# Constructing ANFIS with Sparse Data through Group-Based Rule Interpolation: An Evolutionary Approach

Jing Yang, Changjing Shang, Ying Li, Fangyi Li, Liang Shen and Qiang Shen

*Abstract*—**Adaptive-Network-based Fuzzy Inference System, ANFIS, offers a popular and powerful fuzzy inference mechanism. As with many other advanced data driven techniques, developing an effective ANFIS typically requires sufficient training data. However, in many real-world applications it is not always straightforward to obtain a large amount of representative data that covers the entire problem space to accomplish the required training, seriously restricting the performance of a learned ANFIS. This paper introduces a new ANFIS learning approach through an evolutionary process, which is able to generate an ANFIS with only a small number of training data in a certain problem region, by interpolating well trained ANFISs in the neighbouring regions. Such a process works by firstly producing an initial population of candidate fuzzy rules in the region of data shortage, through interpolating a rule dictionary constructed from trained ANFISs in the neighbourhood regions. The crossover and mutation operations over these candidate rules are then executed in an effort to attain candidates of improved performance. When this genetic learning process terminates the chromosomes in the final population either collectively form or each individually represent a learned ANFIS, depending on whether a single fuzzy rule or a set of fuzzy rules representing an entire ANFIS is implemented with a chromosome within the evolving population. Comparative experimental evaluations on both synthetic and real world datasets are carried out, demonstrating that in spite of data shortage, the proposed interpolation approach is able to produce ANFIS models that significantly outperform those trained using existing learning mechanisms.**

*Index Terms*—**ANFIS, data shortage, fuzzy rule interpolation, ANFIS interpolation, rule dictionary, genetic algorithms.**

## I. Introduction

Fuzzy rule based inference systems have achieved significant successes in performing many real world tasks, including classification, regression and prediction [1], [2], [3]. Along with the development of fuzzy systems different rule generation strategies have been proposed in the literature (e.g., [4], [5], [6], [7]). Among them is the means to construct ANFIS [8] (Adaptive Network based Fuzzy Inference System) which implements fuzzy inference within an adaptive network. ANFIS is one of the most effective and popular fuzzy systems, whilst being structurally simple it is powerful for approximately modelling highly non-linear problems [9]. The classical learning procedure for building ANFISs works via extracting useful knowledge in terms of a set of fuzzy rules directly from training data. As such, normally a large amount of training data is required to generate an effective ANFIS. Unfortunately, for a range of real applications, especially those involving novel problems (say, intelligence data analysis [10] and network security modelling [11]), it is difficult or even impossible to obtain sufficient data in the problem domain to execute the required learning procedure. The shortage of training data significantly degrades the performance and potential of ANFISs.

A number of approaches have been proposed to address this challenging practical issue. One is through transfer learning [12], which works with the assistance of models learned from other relevant domain regions where sufficient data is available, instead of directly learning from data of the problem region concerned [13], [14], [15], [16]. For easy cross referencing, the problem region is hereafter interchangeably termed target domain (TD), while each of the relevant regions is referred to as a source domain (SD). Generally speaking, transfer learning aims to transfer the knowledge accumulated from data in SDs, to support predictive modelling in a certain TD. It first constructs non-linear mappings between the target and the source domains and then, transfers model parameters from the SDs into those in the TD using such mappings. An alternative approach is to utilise fuzzy rule interpolation (FRI) techniques (e.g., [17], [18], [19], [20], [21]), enabling fuzzy inference to be performed with an incomplete or sparse rule base, where a given observation may not match any of the existing rules. FRI works by interpolating a new rule for the unmatched observation using neighbouring rules in the rule base. For instance, each member of the popular family of scale and move transformation based FRI approaches (e.g., [22], [23], [24], [25], [26], [27]) derives an interpolated rule by selecting and transforming rules that are the closest to a given unfired observation, obtaining an inferred result in response to the observation.

Inspired by the aforementioned techniques, a new approach for ANFIS learning is proposed in this paper to cope with the problem of training data shortage, termed ANFIS interpolation. It aims to construct a new ANFIS in the TD (where only limited training data is available) from two well trained ANFISs in the neighbouring SDs (where sufficient data is

J. Yang and F. Li are with School of Computer Science and Engineering, Northwestern Polytechnical University, China and Department of Computer Science, Aberystwyth University, UK. (e-mail: {jiy6, fal2}@aber.ac.uk).

C. Shang and Q. Shen are with Department of Computer Science, Aberystwyth University, UK. (e-mail: {cns, qqs}@aber.ac.uk).

Y. Li is with School of Computer Science and Engineering, Northwestern Polytechnical University, China. (e-mail: lybyp@nwpu.edu.cn).

L. Shen is with School of Information Engineering, Fujian Business University, China. (e-mail: c6c123@gmail.com).

This work is partly supported by the Strategic Partner Acceleration Award (80761-AU201), funded under the Sêr Cymru II programme, UK.

available for training relevant ANFISs).

The approach integrates the general ideas of FRI and transfer learning, while possessing a number of specific characteristics: 1) Unlike traditional FRI approaches, which interpolate one single rule to perform inference for an unfired observation, a group of fuzzy rules are collectively interpolated covering the entire TD. 2) Instead of using Mamdani type of rules [28] for knowledge representation, which almost all existing FRI techniques (other than those reported in [29], [30], [31]) take, Takagi-Sugeno-Kang (TSK) type of rules [32] are utilised. 3) The proposed approach handles situations that are rather different from those dealt with by traditional FRI techniques, where training data (or more generally the rules) regarding the actual problem domain (namely, the TD) are extremely sparse, but there are sufficient data (or rules) in the neighbouring domains (or the SDs). That is, the knowledge about the TD is so poor that traditional FRI methods simply fail to work (with few rules available to carry out interpolation). Combining these characteristics makes it a very difficult task to interpolate a set of accurate rules in an effort to construct a required ANFIS. The proposal is to interpolate a number of candidate fuzzy rules first, forming an initial population which is then modified via an evolutionary optimisation process subsequently. Of course, if there were a fair amount of training data available in the TD, then a traditional FRI that works for TSK models would be sufficient to perform interpolation; that is, there would not be a need for the separation of SDs from the TD in the first place.

In general, evolutionary computation [33], [34] offers a range of optimisation algorithms by analogy to natural evolution processes. Whilst implemented in a stochastic manner, such algorithms perform a highly effective search in the problem hyperspace, efficiently directing the solution to promising regions. Typical evolutionary algorithms such as genetic algorithms (GAs) [35], genetic programming [36] and particle swarm optimizers [37], have been widely employed for a variety of theoretical and practical applications [38], [39]. Applying an evolutionary computation method to aid in building a fuzzy inference system injects learning capacity into the underlying fuzzy systems, which is commonly referred to as evolutionary fuzzy systems in the literature (e.g., [40], [41], [42], [43]). Considering genetic algorithms being the most popular technique for use in developing such systems, the present work utilises a GA to implement ANFIS interpolation.

One of the key points of the proposed work is therefore, to examine how an ANFIS may be encoded using a certain chromosome representation. Without prejudgement, two alternative forms of representing fuzzy rules are considered in this paper, using either individual rule based representation or group of rules (equivalently an entire ANFIS) based representation to construct an initial population for evolving. The initial population is iteratively updated through crossover and mutation operations, subject to the use of a fitness function, in order to determine whether a chromosome may enter into the next loop. Then, chromosomes of the highest fitness, namely rules or ANFISs with the best performance, will be returned when the iterative process terminates. If chromosomes represent individual rules, they collectively form the required ANFIS; otherwise, each returned chromosome is a learned ANFIS. Such a resulting ANFIS will be fine tuned to obtain the final improved ANFIS model, appropriate to deal with approximate inference in the TD concerned.

The remainder of this paper is organised as follows. Section II reviews the relevant background of ANFIS, FRI and GAs for academic completeness. Section III details the proposed evolutionary ANFIS interpolation approach. Experimental results are discussed in Section IV. Finally, Section V concludes the paper and points out identified further research.

## II. Background

Relevant background work is introduced in this section, including an illustration of ANFIS, an overview of FRI and an outline of genetic algorithm. Whilst basic concepts are covered for academic completeness, technical details regarding these topics are beyond the scope of this paper.

### *A. ANFIS*

ANFIS [8] stands for a type of fuzzy inference system that is implemented within the framework of an adaptive network. It provides an efficient way for dealing with highly nonlinear problems and has been widely applied [44]. The information content of an ANFIS includes the network structure and the corresponding parameter learning mechanism. There are totally five layers in a general ANFIS structure. An example of two-input and one-output system is illustrated here, with each input variable of the system described by two fuzzy sets. In this case, the system's rule base contains 4 fuzzy if-then rules of Takagi and Sugeno's type [45], as expressed below:

Rule 1: If $x_1$ is $A_1$ and $x_2$ is $A_3$, then $y_1 = p_1x_1 + q_1x_2 + r_1$
Rule 2: If $x_1$ is $A_1$ and $x_2$ is $A_4$, then $y_2 = p_2x_1 + q_2x_2 + r_2$
Rule 3: If $x_1$ is $A_2$ and $x_2$ is $A_3$, then $y_3 = p_3x_1 + q_3x_2 + r_3$
Rule 4: If $x_1$ is $A_2$ and $x_2$ is $A_4$, then $y_4 = p_4x_1 + q_4x_2 + r_4$

Structurally, such an ANFIS can be illustrated as shown in Fig. 1, where square nodes in the first and fourth layer are the adaptive nodes with modifiable parameters, and circle nodes in the remaining layers involve fixed operations without parameters. In the first layer, each square node is defined with a membership function $\mu_{A_i}(x)$, where $i \in \{1, \cdots, 4\}$, $x$ is the input variable and $A_i$ is a fuzzy set defining an imprecise value of $x$. The most popularly applied function is triangular shaped due to its simplicity, which is defined by

$$\mu_A(x) = \begin{cases} k_1x + b_1 & a_0 \le x \le a_1 \\ k_2x + b_2 & a_1 \le x \le a_2 \\ 0 & otherwise \end{cases} \tag{1}$$

where $k_1 = 1/(a_1 - a_0)$, $b_1 = -a_0/(a_1 - a_0)$, $k_2 = 1/(a_1 - a_2)$, and $b_2 = -a_2/(a_1 - a_2)$ are named premise parameters, with $\{a_0, a_1, a_2\}$ being the three vertexes of the triangular membership function.

The second layer, *Layer 2* multiplies the incoming membership value of each variable and outputs the product: $w_i = \mu_{A_i}(x_1) \times \mu_{A_j}(x_2)$ $(i \in \{1, 2\}, j \in \{3, 4\})$. The output of this layer $w_i$ stands for the firing strength of a certain rule. Then *Layer 3* normalises each rule's firing strength by computing the relative proportion of a given rule's firing strength to the total

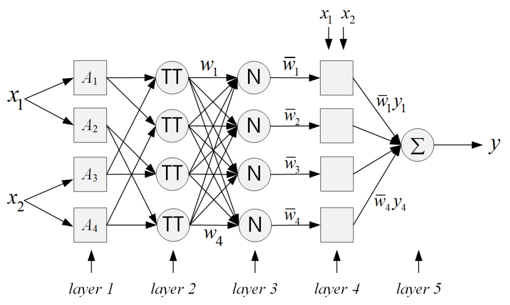

Fig. 1. Architecture of an example ANFIS encoding four rules

of $N$ rules' firing strengths $\bar{w}_i = w_i / \sum_{j=1}^{N} w_j$ (here $N = 4$, $i \in \{1, \cdots, 4\}$). In the fourth layer, each node is a square node with a linear function $\bar{w}_i y_i = \bar{w}_i(p_i x_1 + q_i x_2 + r_i)$, where $p_i, q_i, r_i$ are referred to as consequent parameters. Finally, *Layer 5* is the output layer, computing the overall output in response to the current input, i.e., $y = \sum_i \bar{w}_i y_i$. Each layer within a general ANFIS is summarised in Table I.

