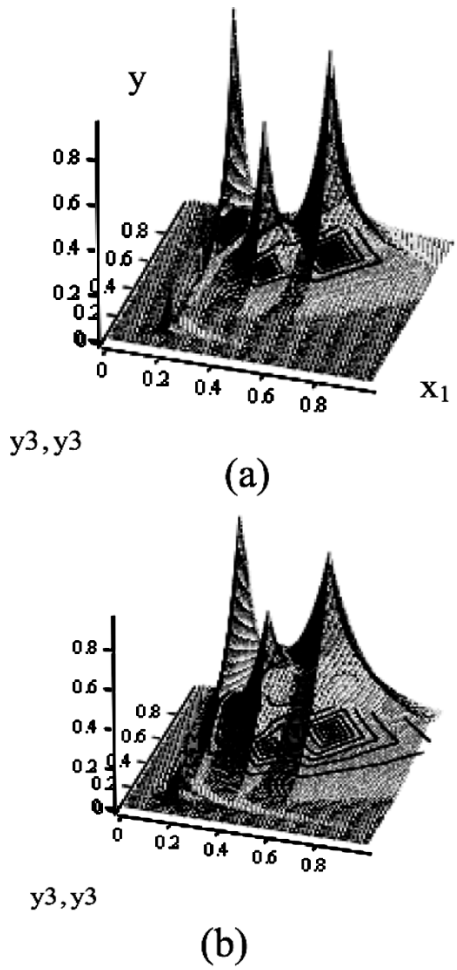

Fig. 9. Input–output characteristics of the network mapping four input-output data $([0.2\ 0.1], 0.2), ([0.6\ 0.8], 0.9), ([0.2\ 0.9], 0.95), ([0.4\ 0.63], 0.65)$ $\tau = 8$ (a) and $\tau = 4$ (b).

without any learning but simply through mapping the problem into the logic setting formed by the proposed network.

An example of such network shown for $\mathrm{N} = 4$ data points and $\tau = 8$ is included in Fig. 9. The four matching neurons are allocated to the individual data points. The high value of the logic modifier (that produces a high-concentration effect, very^4 = very very very very) leads to highly localized receptive fields. Using lower values of the modifiers, say $\tau = 4$(very very) leads to a less visible concentration effect as seen in Fig. 9(b).

V. Evolutionary Development of the Network

A structural optimization becomes a critical feature of the development environment. The heterogeneity of the network (if properly exploited) becomes an evident asset of the architecture that implies its flexibility. The structure optimization however requires a suitable development environment that can cope with the optimization of this nature. The genetic optimization is a sound option to be explored with this regard. While the genetic algorithm is standard to a high extent, we spend less time on the description of the genetic operators and concentrate on the organization of a chromosome as it maps the problem into the optimization environment. Because of the anticipated heterogeneous character of the chromosome, it is advantageous to consider a floating-point version of coding. The benefits of this version of coding over the binary content of the chromosome has been discussed quite intensively in the existing literature [9], [18]. As a prerequisite let us assume that the number of reference neurons and AND neurons is given in advance and these are equal to "h" and "m," respectively, (these could be optimized as well but a straightforward enumeration could be quite appropriate in this setting because of a fairly limited search space). The connections of the AND neurons are "hardwired" to zero and therefore do not need to be optimized. Alluding to the topology of the network, the content of the chromosome shown in Fig. 10 becomes self-explanatory. The thresholding operation of the continuous entries of the neuron is used to select one of the types of the reference neurons. The unit interval is split into equal parts to represent each type of the neuron. Because of this arrangement of the coding, we treat all referential neurons in the same manner (so the likelihood of their appearance in the structure becomes the same). Likewise, we threshold the portion of the chromosome corresponding to the linguistic hedges following the reference neurons. Finally, the weights of the OR neuron assume continuous values from the unit interval and they are taken directly from the corresponding entries of the chromosome.

In the sequel, we discuss the realization of the main functional components of the GA, that is selection, mutation, and crossover operations and discuss the fitness function that is used to guide the optimization process.

1) Selection Process: In this process, we use an elitist ranking selection [1]. This selection mechanism means that individuals to be selected for the next generation are based on their relative rank in the population, as determined by the fitness function. The best individual from each generation is always carried over to the next generation (elitist mechanism) meaning that the solution found so far during the genetic optimization is guaranteed to never disappear.

2) Mutation: The mutation operator is a standard construct encountered in a number of genetic algorithms [8]. Given an individual string $\mathbf{a} = [\mathrm{a_1}, \mathrm{a_2}, \ldots, \mathrm{a_n}]$, we generate a new string $\mathbf{a'} = [\mathrm{a'_1}, \mathrm{a'_2}, \ldots, \mathrm{a'_n}]$ where $\mathrm{a'_i}, \mathrm{i} = 1, 2, \ldots, \mathrm{n}$, is a random number confined in the range of [0, 1] and subject to the following replacement (mutation) rule: $\mathrm{a_i}$ is mutated that is replaced by $\mathrm{a'_i}$ with some probability of mutation $(\mathrm{p_m})$ otherwise the entry of the chromosome is left intact, that is $\mathrm{a'_i} = \mathrm{a_i}$.

3) Crossover: The operation is realized as the BLX-0.5 crossover operator [7], [11] which is carried out as follows. Given are two individuals $\mathbf{a} = [\mathrm{a_1}, \mathrm{a_2}, \ldots, \mathrm{a_n}]$ and $\mathbf{b} = [\mathrm{b_1}, \mathrm{b_2}, \ldots, \mathrm{b_n}]$. The resulting offsprings are formed in the form $\mathbf{a'} = [\mathrm{a'_1}, \mathrm{a'_2}, \ldots, \mathrm{a'_n}]$ and $\mathbf{b'} = [\mathrm{b'_1}, \mathrm{b'_2}, \ldots, \mathrm{b'_n}]$, where $\mathrm{a'_i}, \mathrm{b'_i}, \mathrm{i} = 1, 2, \ldots, \mathrm{n}$, are random numbers located in the range $[\max(0, \min_\mathrm{i} - 0.5\mathrm{I}), \min(1, \max_\mathrm{i} + 0.5\mathrm{I})]$. Here, $\min_\mathrm{i} = \min(\mathrm{a_i}, \mathrm{b_i}), \max_\mathrm{i} = \max(\mathrm{a_i}, \mathrm{b_i})$, and I defined in the form $\mathrm{I} = \max_\mathrm{i} - \min_\mathrm{i}$. This particular crossover operation provides a good balance between using the information of the parents and avoiding premature convergence, cf. [11]. The crossover operator ensures that all values of the generated offspring are confined to the unit interval [0, 1]. The operator is employed with probability of crossover, $\mathrm{p_c}$, otherwise the individuals are left unchanged $\mathbf{a'} = \mathbf{a}$ and $\mathbf{b'} = \mathbf{b}$.

