Feature Selection and Function Approximation Using Adaptive Algorithms

Varun Kumar Ojha

Department of Computer Science
Faculty of Electrical Engineering and Computer Science,
VŠB-Technical University of Ostrava, Czech Republic

Supervisors:

Prof. Dr. Ajith Abraham, Machine Intelligence Research Lab, WA, USA
Prof. Dr. Václav Snášel, VŠB-Technical University of Ostrava, Czech Republic

Abstract

Multiobjective heterogeneous flexible neural tree (HFNT) and multiobjective hierarchical fuzzy inference tree (HFIT) are two novel adaptive algorithms, which were proposed for the feature selection and function approximation after comprehensive literature reviews of the neural network and fuzzy inference system paradigms, respectively. The proposed algorithms were designed as a tree-like model, and the best tree-structure was selected from a topological space by applying a multiobjective evolutionary algorithm that simultaneously minimized both approximation error and tree complexity. Further, the parameter vector of the selected tree, from the Pareto front, was tuned by using a metaheuristic algorithm.

For HFNT, the dynamics of natural selection was exploited to introduce functional heterogeneity in the HFNT nodes, and a diversity index was introduced for creating diverse HFNTs during its tree optimization phase. Subsequently, an evolutionary ensemble of HFNTs was proposed for making use of the final population. On the other hand, the HFIT nodes were low-dimensional type-1 or type-2 fuzzy inference systems, and the tree-like model was a hierarchical arrangement of such nodes.

The performance of both HFNT and HFIT on benchmark datasets was better than the performance of the many important algorithms from the literature. Additionally, both HFNT and HFIT was used for the predictive modeling of the industrial problems, in which the feature selection was a crucial challenge in addition to the prediction. High approximation ability with the simple model generation is the vital contribution of the proposed algorithms for predictive modeling of complex problems.

Keywords: feedforward neural network; fuzzy inference system; multiobjective; metaheuristics; ensemble learning; feature selection.

Contents

1 Introduction

Feature selection plays a crucial role in analyzing the most significant variables in a dataset, and the function approximation is involved in finding the meaningful relationship among the variables of a dataset. Therefore, the goal is *to develop the algorithms that perform these two tasks efficiently and simultaneously.*

1.1 Problem statement

Feature selection and function approximation are usually viewed within the supervised learning paradigm. Two computational intelligence models, feedforward neural network (FNN) and fuzzy inference system (FIS), perform these tasks efficiently when trained/optimized by supplying the training data (X, Y) of N input–output pairs, i.e., $X = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)$ and $Y = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N)$. Each input $\mathbf{x}_i = \langle x_{i1}, x_{i2}, \ldots, x_{ip} \rangle$ is a p-dimensional vector, and it has a corresponding q-dimensional desired output vector $\mathbf{y}_i = \langle y_{i1}, y_{i2}, \ldots, y_{iq} \rangle$.

For the training data (X, Y), a model $\hat{\mathbf{y}}_i = f(\mathbf{x}_i, \mathbf{w})$, where $\hat{\mathbf{y}}_i = \langle \hat{y}_{i1}, \hat{y}_{i2}, \ldots, \hat{y}_{iq} \rangle$ is a q-dimensional model's output corresponding to i-th input vector $\mathbf{x}_i$, produces the predicted output $\hat{Y} = (\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2, \ldots, \hat{\mathbf{y}}_N)$ for all $i = 1, 2, \ldots N$. The predicted output $f : \hat{Y} \longrightarrow \{(\mathbf{x}_i, \mathbf{y}_i) |\ \mathbf{x}_i \in X, \mathbf{y}_i \in Y, i = 1, 2, \ldots N\}$ is then compared to desired output Y by using some error/distance/cost function c_f. Hence, For the problems whose desired outputs are continuous variables, *mean squared error* (MSE) is one of the commonly used cost function, which is expressed as:

$$c_f(Y, \hat{Y}) = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{q} (y_{ij} - \hat{y}_{ij})^2 , \tag{1}$$

where y_{ij} are the desired outputs and $\hat{y}_{ij}$ are the model's outputs, and their differences are summed over N data pairs.