TABLE I
FUNCTIONALITY OF INDIVIDUAL LAYERS IN ANFIS

| Layer | Activity | Parameter |
|---|---|---|
| 1 | Fuzzifying crisp value | Premise parameter |
| 2 | Computing firing strength of each rule | No |
| 3 | Normalising firing strength | No |
| 4 | Computing local output | Consequent parameter |
| 5 | Computing global output | No |

Given the network structure, the only uncertain part of an ANFIS concerns the parameters in the first and fourth layers. These parameters are trained using a hybrid learning method combining gradient descent and Least Square Estimation (LSE), which can be divided into a forward pass and a backward pass. In the forward pass, the input values of the training samples are fed forward to compute the output with the premise parameters being fixed, and the consequent parameters are optimised by LSE. Then, in the backward pass, the consequent parameters are set to be fixed, while the error rates between the computed output and the expected ones are propagated backward, with the premise parameters updated using the gradient descent method. Table II summarises the activities in each pass [8].

TABLE II
TWO PASSES OF ANFIS LEARNING PROCEDURE

| — | Forward Pass | Backward Pass |
|---|---|---|
| Premise parameters | Fixed | Gradient descent |
| Consequent parameters | LSE | Fixed |
| Signals | Node outputs | Error rates |

## B. Fuzzy Rule Interpolation

Fuzzy rule interpolation techniques are developed to perform fuzzy inference with an incomplete or sparse rule base. A sparse rule base means that the problem space is not fully covered by the rules, so that certain input observations cannot match any rule and therefore, no conclusion can be drawn using traditional pattern matching methods (e.g., compositional rule of inference [28]). All existing FRI methods make a reasonable assumption: If a given observation is close to (but does not match) the antecedents of certain rules in the sparse rule base, then the conclusion should also be similar to the consequents of those rules.

Without losing generality, for an unmatched observation, a conventional FRI algorithm typically works by: 1) searching for a small number (usually just two [46]) of the rules closest to the observation within the sparse rule base, and 2) interpolating a new rule by averaging the closest rules found. For instance, suppose that an unfired observation $A^*$ and the $K$ closest rules are given as follows:

$$\begin{array}{ll} observation: & X\ is\ A^* \\ closest\ rules: & If\ X\ is\ A^i,\ then\ Y\ is\ B^i \end{array} \tag{2}$$

where $i = 1, 2, \cdots, K$. Then, the interpolated rule is often generated as the weighted average of the closest rules:

$$\begin{array}{ll} interpolated\ rule: & If\ X\ is\ A',\ then\ Y\ is\ B' \\ where & B' = \sum w_i B^i \\ & A' = \sum w_i A^i \end{array} \tag{3}$$

with $\{w_i | i = 1, 2, \cdots, K\}$ being the weights obtained by computing the similarity degrees between $A^i$ and $A^*$, subject to the constraint that $Rep(A') = Rep(A^*)$, where $Rep(A)$ denotes the representative value of the fuzzy set $A$ [20]. Note that the representative value of a triangular fuzzy set as per Eqn. (1) can be defined as $Rep(A) = (a_0 + a_1 + a_2)/3$ [22].

Following the above general idea of FRI, many FRI methods have been proposed in the literature. However, as indicated previously, almost all such methods are based on the use of Mamdani type of fuzzy model, which differs from the underpinning representation of ANFISs that are a dialect of TSK fuzzy models. This significantly justifies the necessity in devising an approach that enables interpolation of ANFISs, as to be presented later.

## C. Genetic Algorithm

Inspired by observing natural evolution processes, evolutionary computation [33], [34] is in general, proposed to provide an effective and efficient way for searching optimal solution in poorly understood and irregular problem spaces. Typically, evolutionary algorithms work with a population of individuals, in which each individual may be one or a set of potential solutions in the solution space. Among various evolutionary algorithms, GAs may be the most commonly adopted and hence, are utilised in this work also.

Mainly owing to the conceptual simplicity and computational effectiveness GAs gain their popularity. In a GA implementation, each individual of the population is encoded as a chromosome, which may be subsequently modified through crossover and mutation subject to a certain probability. A fitness function is utilised to evaluate the performance of every chromosome within a population. The general implementation of a GA can be summarised in the following steps:

(1) Initialize the population;

(2) Perform crossover and mutation on the population;

(3) Calculate the fitness of each chromosome;

(4) Select a portion of chromosomes for a new population;

(5) Loop to step (2) until a certain stop criterion is met.

Applying GAs for fuzzy systems modelling leads to techniques for building a form of evolutionary fuzzy learning systems. Typical approaches involve the development of procedures for: 1) Parameter tuning – By assuming that the structure of a fuzzy inference system is pre-defined, this procedure adapts the system's parameters [47] with respect to changes in the model input. 2) Rule selection – By encoding a rule base as a fixed-length chromosome, this procedure aims to control the complexity of a fuzzy inference system, leaving less room for redundant, incorrect or badly defined rules to exist [48]. 3) Rule base construction – By encoding both the parameters and the structure of a fuzzy system within each chromosome, this procedure performs parameter estimation and structure identification at the same time [49].

## III. GA for ANFIS Interpolation

This section presents the proposed novel ANFIS interpolation approach, which is implemented via a GA. The problem addressed can be outlined as follows: With only a small number of training data in the target domain $T$, expressed in the form of input-output pairs $\{(\mathbf{x}, y)\}$, the process of ANFIS interpolation is to construct an effective ANFIS $\mathcal{A}_{int}$ over $T$, by interpolating two neighbouring ANFISs, $\mathcal{A}_1$ and $\mathcal{A}_2$, defined on two source domains $S_1$ and $S_2$, respectively. Fig. 2 summarises the entire interpolation process, consisting of 3 main stages: i) population initialisation via observation-guided interpolation of rules embedded in the source ANFISs, based on the method of [30]; ii) evolution via a GA over the initial population; and iii) ANFIS fine-tuning via the standard ANFIS learning method as described in Section II-A (using the given small number of training data in the target domain). The entire evolutionary process will of course depend upon how individuals are to be encoded in a population of chromosomes and how each chromosome may be evaluated. Details of this process are described below.

### A. Chromosome Representation: Two Strategies

How to encode a potential solution with a chromosome is a critical point in GAs. To reflect the fact that an ANFIS is essentially a set of fuzzy rules, there are two strategies that may be taken to represent an ANFIS model within a GA: 1) Encode each underlying rule of the ANFIS as a chromosome, and 2) Encode the entire ANFIS as a chromosome. Both types of chromosome are introduced here.

*1) Rule-based Chromosome Representation:* This representation strategy encodes each rule of an ANFIS as a vector of rule parameters, in the form of (antecedent parameters, consequent parameters). For example, suppose that a rule involving $m$ antecedent variables that take triangular fuzzy sets as values is expressed by:

$$\begin{aligned} &If\ x_1\ is\ A_1\ and\ x_2\ is\ A_2 \ldots and\ x_m\ is\ A_m, \\ &then\ y = p_0 + p_1x_1 + p_2x_2 + \cdots + p_mx_m \end{aligned} \tag{4}$$

where each fuzzy set $A_i$ $(i = 1, 2, \cdots, m)$ contains 3 parameters $(a_{i0}, a_{i1}, a_{i2})$ that respectively denote the three vertices of the triangle. Then, the rule based chromosome can be expressed such that

$$\begin{aligned} &((a_{10}, a_{11}, a_{12}), (a_{20}, a_{21}, a_{22}), \cdots, \\ &(a_{m0}, a_{m1}, a_{m2}),\ p_0, p_1, \cdots, p_m) \end{aligned} \tag{5}$$

In general, using triangular fuzzy sets, if a rule contains $m$ antecedent variables, there will be $3m$ premise parameters and $m + 1$ consequent parameters. Thus, the length of the corresponding chromosome will be $3m + m + 1 = 4m + 1$.

*2) ANFIS-based Chromosome Representation:* In this representation, an entire ANFIS containing $C$ rules is encoded as one chromosome. That is, all these rules are collectively encoded as one vector like the following:

$$(rule\ 1, rule\ 2, \cdots, rule\ C) \tag{6}$$

where each single rule is coded exactly the same as Eqn. (5). In so doing, the length of an ANFIS based chromosome will be $C \times (4m + 1)$, which is much longer than that of a rule based one.

### B. Population Initialisation

An initial population, composed of a number of interpolated rules in the TD, needs to be generated first, in order to start the evolutionary process. This is accomplished by adopting the FRI mechanism of [30], over the given (sparse) training data, with the assistance of a rule dictionary constructed by extracting rules from the two source domain ANFISs $\mathcal{A}_1$ and $\mathcal{A}_2$ (as shown in Fig. 2(b)). Details are as follows.

*1) Constructing a rule dictionary:* A rule dictionary is developed by extracting and then storing rules from the given source ANFISs. In particular, the rule dictionary $D$ is designed with an antecedent part $D_a$ and a consequent part $D_c$, to separately store collected rule antecedents and consequents: $D = \{D_a, D_c\}$. Suppose that $\mathcal{A}_1$ and $\mathcal{A}_2$ consist of $n_1$ and $n_2$ rules, respectively, and the extracted rules are expressed in the following format:

$$\begin{aligned} R_i^{\mathcal{A}_t} : if\ x_1\ is\ A_{i1}^{\mathcal{A}_t}\ and \ldots and\ x_m\ is\ A_{im}^{\mathcal{A}_t}, \\ then\ y_i = \sum_{j=0}^{m} p_{ij}^{\mathcal{A}_t} x_j \end{aligned} \tag{7}$$

where $t \in \{1, 2\}$, $m$ is the number of input variables, and $p_{i0}^{\mathcal{A}_t}$ is the coefficient of the constant term ($x_0 = 1$) in a certain rule consequent.

It follows from the above that the antecedent part of the dictionary $D_a \in R^{m \times n}$, consisting of all the rule antecedents:

$$D_a = [d_1^a\ d_2^a\ \cdots\ d_n^a] \tag{8}$$

where each column $d_i^a = [A_{i,1}^{\mathcal{A}_t}\ A_{i,2}^{\mathcal{A}_t} \cdots A_{i,m}^{\mathcal{A}_t}]^T, t \in \{1, 2\}$ contains all the antecedent fuzzy sets of one particular rule, and $n = n_1 + n_2$ denotes the number of the columns. Similarly, the corresponding consequent part of the dictionary $D_c \in R^{(m+1) \times n}$ is:

$$D_c = [d_1^c\ d_2^c\ \cdots\ d_n^c] \tag{9}$$

where each column $d_i^c = [p_{i,0}^{\mathcal{A}_t}\ p_{i,1}^{\mathcal{A}_t}\ p_{i,2}^{\mathcal{A}_t} \cdots p_{i,m}^{\mathcal{A}_t}]^T, t \in \{1, 2\}$.

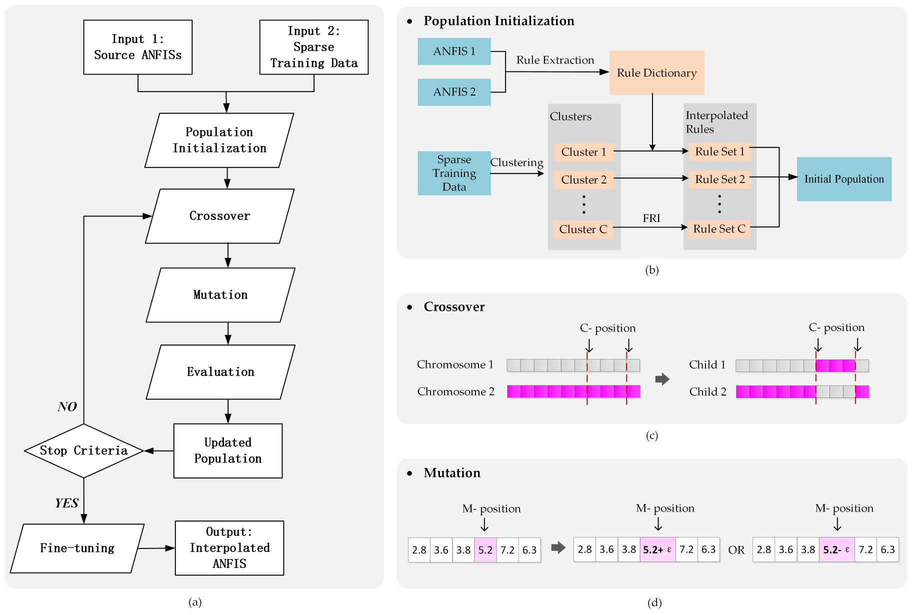

Fig. 2. Flowchart of proposed approach. (a) Overall flowchart. (b) Population initialisation process. (c) Crossover process. (d) Mutation process.