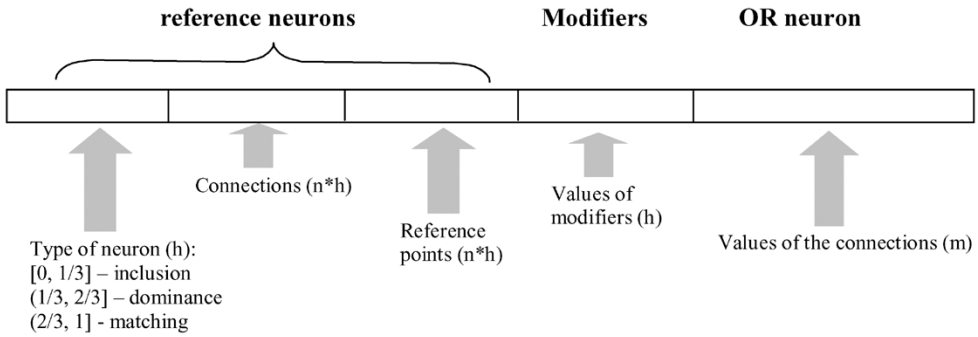

Fig. 10. Organization of a chromosome representing the structure of the network.

4) Fitness Function: The fitness function quantifies how the network approximates the data and is taken as $1-(\mathrm{Q})/(\mathrm{Q}+\varepsilon)$ with Q being a sum of squared errors between the target values (experimental output data) and the corresponding outputs of the network (or any other measure of error). A small positive constant ε standing in the denominator of the above expression assures that the fitness function remains meaningful even for $\mathrm{Q}=0$ (which in practice never occurs).

Two other parameters of the GA involve a size of the population and number of generations. Their values have to be experimented with. Typically, these two parameters are in the range of 100–200 individuals in a population and between 200–500 generations.

VI. Interfaces of Fuzzy Networks

In a nutshell, the fuzzy networks completes a logic-based processing of input signals and realizes a certain logic-driven mapping between input and output spaces. As they interact with a physical world whose manifestation does not usually arise at the level of logic (multivalued) signals, it becomes apparent that there is a need for some interface of the model. Such interfaces are well known in fuzzy modeling [2] (see also [23], [26] and [30]). They commonly arise under a name of fuzzifiers (granular encoders) and defuzzifiers (granular decoders). The role of the encoder is to convert a numeric input coming from the external environment into the internal format of membership grades of the fuzzy sets defined for each input variable. These results of a nonlinear normalization of the input (no matter what original ranges the input variables assume) and a linear increase of the dimensionality of the new logic space in comparison with the original one). The decoder takes the results of the logic processing and transforms them into some numeric values. The layered architecture of the fuzzy models with clearly distinguished interfaces and the logic-processing core is illustrated in Fig. 11.

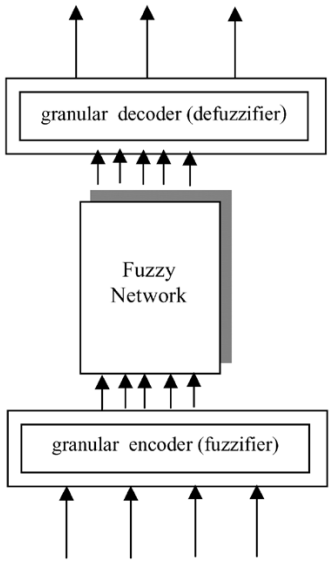

Fig. 11. General layered structure of fuzzy modeling. The use of granular encoders and decoders are essential in the development of communication mechanisms with the modeling environment.

With the design of the interfaces, we exercise two general approaches (in the literature we encounter far more various techniques but they are usually more specialized).

1) *Granulation of individual variables*: This mechanism of granulation is quite common in the realm of fuzzy modeling. In essence, we define several fuzzy sets in the universe of discourse of the variable of interest so that any input is transformed via the membership functions defined there and the resulting membership grades are used in further computations by the model. From the design standpoint, we choose a number of fuzzy sets, type of membership functions and a level of overlap between consecutive fuzzy sets. Some general tendencies along this line are thoroughly reported in the literature. By selecting the number of fuzzy sets (usually between 3 and 9), we position modeling activities at some required level of information granularity (a level of modeling details we are interested in). The type of membership functions helps model the semantics of the information granules. Among many possibilities, we commonly encounter triangular fuzzy sets and Gaussian membership functions. These two types come with an extensive list of arguments that help make a suitable selection with respect to the main objectives of the model (e.g., those concerning a tradeoff between interpretation and accuracy of modeling). The overlap level is essential from different points of view, namely a) semantics of the linguistic terms, b) nonlinear numeric characteristics of the fuzzy model, and c) completeness of the model.
2) *Nonlinear or linear normalization*: Here we transform an original variable defined in some space, say [a, b] (subset of $\mathbf{R}$) is scaled to the unit interval. This could be done with the aid of some mapping $\phi : [\mathrm{a},\mathrm{b}] \rightarrow [0,1]$. The mapping may be chosen to be linear (with some ramping

effect) or nonlinear. In any case, we consider that ϕ is monotonically increasing with the boundary conditions $\phi(a) = 0$ $\phi(b) = 1$. This transformation does not affect the dimensionality of the problem.

So far, we have not discussed any recurrent type of architectures that are used with intent of developing models of dynamic systems. Nevertheless, the general topology outlined in Fig. 11 can be easily extended by introducing auxiliary inputs at the level of the granular encoder.

VII. Interpretation Aspects of the Networks

While each neuron in the network comes with a well-defined semantics and can be easily interpreted, the result of interpretation could lead to a quite lengthy description in case of multivariable systems. To facilitate the process of interpretation and reduce the structure of the detailed logic expression to its essential substructure with the most meaningful topology, we do the pruning of the weakest (unnecessary) connections. The following are the detailed thresholding expressions supporting such pruning activities.

1) For the referential neurons $y = \mathrm{REF}(\mathbf{x}; \mathbf{w}, \mathbf{a})$ we admit the following pruning mechanism: connection w_i is binarized producing a connection $w_i^{\sim}$ assuming Boolean values

$$w_i^{\sim} = \begin{cases} 1, & \text{if } w_i > \lambda \\ 0, & \text{if } w_i \leq \lambda \end{cases} \tag{11}$$

where λ denotes a certain threshold level. Considering that we are concerned with the AND neurons, the connections higher than the assumed threshold are practically eliminated from the computing. Apparently we have $(w_i^{\sim} s x_i) t A \approx 1 t A = A$ where A denotes the result of computing realized by the neuron for the rest of its inputs.

