Engineering Applications of Artificial Intelligence 85 (2019) 845–864

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Heuristic design of fuzzy inference systems: A review of three decades of research

Varun Ojha [a,b,*], Ajith Abraham [c,d], Václav Snášel [e]

[a] ETH Zürich, Switzerland
[b] University of Reading, United Kingdom
[c] University of Pretoria, South Africa
[d] Machine Intelligence Research Labs (MIR Labs), WA, United States
[e] Technical University of Ostrava, Czech Republic

ARTICLE INFO

Keywords:
Evolutionary algorithms
Genetic fuzzy systems
Neuro-fuzzy systems
Hierarchical fuzzy systems
Evolving fuzzy systems
Multi-objective fuzzy systems
Deep fuzzy system

ABSTRACT

This paper provides an in-depth review of the optimal design of type-1 and type-2 fuzzy inference systems (FIS) using five well known computational frameworks: genetic-fuzzy systems (GFS), neuro-fuzzy systems (NFS), hierarchical fuzzy systems (HFS), evolving fuzzy systems (EFS), and multi-objective fuzzy systems (MFS), which is in view that some of them are linked to each other. The heuristic design of GFS uses evolutionary algorithms for optimizing both Mamdani-type and Takagi–Sugeno–Kang-type fuzzy systems. Whereas, the NFS combines the FIS with neural network learning systems to improve the approximation ability. An HFS combines two or more low-dimensional fuzzy logic units in a hierarchical design to overcome the curse of dimensionality. An EFS solves the data streaming issues by evolving the system incrementally, and an MFS solves the multi-objective trade-offs like the simultaneous maximization of both interpretability and accuracy. This paper ofers a synthesis of these dimensions and explores their potentials, challenges, and opportunities in FIS research. This review also examines the complex relations among these dimensions and the possibilities of combining one or more computational frameworks adding another dimension: deep fuzzy systems.

1. Introduction

Research in fuzzy inference systems (FIS) initiated by Zadeh (1988) has drawn the attention of many disciplines over the past three decades. The success of FIS is evident from its applicability and relevance in numerous research areas: control systems (Lee, 1990; Wang and Griffin, 1996), engineering (Precup and Hellendoorn, 2011), medicine (Jain et al., 2017), chemistry (Komiyama et al., 2017), computational biology (Jin and Wang, 2008), finance and business (Bojadziev, 2007), computer networks (Elhag et al., 2015; Gomez and Dasgupta, 2002), fault detection and diagnosis (Lemos et al., 2013), pattern recognition (Melin et al., 2011). These are just a few among numerous FIS's successful applications (Liao, 2005; Castillo and Melin, 2014), which are mainly attributable to the FIS's ability to manage uncertainty and computing for noisy and imprecise data (Zadeh and Kacprzyk, 1992).

The enormous amount of research and innovations in multiple dimensions of FIS propelled its success. These research dimensions realize the concept of: genetic-fuzzy systems (GFS), neuro-fuzzy systems (NFS), hierarchical fuzzy systems (HFS), evolving fuzzy systems (EFS), and multiobjective fuzzy systems (MFS), which are fundamentally relied on two basic fuzzy rule types: Mamdani type (Mamdani, 1974), and Takagi–Sugeno–Kang (TSK) type (Takagi and Sugeno, 1985). Both rule types have "IF X is A THEN Y is B" rule structure, i.e., the rules are in the *antecedent* and *consequent* form. However, the rule types Mamdani and TSK differ in their respective consequent forms. A Mamdani-type rule takes an output action (a class) and TSK-type rule takes a polynomial function as the consequent. Thus, they differ in their approximation ability. The Mamdani-type has a better interpretation ability, and the TSK-type has a better approximation accuracy. For antecedent, both types take a similar form. That is a *rule induction* process take place for input space partition to form antecedent part of a rule. Therefore, the *rule types*, the *rule induction* process, and the *interpretability–accuracy* trade-off govern the FIS's dimensions.

In GFS, researchers investigate mechanisms to encode and optimize the FIS's components. The encoding takes place in the form of genetic vectors and genetic population and the optimization takes place in the form of FIS's structure and parameters adaptation. Herrera (2008), Cordón et al. (2004), and Castillo and Melin (2012) summarized research in GFS with a taxonomy to explain both encoding and structure optimization using a genetic algorithm (GA).

For NFS, researchers investigate network structure formation and parameter optimization (Jang, 1993) and answer the variations in network formation methods and the variations in parameter optimization techniques. Buckley and Yoichi (1995), Andrews et al. (1995),

* Corresponding author at: University of Reading, United Kingdom.
E-mail address: v.k.ojha@reading.ac.uk (V. Ojha).

https://doi.org/10.1016/j.engappai.2019.08.010
Received 18 December 2018; Received in revised form 7 April 2019; Accepted 19 August 2019
Available online xxxx

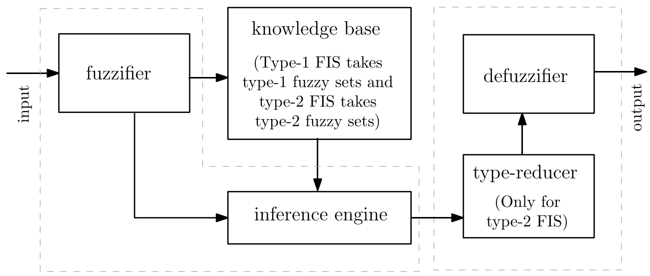

Fig. 1. Typical fuzzy inference system.

and Karaboga and Kaya (2018) offer summaries of such variations. Torra (2002) and Wang et al. (2006) reviewed research in HFS which summarizes the variations in HFS design types and HFS parameter optimization techniques. The EFS research enables the incremental learning ability into FISs (Kasabov, 1998; Angelov and Zhou, 2008), and the MFS research enables FISs to deal with multiple objectives simultaneous (Ishibuchi, 2007; Fazzolari et al., 2013).

This review paper offers a synthesized view of each dimension: GFS, NFS, HFS, EFS, and MFS. The synthesis recognizes these dimensions being linked to each other where the concept of one dimension applies to another. For example, NFS and EFS models can be optimized by GA. Hence, GFS entails its concepts to NFS and EFS. The complexity and concept arises from the synthesis offer a potential to investigate *deep fuzzy systems* (DFS), which may take advantage of GFS, HFS, and NFS simultaneously in a hybrid manner where NFS will offer solutions to network structure formation, HFS may offer solutions to resolving hierarchical arrangement of multiple layers, and GFS may offer solutions to parameter optimization. Moreover, EFS and MFS also play a role in DFS if the goal will be to construct a system for the data stream and to optimize a system for interpretability-accuracy trade-off.

This review walks through each dimension: GFS, NFS, HFS, EFS, and MFS, including a discussion on the standard FIS. First, the rule structure, rule types, and FISs types are discussed in Section 2. A discussion on the FIS's designs describing how various FIS's paradigms emerged through the interaction of FIS with neural networks (NN) and evolutionary algorithms (EA) is given in Section 2.3. Section 3 discusses the GFS paradigm which emerged through FIS and EA combinations. Section 4 describes the NFS paradigm including reference to self-adaptive and online system notions (Section 4.1); basic NFS layers (Section 4.2); and feedforward and feedback architectures (Section 4.3). They are followed by the discussions on the HFS's properties and the HFS's implementations (Section 5). Section 6 summarized the EFS which offers an incremental leaning view in FIS. Section 7 offered the discussions on MFS which covers the Pareto-based multiobjective optimization and the FIS's multiple objectives trade-offs implementations. Followed by the challenges and the future scope in Section 8, and conclusions in Section 9.

2. Fuzzy inference systems

A standard FIS (Fig. 1) is composed of the following components:

(1) a **fuzzifier** unit that fuzzifies the input data;
(2) a **knowledge base** (KB) unit, which contains fuzzy rules of the form IF-THEN, i.e.,
IF a set of conditions (antecedent) is satisfied
THEN a set of conditions (consequent) can be inferred
(3) an **inference engine** module that computes the rules firing strengths to infer knowledge from KB; and
(4) a **defuzzifier** unit that translates inferred knowledge into a rule action (crisp output).

The KB of the FIS is composed of a database (DB) and a *rule-base* (RB). The DB assigns fuzzy sets (FS) to the input variables and the FSs transforms the input variables to fuzzy membership values. For *rule induction*, RB constructs a set of rules fetching FSs from the DB.

In a FIS, an input can be a numeric variable or a linguistic variable. Moreover, an input variable can be singleton [Fig. 2(a)] and non-singleton [Fig. 2(b)]. Accordingly, a FIS is **singleton FIS** if it uses singleton inputs, i.e., FIS uses crisp and precise single value measurements as the input variables, which is the most common practice. However, real-world problems, especially in engineering, measurements are noisy, imprecise, and uncertain. Thus, FIS that uses non-singleton input is a **non-singleton FIS**. In principle, a non-singleton FIS differs with a singleton FIS in their respective input fuzzification processes where a "fuzzifier" transform a non-singleton input or a singleton input to a fuzzy membership value.

A fuzzifier maps a singleton input (crisp input) $X_j \in \mathbf{X}$, $\mathbf{X} = (X_1, X_2, \ldots, X_P)$ for m_{Xj} (a value in X_j) [Fig. 2(a)] to the following membership function for the input fuzzification:

$$\mu_{X_j}(X_j) = \begin{cases} 1, & X_j = m_{Xj} \\ 0, & X_i \neq m_{Xj} \quad \forall X_j \in \mathbf{X} \end{cases} \tag{1}$$

For non-singleton inputs, a fuzzifier maps input X_j (that is considered as noisy, imprecise, and uncertain) onto a Gaussian function (typical choice for numeric variables) as:

$$\mu_{X_j}(X_j) = f(X_j) = \exp\left[-\frac{1}{2}\left(\frac{X_j - m_{Xj}}{\sigma_X}\right)^2\right] \tag{2}$$

where m_{Xj} is input (considered as mean, a value along line X_j) and $\sigma_X \geq 0$ is the standard deviation (std.) that defines the spread of the function μ_{X_j}. The value of the FS at m_{Xj} is $\mu_{X_j}(m_{Xj}) = 1$ and $\mu_{X_j}(X_j)$ decreases from unity as X_j moves away from m_{Xj} (Mouzouris and Mendel, 1997). In general, for a singleton or non-singleton input X_j, the inference engine's output μ_{AX_j} is a combination of fuzzified input $\mu_{X_j}(X_j)$ with an antecedent FS $\mu_{A_j}(X_j)$ as per:

$$\mu_{AX_j}(\bar{X}_j) = \sup\left\{\mu_{X_j}(X_j) \star \mu_{A_j}(X_j)\right\} \tag{3}$$

where $\star$ is *t-norm* operation that can be minimum or product and $\bar{X}_j$ indicate supremum of Eq. (3). Fig. 3 is an example of the product operation in Eq. (3). Fig. 3 evaluates the product of input FS μ_X and the antecedent fuzzy set μ_A that results in $\mu_{AX} = 0.04$ where $m_{Xj} = 2.0$, $\sigma_X = 2.0$, $m_A = 0.0$, and $\sigma_A = 1.5$. The product $\mu_{AX}(\bar{X})$ gives a maximum value at $\bar{X} = 0.72$ (in Fig. 3) which is calculated as:

$$\bar{X} = \frac{m_A\sigma_X^2 + m_X\sigma_A^2}{\sigma_A^2 + \sigma_X^2} \tag{4}$$

The design of RB further distinguishes the type of FISs: a **Mamdani-type FIS** (Mamdani, 1974) or a **Takagi–Sugeno–Kang (TSK)-type FIS** (Takagi and Sugeno, 1985). A TSK-type FIS differs with a Mamdani-type FIS only in the implementation of fuzzy rule's consequent part. In Mamdani-type FIS, a rule's consequent part is an FS, whereas, in TSK-type FIS, a rule's consequent part is a polynomial function.

The DB contains the FSs which are either type-1 fuzzy sets (T1FS) or type-2 fuzzy sets (T2FS). Morover, the FSs are defined using a fuzzy membership functions (MF) and the basic form of which is coined as a T1FS. Whereas, T2FS allows an MF to be fuzzy itself by extending membership value into an additional membership dimension. Hence, the fuzzy set (FS) types also differentiate FIS types: **type-1 FIS** (T1FIS) and the **type-2 FIS** (T2FIS).

For simplicity, this paper is singleton FIS centric and refers non-singleton FIS to the appropriate research. As well as, since Mamdani-type FIS differs with TSK-type FIS only in its consequent part, this paper focuses on TSK-type FIS.

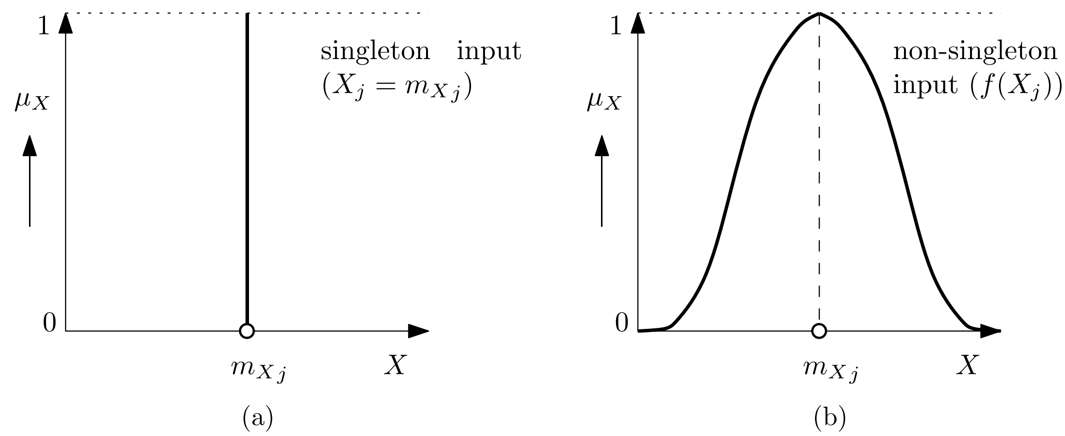

Fig. 2. Input examples: (a) singleton input $\mu_{X_j}(X_j)$ and (b) non-singleton input $\mu_{X_j}(X_j) = f(X_j)$.

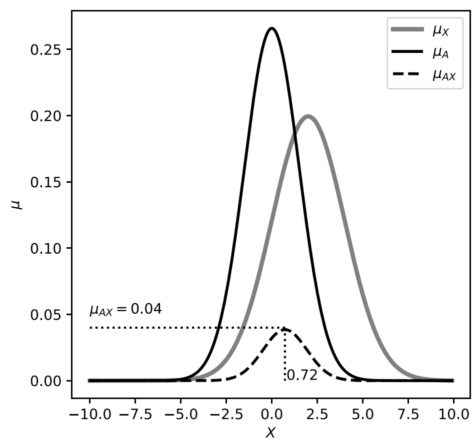

Fig. 3. Product (as a *t-norm* operation) μ_{AX} of input FS μ_X and the antecedent fuzzy set μ_A as per Eq. (3).

2.1. Type-1 fuzzy inference systems

A TSK-type FIS is governed by the "IF–THEN" rule of the form (Takagi and Sugeno, 1985):

$$\mathbf{r}^i : \text{IF } X_1^i \text{ is } A_1^i \text{ and } \cdots \text{ and } X_{p^i}^i \text{ is } A_{p^i}^i \text{ THEN } Y^i \text{ is } B^i, \tag{5}$$

where $\mathbf{r}^i$ is the ith rule in the FIS's RB. The ith rule has A^i as the T1FS, and B^i as a function of inputs $X_1^i, X_2^i, \ldots, X_{p^i}^i$ that returns a crisp output Y^i. At the ith rule, $p^i \le P$ inputs are selected from P inputs. Note that p^i varies from rule-to-rule, and thus, the input dimension at a rule i is denoted as p^i. That is, the subset of inputs to a rule has $p^i \le P$ elements, which leads to a *incomplete rule* because all available inputs may not be present to rule premises (antecedent part). Otherwise, a *complete rule* has all available inputs at its premises. The function B^i, for TSK-type, is commonly expressed as:

$$B^i = c_0^i + \sum_{j=1}^{p^i} c_j^i X_j^i, \tag{6}$$

where X_j^i are the inputs and c_j^i for $j = 0$ to p^i are the *tuneable parameters* at the consequent part of a rule. For Mamdani-type, B^i may be expressed as a "class." The basic building blocks of a FIS is shown in Fig. 1 whose defuzzified crisp output is computed as follows. At first, the inference engine fires the RB's rules, each rule has a firing strength F^i as:

$$F^i = \prod_{j=1}^{p^i} \mu_{A_j^i}(X_j^i), \tag{7}$$

where $\mu_{A_j^i} \in [0,1]$ is the membership value of jth T1FS MF (e.g., Fig. 4(a)) at the ith rule. Assuming firing strength F^i has to be computed for a non-singleton input $\mu_{X_j^i}(X_j^i)$, then firing strength F^i will replace $\mu_{A_j^i}(X_j^i)$ in Eq. (7) by $\mu_{AX_j^i}(X_j^i)$ as per Eq. (3). A detail generalize definition of firing strength computation for non-singleton inputs is given by Mouzouris and Mendel (1997).

The defuzzified output $\hat{Y}$ of T1FIS, as an example, is computed as:

$$\hat{Y} = \frac{\sum_{i=1}^{M} B^i F^i}{\sum_{i=1}^{M} F^i}, \tag{8}$$

where M is the total rules in the RB.

