Multi-Objective Evolutionary Optimisation for Prototype-Based Fuzzy Classifiers

Xiaowei Gu, Miqing Li, Liang Shen, Guolin Tang, Qiang Ni, Taoxin Peng, and Qiang Shen

***Abstract*—Evolving intelligent systems (EISs), particularly, the zero-order ones have demonstrated strong performance on many real-world problems concerning data stream classification, while offering high model transparency and interpretability thanks to their prototype-based nature. Zero-order EISs typically learn prototypes by clustering streaming data online in a "one pass" manner for greater computation efficiency. However, such identified prototypes often lack optimality, resulting in less precise classification boundaries, thereby hindering the potential classification performance of the systems. To address this issue, a commonly adopted strategy is to minimise the training error of the models on historical training data or alternatively, to iteratively minimise the intra-cluster variance of the clusters obtained via online data partitioning. This recognises the fact that the ultimate classification performance of zero-order EISs is driven by the positions of prototypes in the data space. Yet, simply minimising the training error may potentially lead to overfitting, whilst minimising the intra-cluster variance does not necessarily ensure the optimised prototype-based models to attain improved classification outcomes. To achieve better classification performance whilst avoiding overfitting for zero-order EISs, this paper presents a novel multi-objective optimisation approach, enabling EISs to obtain optimal prototypes via involving these two disparate but complementary strategies simultaneously. Five decision-making schemes are introduced for selecting a suitable solution to deploy from the final non-dominated set of the resulting optimised models. Systematic experimental studies are carried out to demonstrate the effectiveness of the proposed optimisation approach in improving the classification performance of zero-order EISs.**

***Index Terms*—classification, evolving intelligent system, fuzzy classifier, multi-objective optimisation, prototype.**

I. Introduction

This work was supported in part by the Strategic Partner Acceleration Award (80761-AU201), funded under the Sêr Cymru II programme, UK; in part by the National Science Foundation for Post-doctoral Scientists (2021M691899), China; in part by the Natural Science Foundation of Shandong Province (ZR2021QG013), China; and in part by the Postdoctoral Innovative Talent Support Plan of Shandong Province (SDBX2021009), China.

X. Gu is with the School of Computing, University of Kent, Canterbury, CT2 7NZ, UK. email: X.Gu@kent.ac.uk

M. Li is with the School of Computer Science, University of Birmingham, Birmingham, B15 2TT, UK. email: m.li.8@bham.ac.uk

L. Shen is with the School of Information Engineering, Fujian Business University, Fuzhou 350506, China. email: liang.shen.18@fjbu.edu.cn

G. Tang is with the School of Management Science and Engineering, Shandong University of Finance and Economics, Jinan, 250014, China, email: guolin_tang@163.com

Q. Ni is with the School of Computing and Communications, Lancaster University, Lancaster, LA1 4WA, UK. email: q.ni@lancaster.ac.uk

T. Peng is with the School of Computing, Edinburgh Napier University, Edinburgh EH10 5DT, UK. email: t.peng@napier.ac.uk

Q. Shen is with the Department of Computer Science, Aberystwyth University, Aberystwyth, SY23 3DB, UK. email: qqs@aber.ac.uk

Corresponding author: Guolin Tang

Manuscript received XXXX XX, 2022; revised XXXX XX, 2022.

CLASSIFICATION is a task aiming to categorise input samples into different classes. As a practically important topic, a wide variety of successful classification algorithms have been developed for real-world applications, addressing challenging problems [1]–[3].

Classification algorithms typically learn from labelled training data to derive a classification model for predicting the class labels of unlabelled new observations. Mainstream classification approaches include, but are not limited to, deep neural networks (DNNs, or artificial neural networks, ANNs) [1], support vector machines (SVMs) [4], k-nearest neighbour (KNN) [5], decision trees (DTs) [6], random forests (RFs) [7], learning vector quantization (LVQ) [8], and evolving intelligent systems (EISs) [9]. Among these, DNNs are arguably the most successful, thanks to the learning abilities inherent in their multi-layered representation, especially for the extensively reported eye-catching performances on visual and speech information processing and recognition [10]. Nevertheless, DNNs are often criticised as being "black box" models, usually containing huge numbers (multi-million or more) of hyper-parameters with no semantically meaningful link to the underlying domain problems. This has greatly restricted the applicability of DNNs for problems that require the classification models to possess a clear transparency and explainability on their reasoning, despite that significant efforts have been made in an attempt to describe their decisions via post-hoc methods [11].

Compared with DNNs, prototype-based approaches, e.g., SVMs, KNN, LVQ and EISs, are preferred in applications where the transparency and explainability of the classification models are critical. They work based on the exploitation of the concept of prototypes, which are representative samples identified from training data [12]. Different prototype-based approaches identify prototypes from data following different algorithmic procedures, thereby resulting in differences in classification performance and model interpretability. In particular, because of the sophisticated, iterative system identification processes underpinning SVMs and LVQ, which are hardly human-comprehensible, these two types of classifier are still regarded as "black box" models. KNN treats all labelled training samples as prototypes, and determines the class label of an unlabelled sample by comparing it with a predefined number of nearest neighbours in the data space. As such, KNN requires no training practically, and its decision-making scheme is simple and easy-to-understand, but the transparency of its inference is limited, especially when the problem size is large.

EISs form another group of prototype-based methods widely

employed for regression and classification problems concerning data streams [9], [13]–[15]. They are normally designed to self-develop and self-evolve from data streams to dynamically model nonstationary problems in real time. Owing to the interpretability of the resultant models EISs have become increasingly popular over the past two decades, with many successful EISs introduced, including but not limited to: dynamic evolving neural-fuzzy inference system [16], evolving Takagi-Sugeno fuzzy model [17], evolving fuzzy rule-based classifier [9], [18], flexible fuzzy inference system [9], [19], sequential adaptive fuzzy inference system [13], parsimonious network based on fuzzy inference system [20], evolving typicality and eccentricity based data analytics classifier [21], evolving possibilistic fuzzy model [22], self-organising fuzzy inference system (SOFIS) [23], recursive maximum correntropy-based evolving fuzzy system [24], and jointly evolving and compressing fuzzy system [14]. To date, EISs have been successfully applied to address various real-world problems, e.g., fault detection in industrial systems [25], online inspection of microfluidic chips for DNA sequencing [26], real-time driver modelling and manoeuvre recognition [27], etc. More details about the latest progress of EISs and their applications can be found in the recently published review papers [2], [3].

EISs, especially, those zero-order ones, identify prototypes from streaming data samples through a non-iterative, "single-pass" process based on their ensemble properties and mutual distances. These prototypes capture and preserve the local data structure and underlying patterns. They build the knowledge bases of the resulting reasoning systems, determining the classification performances of such systems. However, it is often observed that prototypes learned by EISs lack optimality [28], which may greatly reduce the performance of the learned classifiers. The main reason for this is that EISs intend to avoid iterative optimisation in an effort to gain greater computation efficiency.

A commonly followed approach to improve the performance of individual EISs is to construct ensemble fuzzy systems. The concept of ensemble learning is to create a stronger classifier by combining multiple weak classifiers. For instance, an evolving ensemble fuzzy classifier, named parsimonious ensemble (pENsemble), is proposed in [29]. It consists of a dynamic ensemble structure where the base classifiers are weighted according to their classification accuracy. An ensemble pruning scheme is also introduced to pENsemble to remove base classifiers of a lower accuracy. A fuzzily weighted adaptive boosting scheme designed specifically for zero-order EISs to construct stronger ensemble classifiers is introduced in [30]. This novel boosting utilises the confidence scores produced by EISs in both weight updating and ensemble output generation to create more precise classification boundaries. In [31], an online variant of bagging is proposed for constructing ensemble fuzzy classifiers with an autonomous pruning strategy that discards base classifiers with higher errors.

Considering that the prototypes learned by zero-order EISs often lack optimality, an alternative approach to enhance the classification performance of EISs is to employ evolutionary algorithms (EAs) for prototype optimisation [32]. This is similar to evolutionary fuzzy systems [33], which use EAs in the fuzzy rule base design, but differs in the sense that EAs are only involved for optimisation after the prototype identification process is finished. Classification error is the standard performance measure for classification algorithms; a well-trained classification model is expected to predict the class labels of training samples correctly. However, using training error (i.e., classification error on training data) as the sole objective for optimisation may be insufficient because it can cause the classifier to be overfitted, resulting in poorer performance on unseen data samples with unfamiliar patterns.

As EISs learn prototypes from data streams typically by online clustering [2], the optimality of prototypes can be viewed as the optimality of the clusters. Minimising the intra-cluster variance of the clusters obtained during the online learning, by modifying them with historical data, can help zero-order EISs to achieve (locally) optimal partitioning of the underlying data space and attain local optimality [28]. Also, such optimised models are less likely to be overfitting because no label information is involved during the optimisation process. However, the intra-cluster variance is an objective measuring the clustering quality, the locally optimal prototypes obtained by minimising this objective does not necessarily improve the classification precision of the learned models globally.

Based on the above observations, it would be interesting to combine both measures (accuracy and intra-cluster variance) in zero-order EISs, in order to maximise their strengths and minimise their individual weaknesses. With this motivation, this paper introduces a novel approach that exploits training error and intra-cluster variance simultaneously to develop zero-order EISs capable of attaining optimal prototypes with strengthened classification performance. In so doing, the issue of incomparability of solutions (or non-dominated solutions) [34] in multi-objective optimisation may arise. To deal with this problem, five decision-making schemes are considered to determine a suitable solution.

Overall, major contributions of this paper are:

1) A theoretical analysis on the optimality of zero-order EISs is conducted from both classification and data partitioning perspectives to identify the weaknesses of commonly adopted optimisation strategies for EISs;
2) A novel evolutionary optimisation approach considering two disparate but complementary performance measures, namely, training error and intra-variance is proposed to help EISs attain optimality whilst avoiding overfitting, supported with a computational complexity analysis;
3) Five multi-objective decision-making schemes are adopted to identify a suitable solution from a nondominated set in relation to a prescribed application problem, and a recommended scheme is given on the basis of empirical observations from experimental studies.