In case of referential neurons, their reference point a_i requires different treatment depending upon the type of the specific referential operation. For the inclusion operation, $\mathrm{INCL}(x, a_i)$ we can admit the threshold operation in the form

$$\mathrm{INCL}^{\sim}(x, a_i) = \begin{cases} \mathrm{INCL}(x, a_i), \text{if } a_i \leq \mu \\ 1 - x, \text{if} a_i > \mu \end{cases} \tag{12}$$

with μ being some fixed threshold value. In other words, we consider that $\mathrm{INCL}(x, a_i)$ is approximated by the complement of x (where this approximation is implied by the interpretational feasibility rather than being dictated by any formal optimization problem), $\mathrm{INCL}(x, a_i) \approx 1 - x$. For the dominance neuron, we have the expression for the respective binary version of DOM, $\mathrm{DOM}^{\sim}$

$$\mathrm{DOM}^{\sim}(x, a_i) = \begin{cases} \mathrm{DOM}(x, a_i), & \text{if } a_i \leq \mu \\ x, & \text{if } a_i > \mu \end{cases}. \tag{13}$$

2) For the linguistic hedges associated with the AND neurons. Here the hedge $\tau = 0$ leads to the unknown input and as such this input could be completely eliminated. $(x^0 = 1)$.

3) For the OR neuron the weakest connections are eliminated as the corresponding inputs do not impact the output. We have

$$w_i^{\sim} = \begin{cases} 1, & \text{if} w_i > \lambda \\ 0, & \text{if} w_i \leq \lambda \end{cases}. \tag{14}$$

The connection set up to 1 is deemed essential. If we accept a single threshold level of 0.5 and apply this consistently to the all the connections of the network and set up the threshold 0.1 for the inclusion neuron, the statement

$$y = \text{more or less}[[x_1 \text{ included in} 0.6] \text{ or } 0.2 \text{ and } [x_2 \text{ included in } 0.9] \text{ or } 0.7]$$

translates into a concise (yet approximate) version assuming the form of the following logic expression

$$y = \text{more or less } [x_1 \text{ included in } 0.6]$$

The choice of the threshold value could be a subject of a separate optimization phase but we can also admit some arbitrarily values especially if we are focused on the interpretation issues. One should become aware (which is quite intuitive, though) that by increasing the threshold levels and, thus, making more radical changes to the connections of the network, its performance expressed in terms of approximation abilities could be reduced. The size of the network is reduced that enhances its interpretation abilities. Interestingly, the issue of approximation, interpretation tradeoff arises here and through the threshold values, we are provided with an effective instrument to control this important modeling aspect.

VIII. Experimental Studies

The experiments reported in this section are intended to illustrate the development, performance, and interpretation issues of the proposed network. In all experiments, the t-norm is implemented as a product operator while the s-norm treated as a probabilistic sum. Furthermore in training the network we adhere to the 60%–40% data split with 60% of randomly selected data being used for the training of the network. The performance index Q used in the experiments on basis of which the fitness function is developed concerns a RMSE measure given in a standard format ($\mathrm{RMSE} = \sqrt{(1/N)\sum_{k=1}^{N}(y_k - \hat{y}_k)^2}$ with $\hat{y}_k$ being the output of the network). These values concerns the output normalized to the unit interval. In plots and while reporting results, we also show the performance index for the original (not normalized) output variable; its values will be denoted by V.

We start with a low-dimensional synthetic data set $\{(\mathbf{x}(k), y(k))\}$ shown in Fig. 12. These data points are here for illustrative purposes and are not governed by any specific probabilistic characteristics. As shown, there are three inputs and a single output.

In the series of experiments we changed the number of the referential neurons as well as modified the number of AND neurons in the hidden layer. The GA used a population of 100 individuals and was run for 500 generations. The optimal configuration emerged in the form of 3 reference neurons and 3 AND neurons; it leads to the lowest value of the performance index on the

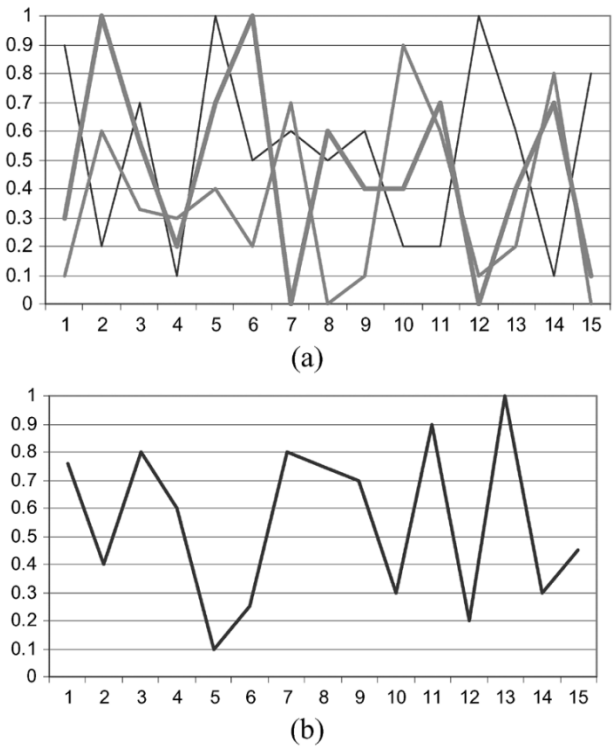

Fig. 12. Synthetic data set (a) input variables ($\mathbf{x}$) and (b) output (y).

training set and this is confirmed by the lowest value achieved on the testing set (as visualized in Fig. 13).

The progress of the optimization is gauged in terms of the fitness function of the population (average fitness) as well as the best individual obtained in the successive generations, see Fig. 14.

The interpretation of the network leads us to an interesting logic description of the data. The referential neurons located in the second layer are of all types (DOM, SIM, and INCL). They come with the following connections and reference points

$$\begin{array}{lll} \text{DOM:} & \mathbf{w} = [0.36\ 0.95\ 0.73] & \mathbf{a} = [0.23\ 0.69\ 0.87] \\ \text{INCL:} & \mathbf{w} = [0.12\ 0.61\ 0.02] & \mathbf{a} = [0.69\ 0.61\ 0.70] \\ \text{SIM:} & \mathbf{w} = [0.02\ 0.16\ 0.96] & \mathbf{a} = [0.90\ 0.29\ 0.48] \end{array}$$

The connections of the AND neurons aggregating the signals of the referential neurons are equal to $[2\ 0.5\ 4], [0.5\ 4\ 0]$, and $[0.5\ 0\ 4]$, respectively. Finally, the connections of the OR neuron are equal to $\mathbf{v} = [0.26\ 0.84\ 0.92]$.