2.2. Type-2 fuzzy inference systems

A T2FS $\tilde{A}$ is characterized by a 3D MF (Mendel, 2013): The x-axis is the primary variable, the y-axis is secondary variable (primary MF denoted by u), and the z-axis is the MF value (secondary MF denoted by μ). Hence, for a singleton input X, a T2FS $\tilde{A}$ is defined as:

$$\tilde{A} = \left\{ \left((X,u), \mu_{\tilde{A}}(X,u) \right) \mid \forall X \in \mathbf{X}, \forall u \in [0,1] \right\}. \tag{9}$$

The MF value μ has a 2D support, called "footprint of uncertainty" of $\tilde{A}$, which is bounded a lower membership function (LMF) $\underline{\mu}_{\tilde{A}}(X)$ and an upper membership function (UMF) $\bar{\mu}_{\tilde{A}}(X)$. A T2FS bounded by an LMF and a UMF is an *interval type-2 fuzzy set* (IT2FS), e.g., a Gaussian function [Eq. (10)] with uncertain mean $m \in [m_1, m_2]$ and std. $\sigma \ge 0$ is an IT2FS (e.g., Fig. 4(b)):

$$\mu_{\tilde{A}}(X, m, \sigma) = \exp\left(-\frac{1}{2} \left(\frac{X - m}{\sigma} \right)^2 \right), \quad m \in [m_1, m_2]. \tag{10}$$

An LMF [Eq. (11)] $\underline{\mu}_{\tilde{A}}(X) \in [0,1]$ and a UMF [Eq. (12)] $\bar{\mu}_{\tilde{A}}(X) \in [0,1]$ of an IT2FS can be defined as (Karnik et al., 1999):

$$\underline{\mu}_{\tilde{A}}(X) = \begin{cases} \mu_{\tilde{A}}(X, m_2, \sigma), & X \le (m_1 + m_2)/2 \\ \mu_{\tilde{A}}(X, m_1, \sigma), & X > (m_1 + m_2)/2 \end{cases} \tag{11}$$

$$\bar{\mu}_{\tilde{A}}(X) = \begin{cases} \mu_{\tilde{A}}(X, m_1, \sigma), & X < m_1 \\ 1, & m_1 \le x \le m_2 \\ \mu_{\tilde{A}}(X, m_2, \sigma), & X > m_2 \end{cases} \tag{12}$$

In Fig. 4(b), a point v along the x-axis of 3D-IT2FS cuts the UMF and LMF along the y-axis, and the value μ of the type-2 MF is taken along the z-axis [dotted line, which a MF in the third dimension in Fig. 4(b) between $\bar{\mu}_{\tilde{A}}(X = v)$ and $\underline{\mu}_{\tilde{A}}(X = v)$]. Considering the IT2FS MF, the ith IF–THEN rule of TSK-type T2FIS, for inputs $\mathbf{X} = (X_1, X_2, \ldots, X_{p^i})$, takes the form:

$$\mathbf{r}^i : \text{IF } X_1^i \text{ is } \tilde{A}_1^i \text{ and } \cdots \text{ and } X_{p^i}^i \text{ is } \tilde{A}_{p^i}^i \text{ THEN } Y^i \text{ is } \tilde{B}^i, \tag{13}$$

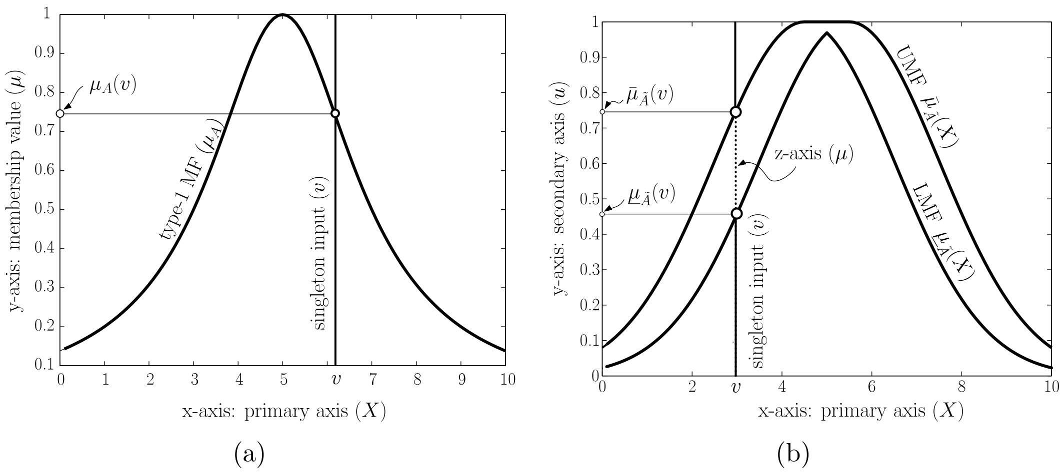

Fig. 4. Fuzzy MF examples: (a) Type-1 MF $\mu_A(X) = 1/[1 + ((X - m)/\sigma)^2]$ with center $m = 5.0$ and width $\sigma = 2.0$. (b)Type-2 MF with fixed $\sigma = 2.0$ and with means $m_1 = 4.5$ and $m_2 = 5.5$. UMF $\bar{\mu}_{\tilde{A}}(X)$ as per Eq. (12) is in solid line and LMF $\underline{\mu}_{\tilde{A}}(X)$ as per Eq. (11) is in dotted line.

where $\tilde{A}^i$ is a T2FS, $\tilde{B}^i$ is a function of $\mathbf{X}$ that returns a pair $[\underline{b}^i, \bar{b}^i]$ called left and right weights of the consequent part of a rule. In TSK, $\tilde{B}^i$ is usually written as:

$$\tilde{B}^i = [c_0^i - s_0^i, c_0^i + s_0^i] + \sum_{j=1}^{p^i} [c_j^i - s_j^i, c_j^i + s_j^i] X_j^i, \tag{14}$$

where X_j^i are the inputs and c_j^i for $j = 0$ to p^i are the rule's consequent part's tunable parameters and s_j^i for $j = 0$ to p^i are its deviation factors. The firing strength $F^i = [\underline{f}^i, \bar{f}^i]$ of IT2FS is computed as:

$$\underline{f}^i = \prod_{j}^{p^i} \underline{\mu}_{\tilde{A}_j^i} \text{ and } \bar{f}^i = \prod_{j}^{p^i} \bar{\mu}_{\tilde{A}_j^i}. \tag{15}$$

At this stage, the inference engine fires the rule and the *type-reducer*, e.g., center of set Y_{cos} as per Eq. (16) reduces the T2FS to T1FS (Karnik et al., 1999; Wu and Mendel, 2009):

$$Y_{cos} = [Y_l, Y_r] \tag{16}$$

where Y_l and Y_r are left and right ends of the interval. For the ascending order of $\underline{b}^i$ and $\bar{b}^i$, y_l and y_r are computed as:

$$Y_l = \frac{\sum_{i=1}^{L} \bar{f}^i \underline{b}^i + \sum_{i=L+1}^{M} \underline{f}^i \underline{b}^i}{\sum_{i=1}^{L} \bar{f}^i + \sum_{i=L+1}^{M} \underline{f}^i} \text{ and } Y_r = \frac{\sum_{i=1}^{R} \underline{f}^i \bar{b}^i + \sum_{i=R+1}^{M} \bar{f}^i \bar{b}^i}{\sum_{i=1}^{R} \underline{f}^i + \sum_{i=R+1}^{M} \bar{f}^i}, \tag{17}$$

where L and R are the switch points determined as per $\underline{b}^L \le Y_l \le \underline{b}^{L+1}$ and $\bar{b}^R \le Y_r \le \bar{b}^{R+1}$, respectively. Subsequently, defuzzified crisp output $\hat{Y} = (Y_l + Y_r)/2$ is computed.

For a *non-singleton interval type-2 FIS*, lower and upper intervals of non-singleton inputs are created. Additionally, similar to the non-singleton input fuzzification μ_{AX} in the case of non-singleton type-1 FIS using input FS μ_X and antecedent FS μ_A shown in Eq. (3), for non-singleton type-2 FIS, both lower $\underline{\mu}_{\tilde{A}X}$ and upper $\bar{\mu}_{\tilde{A}X}$ intervals products are calculated using lower and upper input FSs $\underline{\mu}_X$ and $\bar{\mu}_X$ and lower and upper antecedent FSs $\underline{\mu}_{\tilde{A}}$ and $\bar{\mu}_{\tilde{A}}$. Sahab and Hagras (2011) describe the computation of non-singleton type-2 FIS in detail.

2.3. Heuristic designs of fuzzy systems

The FIS types: Type-1 (Section 2.1) and Type-2 (Section 2.2) follow a similar design procedure and differ only in the type of FSs being used. The heuristic design of FIS can be viewed from its hybridization with neural networks (NN), evolutionary algorithms (EA), and meta-heuristics (MH) (Fig. 5). And, such a confluence offers the following systems (Herrera, 2008):

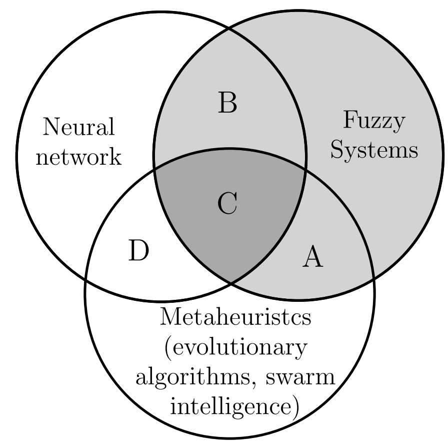

Fig. 5. Spectrum of fuzzy inference system paradigms.

(1) genetic fuzzy systems (A);
(2) neuro-fuzzy systems (B);
(3) hybrid neuro-genetic fuzzy systems (C); and
(4) heuristics design of NNs (D).

This paper discusses the areas A, B, and C of Fig. 5, and the discussion on area D of Fig. 5 is offered by Ojha et al. (2017).

The heuristic design installs learning capabilities into FIS which come from the optimization of its components. The FIS optimization/learning in a supervised environment is common practice. Typically, in **supervised learning**, a FIS is trained/optimized by supplying training data $(\mathbf{X}, \mathbf{Y})$ of N input–output pairs, i.e., $\mathbf{X} = (X_1, X_2, \ldots, X_P)$ and $\mathbf{Y} = (Y_1, Y_2, \ldots, X_Q)$. Each input variable $X_j = \langle x_{j1}, x_{j2}, \ldots, x_{jN} \rangle^T$ is an N-dimensional vector, and it has a corresponding N-dimensional desired output vector $Y_j = \langle y_{j1}, y_{j2}, \ldots, y_{jN} \rangle^T$. For the training data $(\mathbf{X}, \mathbf{Y})$, a FIS model $f(\mathbf{X}, R)$ produces output $\hat{\mathbf{Y}} = (\hat{Y}_1, \hat{Y}_2, \ldots, \hat{Y}_Q)$, where $f : \mathbf{X} \times \mathbf{Y} \to \hat{\mathbf{Y}}$, $R = \{\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_M\}$ is a set of fuzzy M rules, and $\hat{Y}_j = \langle \hat{y}_{j1}, \hat{y}_{j2}, \ldots, \hat{y}_{jN} \rangle^T$ is an N-dimensional model's output, which is compared with the desired output Y_j, $\forall j = 1, 2, \ldots, Q$ and $\forall k = 1, 2, \ldots, N$, by using some error/distance/cost function c_f over model $f(\mathbf{X}, R)$.

The cost function c_f can be a *mean squared error* function or can be an *accuracy* measure, depending on the desired outputs being continuous (regression) or discrete (classification) (Caruana and

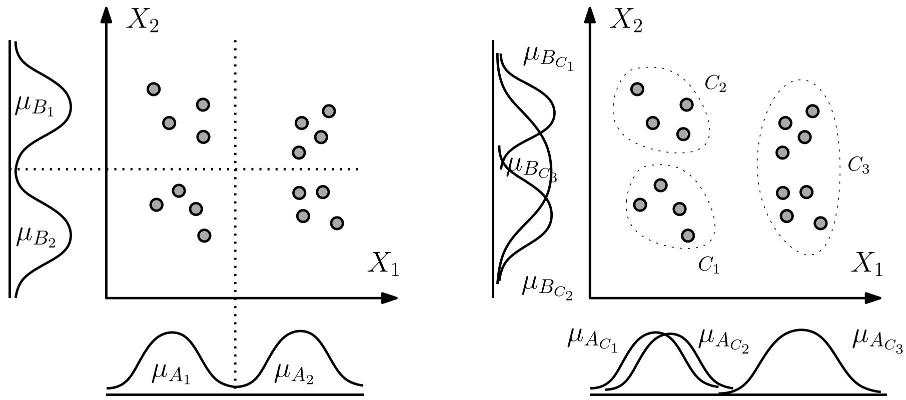

Fig. 6. Input-space partitioning: grid partitioning (left) and clustering partitioning (right). Two-dimensional input-space (inputs X_1 and X_2) is partitioned by assigning MF μ_{A_j} to input X_1 and MF μ_{B_j} to input X_2. In the case of grid partitioning (left), $j = 1, 2$; and in the case of clustering based partitioning (right), $j = C_1, C_2, C_3$.

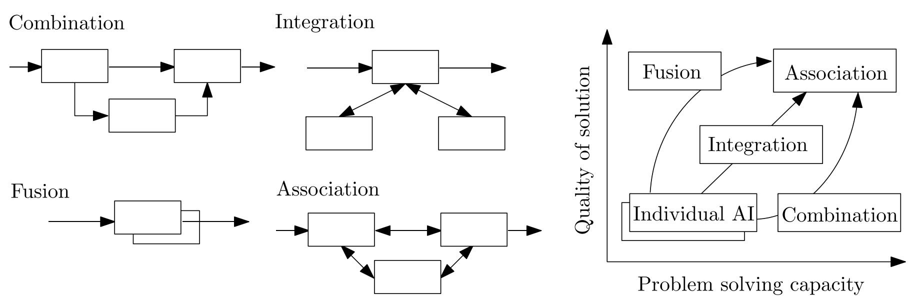

Fig. 7. Synergetic artificial intelligence (Funabashi et al., 1995).

Niculescu-Mizil, 2004). FIS's learning is therefore rely on reducing a cost function c_f by employing strategies for designing and optimizing a FIS model $f(\mathbf{X}, R)$, where model design may be refereed to how the FIS's components interact with each other and optimization may be referred to: RB design, RB parameter learning, and rule selection. In summary, FIS design, optimization, learning, and modeling is viewed as:

(1) the selection of FSs via fuzzy partitioning of input-space;
(2) the design of FIS's rules via an arrangement of rule and inputs;
(3) the optimization of the rule's parameters; and
(4) the inference from the designed FIS.

Often a Gaussian function, a triangular function, or a trapezoidal function are selected as the MF of an FS (Zadeh, 1999). The **input-space partition** corresponding to the MF assignments is one of the most crucial aspects of FIS design. For example, a two-dimension input-space in Fig. 6 having inputs X_1 and X_2 are partitioned using a grid-partitioning method (Jin, 2000; Jang, 1993) or a clustering-based partitioning method (Juang and Lin, 1998; Kasabov and Song, 2002).

Fig. 6 is an example of inputs space partitioning for numerical variables. An example of partitioning for linguistic terms is explained by Cord et al. (2001). Mao et al. (2005) presented an example of input-space partitioning using a binary tree, where the root of the tree takes whole input $\mathbf{X}$ and partition it into two children nodes $X_l \in \mathbf{X}$ and $X_r \in \mathbf{X}$. The partitioned subsets $\{X_l, X_r\} \subset \mathbf{X}$ are assessed for a defined cost function c_f. If the cost c_f is lower than a defined threshold ϵ_{err} than the input-space partitioning stops, else continues.

After the input-space partition, FIS is designed via an arrangement of rules and optimization of rule's parameters for inference from FIS. As per Fig. 5, FIS design can be performed by combining the FIS concept with GA and NN. Such synergy between two or more methods improves the system's approximation capabilities (Funabashi et al., 1995). In this respect, let us revisit four different **synergetic models** (Fig. 7) which indicate four ways of hybridizing artificial intelligence (AI) techniques.

The fuzzy system modeling combined with EA, MH, and NN falls in within the synergetic model: (1) *combination*, when the produced rules are optimized by an EA algorithm or an MH algorithm, and (2) *fusion*, when EA or an NN are used to design FIS, i.e., to construct RB.

3. Genetic fuzzy systems

EA (Back, 1996) and MH (Talbi, 2009) have been effective in FIS optimization (Cordón et al., 2004; Herrera, 2008; Sahin et al., 2012). EA and MH are applied to design, optimize, and learn the fuzzy rules and this gives the notions of evolutionary/genetic fuzzy systems (GFS) whose implementational requirements are:

(1) defining a population structure;
(2) encoding FIS's elements as the individuals in the population;
(3) defining genetic/meta-heuristic operators; and
(4) defining fitness functions relevant to the problem.

3.1. Encoding of genetic fuzzy systems

The questions *how to define a population structure* and *how to encode elements of a FIS* as the individuals (called *chromosome*) of the population opens a diverse implementation of GFS. A FIS has the following elements: input–output variables; rule's premises FSs; rule's consequent FSs and rule's parameters; and the rule set. These elements are combined (encoded to create a vector) in a varied manner that offers diversity in answering the mentioned questions.

Let R be an RB, a set of M rules $\mathbf{r}_i \in R$, $\forall i = 1, 2, \dots, M$, then Fig. 8 represent two basic genetic population structures: $\mathcal{S}_a$ and $\mathcal{S}_b$.

A rule $\mathbf{r}_i \in R$ that has p^i FSs, A_i for T1FS and $\tilde{A}_i$ for T2FS, for $i = 1$ to p^i, the ith rule parameter vector $\mathbf{r}_i$ may be encoded as (Herrera et al., 1995; Ishibuchi et al., 1997b; Ojha et al., 2016):

$$\mathbf{r}_i = \begin{cases} \langle A_1^i, A_2^i, \dots, A_{p^i}^i, c_0^i, c_1^i, \dots, c_{p^i}^i \rangle & \text{for T1FS} \\ \langle \tilde{A}_1^i, \tilde{A}_2^i, \dots, \tilde{A}_{p^i}^i, c_0^i, s_0^i, c_1^i, s_1^i, \dots, c_p^i, s_p^i \rangle & \text{for T2FS} \end{cases} \tag{18}$$

$$\mathcal{S}_a = \begin{pmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \vdots \\ \mathbf{r}_M \end{pmatrix} \qquad \mathcal{S}_b = \begin{pmatrix} \mathbf{r}_{11} & \mathbf{r}_{12} & \dots & \mathbf{r}_{1M} \\ \mathbf{r}_{21} & \mathbf{r}_{22} & \dots & \mathbf{r}_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{r}_{K1} & \mathbf{r}_{K2} & \dots & \mathbf{r}_{KM} \end{pmatrix}$$

Fig. 8. Population structures: $\mathcal{S}_a$ and $\mathcal{S}_b$ where M is total rules in a RB and K is the population size in $\mathcal{S}_b$.

		X_2			
		A_1	A_2	$\dots$	A_k
	A_1	B_1	B_2	$\dots$	B_2
X_1	A_2	B_3	B_1	$\dots$	B_3
	$\vdots$	$\vdots$	$\vdots$	$\ddots$	B_3
	A_k	B_2	B_3	$\dots$	B_2

Fig. 9. Fuzzy decision table for rule contraction (e.g., IFX_1 is A_1 AND X_2 is A_2 THEN Y is B_2) and genetic encoding consisting two input variables X_1 and X_2 and an output Y. The decision table has FSs $A_i, i = 1, 2, \dots$ at the premises part of the rule and at the consequent part of the rule $B_j, j = 1, 2, \dots$ indicate output fuzzy set in the case of Mamdani-type rule and linear equation [see Eq. (6) and (14)].

where A^i has two parameters m_i and σ_i represent center and width of T1FS; and $\tilde{A}^i$ has three parameters m_i, λ, and σ_i represent center, deviation factor, and width respectively. The variable c_j^i for $j = 0$ to p^i are the type-1 rule's consequent weights (parameters) and variable c_j^i and s_j^i for $j = 0$ to p^i are the type-2 rule's consequent weights and weights deviations respectively.

For linguistic fuzzy terms, FS A^i will take a single integer $t_i \in \{0, 1, 2, \dots\}$ (e.g., the integers 0, 1, and 2, respectively may indicate a linguistic term "very small", "small", and "large"). For a Mamdani-type rule, Thrift (1991) and Kim et al. (1995) proposed decision matrix [a rule table as per Eq. 9] for fuzzy rules. Such a decision table can be encoded as a genetic vector for the FIS learning (Hadavandi et al., 2010).

Considering genetic fuzzy populations $\mathcal{S}_a$ in Fig. 8, the *Michigan approach* (Booker, 1982) suggests encoding of a rule $\mathbf{r}_i$ parameters as a chromosome, $C_i = \mathbf{r}_i$ in population $\mathcal{S}_a$, i.e., $\mathcal{S}_a = (\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_M)$ of M rules. Hence, optimization fuzzy system is the reduction of cost function $c_f(\mathcal{S}_a)$ over entire population. In Michigan approach, the optimization of population is met through mutation and crossover of rules, discarding and adding new rules into the population (Ishibuchi et al., 1997b).

Second genetic fuzzy population $\mathcal{S}_b$ in Fig. 8 has each chromosome C_i representing a RB:

$$C_i = R_i = \{\mathbf{r}_{i1}, \mathbf{r}_{i2}, \dots, \mathbf{r}_{iM}\}, \tag{19}$$

a set of M rules/chromosomes for $i = 1, 2, \dots, K$. Thus, the population $\mathcal{S}_b = (R_1, R_2, \dots, R_K)$ for $i = 1$ to K enables both "rule optimization" and "rule selection" opportunities. The rule selection using population $\mathcal{S}_b$ is known as the *Pittsburgh approach* (Smith, 1980) that suggests encoding of fuzzy rule set into a single chromosome, a vectored representation of RB. Pittsburgh approach suggest selecting a subset of m rules from a set (sometime randomly generated) of M rules, $m < M$. In the Pittsburgh approach, the optimization of the population is met through mutation and crossover of the RB and by enabling and disabling the rules in an RB. Hence, the optimization of FIS is the reduction of the cost function $c_f(C_i = R_i)$ of the chromosomes within the population $\mathcal{S}_b$ (Ishibuchi et al., 1997b).