Thus, a model $\hat{\mathbf{y}}_i = f(\mathbf{x}_i, \mathbf{w})$, parameterized by p-dimensional input vector $\mathbf{x}_i$ and an n-dimensional real-valued vector $\mathbf{w}$, under a supervised learning algorithm tries to find a casual relationship between input X and output Y by searching optimum parameter vector $\mathbf{w}^*$ and by minimizing the cost function $c_f : Y \times \hat{Y} \longrightarrow \mathbb{R}_{\geq 0}$. Such process is called function approximation, which may be defined as:

Definition 1 (Function approximation)**.** *Given the dataset (X, Y), the goal in function approximation is to find a function $f(\mathbf{x}_i, \mathbf{w}^*)$ such that $c_f(Y, \hat{Y}^*) \leq c_f(Y, \hat{Y})$ holds for any $\mathbf{w} \in \mathbb{R}^n$, where $\hat{Y}^*$ is computed for $\mathbf{w}^*$ and $\hat{Y}$ is computed for $\mathbf{w}$.*

Let an input vector $\mathbf{x}$ have p input feature. Then, for a set of input features Z, the feature selection may be defined as:

Definition 2 (Feature selection)**.** *Given an input feature set* $Z = \{z_1, z_2, \ldots, z_p\}$ *and the power set* $\mathbf{Z} = \mathcal{P}(Z) - \emptyset$*, a feature selection method finds an optimal set* Z^**, for which a model gives the lowest approximation error, where* $Z^* \in \mathbf{Z}$ *and* $|Z^*| \leq |Z|$*.*

1.2 Objectives

The following objectives were framed in this research for the purpose of developing algorithms for the efficient and simultaneous feature selection and the function approximation:

1) To investigate available metaheuristics based design of FNN and FIS algorithms.

2) To investigate multiobjective optimization methods to adapt the topology and learning parameters, which could lead to a low approximation error and a less complex model.

3) To develop effective data mining algorithms based on the adaptive data structures for the feature selection and the function approximation.

4) To validate the developed algorithms on benchmark datasets and real-world problems.

1.3 Steps towards objectives

1.3.1 Proposal of heterogeneous flexible neural tree

The flexible neural tree (FNT) is an algorithm for the feature selection and function approximation proposed by Chen et al. [1, 2]. FNT perform feature selection and function approximation simultaneously. FNT has neural nodes (FNN-like nodes) and has feedforward connection. Since FNT design is a tree-like structure and evolves using evolutionary algorithms, its size (complexity) can be very large if the tree is created by minimizing only approximation error. Hence, a multiobjective optimization framework for FNTs tree structure optimization was proposed in this research.

For the proposed multiobjective optimization of FNT, a nondominated sorting genetic programming was applied for optimizing *approximation error*, *model complexity*, and *models diversity*, simultaneously. Additionally, adaptation in node was introduced to offer heterogeneity. Thus, the proposed algorithm was named heterogeneous FNT (HFNT).

Finally, HFNT was evaluated on the benchmark problems and an industrial problem (a predictive modeling of pharmaceutical granules).

1.3.2 Proposal of hierarchical fuzzy inference tree

FIS is an efficient way to model inaccurate, imprecise, incomplete, and noisy data [3, 4]. The basic need of FIS design is its optimization of rule base (which can be represented

as a structure, i.e., a connection based model), and the optimization of the rule base parameters. In the past, neural network and hierarchical based FIS design were proposed by researchers.

In this research, a tree-like structure was proposed for FIS design that was a hierarchical arrangement of low-dimensional FISs and resembled FNN-like connection. For this purpose, the node of the tree was designed as type-1 and type-2 FIS. Subsequently, multiobjective evolution algorithm was applied to create a hierarchical fuzzy inference tree (HFIT). The performance of the proposed HFIT was evaluated over six examples including a real-wolrd application (drug dissolution rate prediction).