*2) Interpolating candidate rules:* In order to form an initial population, a number of rules in the TD are required. Traditional means for generating the initial population in a GA usually use randomly set parameters in the rules. Whilst this is practical for situations with sufficient training data it becomes a significant challenge for the current case, where only highly restricted sparse data is available. To address this challenge, FRI is employed to interpolate rules in TD, in an effort to set up an improved initial network, so as to be able to produce a more effective model with less training data. Given the above rule dictionary, a number of candidate rules in the TD can indeed be obtained through interpolation by running the following procedure.

To start with, the given training data $\{(\mathbf{x}, y)\}$ in the TD is divided into $C$ clusters. Here, $C$ stands for the number of the rules in the ANFIS to be interpolated and is determined by $C = \prod_{j=1}^{m} \lfloor \frac{n_1^{(j)} + n_2^{(j)}}{2} \rfloor$, where $n_1^{(j)}$ is the number of membership functions in the $j$th variable of $\mathcal{A}_1$, $n_1 = \prod_{j=1}^{m} n_1^{(j)}$, and $n_2 = \prod_{j=1}^{m} n_2^{(j)}$. Clustering is utilised in order to minimise the derivation of any redundant rules, so that similar training data which belong to one cluster will (eventually) only lead to one rule. It is implemented in the simplest 'variable by variable' fashion (unless otherwise stated). That is, clustering is iteratively applied to all training data according to the first variable of the domain (with variables ordered in any preferable order), and then, for each resulting cluster, it is applied again to the data within the current cluster according to the second variable, etc. Following this 'variable by variable' clustering strategy, the resulting clusters will cover all possible regions of the input variables.

Having obtained the clusters, for each cluster, a number of candidate rules are interpolated to create the initial population. However, the number of training data in each cluster may be very different, and in certain clusters there may be just one datum. For extreme cases where the number of training samples is smaller than $C$, certain clusters are simply empty, covering no data at all, though such situations may be rare. Therefore, instead of just utilising the raw training data contained within the clusters, individual instances are also artificially generated in an effort to enrich the original sparse training data. For this, the centre of each cluster is used as the seed to generate more individuals. In particular, for those clusters without any training data, the seeds are set to be the same as their neighbouring clusters (of course, such initially identical settings will become different through the evolutionary process).

For each cluster $C_k$, the centroid is denoted by $c^{(k)} = (c_1, c_2, \cdots, c_m)^T$ with regard to the $m$ attributes. Use $c^{(k)}$ as a seed, a number of artificial data (denoted by $\{O = (o_1, o_2, \cdots, o_m)^T\}$) can be generated by adding Gaussian white noise with the seed itself being the mean and a small value ($\delta$) being the standard deviation for each attribute. This method borrows practical ideas often adopted in the field of electrical engineering. The number of the individual instances in each cluster is set subject to the constraint in which the sum of cluster sizes will be the size of the required initial population. For implementational simplicity, in this work, all clusters are set to be of an equal size. Such an individual generation process can be illustrated as per Fig. 3. From the resulting data enriched clusters, for each individual instance within a given cluster, a candidate rule can be generated through interpolation that involves the following two steps.

Step 1. Selecting closest rules in rule dictionary:

Given the dictionary of the antecedent part $D_a$, closest rule selection is accomplished by computing the Euclidian distance as defined by Eqn. (10) (although other distance metrics may be used as alternatives if preferred), between the current individual $O$ and every column of $D_a$:

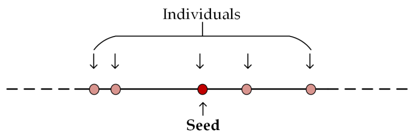

Fig. 3. Individual generation from a seed.

$$d_i = d(d_i^a, O) = \sqrt{\sum_{j=1}^{m} d(A_{ij}^{\mathcal{A}_t}, o_j)^2} \tag{10}$$

where $d(A_{ij}^{\mathcal{A}_t}, o_j) = |Rep(A_{ij}^{\mathcal{A}_t}) - o_j|, t \in \{1,2\}$. Those $K$ columns resulting in the smallest distances are chosen as the closest rules. The index set associated with the $K$ selected columns $\{d_i^a\}$ in $D_a$ is denoted by $\mathcal{K}$. That is, $\forall i \in \mathcal{K}$, $max_i\{d(\{d_i^a\}, O)\} < d(\{d_j^a\}, O), j \notin \mathcal{K}$ and $|\mathcal{K}| = K$.
Step 2. Interpolating rules:

With the obtained closest columns $\{d_i^a;\ i \in \mathcal{K}\}$, the candidate rule for the current individual can be interpolated as the weighted average of the selected rules. This process can be seen as the reconstruction of $O$ using $\{d_i^a\}$. Thus, the weights required for rule interpolation can be obtained by minimising the following reconstruction error:

$$w = \min_{w} ||O - \sum_{i\in\mathcal{K}} Rep(d_i^a) w_i||^2,\ s.t. \sum_{i\in\mathcal{K}} w_i = 1 \tag{11}$$

where $Rep(d_i^a) = [Rep(A_{i,1}^{\mathcal{A}_t})\ Rep(A_{i,2}^{\mathcal{A}_t}) \cdots Rep(A_{i,m}^{\mathcal{A}_t})]^T$, $t \in \{1,2\}$, and $w_i$ denotes the relative weighting of the column $d_i^a$. This is a constraint optimisation problem with solution: $w^{(k)} = (G^{-1}\mathbf{1})/(\mathbf{1}^T G^{-1}\mathbf{1})$, where $G = (O\mathbf{1}^T - X)^T(O\mathbf{1}^T - X)$, $\mathbf{1}$ is a column vector of ones, and the columns of $X$ are the selected $\{\{d_i^a\},\ i \in \mathcal{K}\}$.

Following the principles of FRI which performs reasoning by analogy, the weights $w_i$ derived for the antecedent part are applied to the consequent part to attain similarity. Thus, the newly interpolated rule for the current individual has the following format:

$$\begin{aligned} R: If\ x_1\ is\ A_1\ and \ldots and\ x_m\ is\ A_m, \\ then\ y = \sum_{j=0}^{m} p_j x_j \end{aligned} \tag{12}$$

where the premise and consequent parameters are calculated by Eqn. (13) and Eqn. (14) respectively, with $t \in \{1,2\}$.

$$A_j = \sum_{i\in\mathcal{K}} w_i A_{ij}^{\mathcal{A}_t},\ j = 1, 2, \cdots, m. \tag{13}$$

$$p_j = \sum_{i\in\mathcal{K}} w_i p_{ij}^{\mathcal{A}_t},\ j = 0, 1, 2, \cdots, m. \tag{14}$$

*3) Generating initial population:* By collecting all the interpolated candidate rules, the initial population results. Depending on which of the two chromosome representation strategies is used, there are slightly differences when forming the initial population. Using rule based chromosome representation, all the candidate rules within one cluster form an initial sub-population for this cluster. Therefore, there are $C$ initial sub-populations in total, each of which will initiate an independent evolutionary learning process, as shown in Fig. 4(a). While using ANFIS based chromosome representation, all the candidate rules are collected to form a global initial population, as shown in the Fig. 4(b). Therefore, there is just one initial population in this case.

## C. Crossover and Mutation

For an evolutionary process, from the initial sub-population or the entire population (depending upon which coding style is used for chromosome representation), a subset of chromosomes (of an even cardinality) are randomly selected to perform crossover with a pre-specified crossover probability. The chosen chromosomes are then, randomly paired up. For each (so-called parent) pair $(ch1, ch2)$, the standard 'two-point' crossover is applied, in which a start position and an end position are randomly determined and subsequently, the genes between them are exchanged. In so doing, two child chromosomes are generated per pair. This procedure is illustrated in Fig. 2(c).

After crossover, mutation operation follows. Similar to the crossover operation, a subset of the (sub-)population is selected for mutation with a pre-determined probability. For each chosen chromosome, a mutation position is randomly generated. Then, the mutated chromosome is created by randomly adding or subtracting a small value $\varepsilon$ to the gene at that position. The mutation procedure is illustrated in Fig. 2(d).

Note that as with common approaches in the literature, a larger crossover probability and a smaller mutation probability are empirically assumed.

## D. Fitness Function

Fitness function is used to evaluate the performance of the chromosomes so that better performers can be maintained to enter the next iteration. For prediction and estimation tasks, functions that compute the mean square error or the absolute difference error are the most commonly adopted. In this work, the fitness function is implemented on the basis of the RMSE (Root-Mean-Squared Error), which is defined per chromosome as follows:

$$E = \sqrt{\frac{\sum_{i=1}^{N}(g_k - \mathcal{A}(x_k))^2}{N}} \tag{15}$$

where $N$ is the number of training data, $g_k$ and $\mathcal{A}(x_k)$ are the $k$th expected and estimated output over the evolutionary process, respectively. Suppose that the maximum error permitted is $E_{max}$, then such a fitness function may be specified by

$$F = E_{max} - E \tag{16}$$

For the present application, a meaningful fitness measure should be given for an entire ANFIS instead of a single rule. Thus, further consideration may be due, depending upon which chromosome representation is utilised. For ANFIS based chromosome representation, the fitness of each chromosome can be directly obtained using the fitness function above, as per Eqn. (16). However, if individual rule based chromosome

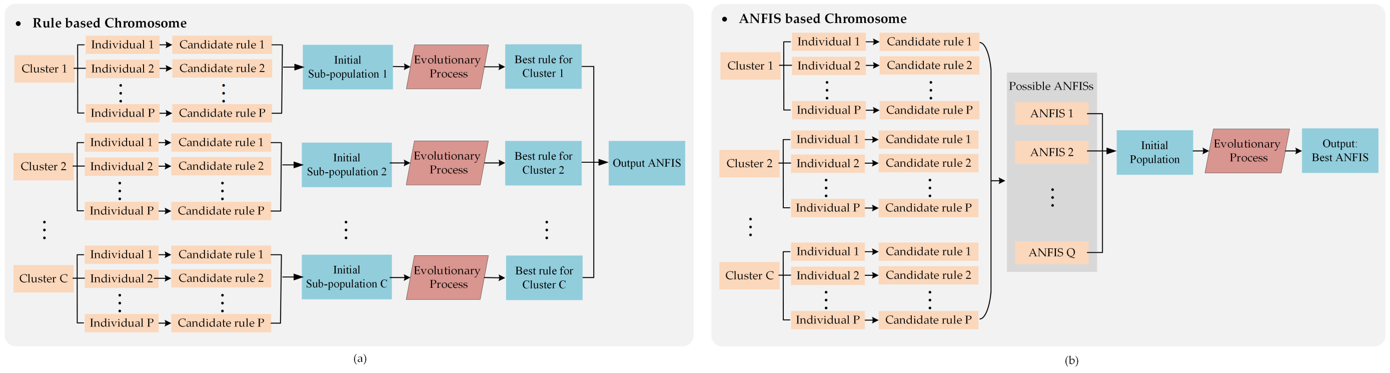

Fig. 4. Type of chromosome representation used in implementation. (a) Rule based chromosome. (b) ANFIS based chromosome.

representation is employed, the fitness of such a chromosome is obtained using a score table that evaluates and records the performance of every candidate rule, as per Table III, where $score_{ij}$ denotes the evaluated (fitness) value of the $j$th candidate rule in the $i$th cluster and $S$ is the size of candidates in each cluster.