Relaying on the population structure $\mathcal{S}_a$ and $\mathcal{S}_b$, numerous literature offers GFS with varied FIS's elements encoding methods: Lee and Takagi (1993) created a *composite chromosome* combining tuple of MF components and rule consequent parameters. Similar composite encoding was performed by Papadakis and Theocharis (2002) for TSK-type rules. Wu and Tan (2006) puts MF's parameter of a type-2 fuzzy rule on a genetic vector. Using the population structure $\mathcal{S}_b$, Ishibuchi et al. (1995) created rules as per Eq. (19), where each rule $\mathbf{r}_{ij}, i = 1$ to $K, j = 1$ to M takes one of three status: 1 if $\mathbf{r}_{ij} \in R_i$, -1 if $\mathbf{r}_{ij} \notin R_i$, and 0 if $\mathbf{r}_{ij}$ was created as a dummy rule.

Hoffmann and Pfister (1997) presented a concept of *messy encoding* by assigning an integer value to FIS's elements while encoding them as a chromosome. For example, the rule IFX_1 is A_2 AND X_2 is A_3 THEN Y_3 is A_1 were encoded as per $\langle 1\ 2\ 2\ 3\ 3\ 1\rangle$ where input variables were $X_1 \rightarrow 1$, $X_2 \rightarrow 2$, $X \rightarrow 3$ and FSs were $A_1 \rightarrow 1$, $A_2 \rightarrow 2$, $A_3 \rightarrow 3$. Hoffmann and Pfister (1997) argued that such an encoding is benefited from GA since the sequence is messed up by GA operations, and thus creates a diverse rule. Melin et al. (2012) amid for obtaining the best rule by assigning a status to TIFIS $\rightarrow 0$ and TIFIS $\rightarrow 1$, Mamdani-type rule $\rightarrow 0$, and TSK-type rule $\rightarrow 1$ apart from assigning an integer value to a FS.

3.2. Training of genetic fuzzy systems

GFS training depended on FIS's encoding, and the GFS training should answer the questions:

(1) Which EA/MH algorithms to be used?
(2) Whether only a few elements of FIS training is sufficient?
(3) How should EA/MH operators be defined for the encoded GFS?

The answer to the first question relies on how an individual chromosome was encoded, as well as; it is a matter of choice from the range of optimization algorithms (Back, 1996; Talbi, 2009). The answer to second questions was investigative by Carse et al. (1996) with four GFS learning schemes: (1) learning MF parameters for fix rules; (2) learning rules by keeping MF parameters fix; (3) learning both MF parameters and rules in stages (one after another); and (4) learning both MF parameters and rules simultaneously. Carse et al. (1996) concluded that learning in both MF and rule is necessary for solving a complex system, and GFS benefits from the cooperation of rules. However, it was left for an empirical evaluation to determine the best performance of stage-wise or simultaneous learning. The answer to the third question is subjective to population definition (Fig. 8) and encoding mechanisms (Section 3.1) since a chromosome (solution vector) can be coded in three ways: a *binary-valued* vector, an *integer-valued* vector, and a *real-valued* vector. Accordingly, an EA/MH optimization as per Algorithm 1 is employed, and the algorithm's operators are chosen and designed.

The binary-values vector and the integer-valued vector optimization is both a *combinatorial* and a *continuous* optimization problem, both of which follow the general procedure as per Algorithm 1. It is a combinatorial optimization when the binary vector and integer vector encoding domain is discrete. That is, the encoding (assignment) of each FIS's element takes either 0 or 1 (Ishibuchi et al., 1995), or takes an integer number (Hoffmann and Pfister, 1997; Tsang et al., 2007), and FIS's fitness depends on finding the best combination of FIS's elements. Hence, a global search algorithm like genetic algorithm (GA) (Goldberg and Holland, 1988), discrete particle swarm optimization (PSO) (Kennedy and Eberhart, 1997), or discrete Ant algorithms (Dorigo et al., 1999) can be employed to optimize binary vector and integer-valued vector. The FIS optimization is a continuous optimization problem when the domain is continuous and FIS optimization is finding the best performing real-valued vector representing the rules parameters (Herrera et al., 1995). Hence, GA (Wright, 1991), PSO (Kennedy, 2011), ACO (Socha and Dorigo, 2008), or a search algorithm (Yang, 2010) can be used for the real-valued vector optimization as per Algorithm 1.

The optimization in a binary or an integer vector invites *crossover operator* like single-point crossover, two-point crossover, and composite crossover; and the *mutation operator* like bit flip, random bit

Algorithm 1 General optimization procedure.

procedure Optimize ($\mathcal{S}$ for $\mathcal{S}_a$ or $\mathcal{S}_b$)
  For $n := 0$, set population $\mathcal{S}^* := \mathcal{S}^n$;
  $\mathcal{S}^n \in \mathbb{R}^{K\times L}$ or $\mathcal{S}^n \in \mathbb{N}^{K\times L}$ or $\mathcal{S}^n \in \mathbb{Z}_2^{K\times L}$
  $K \rightarrow$ individuals (chromosomes) and $L \rightarrow$ parameters (genes)
  $c_f^n :=$ Evaluate ($\mathcal{S}^n$)
  repeat
    $\mathcal{S}^{n+1} :=$ Operator ($\mathcal{S}^n$)
    $c_f^{n+1} :=$ Evaluate ($\mathcal{S}^{n+1}$)
    if ($c_f^{n+1} < c_f^n$) **then** $\mathcal{S}^* := \mathcal{S}^{n+1}$
    end if
    $n := n+1$
  until cost $c_f^n \le c_{f_{min}}$ or iteration $n \ge n_{max}$
  return $\mathcal{S}^*$
end procedure
procedure Evaluate ($\mathcal{S}$)
  if $\mathcal{S}$ is $\mathcal{S}_a$ **then**
    compute cost c_f over $\mathcal{S}$
  else
    compute cost c_f over C_i, i.e., for each chromosome C_i in $\mathcal{S}$
  end if
  return c_f
end procedure
procedure Operator ($\mathcal{S}$)
  if EA **then**
    Apply *Selection*, *Crossover*, *Mutation*, and *Elitism* on $\mathcal{S}$
  else for MH
    Apply MH Operator(s) on $\mathcal{S}$
  end if
  return $\mathcal{S}$
end procedure

resetting, (Goldberg and Holland, 1988). Whereas, real vector invites crossover operators like uniform crossover, arithmetic crossover (Goldberg, 1991; Eshelman and Schaffer, 1993). Ishibuchi et al. (1999) exploited both approaches Pittsburgh and Michigan simultaneously, where for the Pittsburgh approaches they designed mutation operator as the Michigan approach for rule generation.

Typically, as an example, for a one-point crossover and for two selected chromosomes C_{p1} and C_{p2} (also called parents), two new chromosomes C_{o1} and C_{o2} (also called offspring) are produced by swapping elements of the parent chromosomes (a chromosome is vector few elements) as follows:

$$\begin{aligned} C_{p1} &= \{r_{11}, r_{12}, \overset{point}{\downarrow} r_{13}, r_{14}\} \quad \text{parent 1} \\ C_{p2} &= \{r_{21}, r_{22}, \overset{point}{\downarrow} r_{23}, r_{24}\} \quad \text{parent 2} \\ C_{o1} &= \{r_{11}, r_{12}, \mathbf{r_{23}}, \mathbf{r_{24}}\} \quad \text{offspring 1} \\ C_{o2} &= \{r_{21}, r_{22}, \mathbf{r_{13}}, \mathbf{r_{14}}\} \quad \text{offspring 2} \end{aligned} \tag{20}$$

Similarly, as an example, for a one-point mutation, one a chromosome C_{p1} is selected and a new chromosome C_{o1} is produced by replacing an element $\mathbf{r}_{1j}$ of the chromosome C_{p1} by a new element $\mathbf{r_{new}}$ or a random element (e.g., flipping 0 to 1 in binary chromosome, replacing a integer by another integer, and replacing a real-value by another random real-value) as follows:

$$\begin{aligned} C_{p1} &= \{r_{11}, \overset{point}{\downarrow}{r_{12}}, r_{13}, r_{14}\} \quad \text{parent 1} \\ C_{o1} &= \{r_{11}, \mathbf{r_{new}}, r_{13}, r_{14}\} \quad \text{offspring 1} \end{aligned} \tag{21}$$

The real-valued vector encoding of FIS's elements allows a varied FSs to lie on the same genetic vector. Hence, it is necessary to ensure that each gene (dimension) corresponding to a FIS's element takes a value within a defined interval. For example, in Eq. (18), the variables m_1^i and σ_j^i are MF'S parameter, and they need a defined interval like $m_j^i \in [m_{left}, m_{right}]$ and $\sigma_j^i \in [\sigma_{left}, \sigma_{right}]$ to control the MF's shape. Cordón and Herrera (1997) defined *interval of performance* for assuring a boundary for each dimension in the vector.

Martinez-Soto et al. (2010) employed PSO for finding optimal MF parameter of an encoded GFS. Shahzad et al. (2009) combined PSO and GA in a hybrid approach where PSO and GA start with similar populations of rules and swap the best solution iteratively among PSO and GA populations to make communication between both optimizers. Martínez-Soto et al. (2015) extended (Shahzad et al., 2009) hybrid PSO and GA approach to optimize T2FIS, and Valdez et al. (2011) proposed a hybrid approach of PSO-based FIS and GA-based FIS where depending upon their errors, the two rule types were activated and deactivated during the FIS optimization. An *empirical evaluation* of bio-inspired algorithms summarized by Castillo et al. (2012) suggests that ACO outperformed PSO and GA as GFS optimization. Examples of MH-based GFS implementations are chemical optimization (Melin et al., 2013), harmony search (Pandiarajan and Babulal, 2016), artificial bee colony optimization (Habbi et al., 2015), bacteria foraging optimization (Verma and Parihar, 2017).

3.3. Other forms of genetic fuzzy systems

Similar to Michigan approach, also in *iterative rule learning* scheme (Venturini, 1993; Muñoz and Herrera, 1997; Ahn et al., 2007) and *cooperative–competitive rule learning* (Greene and Smith, 1993; Whitehead and Choate, 1996), each rule of an RB are encoded into separate genotypes, and the population of such genotype leads to the formation of RB iteratively. Iterative learning scheme starts with an empty set and adds rules one-by-one to the set by finding an optimum rule from a genetic selection process. For this purpose, the genetic operators such as mutation and crossover are applied over one or two rule(s) to make offspring rule(s), and the quality of the generated rule(s) is(are) evaluated using a predefined rule quality measure. Therefore, iteratively selecting rules according to rule quality measure criteria for forms an optimum RB in an iterative manner (Venturini, 1993).

The cooperative–competitive rule learning is also an RB learning method that determines an optimum RB from competition and cooperation of rules from a genetic/meta-heuristic population. GFS is also implemented as the reinforcement learning system. Juang et al. (2000) proposed symbiotic evolutionary learning of fuzzy reinforcement learning system which uses a cooperative coevolutionary GA for the evolution of fuzzy rules from a population of rules. A reinforcement T2FIS optimization was performed by ACO in Juang and Hsu (2009). Aiming cooperation among FIS's components, Delgado et al. (2004) split the genetic population into four separate populations: RB, individual rules, FSs, and FISs. They proposed a coevolutionary GFS relying on a hierarchical collaborative approach where each population, cooperatively shared application domain fitness as well as the population's individuals.

A *fuzzy tree system*, e.g., TSK rule in Mao et al. (2005), Chien et al. (2002), allows the rules to be implemented as a *binary tree* and an *expression tree* and the rules tree structures to be optimized by genetic programming (GP) (Koza and Rice, 1994). Hoffmann and Nelles (2001) implemented TSK rule as a local linear incremental model tree, where the algorithm incrementally built the tree while partitioning the input-space using a binary tree formation. On the other hand, the *expression tree* approach for fuzzy rule implementation and optimization using rules tree population was performed in Sánchez et al. (2001), Cordón et al. (2002). Their approach also included a mapping of rule-tree parameters (leaf node) onto a vector for its optimization using simulated annealing (Aarts and Korst, 1988).

4. Neuro-fuzzy systems

Since the early 90s (Jang, 1991, 1993; Buckley and Hayashi, 1994; Andrews et al., 1995; Karaboga and Kaya, 2018), neuro-fuzzy systems (NFS) that represent a fusion of both FIS and NN has been forefront among FIS's research dimensions, especially attributed to its data-driven learning ability which does not require prior knowledge of the problem. However, NN needs sufficient training pattern to learn, and a trained NN model does not explain how to interpret its computational behavior, i.e., NN's computational behavior is a "black box," which does not explain how the output was obtained for the input data. On the other hand, FIS requires prior knowledge of the problem and do not have learning ability, but it tells how to interpret its computational behavior, i.e., it explains how the output was obtained for the input data.

The shortcomings of both NN and FIS can be eliminated by combining them while making an NFS (Feuring et al., 1999; Ishibuchi and Nii, 2001). Usually, for the rule extraction from NFS, two types of combinations are practiced (Andrews et al., 1995): *cooperative NFS* and *hybrid NFS*. The cooperative NFS is the simplest approach closer to combination and association synergetic AI (Fig. 7). In cooperative NFS, NN and FIS work independently, and NN determines FIS's parameters from the training data (Sahin et al., 2012). Subsequently, FIS performs the required interpretation of the data. Hybrid NFS is closer to **fusion synergetic** AI (Fig. 7), in which, both NN and FIS are fused to create a model. Working in synergy improve the learning ability of NFS since both NN and FIS are independently capable of approximate to any degree of accuracy (Buckley et al., 1999; Li and Chen, 2000).

NFS are trained in two fundamental manners: supervised learning (See Section 2.3) and reinforcement learning (Lin and Lee, 1994; Moriarty and Mikkulainen, 1996). This paper scope includes supervised learning extensively; whereas, the reinforcement learning for NFS is available in Berenji and Khedkar (1992) through the implementation of generalized approximate reasoning based intelligence-control and in Nauck and Kruse (1993) through model named NEFCON.

4.1. Notions of neuro-fuzzy systems

Self-adaptive/Self-organizing/Self-constructing system. In NFS's context, the *adaptive systems* or the *self-adaptive systems* may refer to the automatic tuning and adjustment of MF's parameters (Jang, 1993; Wang and Lee, 2002). Whereas, a system is non-adaptive if human expert determines the MFs and their parameters. Similarly, *self-organizing systems* (Juang and Lin, 1998; Wang and Rong, 1999) and *self-constructing systems* (Lin et al., 2001) refer to the creation of fuzzy rules and the adaptation of MF's parameters without the intervention of human experts. The implementation of a self-organizing NFS and a self-constructing NFS holds the key to formation appropriate RB (Juang and Lin, 1998; Lin et al., 2001).

There are two leaning aspects of self-adaptive NFS: *structural learning* and *parameter learning* (Lin, 1995). An NFS, therefore, will be self-adaptive if it performs either of these two learning aspects or both of them during learning. In addition to the learning without human intervention, adaptive systems like self-adaptive systems and self-construction systems when strictly refer to online training and incremental learning for every piece of new training data, then the system may be referred to as an evolving fuzzy system (EFS) (see Section 6).

Online learning system/Dynamic learning system. Online learning refers to *sample-by-sample* learning. A learning system is an *online learning system* that adapts its structure and parameters each time it sees a training sample rather than seeing the entire training samples set (batch) at once (Jang, 1993). Similarly, a *dynamic learning system* and a *dynamically changing system* adapts its structure and parameters on receiving new training sample (Wu and Er, 2000; Wu et al., 2001). In a sense, systems that grow their structures by adding MFs nodes and rule nodes are also referred to as the *dynamically growing systems* and the *dynamic evolving systems* (Kasabov and Song, 2002; Kasabov, 2001b). FIS's research dimension EFS encompass online and dynamic learning systems (see Section 6).

Another viewpoint refers to dynamic learning systems as the recurrent fuzzy systems. In other words, the systems which accommodate temporal dependency and whose next (one step ahead) adaptation (learning) is a function of the model's previous output (Jang, 1992; Juang and Lin, 1999). In FIS research, these jargons are used with diverging context.

4.2. Layers of neuro-fuzzy systems

An NFS architecture typically is composed of a maximum of seven layers as shown in Fig. 10 whose layers that can be customized in various forms for both type-1 and type-2 FISs. The type-1 and type-2 FISs only differ in the type of FSs they used. Hence, the variations in type-1 and type-2 NFS architecture depends on the FS type used at the MF layer L_M and the methods used at nodes to performs the computation for type-1 and type-2 FSs. Moreover, the type-reduction that requires for type-2 FIS can be implemented at one of the layer available in the consequent part.

The implementation of NFS architecture categorized into two types of layers: the layers implementing the *antecedent* part and the layers implementing the *consequent* part of a rule. The number of layers in the design of NFS may vary depending upon how the antecedent and consequent part were implemented. Regardless of a layer mention in Fig. 10 explicitly appear or not in an NFS architecture, the functionality of that layer is accommodated in the either of adjacent layers to that layer. Let us discuss the functionality of the typical NFS layers:

Input layer *(L_I)*. A node at the input layer holds $X \in \mathbf{X}$, and primarily has a function $f(X) = X$, i.e., the raw input is fed to the next layer without any manipulation. To the best of our literature knowledge, most models agree to the transfer of inputs to the next layer without any modification. Hence, $a_i^{(1)} = f(X_i); 1 \leq i \leq P$ represents the output a node i of the input layer, where P is the dimension of the input-space. However, models agree to either fuzzify inputs at the membership function layer (L_M) or fuzzify inputs by employing a fuzzy weight to the link connecting input layer (L_I) directly to rule layer (L_R).

The connections/links between L_I and L_M is therefore, not fully connected. Rather, each input is connected to its partitioned FSs. Or in the absence of layer L_M, connection between L_I and L_R are not fully connected. Such partially connections between L_I and L_M or between L_I and L_R play an important role in inducing a diverse rules set.

Membership function layer *(L_M)*. A node at the MF layer L_M, also called *fuzzifier* layer, holds μ, and primarily has a function $f(X) = \mu(a^{(1)}) = \mu(X)$, i.e., a MF $\mu(.)$ is applied on input X. MFs are often problem specific. An MF can be a Gaussian function, a triangular function, or a trapezoidal function. MF layer L_M often refereed as the fuzzification layer that performs fuzzification of the inputs. MF layer is also responsible for the partitioning of the input-space (Fig. 6). The mapping of inputs to MF layer also helps to overcome the *curse of dimensionality* (Brown et al., 1995).

Additionally, whether an MF layer L_M is a separate layer or it acts as a fuzzy weight between the layers L_I and L_R, the MF layer's operation remains the same. The input to an MF layer is $a_i^{(1)} = X_i$ that has been partitioned into p^i FSs with $a_{ij}^{(2)} = \mu_{ij}(a_j^{(1)}); 1 \leq j \leq P$ and $1 \leq j \leq p^i$. Traditionally, inputs partition p^i is kept fixed. However, automatically determining the input-space partition by using clustering based method gives flexibility to NFS's structural adaptation, and such an act is often refereed as structural learning. It also reflects the notions of the self-constructing system (Lin et al., 2001). Examples of clustering for input-space partition are: K-nearest neighbor (Wang and Rong, 1999); mapping constrained agglomerative (Wang and Lee, 2002); evolving clustering (Kasabov, 2001a); and evolving self-organizing map (Deng and Kasabov, 2003).