The remainder of this paper is organised as follows. Section II provides technical background. The proposed optimisation approach is detailed in Section III. An analysis on the computational complexity of the proposed approach is given in Section IV. Experimental studies are presented and results are

TABLE I
LIST OF KEY NOTATIONS AND DEFINITIONS

Notation	Definition
$\boldsymbol{x}$	Data sample
y	Class label of $\boldsymbol{x}$
$\hat{y}$	Predicted class label of $\boldsymbol{x}$
M	Dimensionality of $\boldsymbol{x}$
C	Number of classes
G	Level of granularity
$\mathfrak{R}_c$	The c^{th} IF-THEN fuzzy rule
$\mathbf{P}_c$	Collection of prototypes associated with $\mathfrak{R}_c$
$\boldsymbol{P}_c$	Vectorised $\mathbf{P}_c$
$\boldsymbol{p}_{c,i}$	The i^{th} prototype of $\mathfrak{R}_c$
$\mathbf{C}_{c,i}$	Cluster formed around $\boldsymbol{p}_{c,i}$
P_c	Cardinality of $\mathbf{P}_c$
$\mathbf{X}$	Dataset
K	Cardinality of $\mathbf{X}$
$\boldsymbol{x}_k$	The k^{th} sample of $\mathbf{X}$
y_k	Class label of $\boldsymbol{x}_k$
$\mathbf{X}_c$	Set of data samples of the c^{th} class
K_c	Cardinality of $\mathbf{X}_c$
$\boldsymbol{x}_{c,k}$	The k^{th} sample of $\mathbf{X}_c$
$\mathbf{U}_c$	Set of unique samples of the c^{th} class
U_c	Cardinality of $\mathbf{U}_c$
$\boldsymbol{u}_{c,k}$	The k^{th} sample of $\mathbf{U}_c$
$f_{c,k}$	Occurrence frequency of $\boldsymbol{u}_{c,k}$
$\gamma_{c,G}$	Threshold derived from $\mathbf{X}_c$ at the G^{th} granularity level
$D^{MM}(\cdot)$	Multimodal density
$\lambda_c(\cdot)$	Score of confidence produced by $\mathfrak{R}_c$
$\mathbf{P}$	Collection of prototypes
$\boldsymbol{P}$	Vectorised $\mathbf{P}$
$f_j(\cdot)$	The j^{th} objective function
$\boldsymbol{s}_k$	The k^{th} solution
$\mathbf{F}_k$	The k^{th} Pareto front

discussed in Section V. This paper is concluded by Section VI.

II. PRELIMINARIES

Before presenting the proposed approach, this section provides an overview of the directly relevant background material for academic completeness, including an outline of SOFIS [23] and a brief review of evolutionary multi-objective optimisation. Note that SOFIS is used as the underlying platform to implement the proposed approach due to its popularity, but other zero-order EFSs with a similar operating mechanisms may be employed as an alternative if preferred. A list of the key notations and definitions is summarised in Table I.

A. SOFIS

1) Architecture and Decision-Making Policy: SOFIS is a zero-order prototype-based EIS for classification, with an architecture as depicted in Fig. 1. It is composed of *i)* C data processors (one processor per class); *ii)* a fuzzy rule base consisting of C massively parallel IF-THEN rules (one rule per class), and *iii)* a decision-maker to determine the class labels of testing data.

During the learning stage, each data processor identifies prototypes from data samples of the corresponding class based on their ensemble properties and mutual distances. These prototypes are a selected group of highly representative samples, representing the local patterns of data. They play a key role

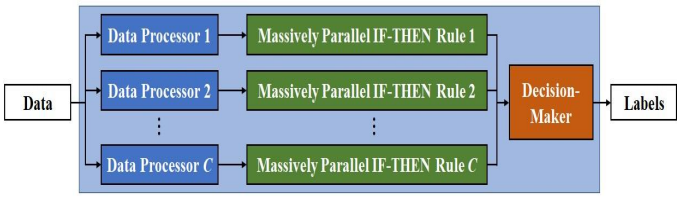


Fig. 1: General architecture of SOFIS [23].

in the system by constructing the massively parallel IF-THEN rules defined in the following form ($c = 1, 2, \ldots, C$) [23]:

$$\mathfrak{R}_c: \; IF \; (\boldsymbol{x} \sim \boldsymbol{p}_{c,1}) \; OR \; (\boldsymbol{x} \sim \boldsymbol{p}_{c,2}) \; OR \; ... \; OR \; (\boldsymbol{x} \sim \boldsymbol{p}_{c,P_c}) \; THEN \; (y = c) \tag{1}$$

where $\boldsymbol{x} = [x_1, x_2, \ldots, x_M]^T$. Since the prototypes are connected by the logical "OR" connectives, $\mathfrak{R}_c$ can be viewed as a parallel ensemble of simpler fuzzy rules sharing the same consequent part as Eq. (2):

$$\mathfrak{R}_{c,i}: \; IF \; (\boldsymbol{x} \sim \boldsymbol{p}_{c,i}) \; \; THEN \; (y = c) \tag{2}$$

During decision-making, given a particular data sample $\boldsymbol{x}$, the activation of $\mathfrak{R}_c$ ($c = 1, 2, \ldots, C$) is produced on the basis of the squared Euclidean distance between $\boldsymbol{x}$ and the nearest prototype of the c^{th} class [23], such that

$$\lambda_c(\boldsymbol{x}) = e^{-||\boldsymbol{x}-\boldsymbol{p}_{c,i^*}||^2}; \; i^* = \underset{i=1,2,..,P_c}{\arg\min} \left(||\boldsymbol{x} - \boldsymbol{p}_{c,i}||^2\right) \tag{3}$$

The class label of $\boldsymbol{x}$ is determined by the IF-THEN rule that produces the highest activation:

$$\hat{y} = c^*; \; c^* = \underset{i=1,2,..,C}{\arg\min} \left(\lambda_c(\boldsymbol{x})\right) \tag{4}$$

The use of such a data-driven threshold guarantees that the learning outcomes of SOFIS is always valid and meaningful. The learning strategy followed by SOFIS is given below (adapted from [23]). By default, the externally controlled level of granularity is set as G, taking a value in terms of a positive integer.

2) Learning Policy: Given a particular dataset, $\mathbf{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_K\}$, SOFIS splits it into C different non-overlapping subsets, denoted by $\mathbf{X}_c = \{\boldsymbol{x}_{c,1}, \boldsymbol{x}_{c,2}, \ldots, \boldsymbol{x}_{c,K_c}\} (c = 1, 2, \ldots, C)$ according to the corresponding class labels, such that $\mathbf{X}_c$ contains only data samples of the c^{th} class. Then, the C subsets are passed on to the corresponding data processors for prototype identification and IF-THEN rule construction.

After the c^{th} data processor has received $\mathbf{X}_c$ ($c = 1, 2, \ldots, C$), it firstly calculates the multimodal density value at each unique data sample of the c^{th} class, denoted as $\boldsymbol{u}_{c,k} \in \mathbf{U}_c$ ($\mathbf{U}_c \subseteq \mathbf{X}_c$) using Eq. (5) [50]:

$$D^{MM}(\boldsymbol{u}_{c,k}) = \frac{f_{c,k}}{1 + \frac{||\boldsymbol{u}_{c,k} - \boldsymbol{\mu}_c||^2}{X_c - ||\boldsymbol{\mu}_c||^2}} \tag{5}$$

where $k = 1, 2, \ldots, U_c$; $f_{c,k} \geq 1$; $\boldsymbol{\mu}_c = \frac{1}{K_c} \sum_{j=1}^{K_c} \boldsymbol{x}_{c,j}$ and $X_c = \frac{1}{K_c} \sum_{j=1}^{K_c} ||\boldsymbol{x}_{c,j}||^2$.

Next, the c^{th} data processor arranges the unique data samples $\mathbf{U}_c$ into a sorted set, denoted as $\mathbf{R}_c$ =

$\{\boldsymbol{r}_{c,1}, \boldsymbol{r}_{c,2}, \ldots, \boldsymbol{r}_{c,U_c}\}$, based on their multimodal density values and mutual distances using Eq. (6) ($c = 1, 2, \ldots, C$) [23]:

$$\boldsymbol{r}_{c,j} = \begin{cases} \underset{\boldsymbol{u} \in \mathbf{U}_c}{\arg\max}(D^{MM}(\boldsymbol{u})), & j = 1 \\ \underset{\substack{\boldsymbol{u} \in \mathbf{U}_c; \\ \boldsymbol{u} \neq \boldsymbol{r}_{c,1}, \ldots, \boldsymbol{r}_{c,j-1}}}{\arg\min} (||\boldsymbol{u} - \boldsymbol{r}_{c,j-1}||^2), & j > 1 \end{cases} \tag{6}$$

Data samples of the c^{th} class with local maximum multimodal density values are identified as raw prototypes following Condition 1 ($j = 1, 2, \ldots, U_c$; $c = 1, 2, \ldots, C$) [23]:

$$\begin{aligned} Cond.1: \; & if \; (D^{MM}(\boldsymbol{r}_{c,j}) > D^{MM}(\boldsymbol{r}_{c,j+1})) \\ & and \; (D^{MM}(\boldsymbol{r}_{c,j}) > D^{MM}(\boldsymbol{r}_{c,j-1})) \\ & then \; (\mathbf{R}_c^* \leftarrow \mathbf{R}_c^* \cup \{\boldsymbol{r}_{c,j}\}) \end{aligned} \tag{7}$$

where $\mathbf{R}_c^* = \{\boldsymbol{r}_{c,1}^*, \boldsymbol{r}_{c,2}^*, \ldots, \boldsymbol{r}_{c,U_c^*}^*\}$ is the collection of local maxima, with $\mathbf{R}_c^* \subseteq \mathbf{R}_c$.