If we proceed with the thresholding operation (assuming the same threshold level of 0.5 for the aggregative neurons and 0.05 and 0.95 for the reference neurons as well as following the pruning mechanism of the rules outlined in Section VII) and reduce the least significant connections then the core part of the network arises in the following format:

Reference neurons

$z_1 = \text{DOM}(x_1, 0.23)$

$z_2 = \text{INCL}(x_1, 0.69)$ and $\text{INCL}(x_2, 0.7)$

$z_3 = \text{SIM}(x_1, 0.9)$ and $\text{SIM}(x_2, 0.29)$

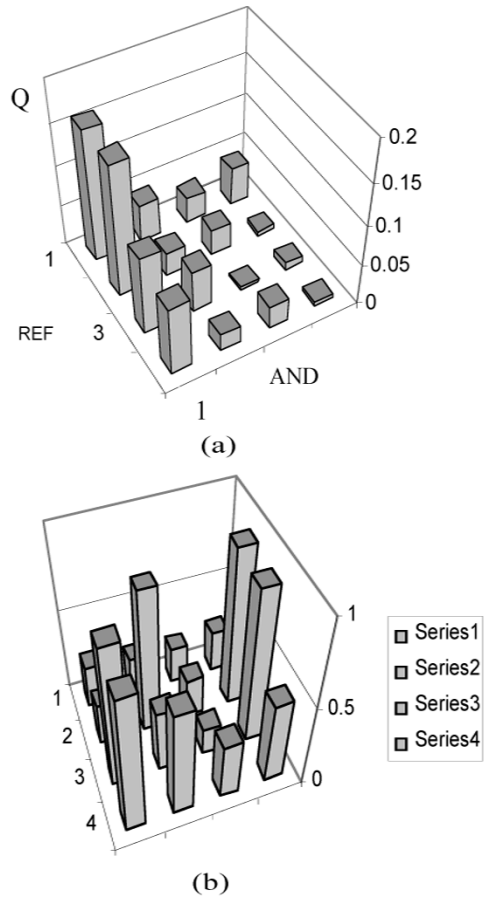

Fig. 13. Performance index (Q) on the (a) training and (b) testing set versus the number of reference and AND neurons.

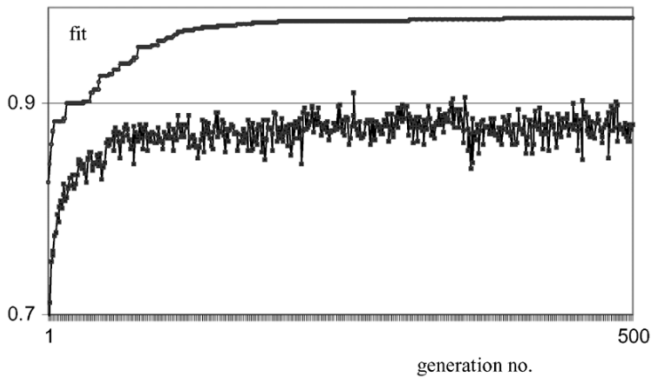

Fig. 14. Fitness function (average and best individual) in successive generations. The results are given for the best configuration (three referential neurons and three AND neurons).

AND neurons

$u_1 =$ very (z_1) and more or less (z_2) and very very (z_3)

$u_2 =$ more or less $(z_1)$and very very (z_2)

$u_3 =$ more or less (z_1)andvery very (z_3)

OR neuron

$y = u_2$ or $u_3 = \{$more or less $(\text{DOM}(x_1, 0.23)$ and very very $[(\text{INCL}(x_1, 0.69)$ and $\text{INCL}(x_2, 0.7)]\}$ or or [more or less $(\text{DOM}(x_1, 0.23)$ and very very $(\text{SIM}(x_1, 0.9)$ and $\text{SIM}(x_2, 0.29))]$.

TABLE I
INPUT VARIABLES AND THE RESULTS OF ITS ENCODING (EITHER USING A COLLECTION OF FUZZY SETS OR 1-OUT-OF-n DECODING)

Variable name and type (continuous, C or discrete, D)	Variable's number (internal to the network)
Number of cylinders (D)	x_1, x_2, x_3, x_4, x_5
Displacement (C)	x_6, x_7, x_8
Horsepower (C)	x_9, x_{10}, x_{11}
Weight (C)	x_{12}, x_{13}, x_{14}
Acceleration (C)	x_{15}, x_{16}, x_{17}
Model year (C)	x_{18}, x_{19}, x_{20}
Origin (D)	x_{21}, x_{22}, x_{23}

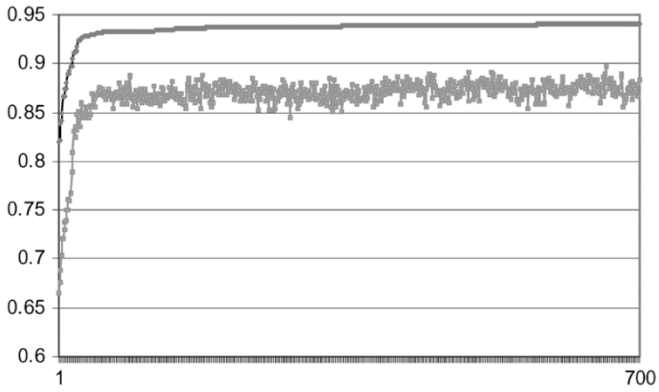

Fig. 15. Genetic optimization of the network; fitness function (best and average) produced in successive generations of the GA.

The two subsequent datasets come from the machine learning repository [17] and concern auto and Boston housing data.

1) Auto Data Set: This dataset deals with the description of various car makes completed in terms of several main characteristics such as number of cylinders, acceleration, and gas consumption. We can develop a logic description of such data in which we predict gas consumption (mpg) as a logic function of the parameters of a vehicle. As the data considered here are either continuous (acceleration) or discrete with a few nominal values, we develop an interface in which any continuous variable is granulated through a collection of fuzzy sets while a discrete variable is coded in the form of 1-out-of-n. For each continuous variable we use three Gaussian membership functions uniformly distributed across the universe of discourse. This selection is primarily dictated by the interpretability of such granules (as they could easily assume some clear semantics such as low, medium, and high). The overlap between two adjacent fuzzy sets is set to 0.5 (which makes these terms easily distinguishable and enhances their semantics). With this form of the interface, we end up with 23 inputs to the network. The output is linearly normalized by converting the range [9, 46.6] into the unit interval. Bearing the encoding used in this network, the details of this mapping clearly identifying each variable are included in Table I.

In the development of the network we use 60% of the dataset (training data) selected at random from the entire set with the rest used for testing purposes. The population consists of 200 individual and the genetic optimization was run for 700 generations. The mutation and crossover rates were equal to 0.10 and 0.80, respectively. As shown in Fig. 15, the genetic optimization proceeds in a fairly typical way where in the first generations most of the optimization takes place. Because of the elitist strategy, the best individual shows up quite quickly and remains for the rest of the course of the GA optimization (the curve saturates quite quickly with some minor enhancements obtained during the later part of the process).