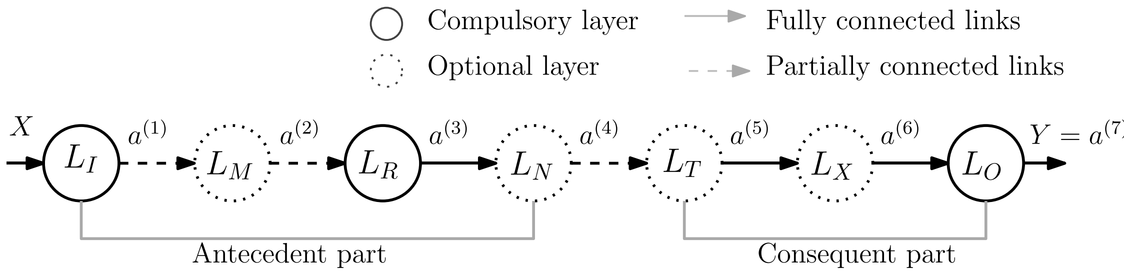

Fig. 10. Neuro-fuzzy system architecture (NFS) with a maximum of seven layers. An NFS receives a vector X as an input and subsequently propagated it through various layers producing outputs $a^{(1)}$ to $a^{(7)}$. The symbols L_I, L_M, L_R, L_N, L_T, L_X, and L_O stand for the NFS layers input, membership function, rule, normalization, term, extra (additional), and output respectively.

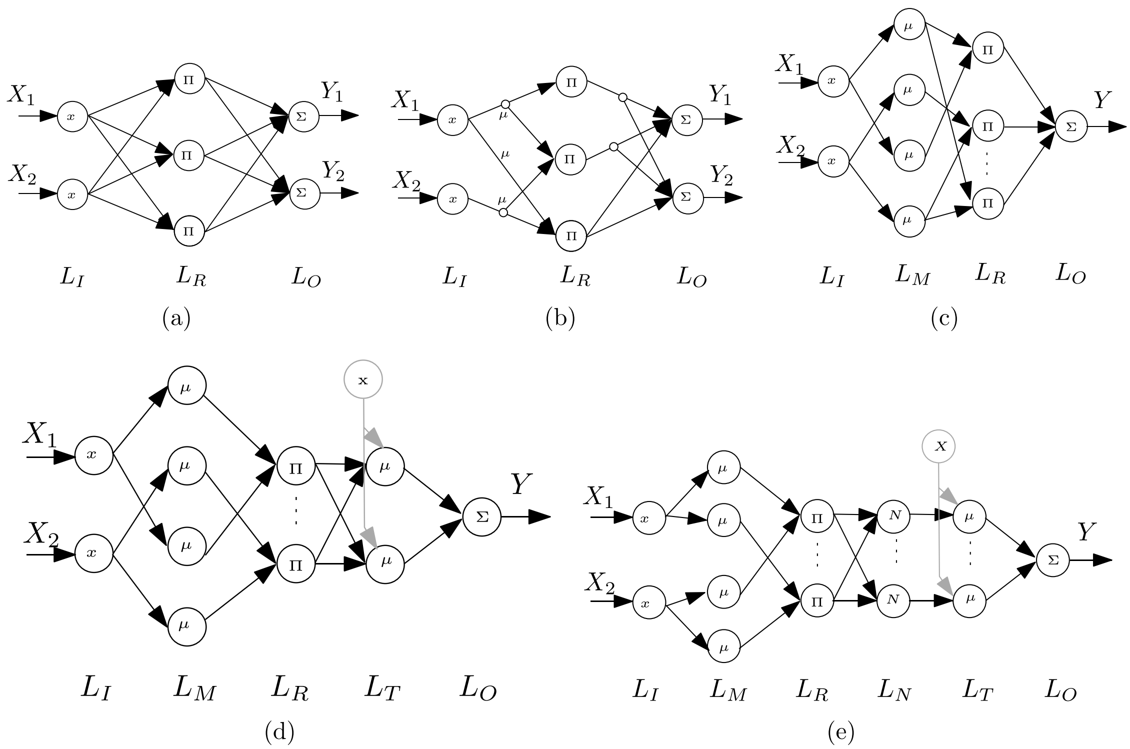

Fig. 11. Feedforward NFS architectures.

Rule layer *(L_R).* A node at the rule layer holds a function $\prod(.)$, and primarily performs $a_j^{(3)} = \prod_{j=1}^{p^i}(a_j^{(2)}) = \prod_{j=1}^{d}\mu_j(X)$, i.e., a rule layer node typically computes *T-norm* of the previous layer's inputs $a_j^{(2)}$. Thus, a node at rule layer represents the antecedent (premises) part of a rule that takes d inputs $a_j^{(2)}; 1 \le j \le d$, where $d \le p^i$ is FS fed to a rule node.

The inputs d to a rule node may or may not be equal to the total number of partitions p^i of an input $a_i^{(1)}$. It also indicates that connections between layer L_M and layer L_R, which are often partly connected, govern the diversity of the rules being formed. It also gives flexibility for a structural adaptation (structural learning) in the fuzzy system being realized. For example, an algorithm may starts with no rule, and it only recruits a rule node if it is necessary during its online leaning (Tung et al., 2011; Juang and Lin, 1998). The output of a rule layer node or in other words antecedent part of a rule is denoted as $a_k^{(3)} = \prod_{j=1}^{d} a_{kj}^{(2)}; 1 \le k \le M$. Hence, the output of M rules (M nodes at the rule layer) can be denoted as $a_k^{(3)}$.

Normalization layer *(L_N).* Normalization layer L_N computes the firing strength of the rules, which is $a_i^{(4)} = a_i^{(3)} / \sum_{k=1}^{M} a_k^{(3)}; i = 1, 2, \dots, M$. Therefore, the number of nodes at the layer L_N is thus equal to the number of nodes at the layer L_R and the connection between L_R and L_N is fully connected.

Term/consequent layer *(L_T).* The nodes as term layer L_T computes consequent part of a rule. Thus, the number of nodes at term layer L_T are the same as the number of nodes at the layer L_R and layer L_N. Each node at this later has a function $\varphi(.)$ and the definition of $\varphi(.)$ depends on the FIS's type implemented, e.g., Mamdani or TSK. In other words, what type of function implemented at the nodes of layer L_T (Horikawa et al., 1992). Assuming that nodes at the layer L_T are constant, then the output $a_k^{(5)} = a_k^{(4)} c_k$, where c is a constant. Another type of consequent/term implementation of TSK (first-order linear equation) node, where $a_k^{(5)} = a_k^{(4)} \left(\sum_{i=1}^{n} x_i c_{ki} + c_{k0}\right)$.

Additional layer *(L_X).* Additional layer L_x is infrequent in NFS architecture design, which performs specific operation $\psi(a^{(5)})$ producing the output $a^{(6)}$. The definition of $\psi(.)$ in Park et al. (2002) is a polynomial neural network. Whether the additional layer L_X is present ($a^{(6)} = \psi(a^{(5)})$) or absent ($a^{(6)} = a^{(5)}$), the input to the output layer L_O is $a^{(6)}$.

Output layer *(L_O).* For a single output problem, output layer L_O holds a single node that usually is the summation of incoming inputs to the node, i.e., $a^{(7)} = \sum_{k=1}^{R} a_k^{(6)}$. Therefore, the output node act as a *defuzzifier*. Hence, the operation at the output layer with a function $\theta(.)$ applied on $a^{(6)}$ is to obtain NFS's output $Y = a^{(7)} = \theta(a^{(6)})$.

4.3. Architectures of neuro-fuzzy systems

4.3.1. Feedforward designs

Feedforward NFS architecture have forward connections from one layer to another and have at least three layers: input, rule, and output. Therefore, the simplest NFS architecture is IRO, i.e., **I**nput, **R**ule, and **O**utput layer architecture.

IRO architecture. Masuoka et al. (1990) represented IRO NFS architecture as a combination of the input-variable-membership net, the rule-net, and the output-variable-membership net. Moreover, the fuzzy rules are directly translated into NNs where the nodes at layer L_I realize rule's antecedent MFs, the node at layer L_R represent fuzzy operation (e.g., AND), and the nodes at layer L_O realize the rule's consequent part. This type of representation can easily be translated back and forth between fuzzy rules and NNs. However, the expert intervention will be required in the NFS construction.

Buckley and Yoichi (1995) showed a design of three-layered IRO NFS architecture and implemented IRO NFS for *discrete fuzzy systems* and *non-discrete fuzzy systems* (Fig. 11(a)). Being a three-layered architecture, their discrete IRO NFS architecture implemented fuzzy rules as the links between layers L_I and L_R, and the layer L_O processed incoming signals from the transfer functions (nodes at layer L_R) using some aggregation function $\theta(.)$. The rules in the discrete IRO NFS is, therefore, can run in parallel. However, for a large input, the rules can grow to huge unmanageable size for a low discrete factor (Buckley and Yoichi, 1995). On the other hand, in a non-discrete IRO fuzzy system, the hidden layer L_R nodes represent the rules and the links between L_I and L_R are set to 1. The nodes at the output layer L_O represent an aggregation of the signals from L_R.

In Buckley and Yoichi (1995) IRO NFS, the fuzzy rules are implemented as a whole either for the links between L_I and L_R or for the nodes at L_R. Whereas, Nauck and Kruse (1997) proposed a three-layered IRO NFS architecture with the link between layers L_I and L_R and between layers L_R and L_O representing MFs also called fuzzy weights. In other words, the links between L_I and L_R fuzzify the inputs before feeding them to nodes at L_R and defuzzify them before feeding them to nodes at L_O.

IRO NFS architecture shown in Fig. 11(b) was proposed for specific problems like classification and approximation bearing abbreviations NEFCLASS (Nauck and Kruse, 1997) and NEFPROX (Nauck and Kruse, 1999) respectively. NFSs are shown in Fig. 11(b) implement the links as the fuzzy weights that improve the NFS *interpretability* since it avoids more than one MFs to be assigned to similar terms (Nauck and Kruse, 1997).

IMRO architecture. NFS design IMRO: input, membership, rule, and output architecture (Fig. 11(c)) directly computes the output $a^{(6)}$ of FIS by assigning weight to the links between layer L_R and L_O (Lin et al., 2001; Wu et al., 2001). The IMRO NFS architecture by Wang and Rong (1999) is a four-layered configuration, where layers L_I and L_M fuzzify the inputs. The layer L_R consists of two nodes: $a_1^{(6)}$ and $a_2^{(6)}$. The first node $a_1^{(6)}$ computes a weighted sum $a_1^{(6)} = a_1^{(3)} = \sum_{i=1}^{m} w_i a^{(2)}; 1 \leq i \leq m$ of the incoming inputs from L_M, where m is the number of nodes at layer L_M, and w_i is the links' weights between L_M from L_R. The weight w_i represent consequent part's FS's center. The second node $a_2^{(6)}$ computes sum $a_2^{(6)} = a_2^{(3)} = \sum_{i=1}^{m} a^{(2)}$ of incoming inputs from L_M, where the link's weight between layers L_M and L_R are set to 1. The output layer L_O node, therefore, realizes $a^{(7)} = a_1^{(6)}/a_2^{(6)}$.

IMRNO/IMRTO architecture. The five-layer NFS architecture (Fig. 11(d)) adds a layer L_N or L_T between the layers L_R and L_O to perform fuzzy quantification via rule normalization or via a fuzzy term nodes (Kasabov et al., 1997; Kim and Kasabov, 1999). Example of an IMRNO NFS architecture with a normalization layer L_N between L_R and L_O is in Kasabov et al. (1997). Whereas, an IMRTO NFS architectures with a term layer L_T is the common practice. The nodes at the layer L_T compute fuzzy outputs, and the links between L_R and L_T represent firing strength (*confidence factor*) of the rules at L_R (Kasabov et al., 1997; Kim and Kasabov, 1999; Kasabov, 2001b; Kasabov and Song, 2002).

Contrary to IMRNO and IMRTO architectures, the five-layered NFS presented by Leng et al. (2006) is an **IRNTO** architecture that has layers L_I, L_R, L_N, L_T, and L_O. In IRNTO model, nodes at layer L_R combine both MF layer L_M and rule layer L_R, and the term layer L_T between L_N and L_O perform a TSK-type consequent operation for the rule.

In general, five-layer NFS architecture implements L_I, L_M, and L_R as its rule's antecedent, where nodes at L_R implements rule's $\prod(.)$ or AND function. The layer L_T and L_O implements the rule's consequent part and perform defuzzification. However, apart from $\prod(.)$ and defuzzyification at layers L_R and L_T example of $\min(.)$ operator at L_R and $\max(.)$ operator at L_T is available in Shann and Fu (1995).

IMRNTO architecture. IMRNTO NFS architecture is the most popular NFS architecture, which is attributed to the efficiency and explicit presence of FIS's components in the architecture (Jang, 1991; Horikawa et al., 1992). ANFIS being the most popular implementation of IMRNTO NFS (Jang, 1993). IMRNTO NFS are six-layered architecture with layers L_I, L_M, L_R, L_N, L_T and L_O. The functioning of the nodes are described in Section 4.2.

IMRNTXO architecture. Beyond IMRNTO NFS architecture, IMRNTXO NFS architecture includes an additional layer that performs certain computation receiving inputs from layer L_T and fed the computed output $a^{(6)}$ to the node(s) at layer L_O. The model: *modified fuzzy polynomial neural network* (Park et al., 2002) is in an example of such seven-layered architecture, where a polynomial NN that implements a polynomial function (like bilinear and biquadratic), which resembles consequent part of TSK type.

Of general NFS architecture in Fig. 10, five variation in NFS architectures formation is shown in Fig. 11 are IRO (three layers), IMRO (four layers), IMRNO/IMRTO/IRNTO (five layers), IMRNTO (six layers), and IMRNTXO (seven layers). The choice of a particular variation in NFS formation has its advantages and disadvantages. For example, IRO architecture limits itself to three layers, and that restricts it to compute entire FIS operations on a few nodes. IRO architecture computes input fuzzification at input layer node that limits it to mix with multiple FSs and when input fuzzification takes place at the links between input and rule layer an input mix with all available FSs for a fully connected network, that limits a proper fuzzy partitioning. However, IRO architectures are easy to implement and they can be translated to fuzzy rules easier than more complex architecture.

The four-layer IMRO architecture solves the fuzzy partition issues that may appear in the layer IRO architecture since it adds a membership layer between input and rule layer. In IMRO architecture, the weight optimization of between the input and membership layer may lead to direct optimization of the FS shapes in addition to a comparatively more variation in rule design (Fig. 11(c)) than IRO architecture.

The five-layer and six-layer architectures IMRNO/IMRTO/IRNTO and IMRNTO add FIS components more explicitly than the three-layer and four-layers architectures. Thus, they offer more efficient ways to design of NFS as a FIS system. In five-layer architecture forth layer is chosen as a normalization layer or term layer, whereas the six-layer architecture uses both normalization and term layers to its architecture. Moreover, seven layer architecture IMRNTXO adds an extra layer for a special purpose such as a polynomial network operation as an extra layer.

The difference among the various architectures is apparent regards to the increasingly explicit presence of the FIS components into the architectures with a higher number of layers than the architectures with a lower number of layers. The explicit presence also offers efficiency and opportunity to optimization NFS architecture to individual FIS component.

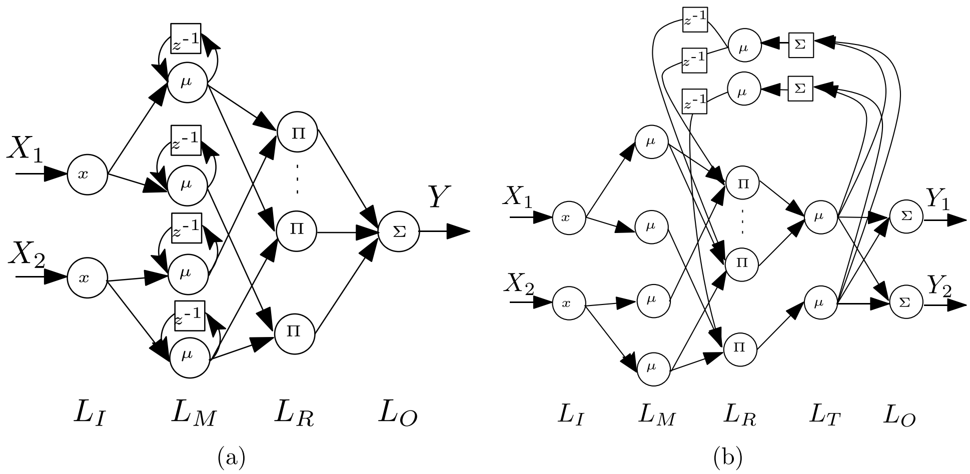

Fig. 12. Feedback NFS architectures.

4.3.2. Feedback/Recurrent designs

Unlike feedforward architecture that models static system and can adapt to a dynamic system through a prepared training set and incremental learning methods, the feedback/recurrent design accommodates dynamic system directly into its structure (model's learning) either via an external feedback mechanism or via an internal mechanism (Mastorocostas and Theocharis, 2002). The recurrent/feedback NFS (RNFS) helps in the implementation of the systems that require its output Y_t at time step t is to be fed as the input Y_{t+1} to the network at next step $t+1$. The external feedback RNFS is the most straightforward implementation of RNFS architecture where rules receive network output directly as its input in the next time step. Whereas, the internal feedback NFS design fits when a system require memory elements to be implemented as an FIS component to define the temporal relation of a dynamic system. That is, in the next step, RNFS's particular layer (e.g., membership, rule, or term layer) receives input y_{t+1} (the output y_t of the previous time step). The example of both recurrent NFS (RNFS) categories are as follows:

External feedback RNFS. Let denote external RNFS architecture design by $\overleftarrow{\mathbf{I}\ldots\mathbf{R}\ldots\mathbf{O}}$, which indicates that the NFS architecture may remain the same as a basic feedforward NFS, but the system incorporates the feedback through one or multiple sources. Such a feedback adaptation can be incorporated through the learning algorithm like temporal backpropagation, e.g., recurrence in ANFIS (Jang, 1992).

Internal feedback RNFS. Internal feedback NFS design $\mathbf{I}\overleftarrow{\mathbf{M}}\mathbf{RO}$ (Lee and Teng, 2000) takes inputs to its MF node as per $a^{(1)}(k+1) = a^{(1)}(k) + a^{(2)}(k-1)$. That is, the recurrence occurs at the MF nodes which enabled the membership layer node to operate as a memory unit that extends the NFS ability for the temporal problems (Fig. 12(a)). Unlike $\mathbf{I}\overleftarrow{\mathbf{M}}\mathbf{RO}$ design, the memory element in the design $\mathbf{IM}\overleftarrow{\mathbf{R}}\mathbf{TO}$ are added at rule layer, and the nodes are called *context element* (Fig. 12(b)) that accommodates both spatial firing from MF nodes and feedback (temporal) firing from term nodes (Juang and Lin, 1999). $\mathbf{IMR}\overleftarrow{\mathbf{T}}\mathbf{O}$ is the third type of internal feedback design implements, where term nodes act as the memory element (Mastorocostas and Theocharis, 2002).

4.3.3. Graph and network based architectures

Apart from the two class of architecture, a general graphical model for information flow was proposed as the *Fuzzy Peri nets* (Looney, 1988). Fuzzy Peri net is directed graph with nodes (neurons) and transition bars (links) that are enabled or disabled when neurons fire. The NFS feedforward and feedback architecture, therefore, can be thought of as the special case of graphical representation. Additionally, examples of FISs combined with *adaptive resonance theory* (ART) to create fuzzy ART architecture is available in Carpenter et al. (1991). Similarly, FISs were also fused with the *min–max network* to create a fuzzy min–max network architecture (Simpson, 1992) and fused with *radial basis function* (RBF) network to created fuzzy RBF architecture (Cho and Wang, 1996).