1.4 Organization of the thesis

In this thesis, two novel algorithms HFIT and HFIT were proposed after the literature review of metaheuristic based design of FNN and FIS. Hence, first, in Chapter 2, state-of-the-art review of FNN optimization is discussed, which provides the ground for identifying the current challenges and future research direction. Subsequently, the proposed multiobjective HFNT is explained in Chapter 3. Chapter 4 provides a detailed state-of-the-art review of metaheuristic-based design of FIS, which is followed by a detail description of the proposed multiobjective HFIT algorithm in Chapter 5. Finally, Chapter 6 offers concise conclusions and future research directions derived from this research.

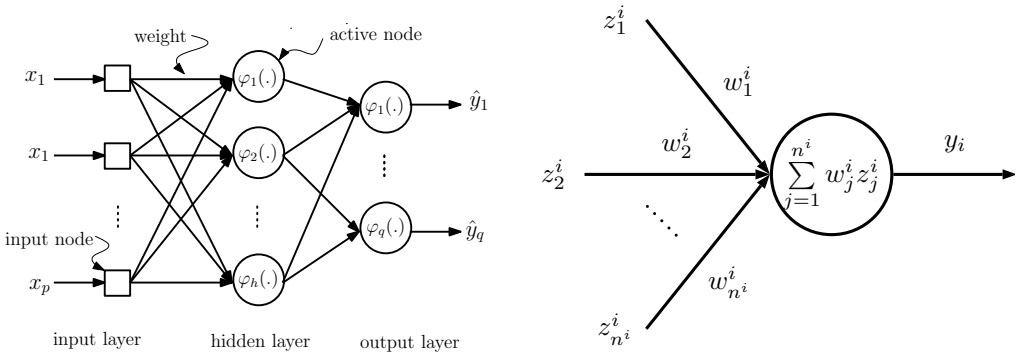
2 Metaheuristic design of feedforward neural network

History of artificial neural network (ANN) goes back to 1943 when McCulloch and Pitts [5] proposed a computational model that imitates the human brain. Feedforward neural networks (FNNs) are the special type of ANN models that are capable of learning and recognizing, and can solve a large range of complex problems [6–8].

Metaheuristics formulates FNN components, such as weights, structure, nodes, etc., into an optimization problem. Metaheuristics uses various heuristics for finding a near-optimum solution. Additionally, multiobjective metaheuristic framework deal with multiple objectives simultaneously, and the existence of multiple objectives in FNN optimization is evident since the minimization of FNNs approximation error is desirable at one hand, and the generalization and model's simplification is at the other.

2.1 Feedforward neural network

Essentially, the FNN in Fig. 1 is a phenotype/structural representation of a function $f(\mathbf{x}, \mathbf{w})$, which is parameterized by p-dimensional input vector $\mathbf{x} = \langle x_1, x_2, \ldots, x_p \rangle$ and n-dimensional real-valued weight vector $\mathbf{w} = \langle w_1, w_2, \ldots, w_n \rangle$. The function $f(\mathbf{x}, \mathbf{w})$ is



(a) Three-layer feedforward neural network (b) Node of the network

Figure 1: Three-layer feedforward neural network (a), where input layer has p input nodes, hidden layer has h activation functions and output layer has q nodes.

a solution to a given problem. Therefore, two tasks involved in solving a problem using FNN are to discover appropriate function $f(\mathbf{x}, \mathbf{w})$ (i.e., architecture optimization) and to discover appropriate weight vector $\mathbf{w}$ (i.e., weight optimization) using some *learning algorithm*.

Network architecture optimization includes the searching of appropriate activation functions $\varphi(\cdot)$ at the nodes, the number of nodes, number of layers, arrangements of the nodes, etc. Therefore, several components of FNN optimization are the connection **weights**; the **architecture** (number of layers in a network, the number of nodes at the hidden layers, the arrangement of the connections between nodes); the **nodes** (activation functions at the nodes); the **learning algorithms** (algorithms training parameters); and the **learning environment**. However, traditionally, the only component that was optimized used to be the connection weights.