Micro-clusters are then formed around the identified local maxima by attracting data samples of the same class within its neighbourhood, resembling Voronoi tessellations as Eq. (8) ($k = 1, 2, \ldots, K_c$; $c = 1, 2, \ldots, C$) [35]:

$$\mathbb{C}_{c,j} \leftarrow \mathbb{C}_{c,j} \cup \{\boldsymbol{x}_{c,j}\}; \; j = \underset{l=1,2,\ldots,U_c^*}{\arg\min} (||\boldsymbol{x}_{c,k} - \boldsymbol{r}_{c,l}^*||^2) \tag{8}$$

where $\mathbb{C}_{c,j}$ is the micro-cluster formed around $\boldsymbol{r}_{c,j}^*$, with the centre (arithmetic mean) of $\mathbb{C}_{c,j}$ denoted as $\hat{\boldsymbol{r}}_{c,j}^*$. The collection of centres of the formed micro-clusters is denoted by $\hat{\mathbf{R}}_c^*$. The multimodal density at the micro-cluster centres are calculated by Eq. (9):

$$D^{MM}(\hat{\boldsymbol{r}}_{c,j}^*) = \frac{|\mathbb{C}_{c,j}|}{1 + \frac{||\hat{\boldsymbol{r}}_{c,j}^* - \boldsymbol{\mu}_c||^2}{X_c - ||\boldsymbol{\mu}_c||^2}} \tag{9}$$

From this, the data-driven distance threshold $\gamma_{c,G}$ is derived using Eq. (10) based on the mutual distances between data samples of the c^{th} class, $\mathbf{X}_c$ and the level of granularity controlled by the user [23], such that

$$\gamma_{c,g} = \frac{1}{\sum_{i=1}^{K_c-1} \sum_{j=i+1}^{K_c} w_{g,i,j}} \sum_{i=1}^{K_c-1} \sum_{j=i+1}^{K_c} w_{g,i,j} ||\boldsymbol{x}_{c,i} - \boldsymbol{x}_{c,j}||^2 \tag{10}$$

where $g = 1, 2, \ldots, G$; $c = 1, 2, \ldots, C$; $w_{g,i,j}$ and $\gamma_{c,0}$ are defined as follows:

$$w_{g,i,j} = \begin{cases} 1, & if \; ||\boldsymbol{x}_{c,i} - \boldsymbol{x}_{c,j}||^2 \leq \gamma_{c,g-1} \\ 0, & else \end{cases} \tag{11}$$

$$\gamma_{c,0} = \frac{1}{K_c(K_c - 1)} \sum_{i=1}^{K_c-1} \sum_{j=i+1}^{K_c} ||\boldsymbol{x}_{c,i} - \boldsymbol{x}_{c,j}||^2 \tag{12}$$

Condition 2 is then used for identifying those more representative micro-clusters that represent the local peaks of multimodal distribution of data:

$$\begin{aligned} Cond.2: \; & if \; \left(D^{MM}(\hat{\boldsymbol{r}}_{c,j}^*) = \underset{\substack{||\hat{\boldsymbol{r}}_{c,j}^* - \boldsymbol{r}||^2 \leq \gamma_{c,G}; \\ \boldsymbol{r} \in \hat{\mathbf{R}}_c^*}}{\max} (D^{MM}(\boldsymbol{r}))\right) \\ & then \; (\mathbf{P}_c^* \leftarrow \mathbf{P}_c^* \cup \{\hat{\boldsymbol{r}}_{c,j}^*\}) \end{aligned} \tag{13}$$

where $j = 1, 2, \ldots, U_c^*$; $c = 1, 2, \ldots, C$; $\mathbf{P}_c^* = \{\boldsymbol{p}_{c,1}^*, \boldsymbol{p}_{c,2}^*, \ldots, \boldsymbol{p}_{c,P_c}^*\}$ is the collection of the centres of those more representative micro-clusters and $P_c = |\mathbf{P}_c^*|$.

Finally, $\mathbf{P}_c^*$ is utilised to form Voronoi tessellations in the data space via Eq. (14) ($c = 1, 2, \ldots, C$):

$$\mathbf{C}_{c,j} \leftarrow \mathbf{C}_{c,j} \cup \{\boldsymbol{x}_{c,k}\}; \; j = \underset{l=1,2,\ldots,P_c}{\arg\min} (||\boldsymbol{x}_{c,k} - \boldsymbol{p}_{c,l}^*||^2) \tag{14}$$

where $\mathbf{C}_{c,j}$ is the shape-free cluster formed around $\boldsymbol{p}_{c,j}^*$. The arithmetic means of the formed clusters are subsequently extracted as the prototypes, denoted as $\mathbf{P}_c$ ($c = 1, 2, \ldots, C$), and the system identification process is completed after building the IF-THEN rules in the form of Eq. (1), using the resulting identified prototypes.

The system identification process of SOFIS is summarised in Algorithm 1, expressed in pseudo code. Note that the learning policy presented in this section is for static data. Interested reader is referred to [23] for more details about the online learning policy of SOFIS.

Algorithm 1 Learning policy of SOFIS [23].

for $c = 1$ to C **do**
  obtain $\mathbf{U}_c$ from $\mathbf{X}_c$;
  calculate D^{MM} at $\mathbf{U}_c$ using (5);
  obtain $\mathbf{R}_c$ by re-ordering $\mathbf{U}_c$ using (6);
  identify $\mathbf{R}_c^*$ from $\mathbf{R}_c$ using Condition 1;
  form Voronoi tessellations around $\mathbf{R}_c^*$ using (8) and obtain $\hat{\mathbf{R}}_c$;
  calculate D^{MM} at $\hat{\mathbf{R}}_c$ using (9);
  derive $\gamma_{c,G}$ from $\mathbf{X}_c$ using (10);
  identify $\mathbf{P}_c^*$ from $\hat{\mathbf{R}}_c$ using Condition 2;
  form clusters around $\mathbf{P}_c^*$ using (14) and obtain $\mathbf{P}_c$;
  construct $\mathbf{R}_c$ from $\mathbf{P}_c$;
end for

B. Evolutionary Multi-objective Optimisation (EMO)

In multi-objective optimisation, there are more than one objective that are required to be optimised. Without loss of generality, it can be expressed as:

$$\min f(\boldsymbol{s}) = (f_1(\boldsymbol{s}), f_2(\boldsymbol{s}), \cdots, f_m(\boldsymbol{s})) \tag{15}$$

where m denotes the number of objectives, and $\boldsymbol{s}$ denotes a solution in the search space $\boldsymbol{S}$ of possible solutions.

In multi-objective optimisation, a solution $\boldsymbol{s}_1$ is said to be better than $\boldsymbol{s}_2$, or commonly referred to as $\boldsymbol{s}_1$ (Pareto) dominates $\boldsymbol{s}_2$, if and only if $\boldsymbol{s}_1$ is not worse than $\boldsymbol{s}_2$ on all the objectives and better on at least one objective. A solution $\boldsymbol{s}$ is called Pareto optimal if there is no solution in $\boldsymbol{S}$ that dominates $\boldsymbol{s}$. A prominent feature in multi-objective optimisation in contrast to single-objective optimisation is that there is no single optimal solution but rather a set of Pareto optimal solutions (called the Pareto front in the objective space), whose size can be prohibitively large or even infinite [37].

A straightforward way to tackle a multi-objective optimisation problem is to convert it into a single-objective optimisation problem by a number of weights, and then solve

it by a global optimisation solver. However, the decision maker (DM) may struggle to specify a number of weights which can confidently reflect their preferences, particularly when more than two objectives are involved. Very recently, it has been reported that even if the DM can confidently specify their weights, the multi-objective solver may find better quality solutions since it can implicitly help the search jump out of the local optima and is much less sensitive to the incommensurability between objectives [38].

As such, a frequently-used approach for multi-objective optimisation is to search for a good approximation of the whole Pareto front, from which the DM can choose their preferred solution to deploy. In particular, evolutionary computation has gained its popularity in solving multi-objective optimisation problems [39], an approach collectively called evolutionary multi-objective optimisation or EMO. This is because the population-based nature of EMO can generate an approximate set of the Pareto front within one execution, with each solution representing a unique trade-off between different objectives.

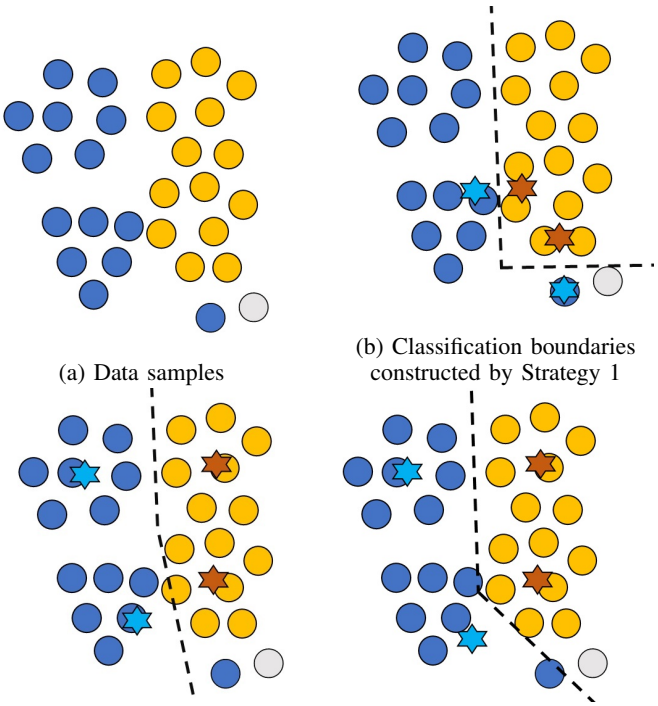
EMO algorithms can be loosely categorised into three classes on the basis of their fitness assignment objectives/rules, namely: Pareto-based, indicator-based and decomposition-based [39]. Pareto-based algorithms, exemplified by NSGA-II [36] and SPEA2 [40], utilise the Pareto dominance relation and a density estimator to distinguish between solutions. Indicator-based algorithms, exemplified by IBEA [41] and SMS-EMOA [42], use an indicator to measure the quality of the entire search population. Decomposition-based algorithms, exemplified by MOEA/D [43] and NSGA-III [44], decompose a multi-objective problem into a number of scalar optimisation problems, with each problem being optimised by a weight vector.

These three classes of EMO algorithms have their own advantages and disadvantages. Pareto-based algorithms work well on problems with two or three objectives but typically fail to scale up to high-dimensional problems [45], [46]. Indicator-based algorithms perform well in terms of convergence, but their performance is dependent on the indicator used and they may struggle to maintain diversity in the potential solutions on certain problems [47]. Decomposition-based algorithms can archive an excellent balance between convergence and diversity on problems with simplex-like Pareto front shapes but may fail on problems with other shapes [43], [44]. Consequently, there is a tendency to explicitly consider multiple objectives in the area, for example using two populations/archives to conduct co-evolution, each being associated with one specific objective [48].