5. Hierarchical fuzzy systems

GFS is a process of empowering FISs for automatic optimization and learning, which focuses on designing FIS's components. NFS is NN inspired and it enables the arrangement of FIS's components into a network-like structure. Whereas, the hierarchical fuzzy systems (HFS) is a hierarchical arrangement of two or more small standard FISs, (say fuzzy logic units — FLU denoted as N_i in Fig. 13) into a hierarchical structure. Hence, HFS invites the following questions:

(1) What are the basic advantages of arranging small FLUs?
(2) What are the possible ways to arrange FLUs?

5.1. Properties of hierarchical fuzzy systems

Let us take Fig. 9 as an example of a standard practice for rule set formation in FISs design. Now assume the rule table in Fig. 9 has $P = 2$ inputs, and each input takes $k = 3$ FSs. Hence, the number of rules will be $k^P = 3^2$, which means that the number of rules grow exponential at the rate of k^P, and subsequently, the number of parameters to be optimized grow exponentially. This phenomenon is known as the *rule explosion* and the *curse of dimensionality*. The rule explosion reduces the basic FIS's property: *interpretation*, i.e., the reasoning as to how the output was obtained for the inputs become unknown. It also led to infeasible *computation* in both space (rule storage space) and time (Torra, 2002). Additional, in both GFS and NFS, the input-space partitioning play a crucial role in the FIS's construction and both GFS and NFS have to employ an external method like clustering to reduce the input space dimensionality. Hoffmann and Nelles (2001) illustrated a GP-based binary-tree like input-space partition that hierarchically partition inputs space, but they form a standalone FIS.

Raju et al. (1991) initiated the design of hierarchical FIS (HFS) that was composed of low-dimensional fuzzy subsystems, called fuzzy logic unit (FLU). One of the arguments for HFS was to overcome the *curse of dimensionality* (Brown et al., 1995) and stop the *rule explosion* by combining several sub-fuzzy systems receiving only a few inputs from the whole set of inputs (Fig. 13) This allows the reduction of fuzzy rules, total system's parameters, and the computation time. Also, the hierarchical design of fuzzy subsystem found to have a universal approximation ability (Wang, 1999; Zeng and Keane, 2005; Wang, 1998). Moreover, HFS offers intelligent control over the system for a dynamically changing domain environment (Karr, 2000). Such a

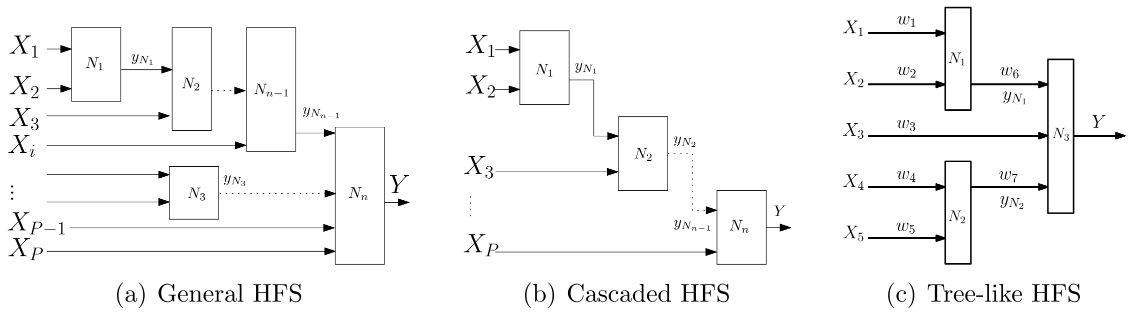

Fig. 13. Typical hierarchical fuzzy systems (HFS): (a) A general combination of low-dimensional fuzzy systems called fuzzy logic units (FLUs) N_i in multiple stages and (b) an incremental combination of low-dimensional FLUs (Chung and Duan, 2000). The inputs X_1, X_2, …, X_P in (a) and (b) are traditionally selected by applying an expert's knowledge (Raju et al., 1991). (c) Tree-like HFS (also called aggregated output HFS) with three FLUs N_1, N_2, and N_3 takes inputs X_1, X_2, X_3, X_4, and X_5 (Ojha et al., 2018).

control may be implemented by allowing one of the FLU in HFS to act as performance checker and optimize entire HFS with its feedback.

Torra (2002) reviewed HFS that presents the following observations for the defining HFS architecture: If some functions are not decomposable then HFS design may not be possible, but for some functions, HFS is proved to be a universal approximator (Wang, 1998). If the system's non-linearities are independent, then separate FLUs can be constructed. If no preference is given to the order (importance) of variables, then a general HFS is trivial to design, else preferred variables should go at beginning stages of hierarchy. If MF for a variable is sharp, then more MFs should be defined for that variable (Wang, 1999). Finally, the interpretability of HFS might become unknown while reasoning (defuzzification) are repeated at multiple stages (Maeda, 1996).

Wang et al. (2006) summarized literature work to investigate the reasoning transparency for the intermediate variable generated by defuzzification at the FLUs at different stages, and concluded that a little work had been done to understand intermediate variables fully. However, in this view, the HFS's interpretability can be improved, provided sufficient monotonicity of FLUs concerning the inputs (Magdalena, 2018). Kouikoglou and Phillis (2009) concluded that under certain conditions (Won et al., 2002), the single-stage HFS's output is monotonic. Hence, it is sufficient for the monotonicity of multi-stage HFS design.

5.2. Implementations of hierarchical fuzzy systems

The classification of HFS types is intuitive since the HFS design is a modular arrangement. Thus, HFS have variety in design and modeling (Lee et al., 2003). A general HFS design is any combination of FLUs in stages (Fig. 13). Special cases of general arrangement can be a cascaded (incremental) design of FLUs (Chung and Duan, 2000) and chain wise FLUs arrangement (Domingo and Sierra, 1997).

Standard HFS. Standard FISs can be transformed to HFS. Joo (2003) transformed standard FIS which has k^P rules for P inputs and k FS. And the FIS was 3-D matrix (cube) with each 2-D slice (rule table as per Fig. 9). Each 2D slice was then transformed to a FLU. Similarly, a type-2 HFS having multiple levels of FLUs implementation was proposed by Hagras (2004) for automatic mobile robot navigation behaviors control. It divided layers (stages) for managing navigation behaviors and as the navigation behaviors in multiple levels. First level accommodated low level behaviors and highest level act as the coordination. An HFS with combining layer wise rule in a hierarchical manner was presented by Fernández et al. (2009) where rules were arranged in two layers and for each layer, KBs were generated by linguistics rule generation method and the rules were selected by GA.

NNs inspired HFS. Joo and Lee (2005) presented a feedforward NN like HFS design that takes the previous layer FLUs input to the THEN part of the rule in current layers. Yu et al. (2007) implemented a hierarchical fuzzy NNs approach and trained hierarchical fuzzy NNs using the backpropagation-like algorithm. Unlike HFS by Joo and Lee (2005) where FLUs are proposed to arranged in a network-like structure, in HFS by Yu et al. (2007), each FLU is an NFS. Mohammadzadeh and Ghaemi (2016) proposed self-structuring hierarchical type-2 NFS (SHT2FNN). Similar to Yu et al. (2007) approach, in SHT2FNN, each FLU is a self-structuring NFS and that the arrangement of FLUs was a cascaded design.

Automatic HFS formation. A majority HFS takes a manual design; whereas, Chen et al. (2007) explained the structural optimization of the HFS where hierarchical arrangements of low-dimensional TSK-type FISs were optimized using probabilistic incremental program evolution (Salustowicz and Schmidhuber, 1997). Ojha et al. (2018) proposed a hierarchical fuzzy inference tree approach (HFIT^M) that has an automatic arrangement of FLUs using GP for type-1 and type-2 TSK FISs. HFIT^M offered automatic selection of the input variables for each FLUs and that the order of input variables are automatically determined along with the HFS structure's automatic determination.

6. Evolving fuzzy systems

Standard FIS, GFS, NFS, and HFS are the concepts of creating a system for modeling and learning from data. Often data are dynamic, i.e., domain environment changes in time. Therefore, any systems relied on the data needs to be updated. Hence, GFS, NFS, and HFS system that embraces and adapt itself to the dynamic nature of data is evolving fuzzy systems (EFS). EFS systems embed provisions for dynamic (online) training of systems for streaming (real-time) data (Angelov, 2009; Kasabov, 1998). In EFS, a system incrementally evolves fuzzy rules incoming new data (Lughofer, 2011). Hence, EFS answers the following questions:

(1) How a fuzzy system adapt to the incoming data stream?
(2) Which components of a fuzzy system are made flexible to evolve?

6.1. Incremental learning of fuzzy systems

Incoming data stream are processed to train and test a system incrementally, i.e., *incremental learning*, *dynamic learning* and *online learning* (Losing et al., 2018). It is a strategy for data-driven training and testing of a system incrementally for unseen data without re-training the system entirely from scratch. Incremental learning, therefore, should take care of *noise* and *concept drift* in the data stream (Schlimmer and Granger, 1986). Noise and concept drift in data may be introduced over time compared to initial training data, i.e., input feature does not describe the output class (Gama et al., 2014). Often for the non-stationary environment, the relation between input features and output class change over time (Elwell and Polikar, 2011). Fuzzy rules are capable of evolving to accommodate noise and concept drift for the data stream (Baruah and Angelov, 2011).

EFS approaches manages the concept drift by first detecting the drift (data shift) and then reacting to the drift. Lughofer and Angelov (2011) describes detection of drift includes tracking of fuzzy rules *age* and evolving the rules' antecedent part using evolving-clustering (Angelov and Filev, 2004a; Angelov, 2010) and evolving-vector quantization (Lughofer, 2008a; Lughofer et al., 2007) and evolving the rules' consequent part. Moreover, the gradual concept drift that is hard to detect can be managed by incremental rule splitting (Lughofer et al., 2018).

6.2. Implementations of evolving fuzzy systems

A fuzzy system, irrespective of its being a standard FIS, a GFS, an NFS, or an HFS when implements incremental learning capability should evolve (alternately we may say modify or refine) itself internally for it is incrementally fed unseen data stream (Angelov and Filev, 2004a). There are two broad categories have been investigated to incorporate EFS concepts in FISs: standard FISs to EFS (Angelov, 2009) and NFS to EFS (Kasabov, 2001a):

FISs → EFS. In standard FISs, incremental learning is offered by adding or removing rules in an RB (vertical direction manipulation), or by adding or removing antecedent part of the rules in an RB (horizontal direction manipulation) as shown in Fig. 14. In Fig. 14, each rule may acquire p^i variables using the evolving clustering methods, i.e., the number of variables are determined automatically; whereas, in traditional clustering methods, the number of clusters has to be predetermined. Moreover, the antecedent part of the rules may expand and contract based on incoming data. Similarly, the number of rules may also be reduced or increased by adding or deleting rules from the RB. Hence, the total rules M^t in the RB are time dependent (Angelov and Filev, 2004a).

Evolving Takagi–Sugeno fuzzy system (eTS) (Angelov and Filev, 2004a; Angelov, 2010) is an example of EFS that modify and update its RB on arrival of every piece of the new data point. It employs online clustering that checks the influence of new data point on the input space partitioning and then it modifies the cluster centers and add new rules in RB. Accordingly, it modifies the rule's consequent parameter. Similarly, flexible fuzzy inference systems (FLEXFIS) (Lughofer, 2008b) rely on the incremental update of cluster centers for the arrivals of new data and accordingly adapt its antecedent and consequent parameters.

The principle of applying incremental clustering to verify new data point and its influence leads to several EFS designs like evolving participatory Kernel recursive least squares model (Lima et al., 2010) that uses participatory learning (Lima et al., 2006). Both eTS and participatory learning concepts were used for evolving a rule having multivariate Gaussian functions at its antecedent part that preserve information between input variable interactions (Lemos et al., 2011). Zhou and Angelov (2007) offers an *evolving self-organizing map* for clustering that replaced the online clustering method in eTS for constructing an evolving EFS classifier.

Lughofer (2013) investigated interpretability aspects such as distinguishability, simplicity, consistency, coverage and completeness, feature importance levels, rule importance levels and interpretation of consequent. They concluded that a very few EFS approach takes care of complexity reduction such as the elimination of redundancies (Lughofer et al., 2011) to improve interpretability conclusion.

NFS → EFS. EFS concept applies to NFS paradigms. NFS is evolved dynamically based on every incoming new data (Kasabov, 2001a; Kasabov and Song, 2002). Such systems are also called *self-evolving* NFS or *adaptive* NFS (Angelov and Filev, 2004b). In self-evolving NFS, the network design has two main parts: antecedent and consequent. For every incoming data, the antecedent part learns new information by using unsupervised means of learning through cluster evolving method, and accordingly, the consequent part weights are updated to accommodate the new information contained in the incoming data.

Incremental learning in NFS is a similar concept as incremental learning in NNs where growing and pruning network architecture may refer to rule addition and deletion in NFS architecture, and learning of weights at the output layer may refer to learning NFS architecture consequent layer parameters (Wang and Kuh, 1992; Feng et al., 2009). As of topological level refinement, evolving NFS architecture may refer to augmented topological concepts (Stanley and Miikkulainen, 2002). Evolving NFS and evolving standard FISs has similar incremental mechanism when it comes to input space partitioning. Both have a major dependency on evolving clustering method (ECM) of inputs space (Kasabov and Song, 2002).

Dynamic evolving neuro-fuzzy inference systems, DENFIS (Kasabov, 2001a; Kasabov and Song, 2002; Kasabov, 2001a) relay on ECM and refine its rule layer (L_R) in its IMRNO/IMRTO architecture (see 4.3.1) by operations like: creating new rule nodes, deleting existing rule nodes, updating existing rules, aggregating two or more rule nodes. Like DENFIS, self-organizing fuzzy neural network, SONFIN (Juang and Lin, 1998) employ clustering methods for input space partitioning and methods of parameter optimization of rules consequent parts. However, it starts with no rule in its structure by examining center of first incoming input data and the first rule and subsequently perform a check on every a new piece of data for the aggregated firing strength of existing rules in the structure an if the aggregated firing strength is found week (i.e., lower than a pre-defined threshold) new rule are added to the structure. Structure adaptation on inputs space clustering are: sequential adaptive FISs (Tung et al., 2011), generalized dynamic NFS (Wu et al., 2001), self-evolving interval type-2 NFS (Juang and Tsao, 2008), self-organizing NFS (Wang and Rong, 1999), recurrent self-evolving NFS with local feedbacks (Juang et al., 2010), and mutually recurrent interval type-2 NFS (Lin et al., 2013).

In summary, the EFS needs the following steps:

Step 1: Construct an initial fuzzy system in *batch mode* or construct EFS in *online mode* from scratch with no rule in RB initial, and add rules as per step 2.
Step 2: Apply *incremental clustering* or an *incremental inputs-space partitioning* mechanism to check on incoming data.
Step 3: Evolve (add, delete, modify) existing EFS rules or rule structure as per step 2.
Step 4: Continue step 2 and step 3 for every new piece of data.

7. Multiobjective fuzzy systems

Multiobjective fuzzy system (MFS) enable a fuzzy system to manage multiple objectives associated with the system, and that system may have been modeled using any of these concepts: standard FIS, GFS, NFS, HFS or EFS. That is, an MFS empowers a FIS to manage multiple objectives which may come from two directions: one, from the problem domain, and two, from the system's own trade-off. This review discusses the objectives inherent in FIS itself.

A data-driven modeling system owns a single objective: *cost function*. The cost function can be the approximation error minimization or the classification accuracy maximization. The minimization or maximization of the cost function is subjected system's parameter optimization. For a FISs, the cost function is subjected to rules and rule's parameters optimization. The primary goal of a FIS is to draw reasoning from the system, i.e., FIS should have *interpretability* property.

Additionally, FIS often gains *complexity* when having numerous rules. For NFS complexity can be the nodes interconnections. The complexity reduction and interpretability improvement are often FIS's objectives. An MFS deals with multiple objectives that are conflicting with each other. Hence, MFS answers the following questions:

(1) What are the multiple objectives that are conflicting associated with the system?
(2) Which two or more objectives associated with the system should be optimized?
(3) How to define the selected two or more objectives functions?
(4) How to manage conflicting objectives?

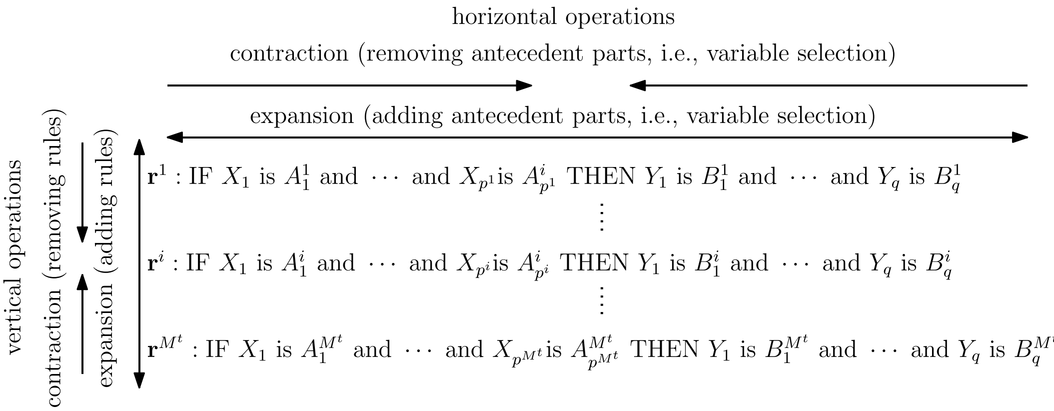

Fig. 14. Evolving fuzzy system: typical dynamic RB learning. The symbols are as follows: p^i and q^i indicates the number of inputs X and outputs Y variable to a rule i, respectively; M^t is the total rules in the RB at time t; A^i is the fuzzy set at the antecedent part of a rule; and B^i is the function of the consequent part of a rule (Angelov and Filev, 2004a).

7.1. Multiobjective trade-offs

Two basic approaches manage the trade-offs of multiple objectives: (1) by aggregating multiple objectives $c_{f_1}, c_{f_2}, \dots, c_{f_m}$ into a single scalar objective function c_f, e.g., sum $c_f = \sum_{i=1}^{m} c_{f_i}$, product $c_f = \prod_{i=1}^{m} c_{f_i}$, or weighted sum $c_f = \sum_{i=1}^{m} c_{f_i} w_i$, etc. (Ishibuchi, 2007); and (2) by optimizing multiple objectives $c_{f_1}, c_{f_2}, \dots, c_{f_m}$ simultaneously (Deb et al., 2002). These two approaches may respectively be called **non-Pareto approach** and **Pareto approach** (Coello et al., 2007). The Pareto-based approach, since optimize function simultaneously, offers a *nondominated* solution where no one objective is dominant than the other, whereas, in non-Pareto approach, one objective may dominate the other. Therefore, a Pareto-based approach is a formidable option to obtain a generalized solution (a FIS) (Zitzler and Thiele, 1999).