TABLE III
SCORE TABLE

| Cluster / Candidate | 1 | 2 | $\cdots$ | C |
|---|---|---|---|---|
| 1 | $score_{11}$ | $score_{21}$ | $\cdots$ | $score_{C1}$ |
| 2 | $score_{12}$ | $score_{22}$ | $\cdots$ | $score_{C2}$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\ddots$ | $\vdots$ |
| S | $score_{1S}$ | $score_{2S}$ | $\cdots$ | $score_{CS}$ |

Note that to capture and reflect the essence of the fitness of a candidate rule, computationally, $score_{ij}$ is calculated as the averaging performance of the $j$th candidate rule in the $i$th cluster across a certain number of randomly generated possible ANFISs. This is necessary because in dealing with practical problems, the number of possible ANFISs can be rather large. For example, suppose that there are 10 rules in the rule base of an ANFIS, and that there are 5 candidates per underlying rule. Then, the number of possible ANFISs will be $5^{10}$, which is a huge figure that requires substantial storage and computation power. Thus, in implementation, just a portion of the possible ANFISs are randomly chosen for evaluation. To balance effectiveness and efficiency, the number of chosen possible ANFISs, denoted by $A_{max}$, may be set empirically (and in this research, $A_{max} = 500$).

The evaluated ANFISs are subsequently sorted according to their RMSEs in descending order. For individual rule-based representation, the score table is initialised with each cell set to zero and, for the $k$th sorted ANFIS ($k \in \{1, 2, \ldots, A_{max}\}$), record the candidate rules used, and the scores of those relevant candidate rules are added by $k$. After doing this for all those created possible ANFISs, the score table results. The procedure for computing the score table is summarised as given in Alg. 1.

**Algorithm 1:** Calculation of Score Table

```
Input:
    Candidate rules;
    C       – Number of clusters;
    S       – Number of candidate rules in each cluster;
    $A_{max}$ – Number of ANFISs for evaluation.
1: Make a zero Score Table of S rows and C columns;
2: Make a zero Counting Table of S rows and C columns;
3: Make $A_{max}$ possible ANFISs randomly and record
   candidate rules used;
4: Evaluate ANFISs and sort them according to RMSE
   in descending order;
5: for $k = 1$ to $k = A_{max}$ do
6:   for $i = 1$ to $i = C$ do
7:     record the candidate rule number $j$ used;
8:     Score Table: $score_{ij} \leftarrow score_{ij} + k$
9:     Counting Table: $count_{ij} \leftarrow count_{ij} + 1$
10:  end for
11: end for
12: for $i = 1$ to $i = C$ do
13:   for $j = 1$ to $j = S$ do
14:     Score Table: $score_{ij} \leftarrow score_{ij}/count_{ij}$
15:   end for
16: end for
Output:
    Score table of all candidate rules
```

## *E. Summary of GA-based ANFIS Interpolation*

The proposed evolutionary ANFIS interpolation algorithms are summarised as Alg. 2.

## *F. ANFIS Interpolation: Contrasting Two Algorithms*

The two algorithms, implemented depending upon which chromosome representation is employed, have already been illustrated in Fig. 4. There exist three major differences between these two implementations: 1) The objects that are evolved during the evolutionary process are different. One using rule based chromosome (termed 'Method 1' hereafter) conducts one evolutionary process per cluster, whilst the other using ANFIS based chromosome (termed 'Method 2') initialises the population directly with ANFISs, conducting the evolutionary

**Algorithm 2:** Evolutionary ANFIS Interpolation

```
Input:
    Source ANFISs in SDs: $\mathcal{A}_1, \mathcal{A}_2$
    Sparse training data in TD
    Population size (ANFIS chromosome): $Q$
  - Population Initialization
1: Extract fuzzy rules $\{R_i\}$ from $\mathcal{A}_1$ and $\mathcal{A}_2$;
2: Construct rule dictionary $D$ by Eqn. (8) and (9);
3: Divide sparse training data into $C$ clusters;
4: for each cluster do
5:   Use centre of each cluster as seed, generate
      a set of individuals;
6:   for each individual do
7:     Interpolate one candidate rule;
8:   end for
9: end for
10: if chromosome is rule type then
11:   for each cluster do
12:     Collect candidate rules within current cluster,
         to form an initial sub-population;
13:   end for
14: else if chromosome is ANFIS type then
15:   for $i = 1$ to $i = Q$ do
16:     for each cluster do
17:       Choose one candidate rules in current cluster;
18:     end for
19:     Form an ANFIS chromosome by all chosen rules;
20:   end for
21:   Collect all ANFIS chromosomes as initial population;
22: end if
  - Crossover and Mutation
23: Choose chromosomes for crossover with probability $p_c$;
24: Do two-point crossover for chosen chromosomes;
25: Choose chromosomes for mutation with probability $p_m$;
26: Do mutation for chosen chromosomes;
  - Chromosome Evaluation
27: if chromosome is rule type then
28:   Evaluate chromosomes using Score Table;
29: else if chromosome is ANFIS type then
30:   Evaluate chromosomes using Eqn. (16);
31: end if
32: Update population by roulette selection;
33: Return to 23 until stop criteria is met;
Output:
    Interpolated ANFIS $\mathcal{A}_{int}$ in TD
```

process only once. This is of course, the fundamental reason for all the differences between these two algorithms. 2) The population initialisation methods are different, as discussed previously in Section III-B3. 3) The fitness measurement means are different, as described in Section III-D.

Both algorithms have their own advantages and disadvantages, as summarised below. In particular, the individual rule-based approach requires less storage – Each chromosome only represents one rule, meaning that it is generally much shorter than a chromosome in ANFIS based method. This also leads to less running time, as to be indicated in an illustrative example later. However, the entire ANFIS-based representation offers a more convenient means for evaluation – The process of assessing the quality of chromosomes to decide whether any of them will enter the next iteration can be directly evaluated in this approach, whilst the evaluation step of rule based chromosomes involves a much more complex procedure (using a score table and using an additional parameter of $A_{max}$).

## G. Fine-tuning of GA-learned ANFIS Model

A GA-learned network (through the process as described above) is used as an intermediate network through the standard ANFIS learning algorithm that was introduced in Section II-A. Note that only the TD data is used in the fine-tuning process, while the SD data is used for training the source ANFISs only. Compared with the use of the original standard ANFIS training procedure, which initialises the network parameters as 'zeros' or 'random values', the proposed ANFIS learning mechanism employs both fuzzy rule interpolation and GA-based evolutionary procedure, in an effort to produce higher quality initial network parameters. In so doing, the sparse training data is used more efficiently in the fine-tuning procedure for generating ANFIS of improved performance.

## H. Complexity Analysis

The time complexity of the proposed methods (for both the rule based and ANFIS based chromosome representation) is analysed here. According to Alg. 2, there are generally three steps in performing evolutionary ANFIS interpolation: population initialisation, crossover and mutation, and chromosome evaluation. As indicated previously, triangular fuzzy sets are utilised in implementing both methods (unless otherwise stated) and hence, the complexity analysis only involves the use of such fuzzy sets. The notations used are listed as follows:

| | |
|---|---|
| $m$ | : number of antecedent attributes |
| $n$ | : number of fuzzy rules in rule dictionary |
| $N$ | : number of sparse training data points |
| $C$ | : number of clusters in training data |
| $P$ | : number of individuals in each cluster |
| $Q$ | : population size for ANFIS based chromosomes |
| $A_{max}$ | : number of ANFISs used in calculating score table |

*1) Rule based chromosome representation:* In the 'Population Initialisation' step, lines 1-2 extract the rule parameters from sources ANFISs. There are totally $n$ rules, with $4m + 1$ parameters in each rule. Thus, running lines 1-2 costs $O(4mn + n)$. Following this, line 3 for clustering takes $O(NCm)$. Lines 4-9 repeat $C \times P$ times implementing candidate rule interpolation, each of which includes three sub-steps: (1) selecting $K$ closest rules (at the cost of $O(n^2)$), (2) calculating weights (at $O(K)$) and (3) generating new rules (at $O(4m+1)$). Therefore, the complexity for computing lines 4-9 is $C \times P \times [O(n^2)+O(K)+O(4m+1)] = O(CPn^2)$. Next, lines 10-13 jointly lead to $C$ initial populations, costing $O(C)$. Hence, the sub-total complexity for this step is $O(4mn+n) + O(NCm) + O(CPn^2) + O(C) = O(NCm) + O(CPn^2)$.

In the 'Crossover and Mutation' step, the crossover and mutation operation repeats $C$ times. Line 23 selects $p_cP$ rules for crossover, costing $O(p_cP)$. In line 24, there are $\lfloor (p_cP)/2 \rfloor$ pairs of chromosomes. For each pair, in the worst case, all the parameters within the two chromosomes are exchanged, taking $O(4m+1)$. Then, in performing the mutation operation, lines 25-26 take $O(p_mP)$. Thus, the sub-total complexity is $C \times [O(p_cP) + \lfloor (p_cP)/2 \rfloor \times O(4m+1) + O(p_mP)] = O(mCP)$.

The rule based chromosome uses a score table to implement the 'Chromosome Evaluation' step. In running Alg. 1, lines 1-2 take $O(SC)$. Line 3 makes $A_{max}$ ANFISs with $C$ rules in each, costing $O(A_{max}C)$. Line 4 costs $O(A_{max}^2)$ for sorting the $A_{max}$ RMSEs. Following this, lines 5-11 repeat $A_{max}C$ times, costing $O(A_{max}C)$. Lines 12-16 repeat $SC$ times with a complexity of $O(1)$ each time. Thus, the sub-total complexity for the 'Chromosome Evaluation' step is $O(SC) + O(A_{max}C) + O(A_{max}^2) = O(A_{max}^2)$.

In summary, the overall complexity for the proposed approach with rule based chromosome is: $O(NCm) + O(CPn^2) + O(mCP) + O(A_{max}^2)$.

*2) ANFIS based chromosome representation:* For the method that exploits ANFIS based chromosome representation, the 'Population Initialisation' step is almost the same as that for the method using rule based chromosomes, expect for its final population construction process as given in lines 15-20, which costs $O(QC)$. Thus, running this 'Population Initialisation' step is at the cost of $O(4mn+n) + O(NCm) + O(CPn^2) + O(QC) = O(NCm) + O(CPn^2) + O(QC)$. Different from the method with rule based chromosome representation, here, the crossover and mutation operation are only implemented once within the population of the size $Q$. Line 23 takes $O(p_cQ)$. In line 24 the crossover operation repeats for $\lfloor (p_cQ)/2 \rfloor$ times, and in the worst case, each crossover incurs exchanges across all the $C(4m+1)$ parameters in each ANFIS based chromosome, thereby costing $O(C(4m+1))$. Following this, lines 25-26 take $O(p_mQ)$. The sub-total complexity for the 'Crossover and Mutation' step is therefore, $O(p_cQ) + \lfloor (p_cQ)/2 \rfloor \times O(C(4m+1)) + O(p_mQ) = O(mCQ)$. Finally in the 'Chromosome Evaluation' step, line 30 evaluates all the $Q$ chromosomes by sorting the related RMSEs, costing $O(Q^2)$. Hence, the overall complexity for the method with ANFIS based chromosome representation is: $O(NCm) + O(CPn^2) + O(mCQ) + O(Q^2)$.