III. Multi-Objective Optimisation and Decision Making for Prototype-Based Classifiers

As aforementioned, a feasible approach to help prototype-based EISs to attain optimality whilst avoiding overfitting is to jointly consider both the training error and the intra-cluster variance in optimisation.

Figure 2 presents an illustrative example, where there exist a number of data samples exclusively belonging to two classes in the data space, respectively represented by dots of two



(a) Data samples

(b) Classification boundaries constructed by Strategy 1

(c) Classification boundaries constructed by Strategy 2

(d) Classification boundaries constructed by Strategy 3

Fig. 2: Illustrative example of classification boundaries formed by prototypes selected with respect to different objectives.

different colours: blue dots for samples of class 1 and yellow for class 2. There is also a grey dot representing a data sample of unknown class, but according to the distribution of coloured samples it can be assumed that this unlabelled sample is very likely belonging to class 2. To create a classification model consisting of four prototypes (two per class; represented by stars in the figure), there are three different strategies that can be considered. Strategy 1 (Fig. 2b) is to address training errors as the main criterion such that the classification boundaries (black dash lines) built by the four prototypes can correctly classify all the training samples. Strategy 2 (Fig. 2c) is to handle intra-cluster variance as the main criterion, where the data space is partitioned by clusters formed around these prototypes in a (locally) optimal manner. Strategy 3 (Fig. 2d) is to consider both criteria together in prototype selection, owing to the recognition from examining Figs. 2b and 2c that the first two strategies are not ideal. This is because the classification boundaries built by Strategy 1 appear to be altered by the "outlier" of class 1 and are less likely to predict the class label of this unlabelled sample correctly, and that the classification boundaries constructed by Strategy 2 are more likely to produce the correct prediction on the unlabelled sample, but would fail to classify the "outlier" to the correct class. In contrast, by considering both objectives together, the classification boundaries constructed by Strategy 3 are able to achieve zero training error, providing more reasonable prediction on the unlabelled sample.

Driven by the above observation, in this section, the two

individual performance measures are utilised together to formulate a bi-objective optimisation problem, for which an effective method for seeking an optimal solution is developed.

A. Optimisation Problem

As shown by Fig. 1, the core of SOFIS is a set of massively parallel IF-THEN rules induced from highly representative prototypes. These prototypes are obtained by creating Voronoi tessellations in the data space, and they play a key role in the internal reasoning and decision-making of SOFIS. Thanks to the nonparametric, prototype-based nature, the optimality of SOFIS is solely dependent upon the positions of such prototypes. Hence, the optimality problem of SOFIS is equivalent to finding the optimal positions for the prototypes within the data space [28].

From the classification perspective, the optimality of SOFIS can be reached by minimising the following objective function [49]:

$$Obj.\ 1:\ f_1(\mathbf{P}) = 1 - \frac{\sum_{k-1}^{K} \mathbb{I}(\hat{y}_k = y_k)}{K} \tag{16}$$

where $f_1(\mathbf{P})$ is an objective widely employed by evolutionary fuzzy systems for optimisation [49]. It directly measures the classification performance of the emerging learned model in terms of how well it fits the training samples. By minimising $f_1(\mathbf{P})$, a set of optimal prototypes that maximise the classification accuracy of SOFIS can be obtained. Note that although a strong classifier is expected to be able to correctly classify all the trainings samples, an optimal solution of $f_1(\mathbf{P})$ may lack the required generalisation ability and perform poorly on unseen samples.

From the data partitioning perspective, however, the optimality of SOFIS can be attained by minimising the following objective function [28]:

$$Obj.\ 2:\ f_2(\mathbf{P}) = \sum_{c=1}^{C}\sum_{j=1}^{P_c}\sum_{k=1}^{K_c} \omega_{c,j,k} ||\boldsymbol{p}_{c,j} - \boldsymbol{x}_{c,k}||^2 \tag{17}$$

where $\mathbf{P} = \{\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_C\}$ is the set of prototypes; and there is ($c = 1, 2, \ldots, C$):

$$\omega = \begin{cases} 1, & if\ \boldsymbol{p}_{c,j} = \underset{\boldsymbol{p}\in\mathbf{P}_c}{\arg\min}(||\boldsymbol{p} - \boldsymbol{x}_{c,k}||^2) \\ 0, & else \end{cases} \tag{18}$$

with $f_2(\mathbf{P})$ measuring the intra-cluster variance in the results of data partitioning. It indicates how well data samples of similar characteristics have been grouped together while data samples of different characteristics have been separated. Thus, a set of optimal prototypes that provide the best depiction of the underlying local patterns and multimodal structure of the training data can be obtained via minimising $f_2(\mathbf{P})$. Nevertheless, $f_2(\mathbf{P})$ is not a direct indicator of the classification performance of the learned model. In other words, achieving an optimal solution of Obj. 2 does not necessarily mean that the classifier will produce the most accurate classification outcomes.

The problem is therefore, to develop an evolutionary optimisation method that jointly maximise the accuracy and minimise the intra-cluster variance of an emerging learned model. The main reason for this study to only consider the optimality of the premise parts (namely, prototypes) of the IF-THEN rules is because the vast majority of zero-order EISs [9], [23], [50] identify prototypes from data samples of different classes separately to tackle the potential class overlap and class imbalance. As a result, each prototype can only belong to one particular class. During the optimisation process, the positions of such prototypes in the data space are updated, leading to the changes of classes they belong to potentially. Although the consequent parts, namely, the class labels of these prototypes are fixed during optimisation, EMO consistently evaluates the impacts of such changes on the classification performance of the system and only selects those solutions with a greater fitness to generate the new population at each iteration, thereby creating better solutions after iteration. Whilst it is possible to optimise the consequent and premise parts of IF-THEN rules together, the performance improvement would be trivial and yet, the computational overheads will be greater. In addition, without imposing any constraints in advance, EMO may also change the class labels of the prototypes identified from minor classes to the major class, bringing extra bias to the model. Therefore, only the prototypes are optimised in this study. However, for zero-order EISs whose prototypes can belong to multiple classes at the same time (e.g., [51]), optimising both the prototypes (premise parameters) and their class labels (consequent parameters) together is needed in order to maximise the performance improvement after optimisation.

B. Optimisation Method

In general, there are four components needed to consider when applying an evolutionary multi-objective optimisation (EMO) algorithm to an application problem: 1) problem representation, i.e., how to represent the problem in a way that the evolutionary algorithm can understand and operate on; 2) initialisation, i.e., how to generate the initial evolutionary population; 3) solution generation, i.e., how to create new solutions; and 4) population maintenance (aka. environmental selection or archiving), i.e., how to select solutions to form the next-generation population. Most EMO algorithms (including those mentioned in Section II-B) are different only on the fourth component [52]; the rest three are flexible from modelling viewpoint, they are up to the user to devise or choose from certain standard procedures. These components are described below, by following the common practices in EMO except for the initialisation where domain heuristic is introduced to help accelerate the search.

1) Problem Representation: SOFIS identifies a set of prototypes from training data during the learning process. A complete set of prototypes represents a full solution to the given classification problem albeit it may not be optimal. That is, a solution consists of all prototypes being concatenated together that jointly form an $L \times 1$ dimensional vector:

$$\boldsymbol{P} = [\boldsymbol{P}_1^T, \boldsymbol{P}_2^T, ..., \boldsymbol{P}_C^T]^T \tag{19}$$

where $\boldsymbol{P}_c^T = [\boldsymbol{p}_{c,1}^T, \boldsymbol{p}_{c,2}^T, ..., \boldsymbol{p}_{c,P_c}^T]$, $c = 1, 2, ..., C$, and $L = M\sum_{c=1}^{C} P_c$.

2) Initialisation: Usually in EMO, without any prior knowledge of the problem domain, an initial population can be generated randomly. However, any intermediate solution learned by SOFIS can be used as a heuristic to form the initial population. For convenience, the first learned solution is taken as the first solution of the initial population, namely, $\boldsymbol{s}_1 \leftarrow \boldsymbol{P}$. Then, the remaining $N-1$ solutions are created from $\boldsymbol{s}_1$ using Eq. (20) ($k = 2, 3, \ldots, N$):

$$\boldsymbol{s}_k = \boldsymbol{P} + \boldsymbol{\epsilon}_k \circ \left[\overbrace{\boldsymbol{\delta}^T, \boldsymbol{\delta}^T, ..., \boldsymbol{\delta}^T}^{\sum_{c=1}^{C} P_c}\right]^T \tag{20}$$

where "$\circ$" denotes Hadamard product; $\boldsymbol{\delta} = [(x_{1,max} - x_{1,min}), (x_{2,max} - x_{2,min}), ..., (x_{M,max} - x_{M,min})]^T$ is an $M \times 1$ dimensional vector, defining a hypercube in the problem space to confine the randomly generated prototypes; $x_{k,max} = \max_{i=1,2,...,K}(x_{i,k})$; $x_{k,min} = \min_{i=1,2,...,K}(x_{i,k})$; and $\boldsymbol{\epsilon}_k$ is an $L \times 1$ dimensional vector whose elements follow the uniform distribution with a value range of $[-1, 1]$.

3) Solution Generation: Considering general problems cases where real-valued variables are involved, the following two most widely used variation operators in EMO are herein used to generate new solutions: simulated binary crossover (SBX) and polynomial mutation (PM) [53]. SBX is a real-parameter crossover operator which borrows the idea of the single-point crossover in binary coding, utilising a distribution index to control the spread of the offspring. As with SBX, PM also uses a distribution index parameter to control the spread of its offspring. Following the practice in the literature [36], the two distribution indices are set to 20. Also following the common practice, a crossover probability 1.0 and a mutation probability $1/L$ (where L is the number of variables) are presumed.

4) Population Maintenance: This is one of the most important components in EMO and is essentially where most EMO algorithms differ [52]. When new solutions are generated, it compares them with the solutions in the old population and decides on which solutions should be preserved in the new population according to their fitness. In theory, any population maintenance method may be employed. Here, the most frequently used population maintenance mechanism NSGA-II [36] is chosen to facilitate multi-objective optimisation. The key rationale for the choice of this golden oldie (invented two decades ago) is that if NSGA-II can help improve the classification performance, then it can be envisaged that modern EMO algorithms would help even more.