Fig. 15 shows two-objectives solution space where solutions lying on Pareto optimal front are feasible solution (Fig. 15(a)), and the solutions within the boundary of may vary in their objective, i.e., a solution (a FIS, R) may be complex but accurate and another solution may be simple but inaccurate (Fig. 15(b)). Hence, no single solution exists that may satisfy both objectives. Therefore, multiobjective optimization takes the form: $\min\{c_{f_1}, c_{f_2}, \dots, c_{f_m}\}$ or $\max\{c_{f_1}, c_{f_2}, \dots, c_{f_m}\}$, i.e., a multiobjective optimization needs to be performed as:
minimize (or maximize)
$\{c_{f_1}(R), c_{f_2}(R), \dots, c_{f_m}(R)\}$
subject to $(R) \in S$,
where $m \geq 2$ is the number of objective functions $c_{f_i} : \mathbb{R}^m \to \mathbb{R}_{\geq 0}$. The vector of objective functions is denoted by $\mathbf{c}_f = \langle c_{f_1}(R), c_{f_2}(R), \dots, c_{f_m}(R)\rangle$. The solution $R = \{\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_M\}$ is a set of M fuzzy rules belonging to the set of solution space S. The set of rules R indicate a solution of GFS, NFS, HFS, or EFS. The word "minimize" or "maximize" indicates the minimization (or maximization) all the objective functions simultaneously.

A nondominated solution is one in which no one objective function can be improved without a simultaneous detriment to at least one of the other objectives of the solution. The nondominated solution is also known as a Pareto-optimal solution.

Definition 1. Pareto-dominance — A solution R_1 is said to dominate a solution R_2 if $\forall i = 1, 2, \dots, m$, $c_{f_i}(R_1) \leq c_{f_i}(R_2)$, and there exists $j \in \{1, 2, \dots, m\}$ such that $c_{f_j}(R_1) < c_{f_j}(R_2)$.

Definition 2. Pareto-optimal — A solution R_1 is called *Pareto-optimal* if there does not exist any other solution that dominates it. Pareto-optimal front is a set of *Pareto-optimal* solutions.

7.2. Implementations of multiobjective fuzzy systems

For FISs together with accuracy $c_{accuracy}$ (performance improvement) of systems the interpretability $c_{interpretability}$ (transparency and reasoning improvement) is always a desirable objective. Additionally, complexity $c_{complexity}$ of system play another important role in improving computational time, as well as, it may play a role in a system's interpretability improvement (Jin, 2000).

Guillaume (2001) summarized the three necessary condition for the interpretability: (1) fuzzy partition must be readable, (2) rule set must be small, (3) rule set must be incomplete. These three necessary conditions were described by Guillaume (2001) to be met by two ways: a good rule induction method and FIS's structure and parameter optimization.

Defining a system's accuracy is subjected to the domain of problem, whereas interpretability definition is a challenging task (Guillaume, 2001; Casillas et al., 2013). Especially when condition (1) set by Guillaume (2001). However, the definition of interpretability and complexity may be straightforward like reduction of rules and parameters in some case, but in some cases, it can be challenging when interpretability and complexity may mean the interaction of rule and interconnections of the node (Ishibuchi, 2007). Hence, based on the definition of a rule vector in Eq. (18), the objectives interpretability (or complexity) can be formulated as:

$$c_{interpretability} = \begin{cases} count_{elements}(\mathbf{r}_i) \\ count_{rules}(\mathcal{S}) \end{cases} \tag{22}$$

where $\mathbf{r}_i$ a rule vector in Eq. (18) and $\mathcal{S}$ is a set (population) of rules.

A question "how the best parameter and best rules are to be selected" arises from the use of $count(\cdot)$ as an interpretability objective function is answered by employing the following methods: (1) variable selection: regularity criteria, geometric criteria, individual discriminant power, and entropy variable index; and (2) rule base optimization: incremental rule generation, rule merging, and statistic based rule evaluation (Guillaume, 2001). Moreover, multiobjective optimization in conjuncture with these methods controls both accuracy and interpretability.

An evolutionary multiobjective algorithm like nondominated sorting GA (NSGA-II) (Deb et al., 2002) or strength Pareto EA (SPEA) (Zitzler and Thiele, 1999) can be applied for optimizing FIS's multiple objectives simultaneously. The **interpretability against accuracy** and **complexity against accuracy** are typical evolutionary multiobjective optimization scenarios (Ishibuchi and Nojima, 2007).

Ishibuchi et al. (1997a) formulated two objectives as the maximization of accuracy and minimization of a number of rules while applying a multiobjective GA for obtaining a set on nondominated solutions. Similarly, Alcalá et al. (2007) considered the number of rule minimization as the interpretability measures and mean squared

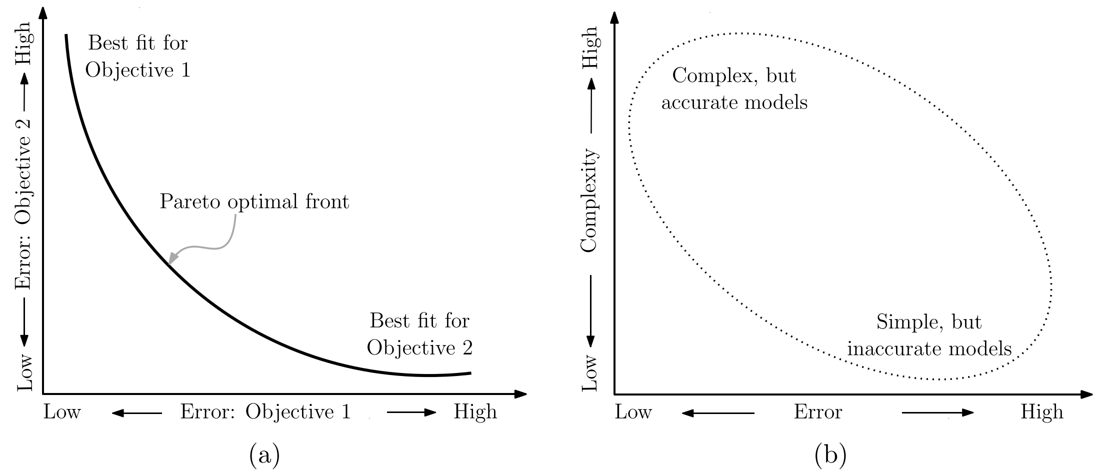

Fig. 15. Multiobjective trade-offs: (a) Solutions on Pareto optimal front in a two-objective solution space. (b) Fuzzy inference system solution space where objective 1 is the error (1-accuracy) of the system and objective 2 is the complexity (interpretability) of the system.

error minimization as the accuracy measure while applying SPEA for optimizing these two objectives simultaneously to conclude that the multiobjective led to the removal of the rules having little importance. Moreover, Pareto-based multiobjective optimization algorithms were used to optimize accuracy–complexity trade-off as a number of rule reduction and the accuracy maximization (Ishibuchi and Nojima, 2006; Gacto et al., 2009; Ducange et al., 2010).

Similarly, in Cordón et al. (2003), Wang et al. (2005), Munoz-Salinas et al. (2008), Alcalá et al. (2009), Antonelli et al. (2011), simultaneous learning of knowledge-base was proposed, which included feature selection, rule complexity minimization together with approximation error minimization, etc. In Antonelli et al. (2012), a co-evolutionary approach that aims towards combining multiobjective approach with single objective approach was presented. In a co-evolutionary approach, first, a multiobjective GA determined a Pareto optimal solution by finding a trade-off between accuracy and rule complexity. Then, a single objective GA was applied to reduce training instances. Fazzolari et al. (2013) summarized research works focused on multiobjective optimization.

Pulkkinen and Koivisto (2010) defined transparency of fuzzy partitions as the interpretability indicator where transparency was described as the MFs number reduction, MF's diversity, MF's normality assurance, and MF's shape symmetry assurance. The transparency–accuracy trade-off was then optimized as multiobjective optimization. Similarly, Rey et al. (2017) took a detailed description of interpretability to put interpretability–accuracy to test. In fact, they took three objectives: maximizing accuracy, maximizing interpretability, and maximizing rule relevance. Rey et al. (2017) measured the accuracy as a means squared error minimization and defined the interpretability in terms of the reduction of (1) number of rules, (2) number of MFs, (3) incoherence among rules (increase rule's coherency), and (4) irreverent rules. A details study on rule coherence and rule relevance are offered by Dubois et al. (1997) and Yen and Wang (1999), respectively.

For NFS, HFS, and the FISs that have structural representation, the structure simplification is one of the objectives which may indeed indicate to a number of rule reduction, parameter reduction, and rule interaction simplification like the number of MFs reduction. Ojha et al. (2018) employed multiobjective genetic programming (MOGA) for the simplification of the model structure while improving accuracy and improving diversity in the rules. These three objectives are conflicting with each other. Therefore, the Pareto optimal set of nondominated solutions offer to chose a solution as desirable in the problem's context.

8. Challenges and opportunities

With the success of FIS and research in FIS's multiple directions like GFS, NFS, HFS, EFS, and MFS comes multiple challenges and multiple opportunities:

Nature of fuzzy systems. The basic FIS's property is its ability to model with an explanation as to how for an input the model achieves its objectives. This FIS's property is referred to as *transparency and interpretability* (Casillas et al., 2013). Although a lot of work has been done to define and preserve transparency and interpretability of a FISs (Guillaume, 2001; Gacto et al., 2011; Cpałka et al., 2014), often with the growing number of fuzzy rules while solving complex problems and with the complex interactions of rules within models (e.g., NFS and HFS), FISs lose their reasoning ability. Hence, preserving transparency and interpretability remains a challenging issue in FIS's modeling (Buckley and Hayashi, 1994; Andrews et al., 1995; Herrera, 2008; Fazzolari et al., 2013).

Additionally, the basic unit of FISs is MF. The designs and the assignments of MFs to inputs variables have to be a careful art since it influences the FIS's reasoning. For the most modeling methods, expert knowledge is required, in both cases: when the input-space partitioning is performed manually, and when the input-space partitioned performed using clustering, number of the clusters has to be predetermined. Therefore, efficient automatic input-space partitioning can play a crucial role in FISs modeling (Jain, 2010). Moreover, research on incorporating FSs like hesitant FSs (Torra, 2010), intuitionistic FSs (Atanassov, 1986) and their type-2 versions (Mendel and John, 2002) to data-driven FIS's modeling present further opportunities.

Nature of data. The quality of the data-driven model relies on the quality of data that is sufficient and balanced (Bellman, 2013). Usually, FISs are good at managing noisy and imprecise data, but most data source offer *unstructured* data (Feldman and Sanger, 2007), and experimental research often produce *heterogeneous* data (Castano and De Antonellis, 2001). Additionally, for pattern recognition problems, the training data supply for generalized extrapolation and modeling are sometimes *insufficient* (Jackson, 1972; Pradlwarter and Schuëller, 2008) and sometimes are *imbalanced* (Chawla et al., 2002; Alshomrani et al., 2015). These issues are dealt with a method like synthetic minority over-sampling technique (SMOTE) (Chawla et al., 2002). Here, rather than generating synthetic samples, training method may be modified for the rule induction sensitive to imbalanced datasets such as the cost-sensitive rule-based system (López et al., 2015) and the metacognitive learning scheme (Das et al., 2015).

Other crucial issues are *high dimensionality* and *abundance*. Some fuzzy systems like HFS offer a solution to curse of dimensionality to some extent. However, the volume of data is a challenge for FISs to maintain its interpretability-accuracy trade-offs (Ishibuchi and Nojima, 2007). In addition to high dimensionality, some training data are *non-stationary* that show drift in concept when data are fed at a regular interval, i.e., data are fed in the stream. The EFS manage using a refinement of the system for evolving FISs over time through incremental learning (Kasabov, 1998; Angelov, 2009), and iterative rule learning algorithm like multi-stage genetic fuzzy system (Muñoz and Herrera, 1997) are potential EFS and may use population-based incremental learning (Baluja, 1994).

Nature of algorithms. The optimization algorithms such as the EAs Cordón et al. (2004), MHs (Castillo and Melin, 2012; Valdez et al., 2014), least squares method (Wang and Mendel, 1992), gradient descent algorithm (Jang, 1993) are exhaustively used for optimization FISs. However, their efficiency is subjective to formulation (encoding) of FISs. A variety of encoding mechanism has been proposed in the past (Section 3) which indicate that FISs being the integration of various decomposable components offer a reach possibility of encoding and their optimization.

Mostly present approaches under MFS treat two objective interpretability (complexity) and accuracy (error) for the simultaneous optimization while applying evolutionary multiobjective. Popular evolutionary multiobjective like NSGA-II (Deb et al., 2002) is good at optimizing two or three objectives, but their performance decreases as the number of objectives increases (Purshouse and Fleming, 2003). Hence, challenges to simultaneously optimize multiple objectives related to FIS such as FIS's interpretability, consistency, coherence, complexity, and accuracy by applying evolutionary multiobjective that deals with multiple objectives like NSGA-III (Deb and Jain, 2014).

Nature of network. The FISs like NFS, HFS, and EFS offer a connectionist model that beers *network structure*. Algorithms built the network structure based on input-space partitioning, and the intuition for input-space partitioning arrive from the problem domain. While doing so, for high dimensional and complex problems, the network structure may grow big enough to lose interpretability. Therefore, multiobjective optimization embedding growing and pruning mechanism may check interpretability–accuracy trade-off. Additionally, connectionist models optimization using EAs needs more attention (Seng et al., 1999).

Rules extracted from connectionist models beers complex interactions, therefore, an algorithm capable of explaining the complex interaction of rules will help to solve complex problems having abundance data and unstructured data without losing the basic FIS's properties. In this view, future research direction **deep fuzzy systems** (DFS) can be defined in two ways:

First straightforward definition, specifically in the context of pattern classification tasks, would be a system whose input data go through a *convolution* process which is coupled with GFS, NFS, HFS, or EFS. This definition is similar (and inspired by) to the deep convolutional neural network (LeCun et al., 2015). Thus it may be termed explicitly as the *deep convolution fuzzy systems*.

Second, a system will be a *deep fuzzy system* if it relies on multiple layers of network architecture as described in Hinton et al. (2012). A multilayer NFS architecture, e.g., by Deng et al. (2017) and a multi-stage HFS architecture, e.g., by Ojha et al. (2018) may fit this definition. The DFS dimension of FIS research is still an open problem to explore and innovate.

9. Conclusions

This paper reviewed five dimensions of fuzzy systems (FIS): genetic fuzzy systems (GFS), neuro-fuzzy systems (NFS), hierarchical fuzzy systems (HFS), evolving fuzzy systems (EFS), and multiobjective fuzzy systems (MFS). The review linked these dimensions since their concepts transcend to another dimension. For example, standard FISs when encoded (formulated) as an optimization problem, entail GFS concept that

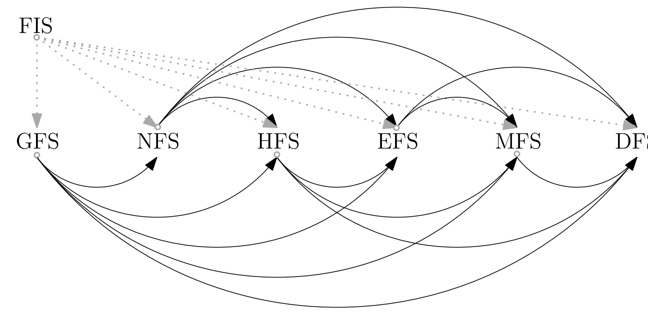

Fig. 16. Fuzzy system complexities and concept entailment.

offers methods and operators to find optimal rule structure. Moreover, FISs can be formulated into other dimensions directly: NFS, HFS, EFS, and MFS. As well as the NFS, HFS, EFS, and MFS when optimized using evolutionary algorithms and metaheuristics are in some sense find links with GFS concept. Similarly, NFS uses the concepts of HFS, EFS, MFS respectively if hierarchical arrangements of NFS are made, evolving NFS are made, and multiobjective optimization of NFS are done.

The GFS, NFS, and HFS also offer deep fuzzy systems (DFS) developments directions. When DFS have both evolving and multiobjective viewpoints, it inherits EFS and MFS concepts. Fig. 16 is a summary of the links between the multiple dimensions of FISs and complexity of concept entailment. The summary in Fig. 16 indicate the challenges and opportunities lie ahead in FISs research: (1) development of efficent rule extraction method as the number of rules grows with the sophistication of the methods; (2) optimal network structure design for rules; (3) creating hybrid optimizations approach like evolutionary algorithm and particle swarm optimization; (4) combining one FIS dimension's concept with another; and (5) development of DFS. Moreover, challenges and opportunities in the treatment of FISs for non-stationary data and multiobjective optimization of interpretability-accuracy trade-off.

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.engappai.2019.08.010.

Acknowledgments

Authors would like to thank the all the anonymous reviewers for the technical comments, which enhanced the contents of the preliminary version of this paper.

References

Aarts, E., Korst, J., 1988. Simulated Annealing and Boltzmann Machines. John Wiley & Sons.
Ahn, H.-S., Chen, Y., Moore, K.L., 2007. Iterative learning control: Brief survey and categorization. IEEE Trans. Syst. Man Cybern. C Appl. Rev. 37 (6), 1099–1121.
Alcalá, R., Ducange, P., Herrera, F., Lazzerini, B., Marcelloni, F., 2009. A multiobjective evolutionary approach to concurrently learn rule and data bases of linguistic fuzzy-rule-based systems. IEEE Trans. Fuzzy Syst. 17 (5), 1106–1122.
Alcalá, R., Gacto, M.J., Herrera, F., Alcalá-Fdez, J., 2007. A multi-objective genetic algorithm for tuning and rule selection to obtain accurate and compact linguistic fuzzy rule-based systems. Int. J. Uncertain. Fuzziness Knowl.-Based Syst. 15 (05), 539–557.
Alshomrani, S., Bawakid, A., Shim, S.-O., Fernández, A., Herrera, F., 2015. A proposal for evolutionary fuzzy systems using feature weighting: dealing with overlapping in imbalanced datasets. Knowl.-Based Syst. 73, 1–17.
Andrews, R., Diederich, J., Tickle, A.B., 1995. Survey and critique of techniques for extracting rules from trained artificial neural networks. Knowl.-Based Syst. 8 (6), 373–389.