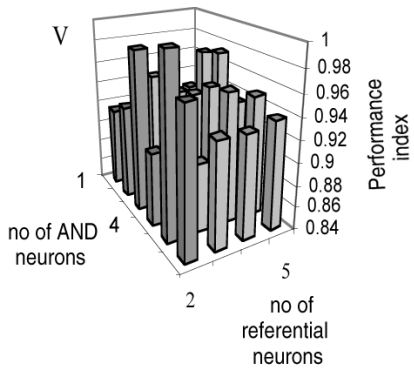

Fig. 16. Performance index V as a function of the number of referential and AND neurons.

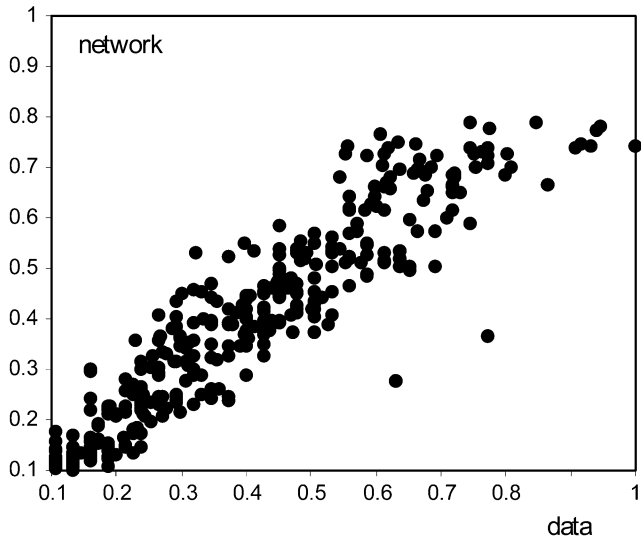

Fig. 17. Network versus data. All data shown with a few discrepancies between the network and data.

Again we found these values to be typical for the experiments and in line with the typical range of the values of the GA parameters encountered in the literature. The experiments were carried out for different combinations of the AND and reference neurons. The performance index treated as a function of these two parameters is shown in Fig. 16. These results lead to the optimal configuration of the neurons with three reference neurons and four AND neurons.

The overall performance of the network (shown for the entire dataset) is presented in Fig. 17 with the experimental data visualized with respect to the corresponding outputs of the network; the cloud of the points obtained in this way locates around the straight line of the slope 45°.

The optimal network comes with the three referential neurons: two inclusion and one dominance neuron. The connections and the reference points of them as well as the connections of the AND and OR neuron are all shown in Figs. 18 and 19. Our intent is to deliver a certain "global" picture as to the meaning of the connections rather than get into details so that we can focus

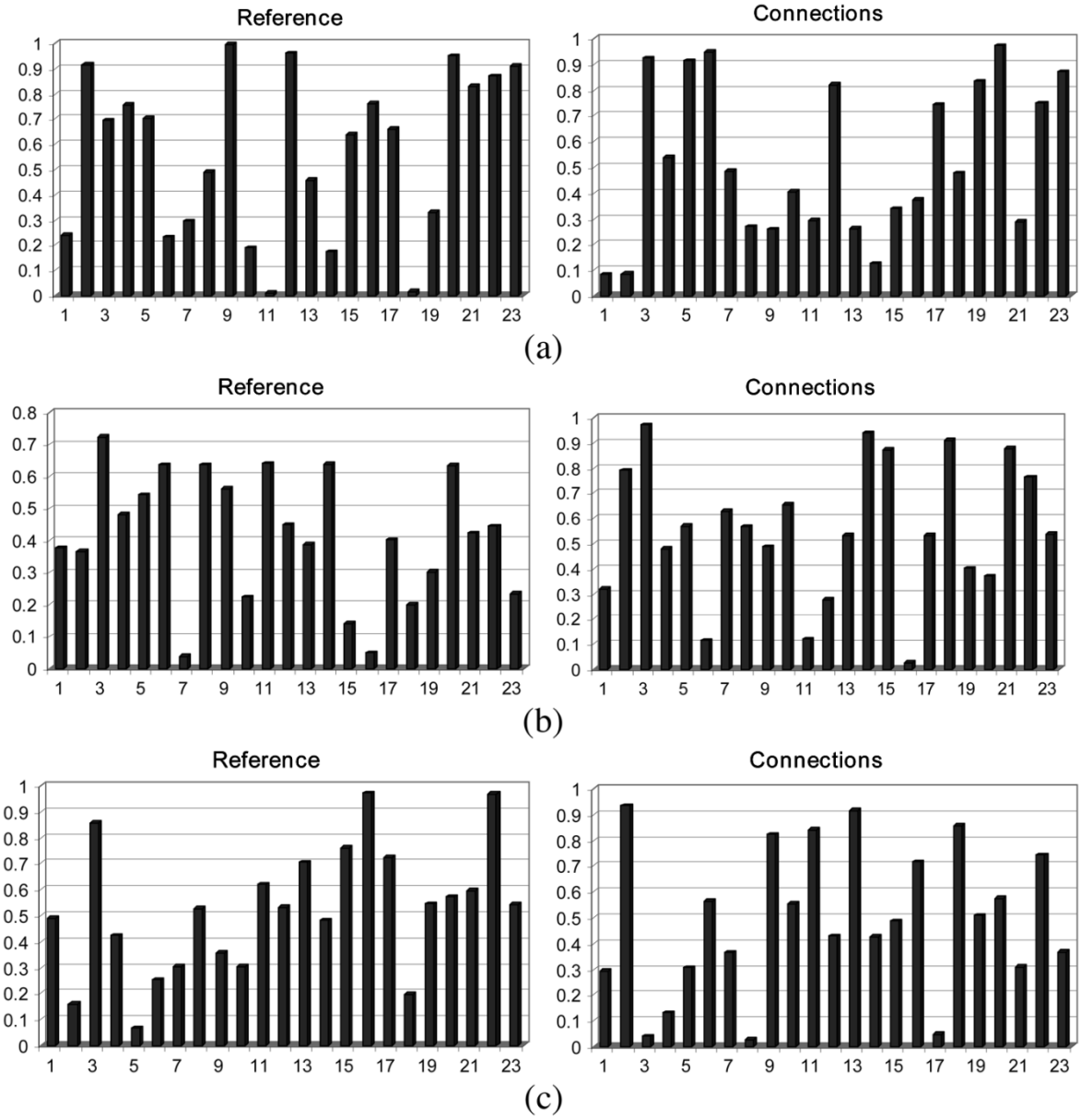

Fig. 18. Connections and reference points of the referential neurons. (a) First INCL neuron. (b) DOM neuron. (c) Second INCL neuron.