TABLE IV
SUMMARY OF COMPLEXITY ANALYSIS

| Main steps | Rule-based chromosome | ANFIS-based chromosome |
|---|---|---|
| Population Initialization | $O(NCm) + O(CPn^2)$ | $O(NCm) + O(CPn^2) + O(QC)$ |
| Crossover and Mutation | $O(mCP)$ | $O(mCQ)$ |
| Chromosome Evaluation | $O(A_{max}^2)$ | $O(Q^2)$ |
| Overall | $O(NCm)+O(CPn^2)+ O(mCP) + O(A_{max}^2)$ | $O(NCm)+O(CPn^2)+ O(mCQ) + O(Q^2)$ |

For clarity, the outcomes of the above computational complexity analysis are summarised in Table IV. Comparing the overall complexity of running the method using rule based chromosome representation and that using ANFIS based one, it can be seen that the first two items ($O(NCm) + O(CPn^2)$) in each are the same. Moreover, in implementations, $A_{max}$ and $Q$ are usually set as the same number because both denote the number of ANFISs to be evaluated. Thus, the only difference between the complexities of these two methods lies in their respective third items: ($O(mCP)$ vs. $O(mCQ)$). From this analysis, it can be concluded that the complexities of the two proposed methods do not differ very much. The time complexity of using the ANFIS based chromosome representation is slightly higher than that of using the rule based one, because that the value of $Q$ is typically larger than that of $P$. This is verified experimentally later.

## IV. EXPERIMENTATION

This section presents a systematic experimental evaluation of the proposed approach. Section IV-A provides the general experimental set-up, including the parameters used, comparative methods employed, and performance index measured. Section IV-B validates the two proposed algorithms by looking into a few synthetic function modelling cases, while Section IV-C shows the effectiveness of the proposed approach in performing TSK regression over ten benchmark datasets.

### *A. Experimental Set-up*

In the experimental studies, triangular membership functions are used in implementing the first layer of an ANFIS due to their popularity and simplicity, unless otherwise stated. Both normalised and unnormalised data are used, reflecting the capability of the proposed approach in dealing with different data representations. Particularly, original data is used without normalisation in the synthetic data experiments for the convenience of result illustration. While in the experiments involving benchmark datasets, all input domains of the original data are normalised to $[0, 1]$. Following the common practice, the crossover and mutation probabilities are chosen as $p_c = 0.8$ and $p_m = 0.2$, respectively, unless otherwise stated. In setting up the initial population, the number of candidate rules of each cluster is set to $P = 5$. Note that $P$ is the number of candidate rules per cluster in the very original population before any crossover and mutation, which differs in principle from the figure $S$ in Table III, that stands for the number of candidate rules in each cluster in the evaluation step.

Regression results using different ANFISs are compared, including: 1) An original ANFIS trained with the classical ANFIS learning method, using the sparse data in the TD only, named as 'Original ANFIS' hereafter; 2) An interpolated ANFIS obtained by the first ANFIS interpolation method [30], named as 'Method 0' (which interpolates one rule with respect to the centre of each cluster without involving evolutionary computation); 3) An interpolated ANFIS obtained by the proposed 'Method 1' (via rule based chromosome representation); and 4) An interpolated ANFIS obtained by the proposed 'Method 2' (via ANFIS based chromosome representation).

RMSE is taken to evaluate the performance of different ANFISs, as per the definition of Eqn. (15), where $N$ now represents the number of the testing data points $\{x_k\}$ in the TD; $g_k$ denotes the corresponding ground truth of the $k$th data point; and $\mathcal{A}(x_k)$ stands for the output of different ANFISs on the data point $x_k$. Obviously, a smaller RMSE indicates a better performance, given otherwise the same conditions while performing statistical analyses.

## B. Experiments on Synthetic Data

In this set of experimental studies, synthetic data is created by sampling each of three non-linear functions, including two 1-D functions (of different complexities) and one 2-D function, the underlying functions used are listed in Table V.

TABLE V
FUNCTIONS USED

| No. | Function | Domain range |
|---|---|---|
| 1 | $cos(x) \cdot x$ | $x \in [-10, 10]$ |
| 2 | $sin(2x)/e^{x/5}$ | $x \in [-10, 10]$ |
| 3 | $sin(x_1/\pi)sin(x_2/\pi)$ | $x_1 \in [-30, 30]$, $x_2 \in [-10, 10]$ |

*1) Illustrative Example:* This is presented in order to show the main working procedures of the proposed ANFIS interpolation approach and the differences between the two types of chromosome representation. The first one dimensional non-linear function [ $y = cos(x) \cdot x$ ] is used for giving the illustrative example, the underlying function is plotted for illustration in Fig. 5. As shown in Fig. 5, the input domain $[-10, 10]$ is divided into three parts to simulate the SDs and the TD. In particular, there are totally 201 data points sampled from this continuous function (with a sampling step of 0.1), in which the data in the left part (67 data points, shown in dashed line) forms the first source domain, which is used for training the first source ANFIS $\mathcal{A}_1$, and the data in the right part (67 data points, shown in dotted line) forms the second source domain for training the corresponding second ANFIS $\mathcal{A}_2$. These two source ANFISs are pre-trained using the standard ANFIS learning algorithm as described in Section II-A, with 4 and 5 fuzzy rules resulted, respectively. The remaining data of the middle part (also 67 data points, shown in real line) forms the target domain, it will be subsequently divided into two sub-parts with a small portion (20%, 13 data points) for training and the rest 80% (54 data points) for testing.

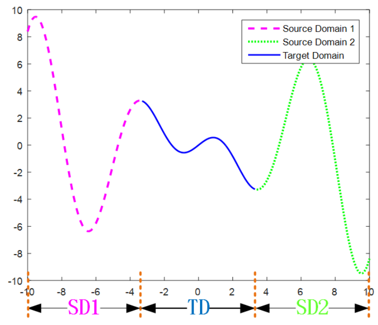

Fig. 5. Illustration of source data and target data.

Following the proposed approach, the fuzzy rules embedded within the two well trained ANFISs $\mathcal{A}_1$ and $\mathcal{A}_2$ are extracted, forming the columns of the rule dictionary, with $4+5 = 9$ rules in total, as shown in Table VI. Note that the rule dictionary is the same for either the rule chromosome based method or the ANFIS chromosome based one.

TABLE VI
RULE DICTIONARY

| Rule | Source ANFIS | Antecedent $(a_0, a_1, a_2)$ | Consequent |
|---|---|---|---|
| 1 | $\mathcal{A}_1$ | $(-12.2, -10.02, -7.83)$ | $14.09x + 149.33$ |
| 2 | $\mathcal{A}_1$ | $(-9.95, -7.82, -5.55)$ | $7.16x + 55.41$ |
| 3 | $\mathcal{A}_1$ | $(-7.79, -5.59, -3.39)$ | $13.83x + 72.83$ |
| 4 | $\mathcal{A}_1$ | $(-5.62, -3.39, -1.2)$ | $10.99x + 40.84$ |
| 5 | $\mathcal{A}_2$ | $(1.75, 3.39, 5.09)$ | $7.16x - 27.67$ |
| 6 | $\mathcal{A}_2$ | $(3.39, 5.05, 6.7)$ | $10.02x - 48.99$ |
| 7 | $\mathcal{A}_2$ | $(5.01, 6.7, 8.33)$ | $6.36x - 36.38$ |
| 8 | $\mathcal{A}_2$ | $(6.71, 8.35, 9.96)$ | $3.30x - 31.56$ |
| 9 | $\mathcal{A}_2$ | $(8.36, 10.01, 11.65)$ | $9.81x - 106.53$ |

Next, the sparse training data (13 data points) is clustered into $C$ clusters, $C = \lfloor(4+5)/2\rfloor = 4$. Details of each cluster are listed in Table VII, from which it can be seen that the number of raw data in each cluster is different, particularly in Cluster 1 there is only one data point. Thus, it can be very difficult to control the size of the initial population while scaling up, if the raw data is directly used. This issue is remedied with artificially generated individual instances, using the cluster centre as the seed by adding Gaussian white noise from it, with the seed itself being the mean and a small number $\delta$ (here $\delta = 0.2$) acting as the standard deviation. For example, the individuals of Cluster 3 are generated, consisting of the cluster centre itself (1.26) and the four randomly generated data points (1.13, 1.37, 0.89 and 1.05). Thus, the number of individuals $P$ in each cluster becomes the same.

TABLE VII
GENERATING INDIVIDUALS FOR EACH CLUSTER ($P = 5$)

| Cluster | Center | Raw data | Generated individuals |
|---|---|---|---|
| 1 | -2.3 | -2.3 | -2.3,-2.43,-2.18,-2.67,-2.5 |
| 2 | -0.62 | 0.2,-1.1,-0.4,-0.6,-1.2 | -0.62,-0.75,-0.5,-0.99,-0.82 |
| 3 | 1.26 | 1.4,0.9,1.5 | 1.26,1.13,1.37,0.89,1.05 |
| 4 | 2.52 | 2.2,3.2,2,2.7 | 2.52,2.38,2.63,2.15,2.31 |

For each generated individual, a candidate rule is interpolated. As such, there are totally $4 \times 5 = 20$ interpolated rules (namely, number of clusters times that of individuals in each cluster), which are subsequently used to construct the initial population. For the method using rule based chromosome representation, the initial population contains 4 initial sub-populations, each with 5 chromosomes. For the method with ANFIS based chromosome representation, one candidate rule in each cluster is chosen and used for constructing possible ANFISs. There are 5 different choices in each cluster, so the number of possible ANFISs is $5^4 = 625$, which is also the size of the initial population. This is a doable number for the current simple illustrative case, however, for more complex problems with tens of clusters, this number will become excessively large and therefore, will require a huge storage space. In order to keep this under control, only a portion of all possible ANFISs are randomly selected as the chromosomes in implementing the initial population. Denote the number of possible ANFISs to take as $Q$, then, as stated previously,

$Q = 500$ is empirically set in the present experimental studies.

In this illustrative example, the length of the individual rule-based chromosome is $(4m+1) = 5$ $(m = 1)$, while that of the entire ANFIS-based chromosome is $5 \times 4 = 20$ (with 4 rules in each ANFIS). From this setup of the initial population, the crossover and mutation operations are followed. The changes incurred to the number of the chromosome during and after one evolutionary iteration is shown in Table VIII. Of course, for rule based chromosome representation, this number is counted within one sub-population. Taking the method running on the entire ANFIS-based chromosome representation as an example, the population is initialised as $Q = 500$ chromosomes at the beginning. Then 400 chromosomes are chosen as parents for crossover with a crossover probability $p_c = 0.8$, resulting in 400 children. After crossover, the number of chromosomes becomes 500+400=900. Similarly, 100 chromosomes are chosen for mutation with a mutation probability $p_m = 0.2$, resulting in 100 new chromosomes, and the number of chromosomes after mutation becomes 900+100=1000. Finally, the evaluation process will select 500 chromosomes out of these 1000 to enter into the next iteration, ensuring that the population size remains the same.

TABLE VIII
NUMBER OF CHROMOSOMES IN ONE (SUB-)POPULATION DURING DIFFERENT PROCESSES WITHIN ONE EVOLUTIONARY ITERATION

| | Initial population | After Crossover | After Mutation | After Evaluation |
|---|---|---|---|---|
| Rule chromosome | 5 | 9 | 10 | 5 |
| ANFIS chromosome | 500 | 900 | 1000 | 500 |

As mentioned previously, the methods for fitness evaluation of the two types of chromosome encoding are different. For the entire ANFIS-based representation, the evaluation is simply done using Eqn. (16) and hence, its illustration is omitted. Only is the evaluation of the individual rule-based chromosomes explained below.