Angelov, P., 2009. Evolving fuzzy systems. Encycl. Complexity Syst. Sci. 3242–3255.
Angelov, P., 2010. Evolving Takagi–Sugeno fuzzy systems from streaming data (eTS+). Evol. Intell. Syst. Method. Appl. 21–50.
Angelov, P.P., Filev, D., 2004a. An approach to online identification of Takagi–Sugeno fuzzy models. IEEE Trans. Syst. Man Cybern. B Cybern. 34 (1), 484–498.
Angelov, P.P., Filev, D., 2004b. Flexible models with evolving structure. Int. J. Intell. Syst. 19 (4), 327–340.
Angelov, P.P., Zhou, X., 2008. Evolving fuzzy-rule-based classifiers from data streams. IEEE Trans. Fuzzy Syst. 16 (6), 1462–1475.
Antonelli, M., Ducange, P., Lazzerini, B., Marcelloni, F., 2011. Learning knowledge bases of multi-objective evolutionary fuzzy systems by simultaneously optimizing accuracy, complexity and partition integrity. Soft Comput. 15 (12), 2335–2354.
Antonelli, M., Ducange, P., Marcelloni, F., 2012. Genetic training instance selection in multiobjective evolutionary fuzzy systems: A coevolutionary approach. IEEE Trans. Fuzzy Syst. 20 (2), 276–290.
Atanassov, K.T., 1986. Intuitionistic fuzzy sets. Fuzzy Sets Syst. 20 (1), 87–96.
Back, T., 1996. Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms. Oxford University Press.
Baluja, S., 1994. Population-based incremental learning. In: Method for Integrating Genetic Search Based Function Optimization and Competitive Learning. Technical Report CMU-CS-94-163, Carnegie Mellon University.
Baruah, R.D., Angelov, P., 2011. Evolving fuzzy systems for data streams: A survey. Wiley Interdiscip. Rev. Data Min. Knowl. Discov. 1 (6), 461–476.
Bellman, R., 2013. Dynamic Programming. Courier Corporation.
Berenji, H.R., Khedkar, P., 1992. Learning and tuning fuzzy logic controllers through reinforcements'. IEEE Trans. Neural Netw. 3 (5), 724–740.
Bojadziev, G., 2007. Fuzzy Logic for Business, Finance, and Management, Vol. 23. World Scientific.
Booker, L.B., 1982. Intelligent Behavior as an Adaptation to the Task Environment (Ph.D. thesis). University of Michigan, Ann Arbor, MI, USA.
Brown, M., Bossley, K., Mills, D., Harris, C., 1995. High dimensional neurofuzzy systems: overcoming the curse of dimensionality. In: Proc. 1995 IEEE Int. Fuzzy Syst., Vol. 4, pp. 2139–2146.
Buckley, J.J., Hayashi, Y., 1994. Fuzzy neural networks: A survey. Fuzzy Sets Syst. 66 (1), 1–13.
Buckley, J.J., Hayashi, Y., Czogała, E., 1999. On the equivalence of neural nets and fuzzy expert systems. Fuzzy Sets Syst. 100, 145–150.
Buckley, J.J., Yoichi, H., 1995. Neural nets for fuzzy systems. Fuzzy Sets Syst. 71 (3), 265–276.
Carpenter, G.A., Grossberg, S., Rosen, D.B., 1991. FuzzyART: Fast stable learning and categorization of analog patterns by an adaptive resonance system. Neural Netw. 4 (6), 759–771.
Carse, B., Fogarty, T.C., Munro, A., 1996. Evolving fuzzy rule based controllers using genetic algorithms. Fuzzy Sets Syst. 80 (3), 273–293.
Caruana, R., Niculescu-Mizil, A., 2004. Data mining in metric space: An empirical analysis of supervised learning performance criteria. In: Proc. 10 ACM SIGKDD, Int. Conf. Knowl. Discovery Data Min. ACM, pp. 69–78.
Casillas, J., Cordón, O., Triguero, F.H., Magdalena, L., 2013. Interpretability Issues in Fuzzy Modeling, Vol. 128. Springer.
Castano, S., De Antonellis, V., 2001. Global viewing of heterogeneous data sources. IEEE Trans. Knowl. Data Eng. 13 (2), 277–297.
Castillo, O., Martínez-Marroquín, R., Melin, P., Valdez, F., Soria, J., 2012. Comparative study of bio-inspired algorithms applied to the optimization of type-1 and type-2 fuzzy controllers for an autonomous mobile robot. Inf. Sci. 192, 19–38.
Castillo, O., Melin, P., 2012. Optimization of type-2 fuzzy systems based on bio-inspired methods: A concise review. Inf. Sci. 205, 1–19.
Castillo, O., Melin, P., 2014. A review on interval type-2 fuzzy logic applications in intelligent control. Inf. Sci. 279, 615–631.
Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W., 2002. SMOTE: synthetic minority over-sampling technique. J. Artif. Intell. Res. 16, 321–357.
Chen, Y., Yang, B., Abraham, A., Peng, L., 2007. Automatic design of hierarchical Takagi–Sugeno type fuzzy systems using evolutionary algorithms. IEEE Trans. Fuzzy Syst. 15 (3), 385–397, 109.
Chien, B.-C., Lin, J.Y., Hong, T.-P., 2002. Learning discriminant functions with fuzzy attributes for classification using genetic programming. Expert Syst. Appl. 23 (1), 31–37.
Cho, K.B., Wang, B.H., 1996. Radial basis function based adaptive fuzzy systems and their applications to system identification and prediction. Fuzzy Sets Syst. 83 (3), 325–339.
Chung, F.-L., Duan, J.-C., 2000. On multistage fuzzy neural network modeling. IEEE Trans. Fuzzy Syst. 8 (2), 125–142.
Coello, C.A.C., Lamont, G.B., Van Veldhuizen, D.A., 2007. Evolutionary Algorithms for Solving Multi-Objective Problems, Vol. 5.
Cord, O., Herrera, F., Hoffmann, F., Magdalena, L., 2001. Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases, Vol. 19. World Scientific.
Cordón, O., Del Jesus, M.J., Herrera, F., Magdalena, L., Villar, P., 2003. A multiobjective genetic learning process for joint feature selection and granularity and contexts learning in fuzzy rule-based classification systems. In: Interpretability Issues in Fuzzy Modeling. Springer, pp. 79–99.
Cordón, O., Gomide, F., Herrera, F., Hoffmann, F., Magdalena, L., 2004. Ten years of genetic fuzzy systems: current framework and new trends. Fuzzy Sets Syst. 1 (141), 5–31.
Cordón, O., Herrera, F., 1997. A three-stage evolutionary process for learning descriptive and approximate fuzzy-logic-controller knowledge bases from examples. Int. J. Approx. Reason. 17 (4), 369–407.
Cordón, O., Moya, F., Zarco, C., 2002. A new evolutionary algorithm combining simulated annealing and genetic programming for relevance feedback in fuzzy information retrieval systems. Soft Comput. 6 (5), 308–319.
Cpałka, K., Łapa, K., Przybył, A., Zalasiński, M., 2014. A new method for designing neuro-fuzzy systems for nonlinear modelling with interpretability aspects. Neurocomputing 135, 203–217.
Das, A.K., Subramanian, K., Sundaram, S., 2015. An evolving interval type-2 neurofuzzy inference system and its metacognitive sequential learning algorithm. IEEE Trans. Fuzzy Syst. 23 (6), 2080–2093.
Deb, K., Jain, H., 2014. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints. IEEE Trans. Evol. Comput. 18 (4), 577–601.
Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. 6 (2), 182–197.
Delgado, M.R., Von Zuben, F., Gomide, F., 2004. Coevolutionary genetic fuzzy systems: A hierarchical collaborative approach. Fuzzy Sets Syst. 141 (1), 89–106.
Deng, D., Kasabov, N., 2003. On-line pattern analysis by evolving self-organizing maps. Neurocomputing 51, 87–103.
Deng, Y., Ren, Z., Kong, Y., Bao, F., Dai, Q., 2017. A hierarchical fused fuzzy deep neural network for data classification. IEEE Trans. Fuzzy Syst. 25 (4), 1006–1012.
Domingo, M., Sierra, C., 1997. A knowledge level analysis of taxonomic domains. Int. J. Intell. Syst. 12 (2), 105–135.
Dorigo, M., Caro, G.D., Gambardella, L.M., 1999. Ant algorithms for discrete optimization. Artif. Life 5 (2), 137–172.
Dubois, D., Prade, H., Ughetto, L., 1997. Checking the coherence and redundancy of fuzzy knowledge bases. IEEE Trans. Fuzzy Syst. 5 (3), 398–417.
Ducange, P., Lazzerini, B., Marcelloni, F., 2010. Multi-objective genetic fuzzy classifiers for imbalanced and cost-sensitive datasets. Soft Comput. 14 (7), 713–728.
Elhag, S., Fernández, A., Bawakid, A., Alshomrani, S., Herrera, F., 2015. On the combination of genetic fuzzy systems and pairwise learning for improving detection rates on intrusion detection systems. Expert Syst. Appl. 42 (1), 193–202.
Elwell, R., Polikar, R., 2011. Incremental learning of concept drift in nonstationary environments. IEEE Trans. Neural Netw. 22 (10), 1517–1531.
Eshelman, L.J., Schaffer, J.D., 1993. Real-coded genetic algorithms and interval-schemata. In: Foundations of Genetic Algorithms, Vol. 2. pp. 187–202.
Fazzolari, M., Alcala, R., Nojima, Y., Ishibuchi, H., Herrera, F., 2013. A review of the application of multiobjective evolutionary fuzzy systems: current status and further directions. IEEE Trans. Fuzzy Syst. 21 (1), 45–65.
Feldman, R., Sanger, J., 2007. The Text Mining HandBook: Advanced Approaches in Analyzing Unstructured Data. Cambridge University Press.
Feng, G., Huang, G.-B., Lin, Q., Gay, R.K.L., 2009. Error minimized extreme learning machine with growth of hidden nodes and incremental learning. IEEE Trans. Neural Netw. 20 (8), 1352–1357.
Fernández, A., del Jesus, M.J., Herrera, F., 2009. Hierarchical fuzzy rule based classification systems with genetic rule selection for imbalanced data-sets. Int. J. Approx. Reason. 50 (3), 561–577.
Feuring, T., Buckley, J.J., Lippe, W.-M., Tenhagen, A., 1999. Stability analysis of neural net controllers using fuzzy neural networks. Fuzzy Sets Syst. 101 (2), 303–313.
Funabashi, M., Maeda, A., Morooka, Y., Mori, K., 1995. Fuzzy and neural hybrid expert systems: Synergetic AI. IEEE Expert 10 (4), 32–40.
Gacto, M.J., Alcalá, R., Herrera, F., 2009. Adaptation and application of multi-objective evolutionary algorithms for rule reduction and parameter tuning of fuzzy rule-based systems. Soft Comput. 13 (5), 419–436.
Gacto, M.J., Alcalá, R., Herrera, F., 2011. Interpretability of linguistic fuzzy rule-based systems: An overview of interpretability measures. Inf. Sci. 181 (20), 4340–4360.
Gama, J., Žliobaite, I., Bifet, A., Pechenizkiy, M., Bouchachia, A., 2014. A survey on concept drift adaptation. ACM Comput. Surv. 46 (4), 44.
Goldberg, D.E., 1991. Real-coded genetic algorithms, virtual alphabets, and blocking. Complex Systems 5 (2), 139–167.
Goldberg, D.E., Holland, .J.H., 1988. Genetic algorithms and machine learning. Mach. Learn. 3 (2), 95–99.
Gomez, J., Dasgupta, D., 2002. Evolving fuzzy classifiers for intrusion detection. In: Proc. 2002 IEEE Workshop on Inf. Assur., Vol. 6. IEEE Computer Press, New York, pp. 321–323.
Greene, D.P., Smith, S.F., 1993. Competition-based induction of decision models from examples. Mach. Learn. 13 (2–3), 229–257.
Guillaume, S., 2001. Designing fuzzy inference systems from data: An interpretability-oriented review. IEEE Trans. Fuzzy Syst. 9 (3), 426–443.
Habbi, H., Boudouaoui, Y., Karaboga, D., Ozturk, C., 2015. Self-generated fuzzy systems design using artificial bee colony optimization. Inf. Sci 295, 145–159.
Hadavandi, E., Shavandi, H., Ghanbari, A., 2010. Integration of genetic fuzzy systems and artificial neural networks for stock price forecasting. Knowl.-Based Syst. 23 (8), 800–808, 152.
Hagras, H.A., 2004. A hierarchical type-2 fuzzy logic control architecture for autonomous mobile robots. IEEE Trans. Fuzzy Syst. 12 (4), 524–539.

Herrera, F., 2008. Genetic fuzzy systems: taxonomy, current research trends and prospects. Evol. Intell. 1 (1), 27–46.
Herrera, F., Lozano, M., Verdegay, J.L., 1995. Tuning fuzzy logic controllers by genetic algorithms. Int. J. Approx. Reason. 12 (3–4), 299–315.
Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Kingsbury, B., et al., 2012. Deep neural networks for acoustic modeling in speech recognition. IEEE Signal Process Mag. 29.
Hoffmann, F., Nelles, O., 2001. Genetic programming for model selection of TSK-fuzzy systems. Inf. Sci. 136 (1–4), 7–28.
Hoffmann, F., Pfister, G., 1997. Evolutionary design of a fuzzy knowledge base for a mobile robot. Int. J. Approx. Reason. 17 (4), 447–469.
Horikawa, S.-i., Furuhashi, T., Uchikawa, Y., 1992. On fuzzy modeling using fuzzy neural networks with the back-propagation algorithm. IEEE Trans. Neural Netw. 3 (5), 801–806.
Ishibuchi, H., 2007. Multiobjective genetic fuzzy systems: review and future research directions. In: IEEE Int. Fuzzy Syst., pp. 1–6.
Ishibuchi, H., Murata, T., Türkşen, I., 1997a. Single-objective and two-objective genetic algorithms for selecting linguistic rules for pattern classification problems. Fuzzy Sets Syst. 89 (2), 135–150.
Ishibuchi, H., Nakashima, T., Kuroda, T., 1999. A hybrid fuzzy genetics-based machine learning algorithm: hybridization of Michigan approach and Pittsburgh approach. In: Int. Conf. Syst. Man Cybern, Vol. 1, pp. 296–301.
Ishibuchi, H., Nakashima, T., Murata, T., 1997b. Comparison of the michigan and pittsburgh approaches to the design of fuzzy classification systems. Electron. Commun. Japan III 80 (12), 10–19.
Ishibuchi, H., Nii, M., 2001. Numerical analysis of the learning of fuzzified neural networks from fuzzy if–then rules. Fuzzy Sets Syst. 120 (2), 281–307.
Ishibuchi, H., Nojima, Y., 2006. Evolutionary multiobjective optimization for the design of fuzzy rule-based ensemble classifiers. Int. J. Hybrid Intell. Syst. 3 (3), 129–145.
Ishibuchi, H., Nojima, Y., 2007. Analysis of interpretability-accuracy deoff of fuzzy systems by multiobjective fuzzy genetics-based machine learning. Int. J. Approx. Reason. 44 (1), 4–31.
Ishibuchi, H., Nozaki, K., Yamamoto, N., Tanaka, H., 1995. Selecting fuzzy if-then rules for classification problems using genetic algorithms. IEEE Trans. Fuzzy Syst. 3 (3), 260–270.
Jackson, D.D., 1972. Interpretation of inaccurate, insufficient and inconsistent data. Geophys. J. R. Astron. Soc. 28 (2), 97–109.
Jain, A.K., 2010. Data clustering: 50 years beyond K-means. Pattern Recognit. Lett. 31 (8), 651–666.
Jain, L.C., Kandel, A., Teodorescu, H.-N.L., 2017. Fuzzy and Neuro-Fuzzy Systems in Medicine. CRC Press.
Jang, J.-S.R., 1991. Fuzzy Modeling using Generalized Neural Networks and Kalman Filter Algorithm, Vol. 91. AAAI, pp. 762–767.
Jang, J.-S.R., 1992. Self-learning fuzzy controllers based on temporal backpropagation. IEEE Trans. Neural Netw. 3 (5), 714–723.
Jang, J.-S.R., 1993. ANFIS: Adaptive-network-based fuzzy inference system. IEEE Trans. Syst. Man Cybern. 23 (3), 665–685, 10471.
Jin, Y., 2000. Fuzzy modeling of high-dimensional systems: complexity reduction and interpretability improvement. IEEE Trans. Fuzzy Syst. 8 (2), 212–221.
Jin, Y., Wang, L., 2008. Fuzzy Systems in Bioinformatics and Computational Biology, Vol. 242. Springer.
Joo, M.G., 2003. A method of converting conventional fuzzy logic system to 2 layered hierarchical fuzzy system. In: IEEE Int. Conf. Fuzzy Syst., Vol. 2. IEEE, pp. 1357–1362.
Joo, M.G., Lee, J.S., 2005. A class of hierarchical fuzzy systems with constraints on the fuzzy rules. IEEE Trans. Fuzzy Syst. 13 (2), 194–203.
Juang, C.-F., Hsu, C.-H., 2009. Reinforcement interval type-2 fuzzy controller design by online rule generation and Q-value-aided ant colony optimization. IEEE Trans. Syst. Man Cybern. B Cybern. 39 (6), 1528–1542.
Juang, C.-F., Lin, C.-T., 1998. An online self-constructing neural fuzzy inference network and its applications. IEEE Trans. Fuzzy Syst. 6 (1), 12–32.
Juang, C.-F., Lin, C.-T., 1999. A recurrent self-organizing neural fuzzy inference network. IEEE Trans. Neural Netw. 10 (4), 828–845.
Juang, C.-F., Lin, J.-Y., Lin, C.-T., 2000. Genetic reinforcement learning through symbiotic evolution for fuzzy controller design. IEEE Trans. Syst. Man Cybern. B Cybern. 30 (2), 290–302.
Juang, C.-F., Lin, Y.-Y., Tu, C.-C., 2010. A recurrent self-evolving fuzzy neural network with local feedbacks and its application to dynamic system processing. Fuzzy Sets Syst. 161 (19), 2552–2568.
Juang, C.-F., Tsao, Y.-W., 2008. A self-evolving interval type-2 fuzzy neural network with online structure and parameter learning. IEEE Trans. Fuzzy Syst. 16 (6), 1411–1424.
Karaboga, D., Kaya, E., 2018. Adaptive network based fuzzy inference system (ANFIS) training approaches: A comprehensive survey. Artif. Intell. Rev. 1–31.
Karnik, N.N., Mendel, J.M., Liang, Q., 1999. Type-2 fuzzy logic systems. IEEE Trans. Fuzzy Syst. 7 (6), 643–658.
Karr, C.L., 2000. A synergistic architecture for adaptive, intelligent control system development. In: 26th Annu. Conf. IEEE Ind. Electron. Soc. (IECON), Vol. 4. IEEE, pp. 2986–2991.
Kasabov, N., 1998. Evolving fuzzy neural networks-algorithms, applications and biological motivation. In: Proc. 4th Int. Conf. Soft Comput., Vol. 1. World Scientific, pp. 127–274.
Kasabov, N., 2001a. Evolving fuzzy neural networks for supervised/unsupervised online knowledge-based learning. IEEE Trans. Syst. Man Cybern. B Cybern. 31 (6), 902–918.
Kasabov, N.K., 2001b. On-line learning, reasoning, rule extraction and aggregation in locally optimized evolving fuzzy neural networks. Neurocomputing 41 (1), 25–45.
Kasabov, N.K., Kim, J., Watts, M.J., Gray, A.R., 1997. FuNN-2-a fuzzy neural network architecture for adaptive learning and knowledge acquisition. Inf. Sci. 101 (3), 155–175.
Kasabov, N.K., Song, Q., 2002. DENFIS: dynamic evolving neural-fuzzy inference system and its application for time-series prediction. IEEE Trans. Fuzzy Syst. 10 (2), 144–154.
Kennedy, J., 2011. Particle swarm optimization. In: Encyclopedia of Machine Learning. Springer, pp. 760–766.
Kennedy, J., Eberhart, R.C., 1997. A discrete binary version of the particle swarm algorithm. In: IEEE Int. Conf. Syst. Man Cybern, Vol.5. IEEE, pp. 4104–4108.
Kim, J., Kasabov, N., 1999. HyFIS: Adaptive neuro-fuzzy inference systems and their application to nonlinear dynamical systems. Neural Netw. 12 (9), 1301–1319.
Kim, J., Moon, Y., Zeigler, B., 1995. Designing fuzzy net controllers using genetic algorithms. IEEE Control Syst. 15 (3), 66–72.
Komiyama, M., Yoshimoto, K., Sisido, M., Ariga, K., 2017. Chemistry can make strict and fuzzy controls for bio-systems: DNA nanoarchitectonics and cell-macromolecular nanoarchitectonics. Bull. Chem. Soc. Jpn. 90 (9), 967–1004.
Kouikoglou, V.S., Phillis, Y.A., 2009. On the monotonicity of hierarchical sum–product fuzzy systems. Fuzzy Sets Syst. 160 (24), 3530–3538.
Koza, J.R., Rice, J., 1994. Genetic Programming II: Automatic Discovery of Reusable Programs, Vol. 40. MIT Press, Cambridge.
LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521 (7553), 436.
Lee, C.-C., 1990. Fuzzy logic in control systems: fuzzy logic controllerI. IEEE Trans. Syst. Man Cybern. 20 (2), 404–418.
Lee, M.-L., Chung, H.-Y., Yu, F.-M., 2003. Modeling of hierarchical fuzzy systems. Fuzzy Sets Syst. 138 (2), 343–361.
Lee, M.A., Takagi, H., 1993. Integrating design stage of fuzzy systems using genetic algorithms. In: 2nd IEEE Int. Conf. Fuzzy Syst.. IEEE, pp. 612–617.
Lee, C.-H., Teng, C.-C., 2000. Identification and control of dynamic systems using recurrent fuzzy neural networks. IEEE Trans. Fuzzy Syst. 8 (4), 349–366.
Lemos, A., Caminhas, W., Gomide, F., 2011. Multivariable gaussian evolving fuzzy modeling system. IEEE Trans. Fuzzy Syst. 19 (1), 91–104.
Lemos, A., Caminhas, W., Gomide, F., 2013. Adaptive fault detection and diagnosis using an evolving fuzzy classifier. Inf. Sci 220, 64–85.
Leng, G., McGinnity, T.M., Prasad, G., 2006. Design for self-organizing fuzzy neural networks based on genetic algorithms. IEEE Trans. Fuzzy Syst. 14 (6), 755–766.
Li, H.-X., Chen, C., 2000. The equivalence between fuzzy logic systems and feedforward neural networks. IEEE Trans. Neural Netw. 11 (2), 356–365.
Liao, S.-H., 2005. Expert system methodologies and applications—a decade review from 1995 to 2004. Expert Syst. Appl. 28 (1), 93–103.
Lima, E., Gomide, F., Ballini, R., 2006. Participatory evolving fuzzy modeling. In: Int. Symp. on Evol. Fuzzy Syst. IEEE, pp. 36–41.
Lima, E., Hell, M., Ballini, R., Gomide, F., 2010. Evolving fuzzy modeling using participatory learning. Evol. Intell. Syst. Method. Appl. 67–86.
Lin, C.-T., 1995. A neural fuzzy control system with structure and parameter learning. Fuzzy Sets Syst. 70 (2), 183–212, 235.
Lin, Y.-Y., Chang, J.-Y., Pal, N.R., Lin, C.-T., 2013. A mutually recurrent interval type-2 neural fuzzy system (MRIT2NFS) with self-evolving structure and parameters. IEEE Trans. Fuzzy Syst. 21 (3), 492–509.
Lin, C.-T., Lee, C.G., 1994. Reinforcement structure/parameter learning for neural-network-based fuzzy logic control systems. IEEE Trans. Fuzzy Syst. 2 (1), 46–63.
Lin, F.-J., Lin, C.-H., Shen, P.-H., 2001. Self-constructing fuzzy neural network speed controller for permanent-magnet synchronous motor drive. IEEE Trans. Fuzzy Syst. 9 (5), 751–759.
Looney, C.G., 1988. Fuzzy petri nets for rule-based decisionmaking. IEEE Trans. Syst. Man Cybern. 18 (1), 178–183.
López, V., Del Río, S., Benítez, J.M., Herrera, F., 2015. Cost-sensitive linguistic fuzzy rule based classification systems under the mapreduce framework for imbalanced big data. Fuzzy Sets Syst. 258, 5–38.
Losing, V., Hammer, B., Wersing, H., 2018. Incremental on-line learning: A review and comparison of state of the art algorithms. Neurocomputing 275, 1261–1274.
Lughofer, E., 2008a. Extensions of vector quantization for incremental clustering. Pattern Recognit. 41 (3), 995–1011.
Lughofer, E.D., 2008b. FLEXFIS: a robust incremental learning approach for evolving Takagi–Sugeno fuzzy models. IEEE Trans. Fuzzy Syst. 16 (6), 1393–1410.
Lughofer, E., 2011. Evolving fuzzy systems-methodologies, advanced concepts and applications, Vol. 53. Springer.
Lughofer, E., 2013. On-line assurance of interpretability criteria in evolving fuzzy systems–achievements new concepts and open issues. Inf. Sci. 251, 22–46.
Lughofer, E., Angelov, 2011. Handling drifts and shifts in on-line data streams with evolving fuzzy systems. Appl. Soft Comput. 11 (2), 2057–2068.