NSGA-II is characterised by two operations, non-dominated sorting and crowding distance [36]. Non-dominated sorting divides candidate solutions (i.e., old population and newly-generated solutions) into a number of layers according to their Pareto dominance relation. Within each layer, all solutions are non-dominated by each other. Then, NSGA-II places those solutions in the superior layers into the new population and identifies the so-called critical layer whose solutions are more than the slots remaining in the new population. For solutions contained within this critical layer, NSGA-II estimates their density by crowding distance, i.e., the average distance of a solution to its left and right neighbours regarding each objective. Finally, NSGA-II selects less crowded solutions in the layer to fill the remaining slots in the new population. Note that each new solution can be transformed back to a set of prototypes for SOFIS to build its fuzzy rule base and perform classification, through a simple reverse operation. The algorithmic procedure of NSGA-II is summarised [36] and given in the Supplementary Material.

It is worth noting that although the optimisation process implemented by the proposed EMO approach has to be performed offline with static historical data, it does not impair the online learning capability of zero-order EISs. In fact, the optimisation process can be triggered at any point of the online learning process of an EIS. The optimised system by EMO on historical training data can continue learning from new observations in a "one pass" manner and self-improve with newly acquired knowledge as normal. The system can even be optimised for multiple times during its entire learning process if sufficient computational resources are provided. Nevertheless, without loss of generality, EMO is only executed at the end of the learning process of SOFIS in this study.

C. Multi-Objective Decision Making

At the end of the optimisation process, EMO returns a set of solutions that are typically non-dominated by each other. An immediate question is therefore, which solution should be used for classifying unseen data samples. Unfortunately, there is no clear answer to this question since both objectives (Eqs. (16) and (17)) measure the fitness of the solutions based on the training data under the assumption that the unseen testing data are generated from the same distribution. Yet, this assumption is not always true in practice, and there is no way to identify the best solution in the absence of prior knowledge.

To address this problem, five decision-making schemes are introduced below, each of which may be taken to choose the solution from the set of solutions returned.

- Scheme 1 is to choose the non-dominated solution with the lowest training error, namely, the best solution in terms of Obj. 1.
- Scheme 2 is to choose the non-dominated solution with the lowest intra-cluster variance, namely, the best solution in terms of Obj. 2.
- Scheme 3 is to consider all the non-dominated solutions obtained for joint decision-making. That is, the class label of a testing sample, $\boldsymbol{x}$ is determined by Eq. (21):

$$\hat{y} = c^*;\ c^* = \underset{c=1,2,...,C}{\arg\max} \left(\prod_{\mathbf{P} \in \mathbf{F}_1} \lambda_{\mathbf{P},c}(\boldsymbol{x}) \right) \tag{21}$$

  where $\lambda_{\mathbf{P},c}(\boldsymbol{x})$ is the c^{th} score of confidence produced by SOFIS with the solution $\mathbf{P}$; $\mathbf{F}_1$ denotes the first Pareto front.
- Scheme 4 is to use the existing Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) [54] method, which is a well-known technique in multi-objective decision making to choose the solution.
- Scheme 5 is to use another existing, well-known technique in multi-objective decision making, termed Analytical Hierarchy Process (AHP) [55] to choose the solution.

Note that in multi-objective optimisation, certain decision makers may prefer the adoption of extreme solutions (i.e., solutions with the best values on one specific objective) [56], [57]. The first two schemes above can be viewed as such an approach, choosing the best solution on each objective. This is a commonly used strategy in the literature (e.g., [58], [59]). Since the output of an EMO algorithm is a set of non-dominated solutions, information contained within all of such potential solutions can be exploited following the idea of ensemble learning, which is what Scheme 3 does. TOPSIS and AHP are widely used techniques in the field of multi-objective decision-making. TOPSIS is based on the concept that the chosen solution is expected to have the shortest geometric distance from the underlying best solution and the longest geometric distance from the worst solution [54]. AHP works by ranking the solutions through pair-wise comparisons in conjunction with a set of weights which represent the relative importance of objectives [55]. In this study, both Obj. 1 and Obj. 2 are equally weighted according to the common practice under the circumstance where the relative importance of these objectives cannot be quantified.

IV. Computational Complexity Analysis

The computational complexity of SOFIS and EMO is analysed in this section.

For SOFIS, the complexity of calculating the multimodal density values, D^{MM} at the unique data samples of the c^{th} class, $\mathbf{U}_c$ by Eq. (5) is $O(MU_c)$, and that of converting $\mathbf{U}_c$ to the sorted set $\mathbf{R}_c$ by Eq. (6) is $O(\frac{MU_c(U_c-1)}{2})$. The computational complexity of identifying raw prototypes using Condition 1 is $O(U_c)$, and the complexity of forming micro-clusters around these raw prototypes by Eq. (8) is $O(MU_c^*K_c)$. The complexity of estimating the data-driven threshold, $\gamma_{c,G}$ by Eq. (10) is $O(MK_c^2)$, and the complexity for identifying these more representative micro-clusters by Condition 2 is $O(U_c^*)$. Finally, the computational complexity of forming Voronoi tessellation by Eq. (14), extracting prototypes, $\mathbf{P}_c$ and constructing the IF-THEN rule, $\mathfrak{R}_c$ is $O(MP_cK_c)$. Considering that the dataset is composed of data samples of C different classes, the overall complexity of the learning process of SOFIS is $O(M\sum_{c=1}^{C}K_c^2)$.

For EMO, the computational complexity of population initialisation by Eq. (20) is $O(MNL)$, where $L=\sum_{c=1}^{C}P_c$; N denotes the size of the population used in the EMO algorithm. The complexity of generating new solutions via SBX and PM at each optimisation iteration is $O(MNL)$ as well. Given a particular solution, the computational complexity of measuring its fitness based on Obj. 1 is $O(MKL)$ and that for Obj. 2 is $O(M\sum_{c=1}^{C}P_cK_c)$. The fitness assignment in NSGA-II (i.e., nondominated sorting and the calculation of crowding distance) requires $O(mN^2)$, where m denotes the number of objectives and $m=2$ in this study. Therefore, the computational complexity of an iteration in the EMO algorithm is $O(N^2)$ or $O(MNKL)$, whichever is greater. Therefore, the overall complexity of the optimisation process implemented by EMO is $O(MNKLI)$ if $MKL \geq N$, or $O(N^2I)$ otherwise, where I is the maximum iteration number.

V. Experimental Investigation

Systematic experimental studies based on widely used benchmark datasets are presented in this section to demonstrate the efficacy of the proposed optimisation approach, with SOFIS serving as the implementation platform.

A. Configuration

1) Datasets used: In this work, a total of 20 popular numerical datasets are employed for experimental investigation, including: *(1)* abalone (AB); *(2)* cardiotocography (CA); *(3)* epileptic seizure recognition (ES); *(4)* German credit (GC); *(5)* gesture phase segmentation (GP); *(6)* image segmentation (IS); *(7)* letter recognition (LR); *(8)* mammography (MA); *(9)* multiple features (MF); *(10)* MAGIC gamma telescope (MG); *(11)* occupancy detection (OD); *(12)* optical recognition of handwritten digits (OR); *(13)* page-blocks (PB); *(14)* pen-based recognition of handwritten digits (PR); *(15)* phishing websites (PW); *(16)* shill bidding (SB); *(17)* seismic (SE); *(18)* semeion handwritten digit (SH); *(19)* texture (TE); and *(20)* wilt (WI). Furthermore, four remote sensing image sets for land-use classification are used to evaluate the proposed optimisation approach on high-dimensional problems, namely, *(1)* OPTIMAL-31 (OPT); *(2)* RSSCN7 (RSS); *(3)* UCMerced (UCM) and *(4)* WHU-RS19 (WHU). Key information of these two groups of datasets is summarised in Supplementary Tables S1 and S2, respectively, with web links to these datasets given in Supplementary Table S3.

2) Numerical classification problems: Regarding experiments on numerical benchmark datasets, for the IS, OD, OR, PR, and WI datasets, the original training-testing splits are used. For the other 15 datasets, namely, AB, CA, ES, GC, GP, LR, MA, MF, MG, PB, PW, SB, SE, SH and TE, 50% of the data samples are randomly selected for building the training sets and the rest for testing. The classification performance of the proposed approach is evaluated on the 20 datasets in terms of classification error (Err), namely, Obj. 1. Considering that some of the datasets are imbalanced, namely, classes are not distributed equally, balanced classification accuracy (Bac) is also employed as the performance measure for better evaluation [60].

3) Remote sensing image classification problems: In the experiments on the four image datasets, three mainstream DCNNs (namely, ResNet50 [61], DenseNet121 [62] and InceptionV3 [63]) are utilised for feature extraction after fine-tuning on the NWPU45 dataset, following the same procedure as described in [64]. Each fine-tuned DCNNs extracts 1024×1 dimensional high-level feature vectors per remote sensing image, and the resulting feature vectors are combined together into a more descriptive 1024×1 dimensional representation by arithmetic mean (over the three sets of vectors). Following the common practice in the literature, the training-testing split ratio of OPT is set to $8:2$; and for the RSS, UCM and WHU datasets, two different split ratios are considered for each, i.e., $2:8$ and $5:5$, $5:5$ and $8:2$, and $4:6$ and $6:6$, respectively.

4) State-of-the-art (SOTA) methods for comparison: In this work, the following eight SOTA single-model classification algorithms are taken for comparison: *(1)* Zero-order autonomous

learning multi-model classifier (ALMMo0) [50]; *(2)* eClass0 classifier [9]; *(3)* Support vector machine (SVM) [4]; *(4)* k-nearest neighbour (KNN) classifier [5]; *(5)* Decision tree (DT) [6]; *(6)* Multilayer perceptron (MLP); *(7)* Sequence classifier (SEQ) [65], and *(8)* Sequence-dictionary-based KNN (SDKNN) classifier [65].

Furthermore, the following six ensemble models are also used for performance comparison: *(9)* random forest (RF) [7]; *(10)* fuzzily weighted AdaBoost-based SOFIS ensemble (FWADBSOFIS) [30]; *(11)* AdaBoost.M2-based KNN ensemble (ADBKNN) [66]; *(12)* SAMME-based KNN ensemble (SAMKNN) [67]; *(13)* AdaBoost.M2-based SVM ensemble (ADBSVM) [66], and; *(14)* SAMME-based SVM ensemble (SAMSVM) [67].