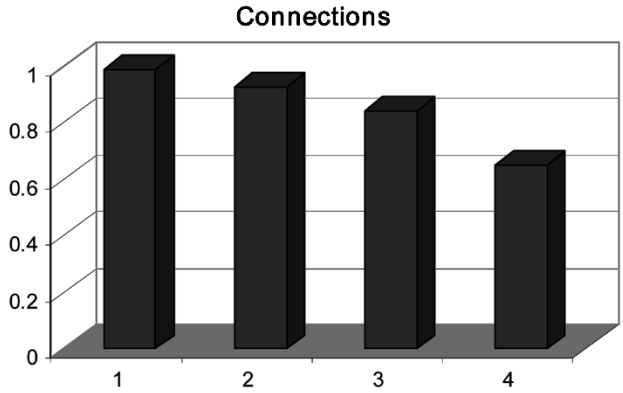

Fig. 19. Connections of the OR neuron.

on the pruning of the network and produce a compact description of data.

The reference neurons give rise to highly heterogeneous space; remarkably different values of the connections corresponding with the specific inputs have been selected. The linguistic hedges (τ) associated with the three AND neurons are determined to be equal to [0.5 2 2], [2 2 0.5], [0.5 0 0], and [1 1 4]. In case of the third neuron, two of its inputs become irrelevant (unknown).

We observe a substantial diversity in the values of the connections. For instance, in the third AND neuron we have only a single significant connection while the remaining are set up to 1 (reflecting the unknown aspect of the corresponding input to this neuron).

Using the thresholding operation, the core part of the network can be easily revealed. The arbitrarily assumed value of the threshold level for the connections is taken as 0.3. In this case, all the connections of the OR neuron are retained. The subsets

TABLE II
SUBSETS OF THE VARIABLES RETAINED BY THE REFERENCE NEURONS; INCLUDED ARE THE VALUES OF THE ASSOCIATED REFERENCE VALUES OFTHE REFERENCE POINT

Reference neuron	Variable number	Reference point
INCL-1	1, 2, 8, 9, 11, 13, 21	0.24, 0.91, 0.49, 0.99, 0.01, 0.46, 0.17
DOM	6, 11, 12, 16	0.63, 0.64, 0.45, 0.05
INCL-2	1, 3, 4, 8, 17	0.49, 0.86, 0.42, 0.53, 0.73

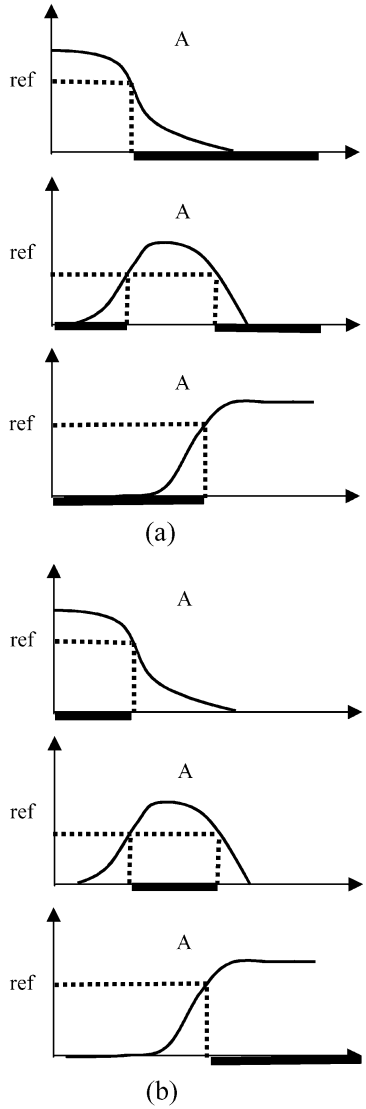

Fig. 20. Regions (intervals) of input variable resulting from the satisfaction of the logic predicates of inclusion and dominance and induced by different forms of the membership function. (a) Inclusion, INCL(x is A, ref) and (b) dominance, DOM(x is A, ref).

of the variables of the retained by the corresponding reference neurons are included in Table II.

The logic description of data can be directly inferred from the network developed so far. A few observations are helpful in this regard. In case of continuous variable that has been discretized (quantized) in terms of fuzzy sets, the logic expression INCL(A, ref) where the satisfaction of this predicate attains high-truth values (preferably close to 1) induces some corresponding regions positioned in the original input variable as visualized in Fig. 20. Note that the location of the region depends upon the form of the membership function.

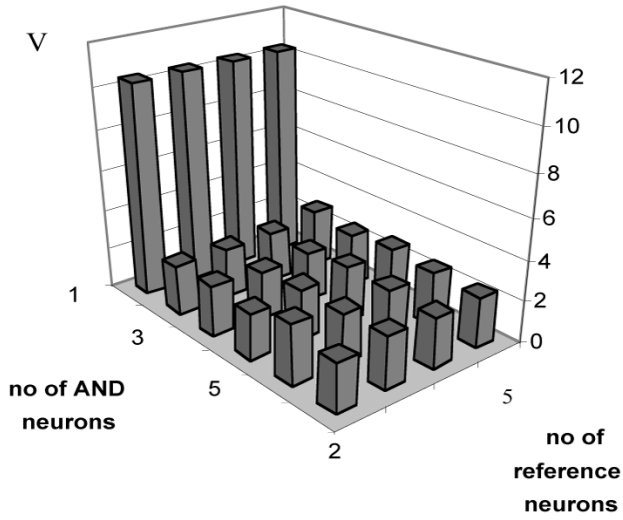

Fig. 21. Values of the performance index Q for the training set.

For the discrete inputs with the 1-out of-n decoding, the interpretation of the logic expression INCL(x is $\{\xi\}$, ref) with ξ being a discrete value of the variable gives rise to the expression

x is ξ with confidence level equal to ref.