Having produced four sub-populations (each with 10 chromosomes), a score table of size $10 \times 4$ is computed according to Alg. 1. Fig. 6 presents four instances of computed tables. Firstly, the score table is initialised as a zero table of size $10 \times 4$, as illustrated in Table (a) of this figure. Then, 500 $(A_{max})$ possible ANFISs are randomly generated, with the rules used in these ANFISs recoded. The resulting ANFISs are evaluated using Eqn. (15) and are sorted in descending order according to their RMSEs. In this illustrative example, the first ANFIS (with the largest RMSE) is composed of 4 candidate rules indexed by candidates $[3, 10, 6, 4]$. That is, the first rule of this ANFIS is the third candidate rule (or chromosome) in cluster 1, the second rule is the tenth candidate in cluster 2, the third rule is the sixth candidate in cluster 3, and the last rule is the fourth candidate rule in cluster 4. The value in each of the four corresponding locations (or cells) in the score table is therefore, added by 1, resulting in Table (b) of Fig. 6. Next, given the ordered indices $[1, 7, 6, 10]$ of the second ANFIS, the values in their corresponding locations are added by 2, leading to Table (c) of Fig. 6. This process is repeated for all of the 500 ANFISs. Averaging over such 500 tables results in the final score table that consists of the scores or fitness of all candidate rules, as shown in Table (d) of Fig. 6, where the candidate rules with the largest average score in each cluster are highlighted in bold. Of course, here, the scores are the averages of how many times a certain candidate rule is utilised by the randomly generated (500) ANFISs.

| Candidate \ Cluster | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 |
| 9 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 |

(a)

| Candidate \ Cluster | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 |
| 3 | 1 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 1 |
| 5 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 1 | 0 |
| 7 | 0 | 0 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 |
| 9 | 0 | 0 | 0 | 0 |
| 10 | 0 | 1 | 0 | 0 |

(b)

| Candidate \ Cluster | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 2 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 |
| 3 | 1 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 1 |
| 5 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 3 | 0 |
| 7 | 0 | 2 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 |
| 9 | 0 | 0 | 0 | 0 |
| 10 | 0 | 1 | 0 | 2 |

(c)

| Candidate \ Cluster | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 221.3 | 311.1 | 244.2 | 373.5 |
| 2 | 281.6 | 258.1 | 217.5 | 258.1 |
| 3 | 220.2 | 333.2 | 152.7 | **409.0** |
| 4 | 265.9 | 169.3 | 339.1 | 148.3 |
| 5 | 178.2 | 229.6 | 287.3 | 245.0 |
| 6 | 260.0 | 245.1 | 164.3 | 312.1 |
| 7 | 236.1 | 159.7 | **358.2** | 132.9 |
| 8 | **296.1** | 313.2 | 222.2 | 197.2 |
| 9 | 268.9 | **355.1** | 283.2 | 373.7 |
| 10 | 283.5 | 131.2 | 250.1 | 116.1 |

(d)

Fig. 6. Example for score table computation

*2) Accuracy and Runtime Performance:* The experimental results of all three function modelling case studies are presented here. The classical 5×5-fold cross validation is applied in the experiments to statistically evaluate the performance of different ANFISs. The mean and standard deviation values of RMSE using different ANFISs are listed in the first half of Table IX, while the visual result of one randomly selected fold regarding the first function modelling problem is shown in Fig. 7 as an illustrative example.

As can be seen from the results, the 'Original ANFIS' gives the worst outcomes (which is not surprising due to data shortage in the TD). While the existing interpolation-aided approach (of [30]), i.e., 'Method 0' shows an already significantly improved performance over the 'Original ANFIS', the GA-based algorithms produce further enhanced results. Quantitatively, running either 'Method 1' or 'Method 2' leads to smaller RMSE values with a narrower standard deviation, as shown in Table IX. Qualitatively, the shape of the estimated function by either of the proposed methods is much closer to that of the ground truth, as depicted in Fig. 7.

Comparing the two proposed algorithms themselves, the results demonstrate that 'Method 1' (which represents individual rules as chromosomes) performs slightly better than 'Method 2' (which expresses an entire ANFIS as one chromosome), for this function modelling problem. The likely reason for this is that, as the rule based chromosomes are shorter than the ANFIS based ones, there are more opportunities for crossover and mutation operations to take effect under the

TABLE IX
ACCURACY (MEAN ± STANDARD DEVIATION) AND RUNNING TIME (SECONDS) OF DIFFERENT METHODS ON SYNTHETIC DATA

| Methods | Function 1 | Function 2 | Function 3 | Run time |
|---|---|---|---|---|
| Original ANFIS | 0.913 ± 1.184 | 1.329 ± 1.153 | 0.468 ± 0.244 | 0.32 ± 0.21 |
| Method 0 | 0.364 ± 0.406 | 0.905 ± 0.743 | 0.351 ± 0.148 | 0.65 ± 0.26 |
| Method 1 | **0.294** ± 0.401 | **0.691** ± 0.461 | 0.334 ± 0.147 | 40.21 ± 27.33 |
| Method 2 | 0.317 ± 0.441 | 0.794 ± 0.552 | **0.325** ± 0.096 | 55.42 ± 29.70 |
| LR | 1.243 ± **0.055** | 0.772 ± **0.022** | 0.471 ± **0.007** | 3.04 ± 2.52 |
| SVR | 1.234 ± 0.064 | 0.789 ± 0.030 | 0.470 ± **0.007** | 3.44 ± 2.13 |
| CART | 1.302 ± 0.329 | 1.394 ± 0.235 | 0.393 ± 0.037 | 3.32 ± 1.75 |
| RF | 0.547 ± 0.133 | 0.805 ± 0.167 | 0.408 ± 0.024 | 3.61 ± 0.78 |
| E-ANFIS | 1.552 ± 0.478 | 0.836 ± 0.048 | 0.476 ± 0.012 | 359.21 ± 126.42 |

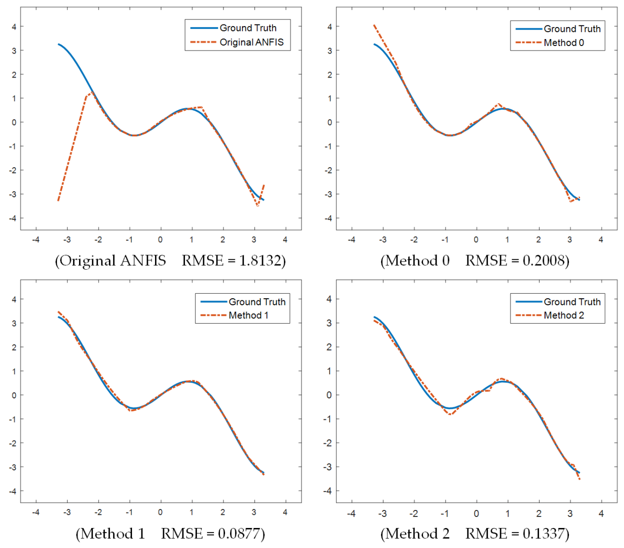

Fig. 7. Visual illustration of one-fold result by different ANFISs

same probabilistic set up.

Both implementations for the proposed ANFIS interpolation approach are also compared with five conventional machine learning methods: linear regression (LR), support vector regression (SVR), classification and regression tree (CART), random forest (RF), and an evolutionary ANFIS method (E-ANFIS [50]) in which the genetic algorithm is used to learn the network parameters (without interpolation). The evaluation results are listed in the second half of Table IX. From the accuracy over the three function modelling case studies, it can be observed that the proposed ANFIS interpolation methods ('Method 1' or 'Method 2') perform better in terms of mean error values. Although the classical machine learning methods (especially the LR approach) may give better standard deviation values, their overall accuracy is lower than what is achieved by the proposed methods.

The average running time (also showing in the 'Mean ± Standard deviation' form) of different methods over the three function modelling cases is shown in the last column of Table IX. Comparing the two proposed methods with other approaches, as can be expected, they run faster than 'E-ANFIS' but incur more computation than the rest. However, the considerably improved accuracy they gain over the other approaches justifies this increase in computational cost. Comparing the two proposed methods themselves, it can be seen that the method employing individual rule-based chromosome representation consumes less time, this conforms to the theoretical analysis presented in Section III-H.

## C. Experiments on Real Data

A number of investigations are carried out here, looking into the issues regarding accuracy, amount of training data and key GA parameters (namely, crossover and mutation rates).

*1) Datasets and Experimental Environment:* Table X lists the ten popular benchmark real-world regression datasets, taken from the KEEL data repository [51], which are used here to further evaluate the performance of both GA-based ANFIS interpolation algorithms.

TABLE X
DATASETS USED IN EXPERIMENTAL STUDY

| **Dataset** | **Attribute No.** | **Instance No.** |
|---|---|---|
| Diabetes | 2 | 43 |
| Plastic | 2 | 1650 |
| Quake | 3 | 2178 |
| Laser | 4 | 993 |
| AutoMPG6 | 5 | 392 |
| Delta-ail | 5 | 7129 |
| Friedman | 5 | 1200 |
| Dee | 6 | 365 |
| Delta-elv | 6 | 9517 |
| ANACALT | 7 | 4052 |

To conduct this set of experimental investigations, the SDs and TD are created in a similar manner to the synthetic data experiments, by splitting each entire dataset into three parts according to one of the input variables. For example, the Quake dataset has 3 input variables ('Longitude', 'Latitude' and 'Depth'). This is divided into three sub-datasets with regard to the values of the variable 'Longitude'. In particular, instances whose 'Longitude' value is smaller than $-40$ form the first SD with 593 instances, those whose 'Longitude' value is between $[-40, 92]$ form the TD with 332 instances, and the remaining $1253$ instances form the second SD. The same as the synthetic data experiments reported previously, only 20% (66 instances) of the data in TD are used for training, with the remaining 80% (266 instances) used for testing.

*2) Accuracy Analysis:* The 5×5-fold cross validation results over the ten datasets are shown in Table XI, with the average RMSE values listed in the last row and the best results indicated in bold. Similar observation can be made from Table XI to those results obtained from the synthetic data experiments. The 'Original ANFIS' gives the poorest performance as expected, and the other three interpolated ANFISs improve the inference results obviously. Amongst the three interpolation-based methods, both GA-based implementations outperform 'Method 0' for most cases, though there is no clear winner between the two themselves.

*3) Comparison with other machine learning methods:* As with the experiments on synthetic data, the proposed approach is also compared with five conventional machine learning methods: LR, SVR, CART, RF, and E-ANFIS. Different from those earlier experiments (where triangular membership functions and a very simple clustering method were used), to enrich the experimental investigation, Gaussian membership functions and fuzzy c-means clustering are utilised here, in implementing the proposed ANFIS interpolation methods. The results are listed in Table XII with the best highlighted in bold.

The proposed approach outperforms the others in a substantial majority of cases. For the three particular cases (of LR on the Quake dataset, SVR on Dee, and RF on AutoMPG6) where an existing method shows the best outcome, the accuracy of the proposed approach is close. In particular, the proposed approach with ANFIS based chromosome representation gives by far, the best mean value of the overall RMSE averaged across the ten datasets, and the proposed with rule based chromosome representation achieves the second best (as shown in the second last row of Table XII). This superiority in performance remains to be true taking into consideration the standard deviations. Note however, that in terms of the overall averaged standard deviation measure, the classical SVR has the least value, demonstrating its excellent stability.

The last row of Table XII shows the running time performance of different methods. Similar observations as per those on the synthetic data case studies can be drawn: The proposed methods are faster than E-ANFIS, but slower than the rest, since the evolutionary process incurs more computation. Between the two proposed methods themselves, it is clear that the implementation with ANFIS-based chromosome representation consumes more time. Again, this is expected, reflecting the result of theoretical analysis on computational complexity.

*4) Effects of Sparsity in Training Data:* This particular experimentation is conducted to investigate the performance of different ANFISs in response to the use of different percentages of training data, instead of just the 20% and 80% split. To focus on the discussion about the consequences of varying the amount of training data, only the results on the 'Quake' dataset are shown here, as given in Fig. 8.