5) Parameter settings: In running the experiments, ALMMo0, eClass0, and SEQ classifiers use the recommended parameter settings given in [9], [50], and [65], respectively. SVM uses the linear kernel with the box constraint set to be 1. For KNN and SDKNN the parameter k is set to 10. The maximum split of DT is set to be 50. MLP has three hidden layers with 20 neurons per layer.

For the ensemble classifiers compared, the number of base classifiers employed in each is set to 20 uniformly, to have a fair comparison. FWADBSOFIS follows the exact same parameter settings as [30]. The base classifiers (namely, DT, KNN and SVM) of the other five ensemble models retains the aforementioned experimental settings.

Regarding the multi-objective optimisation algorithm introduced in this work, the population size is set to 100 and the number of generations is set to 1000. The algorithm is implemented on the MATLAB2020b platform, and the performance evaluation is conducted on a desktop with dual core i7 processor 2.60GHz×2 and 32.0GB. All numerical results reported in this section are obtained after 25 Monte Carlo simulations.

B. Results

To verify the effectiveness of the multi-objective optimisation approach the following 10 datasets are used: CA, IS, MF, OD, OR, PB, PR, PW, SH and WI. The 10 datasets covers classification problems of various characteristics, including some challenging ones, such as large scale, class imbalance, high dimensionality, non-linear separability, etc., and hence, they are suitable for performance demonstration.

Firstly, SOFIS learns from the training set to extract a set of prototypes. Then, the multi-objective optimisation algorithm (as presented in Section III-B) is used to optimise the learned prototypes, by minimising the two objectives (Obj. 1 and Obj. 2) iteratively until the maximum iteration number is reached. During the experiments, the level of granularity, G varies from 3 to 6.

The overall average Err rates on the testing samples over the 10 datasets resulted from running the optimised SOFIS in conjunction with each of the five different decision-making schemes are reported in Table II, with the average Err rates of running the original SOFIS itself also given as the baseline for contrasting. For further verification, single-objective optimisation (SOO) is also run, following the same experimental protocols to compare against the multi-objective optimisation (MOO) approach. For fairness, an evolutionary algorithm is used to implement SOO with the same specifications on problem representation, initialisation and solution generation. In other words, only the component population maintenance procedure is different from the multi-objective optimisation approach. SOO optimises the two objectives separately, and the obtained best solutions are denoted by best Obj. 1 or best Obj. 2 in Table II. For clarity, the average performance improvements (in percentage) after optimisation using MOO or SOO are compared with those achieved the baseline, as shown in Table III.

Note that in Tables II and III, for MOO, Best Obj. 1 means the choice of the solution with the least Err. Best Obj. 2 means that with the least intra-cluster variance, Joint means that after considering the ensemble of all the solutions obtained, and AHP and TOPSIS mean the choices made with running the corresponding popular decision making methods. For SOO, Best Obj. 1 means the best solution obtained when only optimising the objective Err and Best Obj. 2 means the best solutions obtained when only optimising the objective intra-cluster variance.

The average training errors of SOFIS after optimisation by MOO or SOO approaches are reported in Supplementary Table S4, where the non-dominated solutions of best Obj. 1 and Obj. 2 obtained with MOO are considered. The average performance improvements (in percentage) over the baseline are given in the same table. Further detailed results obtained are given by Supplementary Tables S5 and S6. The average number of prototypes identified by SOFIS with different levels of granularity on the 10 benchmark problems are depicted in Fig. 3.

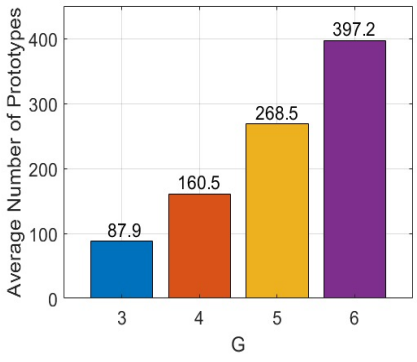


Fig. 3: Average number of prototypes identified by SOFIS at different levels of granularity.

It can be seen from Tables II and III that the MOO approach greatly improves the performance of SOFIS over the 10 benchmark problems. Generally, the performance improvement is greater when the knowledge base of SOFIS contains less prototypes. For example, after being optimised by MOO, given $G = 3$ the performance of SOFIS with the best non-dominated solution of Obj. 1 is improved by 63.59% over the baseline, whilst the performance is improved by 9.18% if $G = 6$. This is because the data space can only be partitioned coarsely by a smaller amount of prototypes, and optimising these prototypes

TABLE II
PERFORMANCE COMPARISON BETWEEN MULTI-OBJECTIVE OPTIMISATION (MOO) WITH DIFFERENT DIFFERENT DECISION MAKING SCHEMES AND SINGLE-OBJECTIVE OPTIMISATION (SOO).

G	MOO					SOO		Baseline
	Best Obj. 1	Best Obj. 2	Joint	TOPSIS	AHP	Best Obj. 1	Best Obj. 2	
3	0.1123	0.1261	0.1179	0.1149	0.1184	0.1290	0.1421	0.1544
4	0.1009	0.1104	0.1052	0.1022	0.1051	0.1211	0.1227	0.1248
5	0.0960	0.1004	0.0975	0.0969	0.0981	0.1144	0.1084	0.1030
6	0.0932	0.0971	0.0947	0.0936	0.0947	0.0950	0.1059	0.0986

TABLE III
PERFORMANCE IMPROVEMENTS USING MULTI-OBJECTIVE OPTIMISATION (MOO) WITH DIFFERENT DECISION MAKING SCHEMES AGAINST USING SINGLE-OBJECTIVE OPTIMISATION (SOO).

G	MOO					SOO	
	Best Obj. 1	Best Obj. 2	Joint	TOPSIS	AHP	Best Obj. 1	Best Obj. 2
3	+63.59%	+32.06%	+51.68%	+58.91%	+52.57%	+40.06%	+9.06%
4	+37.67%	+20.03%	+30.54%	+34.78%	+31.74%	+18.02%	+2.84%
5	+17.59%	+9.83%	+14.95%	+15.67%	+14.74%	+3.00%	-2.29%
6	+9.18%	+3.20%	+6.84%	+8.57%	+7.19%	+4.63%	-5.95%

TABLE IV
PERFORMANCE COMPARISON AMONGST DIFFERENT CLASSIFIERS

Algorithm	Err		Bac	
	Average	Rank	Average	Rank
MOOSOFIS	0.1240	2	0.8040	2
SOFIS	0.1328	7	0.8028	3
ALMMo0	0.1531	10	0.7880	6
eClass0	0.3295	16	0.6626	16
SVM	0.2155	15	0.7404	15
KNN	0.1260	4	0.7844	7
DT	0.1723	12	0.7680	11
MLP	0.1733	13	0.7434	14
SEQ	0.1483	8	0.7933	5
SDKNN	0.1750	14	0.7482	13
RF	0.1254	3	0.8074	1
FWADBSOFIS	0.1209	1	0.8018	4
ADBKNN	0.1316	6	0.7817	9
SAMKNN	0.1284	5	0.7840	8
ADBSVM	0.1535	11	0.7660	12
SAMSVM	0.1496	9	0.7743	10

can cause greater changes on the classification boundaries, resulting in greater performance improvement.

Comparing amongst the five decision-making schemes, it is revealed that in general, Scheme 1 demonstrates the best classification performance over the 10 benchmark problems, whilst Scheme 2 performs the worst. The main reason for this is that Obj. 2 measures the optimality of data partitioning, not the classification performance. Hence, the best solution of Obj. 2 may not be the best option to be used for classifying testing samples. However, this is not universal. As shown by other results given in Supplementary Table S5, depending on the nature of data, Scheme 2 can occasionally outperform Scheme 1 on some benchmark problems, e.g., regarding OR and PR. In addition, Schemes 4 and 5 outperform Schemes 2 and 3, ranking the second and third places amongst the five decision-making schemes in terms of minimising classification errors. Based on these observations, Scheme 1 is recommended for decision-making as it offers the best classification performance overall.

Importantly, Tables II and III also demonstrate that the SOFIS optimised by MMO (MOOSOFIS) outperforms the SOFIS optimised by SSO (SOOSOFIS) on benchmark problems with various experimental settings, thanks to the two different but complementary objectives used for optimisation. Furthermore, SOOSOFIS performs better using Obj. 1 (namely, classification accuracy), which is in line with the observation that Scheme 1 generally offers better performance than Scheme 2. Therefore, it is experimentally confirmed that the MOO approach can effectively help prototype-based classifiers to achieve better performance, by minimising the two objectives simultaneously. This conclusion also coincides with the recent studies [38], [56], [57] that, even when there is only one objective of concern, considering a certain other objective in the evolutionary optimisation process (and hence, multi-objectivisation) can help significantly improve the result. The incomparability between solutions (i.e., Pareto non-dominated to each other) in multi-objective optimisation can help the search jump out of the local optima of the primary objective (thus having a high chance to find the global optimum).