Lughofer, E., Angelov, P., Zhou, X., 2007. Evolving single-and multi-model fuzzy classifiers with flexfis-class. In: IEEE Int. Conf. Fuzzy Syst. IEEE, pp. 1–6.
Lughofer, E., Bouchot, J.-L., Shaker, A., 2011. On-line elimination of local redundancies in evolving fuzzy systems. Evol. Syst. 2 (3), 165–187.
Lughofer, E., Pratama, M., Skrjanc, I., 2018. Incremental rule splitting in generalized evolving fuzzy systems for autonomous drift compensation. IEEE Trans. Fuzzy Syst. 26 (4), 1854–1865.
Maeda, H., 1996. An investigation on the spread of fuzziness in multi-fold multi-stage approximate reasoning by pictorial representation—under sup-min composition and triangular type membership function. Fuzzy Sets Syst. 80 (2), 133–148.
Magdalena, L., 2018. Do hierarchical fuzzy systems really improve interpretability?. In: Int. Conf. Inf. Process. Manage. Uncertainty in Knowledge-Based Syst. Springer, pp. 16–26.
Mamdani, E.H., 1974. Application of fuzzy algorithms for control of simple dynamic plant. In: Proc. Inst. Electr. Eng., Vol. 121. IET, pp. 1585–1588.
Mao, J., Zhang, J., Yue, Y., Ding, H., 2005. Adaptive-tree-structure-based fuzzy inference system. IEEE Trans. Fuzzy Syst. 13 (1), 1–12.
Martinez-Soto, R., Castillo, O., Aguilar, L.T., Melin, 2010. Fuzzy logic controllers optimization using genetic algorithms and particle swarm optimization. In: Mexican Int. Conf. Artif. Intell.. Springer, pp. 475–486.
Martínez-Soto, R., Castillo, O., Aguilar, L.T., Rodriguez, A., 2015. A hybrid optimization method with PSO and GA to automatically design type-1 and type-2 fuzzy logic controllers. Int. J. Mach. Learn. Cybern. 6 (2), 175–196.
Mastorocostas, P.A., Theocharis, J.B., 2002. A recurrent fuzzy-neural model for dynamic system identification. IEEE Trans. Syst. Man Cybern. Cybern. 32 (2), 176–190.
Masuoka, R., Watanabe, N., Kawamura, A., Owada, Y., Asakawa, K., 1990. Neurofuzzy system-fuzzy inference using a structured neural network. In: Proc. Int. Conf. Fuzzy Logic & Neural Netw., pp. 173–177.
Melin, P., Mendoza, O., Castillo, O., 2011. Face recognition with an improved interval type-2 fuzzy logic sugeno integral and modular neural networks. IEEE Trans. Syst. Man Cybern. A Syst. Hum. 41 (5), 1001–1012.
Melin, P., Olivas, F., Castillo, O., Valdez, F., Soria, J., Valdez, M., 2013. Optimal design of fuzzy classification systems using PSO with dynamic parameter adaptation through fuzzy logic. Expert Syst. Appl. 40 (8), 3196–3206.
Melin, P., Sánchez, D., Castillo, O., 2012. Genetic optimization of modular neural networks with fuzzy response integration for human recognition. Inf. Sci. 197, 1–19.
Mendel, J.M., 2013. On KM algorithms for solving type-2 fuzzy set problems. IEEE Trans. Fuzzy Syst. 21 (3), 426–446.
Mendel, J.M., John, R.B., 2002. Type-2 fuzzy sets made simple. IEEE Trans. Fuzzy Syst. 10 (2), 117–127.
Mohammadzadeh, A., Ghaemi, S., 2016. A modified sliding mode approach for synchronization of fractional-order chaotic/hyperchaotic systems by using new self-structuring hierarchical type-2 fuzzy neural network. Neurocomputing 191, 200–213.
Moriarty, D.E., Mikkulainen, R., 1996. Efficient reinforcement learning through symbiotic evolution. Mach. Learn. 22 (1–3), 11–32.
Mouzouris, G.C., Mendel, J.M., 1997. Nonsingleton fuzzy logic systems: theory and application. IEEE Trans. Fuzzy Syst. 5 (1), 56–71.
Muñoz, González A., Herrera, F., 1997. Multi-stage genetic fuzzy systems based on the iterative rule learning approach. In: Mathware & Soft Comput., Vol. 4. Polytechnic University of Catalonia.
Munoz-Salinas, R., Aguirre, E., Cordón, O., García-Silvente, M., 2008. Automatic tuning of a fuzzy visual system using evolutionary algorithms: single-objective versus multiobjective approaches. IEEE Trans. Fuzzy Syst. 16 (2), 485–501.
Nauck, D., Kruse, R., 1993. A fuzzy neural network learning fuzzy control rules and membership functions by fuzzy error backpropagation. In: IEEE Int. Conf. Neural Netw.. IEEE, pp. 1022–1027.
Nauck, D., Kruse, R., 1997. A neuro-fuzzy method to learn fuzzy classification rules from data. Fuzzy Sets Syst. 89 (3), 277–288.
Nauck, D., Kruse, R., 1999. Neuro-fuzzy systems for function approximation. Fuzzy Sets Syst. 101 (2), 261–271, 262.
Ojha, V.K., Abraham, A., Snášel, V., 2016. Metaheuristic tuning of type-II fuzzy inference systems for data mining. In: IEEE Int. Conf. Fuzzy Syst.. IEEE, pp. 610–617.
Ojha, V.K., Abraham, A., Snášel, V., 2017. Metaheuristic design of feedforward neural networks: A review of two decades of research. Eng. Appl. Artif. Intell. 60, 97–116.
Ojha, V.K., Snášel, V., Abraham, A., 2018. Multiobjective programming for type-2 hierarchical fuzzy inference trees. IEEE Trans. Fuzzy Syst. 26 (2), 915–936.
Pandiarajan, K., Babulal, C., 2016. Fuzzy harmony search algorithm based optimal power flow for power system security enhancement. Int. J. Electr. Power Energy Syst. 78, 72–79.
Papadakis, S.E., Theocharis, J., 2002. A GA-based fuzzy modeling approach for generating TSK models. Fuzzy Sets Syst. 131 (2), 121–152.
Park, B.-J., Pedrycz, W., Oh, S.-K., 2002. Fuzzy polynomial neural networks: hybrid architectures of fuzzy modeling. IEEE Trans. Fuzzy Syst. 10 (5), 607–621.
Pradlwarter, H., Schuëller, G., 2008. The use of kernel densities and confidence intervals to cope with insufficient data in validation experiments. Comput. Methods Appl. Mech. Engrg. 197 (29–32), 2550–2560.
Precup, R.-E., Hellendoorn, H., 2011. A survey on industrial applications of fuzzy control. Comput. Ind. 62 (3), 213–226.
Pulkkinen, P., Koivisto, H., 2010. A dynamically constrained multiobjective genetic fuzzy system for regression problems. IEEE Trans. Fuzzy Syst. 18 (1), 161–177.
Purshouse, R.C., Fleming, P.J., 2003. Evolutionary many-objective optimisation: An exploratory analysis. In: Evolutionary Computation, 2003. CEC'03. The 2003 Congress on, Vol. 3. IEEE, pp. 2066–2073.
Raju, G., Zhou, J., Kisner, R.A., 1991. Hierarchical fuzzy control. Internat. J. Control 54 (5), 1201–1216.
Rey, M.I., Galende, M.J., Sainz-Palmero, G., 2017. Multi-objective based fuzzy rule based systems (FRBSs) for trade-off improvement in accuracy and interpretability: A rule relevance point of view. Knowl.-Based Syst. 127, 67–84.
Sahab, N., Hagras, H., 2011. Adaptive non-singleton type-2 fuzzy logic systems: A way forward for handling numerical uncertainties in real world applications. Int. J. Comput. Commun. Control 6 (3), 503–529.
Sahin, S., Tolun, M.R., Hassanpour, R., 2012. Hybrid expert systems: A survey of current approaches and applications. Expert Syst. Appl. 39 (4), 4609–4617.
Salustowicz, R., Schmidhuber, J., 1997. Probabilistic incremental program evolution. Evol. Comput. 5 (2), 123–141.
Sánchez, L., Couso, I., Corrales, J.A., 2001. Combining GP operators with SA search to evolve fuzzy rule based classifiers. Inf. Sci. 136 (1–4), 175–191.
Schlimmer, J.C., Granger, R.H., 1986. Incremental learning from noisy data. Mach. Learn. 1 (3), 317–354.
Seng, T.L., Khalid, M.B., Yusof, R., 1999. Tuning of a neuro-fuzzy controller by genetic algorithm. IEEE Trans. Syst. Man Cybern. B Cybern. 29 (2), 226–236.
Shahzad, M., Zahid, S., Farooq, M., 2009. A hybrid GA-PSO fuzzy system for user identification on smart phones. In: Proc. 11th Annu. Conf. Genetic and Evol. Comput.. ACM, pp. 1617–1624.
Shann, J., Fu, H., 1995. A fuzzy neural network for rule acquiring on fuzzy control systems. Fuzzy Sets Syst. 71 (3), 345–357.
Simpson, P.K., 1992. Fuzzy min–max neural networks–part 1: Classification. IEEE Trans. Neural Netw. 3 (5), 776–786.
Smith, S.F., 1980. A Learning System Based on Genetic Adaptive Algorithms (Ph.D. thesis). University of Pittsburgh, Pittsburgh, PA, USA.
Socha, K., Dorigo, M., 2008. Ant colony optimization for continuous domains. Eur. J. Oper. Res. 185 (3), 1155–1173.
Stanley, K.O., Miikkulainen, R., 2002. Evolving neural networks through augmenting topologies. Evol. Comput. 10 (2), 99–127.
Takagi, T., Sugeno, M., 1985. Fuzzy identification of systems and its applications to modeling and control. IEEE Trans. Syst. Man Cybern. 15 (1), 116–132.
Talbi, E.-G., 2009. Metaheuristics: From Design To Implementation, Vol. 74. John Wiley & Sons.
Thrift, P.R., 1991. Fuzzy logic synthesis with genetic algorithms. In: Proc. 4th Int. Conf. Genetic Algorithms, pp. 950–513.
Torra, V., 2002. A review of the construction of hierarchical fuzzy systems. Int. J. Intell. Syst. 17 (5), 531–543.
Torra, V., 2010. Hesitant fuzzy sets. Int. J. Intell. Syst. 25 (6), 529–539.
Tsang, C.-H., Kwong, S., Wang, H., 2007. Genetic-fuzzy rule mining approach and evaluation of feature selection techniques for anomaly intrusion detection. Pattern Recognit. 40 (9), 2373–2391.
Tung, S.W., Quek, C., Guan, C., 2011. SaFIN: A self-adaptive fuzzy inference network. IEEE Trans. Neural Netw. 22 (12), 1928–1940.
Valdez, F., Melin, P., Castillo, O., 2011. An improved evolutionary method with fuzzy logic for combining particle swarm optimization and genetic algorithms. Appl. Soft Comput. 11 (2), 2625–2632.
Valdez, F., Melin, P., Castillo, O., 2014. A survey on nature-inspired optimization algorithms with fuzzy logic for dynamic parameter adaptation. Expert Syst. Appl. 41 (14), 6459–6466.
Venturini, G., 1993. SIA: A supervised inductive algorithm with genetic search for learning attributes based concepts. In: Eur. Conf. Machine Learning. Springer, pp. 280–296.
Verma, O.P., Parihar, A.S., 2017. An optimal fuzzy system for edge detection in color images using bacterial foraging algorithm. IEEE Trans. Fuzzy Syst. 25 (1), 114–127.
Wang, L.-X., 1998. Universal approximation by hierarchical fuzzy systems. Fuzzy Sets Syst. 93 (2), 223–230.
Wang, L.-X., 1999. Analysis and design of hierarchical fuzzy systems. IEEE Trans. Fuzzy Syst. 7 (5), 617–624.
Wang, E.-C., Kuh, A., 1992. A smart algorithm for incremental learning. In: Int. Jt. Conf. Neural Netw., Vol. 3. IEEE, pp. 121–126.
Wang, J.-S., Lee, C.G., 2002. Self-adaptive neuro-fuzzy inference systems for classification applications. IEEE Trans. Fuzzy Syst. 10 (6), 790–802.
Wang, L.-X., Mendel, J.M., 1992. Fuzzy basis functions, universal approximation, and orthogonal least-squares learning. IEEE Trans. Neural Netw. 3 (5), 807–814.
Wang, Y., Rong, G., 1999. A self-organizing neural-network-based fuzzy system. Fuzzy Sets Syst. 103 (1), 1–11.
Wang, D., Zeng, X.-j., Keane, J., 2006. A survey of hierarchical fuzzy systems. Int. J. Comput. Cogn. 4 (1), 18–29.
Wang, S., Jin, Y., Wei, W., Man, K.-F., 2005. Multi-objective hierarchical genetic algorithm for interpretable fuzzy rule-based knowledge extraction. Fuzzy Sets Syst. 149 (1), 149–186.
Wang, K., Griffin, M. F, 1996. An approach to fuzzy control of nonlinear systems: stability and design issues. IEEE Trans. Fuzzy Syst. 4 (1), 14–23.

Whitehead, B.A., Choate, T.D., 1996. Cooperative-competitive genetic evolution of radial basis function centers and widths for time series prediction. IEEE Trans. Neural Netw. 7 (4), 869–880.
Won, J.M., Park, S.Y., Lee, J.S., 2002. Parameter conditions for monotonic Takagi–Sugeno–Kang fuzzy system. Fuzzy Sets Syst. 1 (2), 135–146.
Wright, A.H., 1991. Genetic algorithms for real parameter optimization. In: Foundations of Genetic Algorithms, Vol. 1. Elsevier, pp. 205–218.
Wu, S., Er, M.J., 2000. Dynamic fuzzy neural networks-a novel approach to function approximation. IEEE Trans. Syst. Man Cybern. B Cybern. 30 (2), 358–364.
Wu, S., Er, M.J., Gao, Y., 2001. A fast approach for automatic generation of fuzzy rules by generalized dynamic fuzzy neural networks. IEEE Trans. Fuzzy Syst. 9 (4), 578–594.
Wu, D., Mendel, J.M., 2009. Enhanced Karnik–Mendel algorithms. IEEE Trans. Fuzzy Syst. 17 (4), 923–934.
Wu, D., Tan, W.W., 2006. Genetic learning and performance evaluation of interval type-2 fuzzy logic controllers. Eng. Appl. Artif. Intell. 19 (8), 829–841.
Yang, X.-S., 2010. Nature-Inspired Metaheuristic Algorithms. Luniver Press.
Yen, J., Wang, L., 1999. Simplifying fuzzy rule-based models using orthogonal transformation methods. IEEE Trans. Syst. Man Cybern. Part B Cybern. 29 (1), 13–24.
Yu, W., Moreno-Armendariz, M.A., Rodriguez, F.O., 2007. System identification using hierarchical fuzzy neural networks with stable learning algorithm. J. Intell. Fuzzy Syst. 18 (2), 171–183.
Zadeh, L.A., 1988. Fuzzy logic. Computer 21 (4), 83–93.
Zadeh, L.A., 1999. Fuzzy sets as a basis for a theory of possibility. Fuzzy Sets Syst. 100 (1), 9–34.
Zadeh, L.A., Kacprzyk, J., 1992. Fuzzy Logic for the Management of Uncertainty. John Wiley & Sons.
Zeng, X.-J., Keane, J.A., 2005. Approximation capabilities of hierarchical fuzzy systems. IEEE Trans. Fuzzy Syst. 13 (5), 659–672.
Zhou, X., Angelov, P, 2007. Autonomous visual self-localization in completely unknown environment using evolving fuzzy rule-based classifier. In: IEEE Symp. Comput. Intell. in Secur. and Defense Appl.. IEEE, pp. 131–138.
Zitzler, E., Thiele, L., 1999. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. IEEE Trans. Evol. Comput. 3 (4), 257–271.