It can be seen that in general, the RMSE values decrease as the percentage of training data increases, independent of which learning method is used. Importantly, also independent of what percentage of training data is utilised, the three interpolated ANFISs all remarkably outperform the 'Original ANFIS' (until training data reaches 90%). Furthermore, the RMSE values of the three interpolated ANFISs generally decline much less rapidly than the RMSE of the 'Original ANFIS', indicating that the interpolated ANFISs are less sensitive to the decrease of training data. Indeed, the performances of these three methods using $30\%$ data are very close to those using $90\%$ data. When the available data percentage is 20%, the results are of course the same as discussed in the preceding sub-section, with the three interpolated ANFISs significantly beating the original and the two GA-based methods outperforming 'Method 0'. When only $10\%$ of data is available for training, the performances of the interpolated methods degrade but still beat the 'Oriainl ANFIS' trained with the conventional method substantially. Most interestingly, even in this case, either of the two proposed methods retains its superior performance over 'Method 0', especially when rule based chromosome representation is used. Overall, the proposed ANFIS interpolation algorithms via evolutionary computation achieve a much better result while using less training data than the existing techniques.

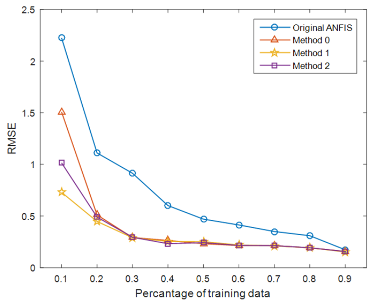

Fig. 8. Performance vs. percentage of training data used

*5) Effects of Crossover and Mutation Probability:* The crossover probability $p_c$ and mutation probability $p_m$ are two important parameters in the evolutionary process. Generally, larger crossover probability and smaller mutation probability are a common choice for most evolutionary problems. In the above-reported experimental investigations, these two parameters are set to $p_c = 0.8, p_a = 0.2$ respectively. In order to examine the effects on performance of using different $p_c$ and $p_m$, these probabilities are varied from 0.1 to 0.9 (with step=0.1) in this experimentation. This part of the experimental studies is focussed on the use of the entire ANFIS based chromosome representation since the number of the individual rule-based chromosome in each sub-population is very small (only 5 in the experiments, so the results of using different $p_c$ and $p_m$ do not differ much in the first place). The resulting box-plot over the 'Quake' dataset is shown in Fig. 9. In conformation with common knowledge in evolutionary computation the results reveal that larger $p_c$ as well as smaller $p_m$ give a better value for both the median and the interquartile range.

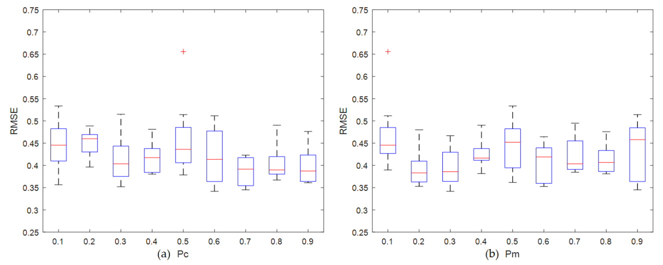

Fig. 9. (a) Performance vs. crossover rate. (b) Performance vs. mutation rate.

TABLE XI
EXPERIMENTAL RESULTS USING DIFFERENT ANFISS ON REAL WORLD DATASETS

| Dataset | Mean ± Standard deviation of ANFISs | | | |
|---|---|---|---|---|
| | Original ANFIS | Method 0 | Method 1 | Method 2 |
| Diabetes | $1.527 \pm 0.829$ | $1.058 \pm 0.497$ | $\mathbf{0.989 \pm 0.359}$ | $1.011 \pm 0.451$ |
| Plastic | $2.003 \pm 0.246$ | $1.843 \pm 0.105$ | $1.836 \pm 0.093$ | $\mathbf{1.828 \pm 0.078}$ |
| Quake | $1.264 \pm 0.641$ | $0.542 \pm 0.248$ | $\mathbf{0.458 \pm 0.175}$ | $0.498 \pm 0.225$ |
| Laser | $12.087 \pm 4.739$ | $3.757 \pm 2.724$ | $3.476 \pm \mathbf{1.069}$ | $\mathbf{3.279} \pm 1.262$ |
| AutoMPG6 | $12.770 \pm 1.431$ | $5.120 \pm 1.038$ | $\mathbf{4.776} \pm 1.015$ | $4.987 \pm \mathbf{0.843}$ |
| Delta-ail | $3.53 \times 10^{-4} \pm 1.09 \times 10^{-4}$ | $1.78 \times 10^{-4} \pm 1.06 \times 10^{-5}$ | $\mathbf{1.73 \times 10^{-4} \pm 8.87 \times 10^{-6}}$ | $1.75 \times 10^{-4} \pm 9.49 \times 10^{-6}$ |
| Friedman | $5.177 \pm 0.633$ | $3.004 \pm 0.409$ | $\mathbf{2.900 \pm 0.335}$ | $2.930 \pm 0.405$ |
| Dee | $1.017 \pm 0.339$ | $0.870 \pm 0.225$ | $\mathbf{0.824 \pm 0.188}$ | $0.839 \pm 0.191$ |
| Delta-elv | $0.0034 \pm 6.195 \times 10^{-4}$ | $0.0020 \pm 1.69 \times 10^{-4}$ | $\mathbf{0.0018} \pm 7.11 \times 10^{-5}$ | $\mathbf{0.0018 \pm 5.90 \times 10^{-5}}$ |
| ANACALT | $1.209 \pm 0.966$ | $0.994 \pm 0.768$ | $0.999 \pm 0.675$ | $\mathbf{0.986 \pm 0.681}$ |
| Average | $3.706 \pm 0.982$ | $1.719 \pm 0.601$ | $\mathbf{1.626 \pm 0.391}$ | $1.635 \pm 0.412$ |

TABLE XII
ACCURACY (MEAN ± STANDARD DEVIATION) AND RUNNING TIME (SECONDS) OF DIFFERENT METHODS ON REAL DATA

| Dataset | Results of different methods | | | | | | |
|---|---|---|---|---|---|---|---|
| | LR | SVR | CART | RF | E-ANFIS | Method 1 | Method 2 |
| Diabetes | $0.533 \pm 0.100$ | $0.441 \pm 0.052$ | $0.779 \pm 0.130$ | $0.391 \pm 0.052$ | $1.172 \pm 1.100$ | $\mathbf{0.362 \pm 0.045}$ | $0.377 \pm 0.050$ |
| Plastic | $1.623 \pm 0.044$ | $2.285 \pm \mathbf{0.037}$ | $2.245 \pm 0.128$ | $1.886 \pm 0.069$ | $1.865 \pm 0.348$ | $1.620 \pm 0.046$ | $\mathbf{1.603} \pm 0.056$ |
| Quake | $\mathbf{0.194 \pm 0.006}$ | $0.206 \pm 0.007$ | $0.243 \pm 0.015$ | $0.195 \pm 0.008$ | $0.259 \pm 0.153$ | $0.198 \pm \mathbf{0.006}$ | $0.199 \pm 0.007$ |
| Laser | $6.934 \pm 1.337$ | $15.522 \pm \mathbf{0.435}$ | $12.872 \pm 2.685$ | $7.680 \pm 0.801$ | $5.644 \pm 2.636$ | $4.547 \pm 1.632$ | $\mathbf{4.030} \pm 1.539$ |
| AutoMPG6 | $3.569 \pm 0.685$ | $3.336 \pm 0.353$ | $4.044 \pm 0.559$ | $\mathbf{3.282 \pm 0.331}$ | $5.625 \pm 1.275$ | $3.628 \pm 0.696$ | $3.575 \pm 0.572$ |
| Delta-ail | $1.55 \times 10^{-4} \pm 1.48 \times 10^{-6}$ | $1.59 \times 10^{-4} \pm 1.66 \times 10^{-6}$ | $1.91 \times 10^{-4} \pm 7.33 \times 10^{-6}$ | $1.66 \times 10^{-4} \pm 5.13 \times 10^{-6}$ | $1.64 \times 10^{-4} \pm 3.86 \times 10^{-5}$ | $\mathbf{1.50 \times 10^{-4}} \pm \mathbf{1.45 \times 10^{-6}}$ | $1.51 \times 10^{-4} \pm 1.46 \times 10^{-6}$ |
| Friedman | $2.795 \pm 0.049$ | $2.771 \pm \mathbf{0.047}$ | $7.224 \pm 0.483$ | $2.771 \pm 0.156$ | $2.408 \pm 1.325$ | $2.041 \pm 0.075$ | $\mathbf{2.030} \pm 0.052$ |
| Dee | $0.424 \pm 0.023$ | $\mathbf{0.416 \pm 0.021}$ | $1.016 \pm 0.063$ | $0.482 \pm 0.035$ | $0.810 \pm 0.366$ | $0.477 \pm 0.048$ | $0.486 \pm 0.064$ |
| Delta-elv | $1.51 \times 10^{-3} \pm 1.17 \times 10^{-5}$ | $1.53 \times 10^{-3} \pm 1.23 \times 10^{-5}$ | $1.95 \times 10^{-3} \pm 5.19 \times 10^{-5}$ | $1.51 \times 10^{-3} \pm 1.26 \times 10^{-5}$ | $1.55 \times 10^{-3} \pm 1.78 \times 10^{-4}$ | $\mathbf{1.48 \times 10^{-3}} \pm \mathbf{1.03 \times 10^{-5}}$ | $\mathbf{1.48 \times 10^{-3}} \pm 1.09 \times 10^{-5}$ |
| ANACALT | $0.405 \pm \mathbf{0.020}$ | $0.333 \pm 0.021$ | $0.299 \pm 0.038$ | $0.292 \pm 0.043$ | $0.357 \pm 0.100$ | $0.280 \pm 0.029$ | $\mathbf{0.279} \pm 0.028$ |
| Average | $1.648 \pm 0.227$ | $2.531 \pm \mathbf{0.097}$ | $2.873 \pm 0.409$ | $1.698 \pm 0.149$ | $1.814 \pm 0.811$ | $1.316 \pm 0.258$ | $\mathbf{1.258} \pm 0.236$ |
| Run time | 1.46 ± 0.15 | 13.34 ± 15.57 | 2.38 ± 1.85 | 3.47 ± 2.5 | 225.3 ± 171.65 | 35.15 ± 20.2 | 64.91 ± 36.55 |

## V. CONCLUSION

This paper has presented a new ANFIS interpolation approach via evolutionary computation (implemented with GAs), in an effort to improve the learning of ANFIS when there is significant shortage of training data for the problem concerned. Different from typical existing FRI algorithms, the concept of 'ANFIS interpolation' means the interpolation of an entire inference system, or a whole group of rules in the target region. This is enabled with the assistance of well-trained ANFISs in the neighbouring regions. Two forms of chromosome representation are considered: one encoding individual rules and the other expressing an entire possible ANFIS. The proposed approach has been tested on both three function modelling problems with synthetic data and real world regression problems involving 10 benchmark datasets, demonstrating its ability in significantly improving the inference performance when compared with existing techniques.

This work only involves two source ANFISs in the interpolation procedure, but the framework can be extended to interpolation of multiple ANFISs. Currently, the simplest evolutionary process (genetic algorithm) is utilised, implementing the underlying ideas with more advanced evolutionary algorithms may lead to further strengthened interpolative results. Purely for convenience in implementation, the sizes of different clusters are herein assumed to be the same, but there is no reason why an adaptive mechanism cannot be introduced to form different cluster sizes, reflecting the amounts of training data scales given per cluster. This may further improve the learning performance. Also, expanding and evaluating the proposed work with more real world tasks forms a necessary direction of continuing the present work. A particular focus on the practical application of this research is for image modelling and analysis, especially in the areas of remote sensing [52], [53] and medical diagnosis [54], [55].