With these interpretation mechanisms in mind, the network translates into the following expression:

$$\begin{aligned}&\text{mpg}\\&\quad=\text{INCL}\,(4\ \text{cylinders}, 0.24)\ \text{and INCL}(6\ \text{cylinders}, 0.91)\\&\qquad\text{and INCL}\,(\text{displacement is large}, 0.49)\ \text{and}\\&\qquad\text{INCL}\,(\text{horsepower is high}, 0.01)\ \text{and}\\&\qquad\text{INCL}\,(\text{weight is medium}, 0.46)\ \text{and}\\&\qquad\text{INCL}\,(\text{origin}=\text{Japanese}, 0.17)\}\\&\text{or}\\&\quad\{\text{DOM}\,(\text{displacement is small}, 0.63)\ \text{and}\\&\qquad\text{DOM}\,(\text{horsepower is high}, 0.64)\ \text{and}\\&\qquad\text{DOM}\,(\text{weight is low}0.45)\ \text{and}\\&\qquad\text{DOM}\,(\text{acceleration is medium}, 0.05)\}\\&\text{or}\\&\quad\{\text{INCL}\,(4\ \text{cylinders}, 0.49)\ \text{and INCL}\,(6\ \text{cylinders}, 0.86)\\&\qquad\text{and INCL}\,(8\ \text{cylinders}, 0.42)\ \text{and}\\&\qquad\text{and INCL}(\text{displacement is low}, 0.53)\\&\qquad\text{and INCL}(\text{acceleration is high}, 0.73)\}.\end{aligned}$$

Higher values of mpg (let us stress that their values are normalized to the unit interval) are associated with higher truth values of the compound logic expression describing the data.

2) Boston Housing Data: The dataset concerns the price of real estate in the Boston area providing several features of the houses (age, distance to employment centers, number of rooms, student-teacher ratio, etc) and their prices. In our discussion, the

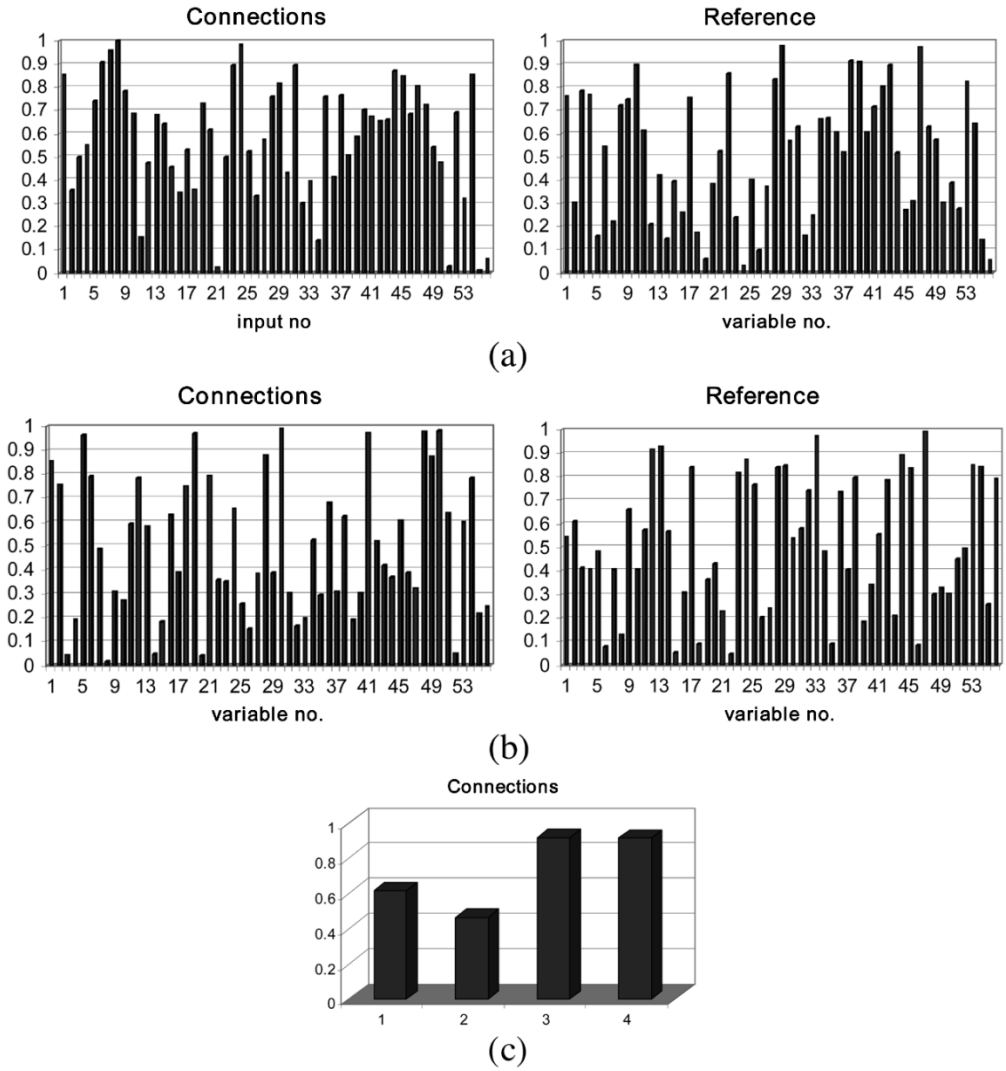

Fig. 22. Parameters of the fuzzy neurons. (a) Inclusion neuron. (b) Similarity. (c) OR neuron in the output layer.

price is treated as the dependent variable whose logic relationship with the inputs we are interested in. The setup of the genetic optimization is the same as in the previous case. With the same coding (3 fuzzy sets per variable and 1-out of-n coding) we end up with 56 input variables. The optimization with respect to the number of reference and AND neurons leads to the configuration of 2–4 (2 reference neurons, 4 AND neurons), refer to Fig. 21.

The parameters of the neurons (reference points and their connections) are illustrated in Fig. 22. The optimal configuration leading to the minimal value of the performance index on the training set involves four AND neurons and two reference neurons (one inclusion and one similarity neuron). The performance index V is equal to 2.283; its value on the testing set is equal to 2.468. Bearing in mind the original range of the output that is [5, 50] this amounts to about 3.4% and 3.5% of its total range (that $\sqrt{2.283}/45$ and $\sqrt{2.468}/45$).

The connections of the AND neurons are equal to [0 1], [4 1], [0.5 0], and [0.5 0]. It becomes evident that some inputs are irrelevant (those associated with zeros that is being regarded as unknown). For the OR neuron, one of the connections is subject to pruning as its value is lower than 0.5.

IX. Conclusion

Being motivated by the genuine need of constructing networks that exhibit plasticity and retaining interpretability, we have developed a heterogeneous structured composed of logic neurons. The two main categories of aggregative and reference neurons are deeply rooted in the fundamental operations encountered in fuzzy sets (including logic operations, linguistic modifiers, and logic reference operations). The direct interpretability of the network we addressed in the study helps develop a logic description of data. As the network takes advantage of using various neurons, this imposes an immediate requirement of structural optimization and leads to the utilization of the mechanisms of genetic optimization (genetic algorithms). This study was aimed at addressing the fundamental conceptual and algorithmic (design) issues of the networks. There are a number of advanced issues worth pursuing including.