Another interesting observation made from Tables II, III and Fig. 3 is that regardless of the schemes used for decision-making, the Err rates of SOFIS, MOOSOFIS and SOOSOFIS can be reduced by increasing the level of granularity, G. This is because that with a higher level of granularity, SOFIS is able to construct finer and more precise classification boundaries by identifying more prototypes from data. After the iterative optimisation process implemented by MOO or SOO, the optimised SOFIS can achieve even greater classification performance. On the other hand, as aforementioned, optimising a larger knowledge base (namely, a greater amount of prototypes) is more computationally expensive and brings smaller performance improvement. Thus, a trade-off between

TABLE V
PERFORMANCE COMPARISON ON REMOTE SENSING LAND-USE PROBLEMS

Dataset	OPT	RSS		UCM		WHR	
	8:2	2:8	5:5	5:5	8:2	4:6	6:4
MOOSOFIS	0.0011 (0.0019)	0.1092 (0.0065)	0.0843 (0.0068)	0.0402 (0.0050)	0.0307 (0.0076)	0.0370 (0.0080)	0.0251 (0.0101)
CaffeNet [71]	-	0.1443 (0.0095)	0.1174 (0.0062)	0.0602 (0.0067)	0.0498 (0.0081)	0.0489 (0.0120)	0.0376 (0.0056)
VGG-VD-16 [71]	-	0.1602 (0.0087)	0.1282 (0.0094)	0.0586 (0.0069)	0.0479 (0.0120)	0.0456 (0.0060)	0.0395 (0.0091)
GoogLeNet [71]	-	0.1745 (0.0111)	0.1416 (0.0092)	0.0630 (0.0060)	0.0569 (0.0089)	0.0688 (0.0082)	0.0529 (0.0133)
SalM3LBP-CLM [72]	-	-	-	0.0579 (0.0075)	0.0425 (0.0080)	0.0465 (0.0076)	0.0362 (0.0082)
ARCNet-VGG16 [73]	0.0730 (0.0035)	-	-	0.0319 (0.0014)	0.0088 (0.0040)	0.0250 (0.0049)	0.0025 (0.0025)
GBNet [69]	0.0672 (0.0027)	-	-	0.0268 (0.0032)	0.0075 (0.0050)	0.0295 (0.0019)	0.0143 (0.0048)
EfficientNet-B3-Basic [70]	0.0524 (0.0026)	0.0794 (0.0039)	0.0561 (0.0010)	0.0237 (0.0006)	0.0127 (0.0020)	0.0272 (0.0024)	0.0232 (0.0010)
EfficientNet-B3-Attn-2 [70]	0.0414 (0.0022)	0.0670 (0.0019)	0.0383 (0.0023)	0.0210 (0.0036)	0.0079 (0.0022)	0.0140 (0.0040)	0.0132 (0.0093)

the computational overheads of the optimisation process and the classification performance of the optimised model has to be made when setting the level of granularity. In this study, the recommended value for the level of granularity is empirically set to be $G = 9$. However, a globally optimal setting does not exist theoretically whilst an empirically set parameter may perform differently from problem to problem, depending on the nature of data.

C. Further Performance Assessment

For better evaluation, the classification performance of MOOSOFIS attained by the proposed approach is compared with 14 aforementioned single-model and ensemble classifiers on all 20 numerical benchmark problems listed in Section V.A. The level of granularity for SOFIS is set to 9, following the recommended setting. The average results over the 20 datasets, in terms of Err and Bac, obtained by MOOSOFIS, SOFIS and the 14 competitors are presented in Table IV, and the ranks per measure are also given in the same table. More detailed results can be found in Supplementary Table S7 and S8. It can be seen that MOOSOFIS is ranked the second amongst the 16 classifiers on both performance measures. Such a performance surpasses all single-model classification systems, outperforming all the ensemble classifiers on at least one performance measure. This example demonstrates the strong performance achieved by MOOSOFIS, reflecting the efficacy of the proposed approach.

To reveal the statistical significance of the superior performance achieved by MOOSOFIS, over the other 15 single-model and ensemble classifiers, pairwise Wilcoxon rank tests [68] are conducted. The outcomes of the pairwise tests in terms of p-value are reported in Supplementary Table S9, where the cascaded classification results by each classification approach across the 25 Monte-Carlo experiments are used. It can be observed that 88.67% of the p-values returned by the pairwise Wilcoxon tests are below the level of significance specified by $\alpha = 0.05$. This suggests that the performance of MOOSOFIS is significantly better than the others.

Finally, experiments on the four real-world remote sensing image sets for land-use classification are conducted, in an effort to evaluate the effectiveness of the proposed optimisation approach on high-dimensional problems. The results (again, in terms of Err) on these four problems are reported in Table V. For comparison, the results obtained by the relevant SOTA approaches in the literature are given in the same table. It can be seen that MOOSOFIS is able to produce highly accurate predictions on the land-use categories of testing images over all four datasets. Although the employed feature descriptors in the present experiments are not fine-tuned on any of these datasets, the performance of MOOSOFIS is on par with that attainable by the best performing DNN models, including GB-Net [69], EfficientNet-B3-Basic [70], EfficientNet-B3-Basic [70]. This, once again, demonstrates the strong performance of MOOSOFIS.

VI. CONCLUSION AND FURTHER RESEARCH

This paper has presented a multi-objective optimisation approach for optimising prototype-based fuzzy classification systems. It significantly improves the performance of the underlying prototype-based classifier, by iteratively minimising the two different but complementary objectives (namely, accuracy and intra-cluster variance) simultaneously, especially when the model size is compact. The systematic experimental studies have shown that implemented with SOFIS as the base classifier, the approach can outperform a wide range of (14) single-model and ensemble competitors. Whilst the present work is implemented with SOFIS as its basis, there is no reason why other prototype-based classifiers cannot gain similar benefits from the approach, given that SOFIS is simply taken as an example owing to its relative popularity.

In spite of the success as evaluated by the experimental investigations, some limitations remain with the proposed approach that can benefit from further improvement. Firstly, this

study employs the Pareto-based EMO algorithm NSGA-II as the backbone, but there exist other classes of EMO algorithms that have been more recently developed, e.g., indicator-based algorithm SMS-EMOA [42], decomposition-based algorithm NSGA-III [44], and bi-criterion-based algorithm BCE [48]. Utilising a more advanced algorithm or designing a customised method based on the problem's characteristics can be expected to further improve the performance.

Secondly, this study only considers two measures in the optimisation process. One measures classification error on the training samples and the other measures the intra-cluster variance of the training data partitioning. It is clearly shown by numerical results that minimising these objectives together can improve the classification performance greatly. Nevertheless, there are other measures such as the correntropy loss [74] and F1 score [75]. Their employment either as an alternative or as additional joint criterion may produce further improved outcomes. In addition, objectives evaluating the interpretability of the model can also be considered [33].

Lastly, another question not being addressed sufficiently by the present work is how to select the best non-dominated solution for classification. it would be very useful to devise a novel decision making method to choose a more suitable solution to deploy. A possible way is to incorporate certain domain knowledge (e.g., relative importance of different objectives) of the problem in the decision making mechanism.

REFERENCES

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. Crambridge, MA: MIT Press, 2016.

[2] D. Leite, I. Skrjanc, and F. Gomide, "An overview on evolving systems and learning from stream data," *Evol. Syst.*, vol. 11, no. 2, pp. 181–198, 2020.

[3] I. Skrjanc, J. Iglesias, A. Sanchis, D. Leite, E. Lughofer, and F. Gomide, "Evolving fuzzy and neuro-fuzzy approaches in clustering, regression, identification, and classification: a survey," *Inf. Sci. (Ny).*, vol. 490, pp. 344–368, 2019.

[4] N. Cristianini and J. Shawe-Taylor, *An introduction to support vector machines and other kernel-based learning methods*. Cambridge: Cambridge University Press, 2000.

[5] P. Cunningham and S. Delany, "K-nearest neighbour classifiers," *Mult. Classif. Syst.*, vol. 34, pp. 1–17, 2007.

[6] J. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, 1986.

[7] L. Breiman, "Random forests," *Mach. Learn. Proc.*, vol. 45, no. 1, pp. 5–32, 2001.

[8] T. Kohonen, "The self-organizing map," *Neurocomputing*, vol. 21, no. 1–3, pp. 1–6, 1998.

[9] P. Angelov and X. Zhou, "Evolving fuzzy-rule based classifiers from data streams," *IEEE Trans. Fuzzy Syst.*, vol. 16, no. 6, pp. 1462–1474, 2008.

[10] Z. Zhou and J. Feng, "Deep forest: towards an alternative to deep neural networks," in *International Joint Conference on Artificial Intelligence*, 2017, pp. 3553–3559.

[11] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nat. Mach. Intell.*, vol. 1, no. 5, pp. 206–215, 2019.

[12] D. Chen, Q. Yang, J. Liu, and Z. Zeng, "Selective prototype-based learning on concept-drifting data streams," *Inf. Sci. (Ny).*, vol. 516, pp. 20–32, 2020.

[13] H. Rong, N. Sundararajan, G. Huang, and P. Saratchandran, "Sequential adaptive fuzzy inference system (SAFIS) for nonlinear system identification and prediction," *Fuzzy Sets Syst.*, vol. 157, no. 9, pp. 1260–1275, 2006.

[14] H. Huang, H. Rong, Z. Yang, and C. Vong, "Jointly evolving and compressing fuzzy system for feature reduction and classification," *Inf. Sci. (Ny).*, vol. 579, pp. 218–230, 2021.

[15] Z. Yang, H. Rong, P. Angelov, and Z. Yang, "Statistically evolving fuzzy inference system for non-Gaussian noises," *IEEE Trans. Fuzzy Syst.*, DOI: 10.1109/TFUZZ.2021.3090898, 2021.

[16] N. Kasabov and Q. Song, "DENFIS: dynamic evolving neural-fuzzy inference system and its application for time-series prediction," *IEEE Trans. Fuzzy Syst.*, vol. 10, no. 2, pp. 144–154, 2002.

[17] P. Angelov and D. Filev, "An approach to online identification of Takagi-Sugeno fuzzy models," *IEEE Trans. Syst. Man, Cybern. - Part B Cybern.*, vol. 34, no. 1, pp. 484–498, 2004.

[18] P. Angelov, E. Lughofer, and X. Zhou, "Evolving fuzzy classifiers using different model architectures," *Fuzzy Sets Syst.*, vol. 159, no. 23, pp. 3160–3182, 2008.

[19] E. Lughofer, "FLEXFIS: a robust incremental learning approach for evolving Takagi-Sugeno fuzzy models," *IEEE Trans. Fuzzy Syst.*, vol. 16, no. 6, pp. 1393–1410, 2008.

[20] M. Pratama, S. Anavatti, P. Angelov, and E. Lughofer, "PANFIS: a novel incremental learning machine," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 25, no. 1, pp. 55–68, 2014.

[21] D. Kangin, P. Angelov, and J. A. Iglesias, "Autonomously evolving classifier TEDAClass," *Inf. Sci. (Ny).*, vol. 366, pp. 1–11, 2016.

[22] L. Maciel, R. Ballini, and F. Gomide, "Evolving possibilistic fuzzy modeling for realized volatility forecasting with jumps," *IEEE Trans. Fuzzy Syst.*, vol. 25, no. 2, pp. 302–314, 2017.

[23] X. Gu and P. Angelov, "Self-organising fuzzy logic classifier," *Inf. Sci. (Ny).*, vol. 447, pp. 36–51, 2018.

[24] H. Rong, Z. Yang, and P. Wong,"Robust and noise-insensitive recursive maximum correntropy-based evolving fuzzy system," *IEEE Trans. Fuzzy Syst.*, vol. 28, no. 9, pp. 2277–2284, 2019.