## REFERENCES

[1] A. Esfahanipour and W. Aghamiri, "Adapted neuro-fuzzy inference system on indirect approach tsk fuzzy rule base for stock market analysis," *Expert Syst. Appl.*, vol. 37, no. 7, pp. 4742–4748, 2010.

[2] J. Alcalá-Fdez and J. M. Alonso, "A survey of fuzzy systems software: Taxonomy, current research trends, and prospects," *IEEE Trans. Fuzzy Syst.*, vol. 24, no. 1, pp. 40–56, 2016.

[3] Y. Jiang, Z. Deng *et al.*, "Recognition of epileptic eeg signals using a novel multiview tsk fuzzy system," *IEEE Trans. Fuzzy Syst.*, vol. 25, no. 1, pp. 3–20, 2017.

[4] N. K. 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.

[5] K. B. Cho and B. H. Wang, "Radial basis function based adaptive fuzzy systems and their applications to system identification and prediction," *Fuzzy Sets Syst.*, vol. 83, no. 3, pp. 325–339, 1996.

[6] J.-H. Chiang and P.-Y. Hao, "Support vector learning mechanism for fuzzy rule-based modeling: a new approach," *IEEE Trans. Fuzzy Syst.*, vol. 12, no. 1, pp. 1–12, 2004.

[7] T. Chen, C. Shang, P. Su, and Q. Shen, "Induction of accurate and interpretable fuzzy rules from preliminary crisp representation," *Knowl. Based Syst.*, vol. 146, pp. 152–166, 2018.

[8] J.-S. Jang, "Anfis: adaptive-network-based fuzzy inference system," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, no. 3, pp. 665–685, 1993.

[9] C.-C. Chuang, S.-F. Su, and S.-S. Chen, "Robust tsk fuzzy modeling for function approximation with outliers," *IEEE Trans. Fuzzy Syst.*, vol. 9, no. 6, pp. 810–821, 2001.

[10] S. Jin, R. Diao, C. Quek, and Q. Shen, "Backward fuzzy rule interpolation," *IEEE Trans. Fuzzy Syst.*, vol. 22, no. 6, pp. 1682–1698, 2014.

[11] N. Naik, R. Diao, and Q. Shen, "Dynamic fuzzy rule interpolation and its application to intrusion detection," *IEEE Trans. Fuzzy Syst.*, vol. 26, no. 4, pp. 1878–1892, 2018.

[12] S. J. Pan, Q. Yang *et al.*, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, 2010.

[13] H. Zuo, G. Zhang, W. Pedrycz, V. Behbood, and J. Lu, "Fuzzy regression transfer learning in takagi–sugeno fuzzy models," *IEEE Trans. Fuzzy Syst.*, vol. 25, no. 6, pp. 1795–1807, 2017.

[14] H. Zuo, G. Zhang, V. Behbood, and J. Lu, "Granular fuzzy regression domain adaptation in takagi–sugeno fuzzy models," *IEEE Trans. Fuzzy Syst.*, vol. 26, no. 2, pp. 847–858, 2018.

[15] Z. Deng, Y. Jiang, F.-L. Chung, H. Ishibuchi, and S. Wang, "Knowledge-leverage-based fuzzy system and its modeling." *IEEE Trans. Fuzzy Syst.*, vol. 21, no. 4, pp. 597–609, 2013.

[16] J. Shell and S. Coupland, "Fuzzy transfer learning: methodology and application," *Inf. Sci.*, vol. 293, pp. 59–79, 2015.

[17] D. Tikk, I. Joó, L. T. Kóczy, P. Várlaki, B. Moser, and T. D. Gedeon, "Stability of interpolative fuzzy kh controllers," *Fuzzy Sets Sys.*, vol. 125, no. 1, pp. 105–119, 2002.

[18] L. Kóczy and K. Hirota, "Approximate reasoning by linear rule interpolation and general approximation," *Int. J. Approx. Reason.*, vol. 9, no. 3, pp. 197–225, 1993.

[19] P. Baranyi, L. T. Kóczy, and T. D. Gedeon, "A generalized concept for fuzzy rule interpolation," *IEEE Trans. Fuzzy Syst.*, vol. 12, no. 6, pp. 820–837, 2004.

[20] Z. Huang and Q. Shen, "Fuzzy interpolative reasoning via scale and move transformations," *IEEE Trans. Fuzzy Syst.*, vol. 14, no. 2, pp. 340–359, 2006.

[21] S.-M. Chen and Y.-C. Chang, "Weighted fuzzy rule interpolation based on ga-based weight-learning techniques," *IEEE Trans. Fuzzy Syst.*, vol. 19, no. 4, pp. 729–744, 2011.

[22] Z. Huang and Q. Shen, "Fuzzy interpolation and extrapolation: A practical approach," *IEEE Trans. Fuzzy Syst.*, vol. 16, no. 1, pp. 13–28, 2008.

[23] L. Yang and Q. Shen, "Adaptive fuzzy interpolation," *IEEE Trans. Fuzzy Syst.*, vol. 19, no. 6, pp. 1107–1126, 2011.

[24] L. Yang, F. Chao, and Q. Shen, "Generalised adaptive fuzzy rule interpolation," *IEEE Trans. Fuzzy Syst.*, vol. 25, pp. 839–853, 2017.

[25] C. Chen, MacParthalain, L. Y. N., Price, P., C. Quek, and Q. Shen, "Rough-fuzzy rule interpolation," *Inf. Sci.*, vol. 351, pp. 1–17, 2016.

[26] F. Li, C. Shang, Y. Li, J. Yang, and Q. Shen, "Fuzzy rule based interpolative reasoning supported by attribute ranking," *IEEE Trans. Fuzzy Syst.*, vol. 26, no. 5, pp. 2758–2773, 2018.

[27] F. Li, Y. Li, C. Shang, and Q. Shen, "Fuzzy knowledge-based prediction through weighted rule interpolation," *IEEE Trans. Cybern.*, vol. 50, no. 10, pp. 4508–4517, 2020.

[28] E. Mamdani, "Application of fuzzy logic to approximate reasoning using linguistic synthesis," *IEEE Trans. Comput.*, vol. 26, no. 12, pp. 1182–1191, 1977.

[29] J. Li, L. Yang, Y. Qu, and G. Sexton, "An extended takagi–sugeno–kang inference system (tsk+) with fuzzy interpolation and its rule base generation," *Soft Comput.*, vol. 22, no. 10, pp. 3155–3170, 2018.

[30] J. Yang, C. Shang, Y. Li, F. Li, and Q. Shen, "Anfis construction with sparse data via group rule interpolation," *IEEE Trans. Cybern.*, doi: 10.1109/TCYB.2019.2952267.

[31] T. Chen, C. Shang, J. Yang, F. Li, and Q. Shen, "A new approach for transformation-based fuzzy rule interpolation," *IEEE Trans. Fuzzy Syst.*, vol. 28, no. 12, pp. 3330–3344, 2020.

[32] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Trans. Syst., Man, Cybern.*, no. 1, pp. 116–132, 1985.

[33] T. Bäck, U. Hammel, and H.-P. Schwefel, "Evolutionary computation: Comments on the history and current state," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 3–17, 1997.

[34] T. Bäck, D. B. Fogel, and Z. Michalewicz, *Evolutionary computation 1: Basic algorithms and operators*. CRC press, 2018.

[35] J. Petke, S. O. Haraldsson, M. Harman, W. B. Langdon, D. R. White, and J. R. Woodward, "Genetic improvement of software: a comprehensive survey," *IEEE Trans. Evol. Comput.*, vol. 22, no. 3, pp. 415–432, 2017.

[36] A. Agapitos, R. Loughran *et al.*, "A survey of statistical machine learning elements in genetic programming," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 6, pp. 1029–1048, 2019.

[37] Y. Zhang, S. Wang, and G. Ji, "A comprehensive survey on particle swarm optimization algorithm and its applications," *Mathematical Problems in Engineering*, vol. 2015, 2015.

[38] R. V. Kulkarni and G. K. Venayagamoorthy, "Particle swarm optimization in wireless-sensor networks: A brief survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 41, no. 2, pp. 262–267, 2010.

[39] B. Xue, M. Zhang, W. N. Browne, and X. Yao, "A survey on evolutionary computation approaches to feature selection," *IEEE Trans. Evol. Comput.*, vol. 20, no. 4, pp. 606–626, 2015.

[40] Y. Shi, R. Eberhart, and Y. Chen, "Implementation of evolutionary fuzzy systems," *IEEE Trans. Fuzzy Syst.*, vol. 7, no. 2, pp. 109–119, 1999.

[41] O. Cordón, "A historical review of evolutionary learning methods for mamdani-type fuzzy rule-based systems: Designing interpretable genetic fuzzy systems," *Int. J. Approx. Reason.*, vol. 52, no. 6, pp. 894–913, 2011.

[42] P. Angelov, D. P. Filev, and N. Kasabov, *Evolving intelligent systems: methodology and applications*. John Wiley & Sons, 2010, vol. 12.

[43] X. Gu, Q. Shen, and P. Angelov, "Particle swarm optimized autonomous learning fuzzy system," *IEEE Trans. Cybern.*, doi: 10.1109/TCYB.2020.2967462.

[44] Z. J. Viharos and K. B. Kis, "Survey on neuro-fuzzy systems and their applications in technical diagnostics and measurement," *Measurement*, vol. 67, pp. 126–136, 2015.

[45] T. Takagi and M. Sugeno, "Derivation of fuzzy control rules from human operator's control actions," *IFAC Proceedings Volumes*, vol. 16, no. 13, pp. 55–60, 1983.

[46] F. Li, C. Shang, Y. Li, J. Yang, and Q. Shen, "Interpolation with just two nearest neighbouring weighted fuzzy rules," *IEEE Trans. Fuzzy Syst.*, vol. 28, no. 9, pp. 2255–2262, 2020.

[47] F. Herrera, M. Lozano, and J. L. Verdegay, "Tuning fuzzy logic controllers by genetic algorithms," *Int. J. Approx. Reason.*, vol. 12, no. 3-4, pp. 299–315, 1995.

[48] H. Ishibuchi, K. Nozaki, N. Yamamoto, and H. Tanaka, "Selecting fuzzy if-then rules for classification problems using genetic algorithms," *IEEE Trans. Fuzzy Syst.*, vol. 3, no. 3, pp. 260–270, 1995.

[49] O. Cord *et al.*, *Genetic fuzzy systems: evolutionary tuning and learning of fuzzy knowledge bases*. World Scientific, 2001, vol. 19.

[50] S. Mousavi, S. Mirinezhad, and V. Lyashenko, "An evolutionary-based adaptive neuro-fuzzy expert system as a family counselor before marriage with the aim of divorce rate reduction," *Artigence*, vol. 1, pp. 1–16, 05 2018.

[51] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, and F. Herrera, "Keel data-mining software tool: data set repository, integration of algorithms and experimental analysis framework." *Journal of Multiple-Valued Logic & Soft Computing*, vol. 17, 2011.

[52] J. Yang, Y. Li, J. Chan, and Q. Shen, "Image fusion for spatial enhancement of hyperspectral image via pixel group based non-local sparse representation," *Remote Sensing*, vol. 9, no. 1, p. 53, 2017.

[53] H. Zhang, Y. Li, Y. Jiang, P. Wang, Q. Shen, and C. Shen, "Hyperspectral classification based on lightweight 3d-cnn with transfer learning," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 8, pp. 5813–5828, 2019.

[54] N. MacParthalain, R. Jensen, Q. Shen, and R. Zwiggelaar, "Fuzzy-rough approaches for mammographic risk analysis," *Intelligent Data Analysis*, vol. 14, no. 2, pp. 225–244, 2010.

[55] F. Li, C. Shang, Y. Li, and Q. Shen, "Interpretable mammographic mass classification with fuzzy interpolative reasoning," *Knowledge-Based Systems*, vol. 191, p. 105279, 2020.