- Interpretation mechanisms with an emphasis placed on the systematic pruning process that should be guided by some well-defined development criteria. The two potential candidates could involve a criterion of accuracy (which ex-

presses how much the pruning affects the performance index) and a criterion of interpretability. This one could be more difficult to interpret and attach a tangible interpretation; as one among possible alternatives we can count the number of connections dropped and link it to the structural measure of complexity (such as, e.g., Akaike criterion in system identification) of the reduced network.

Developing templates of networks for mapping domain knowledge onto a network's topology; one can envision that building a number of "templates" could be of substantial benefit

- The realization of the templates may trigger more research into hierarchical networks composed of a collection of specialized subnetworks.

REFERENCES

[1] J. E. Baker, "Adaptive selection methods for genetic algorithms," in *Proc. 1st Int. Conf. Genetic Algorithms*, 1985, pp. 101–111.

[2] A. Bargiela and W. Pedrycz, *Granular Computing: An Introduction*. Norwell, MA: Kluwer, 2002.

[3] J. Casillas *et al.*, Ed., *Interpretability Issues in Fuzzy Modeling*, Berlin, Germany: Springer-Verlag, 2003.

[4] J. A. Dickerson and M. S. Lan, "Fuzzy rule extraction from numerical data for function approximation," *IEEE Trans. Syst., Man, Cybern.*, vol. 26, pp. 119–129, Aug. 1995.

[5] A. Di Nola, W. Pedrycz, and S. Sessa, "Fuzzy relational structures: The state of art," *Fuzzy Sets Syst.*, vol. 75, pp. 241–262, 1995.

[6] A. F. Gomez-Skarmeta, M. Delgado, and M. A. Vila, "About the use of fuzzy clustering techniques for fuzzy model identification," *Fuzzy Sets Syst.*, vol. 106, pp. 179–188, 1999.

[7] L. J. Eshelman and J. D. Schaffer, "Real-coded genetic algorithms and interval schemata," in *Foundations of Genetic Algorithms 2*. San Mateo, CA: Morgan Kaufman, 1993, pp. 187–202.

[8] J. Fodor and M. Roubens, *Fuzzy Preference Modeling and Multicriteria Decision Support*, MA: Kluwer, 1994.

[9] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[10] ——, "Real-coded genetic algorithms, virtual alphabets, and blocking," *Complex Syst.*, no. 5, pp. 139–167, 1991.

[11] F. Herrera, M. Lozano, and J. L. Verdegay, "Tackling real-coded genetic algorithms: Operators and tools for behavioral analysis," *Artif. Intell. Rev.*, vol. 12, pp. 265–319, 1998.

[12] K. Hirota and W. Pedrycz, "OR/AND neuron in modeling fuzzy set connectives," *IEEE Trans. Fuzzy Syst.*, vol. 2, pp. 151–161, May 1994.

[13] ——, "Fuzzy relational compression," *IEEE Trans. Syst., Man, Cybern. B*, vol. 29, pp. 407–415, June 1999.

[14] J. Jang, C. T. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*. Englewood Cliffs, NJ: Prentice-Hall, 1997.

[15] B. Kosko, *Neural Networks and Fuzzy Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1991.

[16] T. Lin and C. S. Lee, *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1996.

[17] C. J. Merz and P. M. Murphy. UCI Repository of Machine Learning Databases. Dept. Inform. Comput. Sci., Univ. California, Irvine. [Online]. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html

[18] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd ed. Berlin, Germany: Springer-Verlag, 1996.

[19] S. Mitra and S. K. Pal, "Logical operation based fuzzy MLP for classification and rule generation," *Neural Netw.*, vol. 7, pp. 353–373, 1994.

[20] ——, "Fuzzy multiplayer perceptron, inferencing, and rule generation," *IEEE Trans. Neural Networks*, vol. 6, pp. 51–63, Jan. 1995.

[21] S. K. Pal and S. Mitra, *Neuro-Fuzzy Pattern Recognition*. New York: Wiley, 1999.

[22] W. Pedrycz, "Approximate solutions of fuzzy relational equations," *Fuzzy Sets Syst.*, vol. 26, pp. 183–202, 1988.

[23] ——, "A fuzzy cognitive structure for pattern recognition," *Pattern Recognit. Lett.*, vol. 9, pp. 305–313, 1989.

[24] ——, "Processing in relational structures: Fuzzy relational equations," *Fuzzy Sets Syst.*, vol. 40, pp. 77–106, 1991.

[25] ——, "Neurocomputations in relational systems," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 13, pp. 289–297, Mar. 1991.

[26] ——, "Selected issues of frame of knowledge representation realized by means of linguistic labels," *J. Intell. Syst.*, vol. 7, pp. 155–169, 1992.

[27] ——, "Fuzzy neural networks and neurocomputations," *Fuzzy Sets Syst.*, vol. 56, pp. 1–28, 1993.

[28] W. Pedrycz and A. Rocha, "Knowledge-based neural networks," *IEEE Trans. Fuzzy Syst.*, vol. 1, pp. 254–266, Nov. 1993.

[29] W. Pedrycz, P. Lam, and A. F. Rocha, "Distributed fuzzy modeling," *IEEE Trans. Syst., Man, Cybern.*, vol. 25, pp. 769–780, May 1995.

[30] W. Pedrycz, "Interfaces of fuzzy models: A study in fuzzy information processing," *Inform. Sci.*, vol. 90, pp. 231–280, 1996.

[31] W. Pedrycz and F. Gomide, *An Introduction to Fuzzy Sets: Analysis and Design*. Cambridge, MA: MIT Press, 1998.

[32] M. Setnes, R. Babuska, and H. Vebruggen, "Rule-based modeling: Precision and transparency," *IEEE Trans. Syst., Man, Cybern. C*, vol. 28, pp. 165–169, Feb. 1998.

[33] T. Sudkamp and R. J. Hammel II, "Rule base completion in fuzzy models," in *Fuzzy Modeling: Paradigms and Practice*, W. Pedrycz, Ed. Norwell, MA: Kluwer, 1996, pp. 313–330.

Witold Pedrycz (M'88–SM'94–F'99) is a Professor and Canada Research Chair (CRC) in the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, Canada. He is actively pursuing research in computational intelligence, fuzzy modeling, knowledge discovery and data mining, fuzzy control including fuzzy controllers, pattern recognition, knowledge-based neural networks, relational computation, bioinformatics, and software engineering. He has published numerous papers in this area. He is also an author of eight research monographs covering various aspects of computational intelligence and software engineering.

Dr. Pedrycz has been a member of numerous program committees of IEEE conferences in the area of fuzzy sets and neurocomputing. He currently serves as an Associate Editor of IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS Part A and B, and IEEE TRANSACTIONS ON FUZZY SYSTEMS. He is Editor-in-Chief of *Information Sciences* and President-Elect of North American Fuzzy Information Processing Society (NAFIPS) and International Fuzzy System Association (IFSA).