[25] A. Lemos, W. Caminhas, and F. Gomide, "Adaptive fault detection and diagnosis using an evolving fuzzy classifier," *Inf. Sci. (Ny).*, vol. 220, pp. 64–85, 2013.

[26] E. Lughofer, E. Weigl, W. Heidl, C. Eitzinger, and T. Radauer, "Integrating new classes on the fly in evolving fuzzy classifier designs and their application in visual inspection," *Appl. Soft Comput.*, vol. 35, pp. 558–582, 2015.

[27] G. Andonovski, O. Sipele, J. A. Iglesias, A. Sanchis, E. Lughofer, and I. Skrjanc, "Detection of driver maneuvers using evolving fuzzy cloud-based system," in *IEEE Symposium Series on Computational Intelligence*, 2021, pp. 700–706.

[28] X. Gu, P. Angelov, and H. Rong, "Local optimality of self-organising neuro-fuzzy inference systems," *Inf. Sci. (Ny).*, vol. 503, pp. 351–380, 2019.

[29] M. Pratama, W. Pedrycz, and E. Lughofer,"Evolving ensemble fuzzy classifier," *IEEE Trans. Fuzzy Syst.*, vol. 26, no. 5, pp. 2552–2567, 2018.

[30] X. Gu and P. Angelov, "Multi-class fuzzily weighted adaptive boosting-based self-organising fuzzy inference ensemble systems for classification," *IEEE Trans. Fuzzy Syst.*, DOI: 10.1109/TFUZZ.2021.3126116, 2021.

[31] E. Lughofer, M. Pratama, and I. Skrjanc, "Online bagging of evolving fuzzy systems," *Inf. Sci. (Ny).*, vol. 570, pp. 16–33, 2021.

[32] X. Gu, Q. Shen, and P. Angelov, "Particle swarm optimized autonomous learning fuzzy system," *IEEE Trans. Cybern.*, vol. 51, no. 11, pp. 5352–5363, 2021.

[33] A. Fernandez, F. Herrera, O. Cordon, M. Jose Del Jesus, and F. Marcelloni, "Evolutionary fuzzy systems for explainable artificial intelligence: why, when, what for, and where to?," *IEEE Comput. Intell. Mag.*, vol. 14, no. 1, pp. 69–81, 2019.

[34] B. Li, J. Li, K. Tang, and X. Yao, "Many-objective evolutionary algorithms: a survey," *ACM Comput. Surv.*, vol. 48, no.1, pp. 1-35, 2015.

[35] A. Okabe, B. Boots, K. Sugihara, and S. Chiu, *Spatial tessellations*: *concepts and applications of Voronoi diagrams*, 2nd ed. Chichester, England: John Wiley & Sons., 1999.

[36] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, 2002.

[37] K. Miettinen, "Nonlinear Multiobjective Optimization," *Kluwer Academic Publishers*, Boston.

[38] T. Chen and M. Li, "The weights can be harmful: Pareto search versus weighted search in multi-objective search-based software engineering," *ACM Trans. Softw. Eng. Methodol.*, DOI:10.1145/3514233, 2022.

[39] M. Emmerich and A. Deutz, "A tutorial on multiobjective optimization: fundamentals and evolutionary methods," *Nat. Comput.*, vol. 17, no. 3 pp. 585-609, 2018.

[40] E. Zitzler, M. Laumanns, and L. Thiele,"SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization," in *Evolutionary Methods for Design, Optimisation and Control*, 2002, pp. 95–100.

[41] E. Zitzler and S. Künzli, "Indicator-based selection in multiobjective search," In *International Conference on Parallel Problem Solving from Nature*, 832–842, 2004.

[42] N. Beume, B. Naujoks, and Emmerich, "SMS-EMOA: multiobjective selection based on dominated hypervolume," *Eur. J. Operat. Res.*, vol. 181, no. 3, 1653–1669, 2007.

[43] Q. Zhang and H. Li. "MOEA/D: a multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 66, pp. 712–731, 2007.

[44] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference point-based nondominated sorting approach, part I: solving problems with box constraints," *IEEE Trans. Evol. Comput.*n, vol. 18, no. 4, pp. 577–601, 2014.

[45] H. Ishibuchi, N. Tsukamoto, and Y. Nojima, "Evolutionary many-objective optimization: a short review," In *IEEE Congress on Evolutionary Computation*, pp. 2419–2426, 2008.

[46] M. Li, S. Yang, X. Liu, and R. Shen, "A comparative study on evolutionary algorithms for many-objective optimization," In *International Conference on Evolutionary Multi-Criterion Optimization*, pp. 261-275, 2013.

[47] J. Falcón-Cardona, and C. Coello, "Indicator-based multi-objective evolutionary algorithms: a comprehensive survey," *ACM Comput. Surv.*, vol. 53, no. 2, pp. 1–35, 2020.

[48] M. Li, S. Yang, and X. Liu, "Pareto or non-Pareto: bi-criterion evolution in multi-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 20, no. 5, pp. 645-665, 2016.

[49] M. Fazzolari, R. Alcala, Y. Nojima, H. Ishibuchi, and F. Herrera, "A review of the application of multiobjective evolutionary fuzzy systems: current status and further directions," *IEEE Trans. Fuzzy Syst.*, vol. 21, no. 1, pp. 45–65, 2013.

[50] P. Angelov and X. Gu, "Autonomous learning multi-model classifier of 0-order (ALMMo-0)," in *IEEE Conference on Evolving and Adaptive Intelligent Systems*, 2017, pp. 1–7.

[51] X. Gu, P. Angelov, and Q. Shen, "Self-organizing fuzzy belief inference system for Classification," *IEEE Trans. Fuzzy Syst.*, DOI: 10.1109/TFUZZ.2022.3179148, 2022.

[52] M. Li, "Is our archiving reliable? multiobjective archiving methods on "simple" artificial input sequences," *ACM Trans. Evol. Learn. Optim.*, vol. 1, no. 3, pp. 1-19, 2021.

[53] K. Deb, *Multi-objective optimization using evolutionary algorithms*, John Wiley, New York, 2001.

[54] C. Hwang, Y. Lai, and T. Liu, "A new approach for multiple objective decision making," *Comput. Oper. Res.*, vol. 20, no. 8, pp. 889–899, 1993.

[55] E. Forman and S. Gass, "The analytic hierarchy process - an exposition," *Oper. Res.*, vol. 49, no. 4, pp. 469–486, 2001.

[56] M. Li, T. Chen, and X. Yao, "How to evaluate solutions in Pareto-based search-based software engineering: a critical review and methodological guidance," *IEEE Trans. Softw. Eng.*, vol. 48, no. 5, pp. 1771–1799, 2022.

[57] X. Ma, Z. Huang, X. Li, Y. Qi, L. Wang, and Z. Zhu, "Multiobjectivization of single-objective optimization in evolutionary computation: a survey," *IEEE Trans. Cybern.*, DOI: 10.1109/TCYB.2021.3120788, 2021.

[58] M. Fazzolari, R. Alcala, and F. Herrera, "A multi-objective evolutionary method for learning granularities based on fuzzy discretization to improve the accuracy-complexity trade-off of fuzzy rule-based classification systems: D-MOFARC algorithm," *Appl. Soft Comput.*, vol. 24, pp. 470–481, 2014.

[59] M. Rey, M. Galende, M.Fuente, and G. Sainz-Palmero, "Multi-objective based fuzzy rule based systems (FRBSs) for trade-off improvement in accuracy and interpretability: a rule relevance point of view.," *Knowledge-Based Syst.*, vol. 127, pp. 67–84, 2017.

[60] K. Brodersen, C. Ong, K. Stephan, and J. Buhmann, "The balanced accuracy and its posterior distribution," in *International Conference on Pattern Recognition*, 2010, pp. 3121–3124.

[61] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[62] G. Huang, Z. Liu, L. van der Maaten, and K. Weinberger, "Densely connected convolutional networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4700–4708.

[63] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.

[64] X. Gu, C. Zhang, Q. Shen, J. Han, P. Angelov, and P.Atkinson, "A self-training hierarchical prototype-based ensemble framework for remote sensing scene classification," *Inf. Fusion*, vol. 80, pp. 179–204, 2022.

[65] R. Patro, S. Subudhi, P. Biswal, and F. Dell'Acqua, "Dictionary-based classifiers for exploiting feature sequence information and their application to hyperspectral remotely sensed data," *Int. J. Remote Sens.*, vol. 40, no. 13, pp. 4996–5024, 2019.

[66] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 119–139, 1997.

[67] J. Zhu, H. Zou, S. Rosset, and T. Hastie, "Multi-class AdaBoost," *Stat. Interface*, vol. 2, no. 3, pp. 349–360, 2009.

[68] F. Wilcoxon, "Individual comparisons of grouped data by ranking methods," *J. Econ. Entomol.*, vol. 39, no. 6, pp. 269–270, 1946.

[69] H. Sun, S. Li, X. Zheng, and X. Lu, "Remote sensing scene classification by gated bidirectional network," *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 1, pp. 82–96, 2020.

[70] H. Alhichri, A. Alswayed, Y. Bazi, N. Ammour, and N. Alajlan, "Classification of remote sensing images using EfficientNet-B3 CNN model with attention," *IEEE Access*, vol. 9, pp. 14078–14094, 2021.

[71] G. Xia et al., "AID: a benchmark dataset for performance evaluation of aerial scene classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 55, no. 7, pp. 3965–3981, 2017.

[72] X. Bian, C. Chen, L. Tian, and Q. Du, "Fusing local and global features for high-resolution scene classification," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 10, no. 6, pp. 2889–2901, 2017.

[73] Q. Wang, S. Liu, and J. Chanussot, "Scene classification with recurrent attention of VHR remote sensing images," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 2, pp. 1155–1167, 2019.

[74] R. Pokharel and J. Príncipe, "Kernel classifier with correntropy loss," in *International Joint Conference on Neural Networks*, 2012, pp. 10–15.

[75] M. Sokolova, N. Japkowicz, and S. Szpakowicz, "Beyond accuracy, F-score and ROC: a family of discriminant measures for performance evaluation," in *Australian Joint Conference on Artificial Intelligence*, 2006, pp. 1015–1021