2.2 Metaheuristic formulation of FNN components

The basic form of FNN optimization is the act of searching its weights (free parameters of FNN) such that the cost function (1) can be minimized. However, the goodness (performance) of FNN cost function depends not only on finding optimum weights, but finding optimum architecture, activation function, parameter setting of learning algorithm, and training environment are equally important. To apply metaheuristic algorithms for optimizing FNN, using some strategy, the FNN components (*phenotype*) need to be formulated into a vector (*genotype*) form.

Fig. 2 (a *Venn diagram*) illustrates the spectrum metaheuristics optimization of FNN components: weights, architecture, activation function, and learning rule's parameters. In Fig. 2, area "a1" indicates optimization of weights; area "a2" indicates optimization of weights and architecture; area "a3" indicates optimization of weights, architecture, and

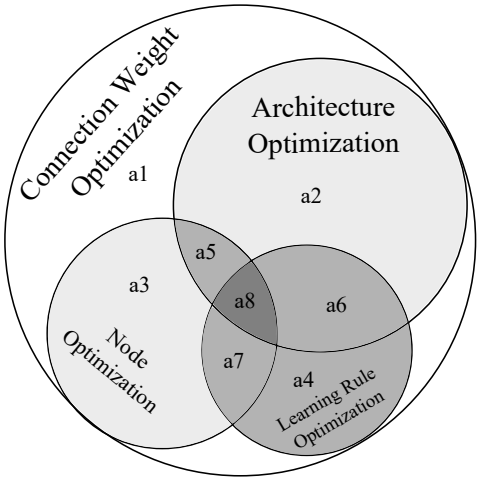


Figure 2: Spectrum of metaheuristic design for FNN.

activation function; and the areas "a4" to "a8" indicate all other possible combinations.

Examining Fig. 2, one can say that the strength and complexity of optimization increases from area denoted "a1" to "a8," where "a1" is the simplest approach and "a8" is the most sophisticated approach.

3 Multiobjective heterogeneous flexible neural trees

Machine learning algorithms are inherently multiobjective in nature, where approximation error minimization and model's complexity simplification are two conflicting objectives. A multiobjective genetic programming (MOGP) based framework was proposed for creating a heterogeneous flexible neural tree (HFNT^{M}), which is a tree-like flexible FNN model. MOGP guided an initial HFNT^{M} population towards Pareto-optimal solutions, where the final population was used for making an ensemble system.

A diversity index measure along with approximation error and complexity was introduced to maintain diversity among the candidates in the population. Hence, the ensemble was created by using accurate, structurally simple, and diverse candidates from MOGP final population. Differential evolution algorithm was applied to fine-tune the underlying parameters of selected candidates.

A comprehensive test on benchmark and real-world (an industrial problem, where the data represented a dynamic environment of die filing process) datasets proved the efficiency of the proposed HFNT^{M} approach over other available prediction models. Moreover, heterogeneous creation of HFNT^{M} proved to be efficient in making ensemble system from the final population. Therefore, the key contributions of this chapter is as follows:

1) A heterogeneous flexible neural tree (HFNT) for function approximation and feature selection was proposed.

2) Application of NSGA-II-based multiobjective genetic programming framework for HFNT was proposed. Hence, notation HFNT^{M} was used in this chapter.

3) Alongside approximation error and tree complexity, a diversity index was introduced to maintain diversity among the candidates in the population.

4) HFNT was found competitive with other algorithms when compared and cross-validated over classification, regression, and time-series datasets.

5) The proposed evolutionary weighted ensemble of HFNTs final population further improved its performance.

3.1 Heterogeneous flexible neural tree

HFNT is analogous to a multilayer feedforward neural network that has over-layer connections and activation function at the nodes. HFNT construction has two phases [2]: 1) the *tree construction* phase, in which evolutionary algorithms are applied to construct tree-like structure; and 2) the *parameter-tuning* phase, in which genotype of HFNT (underlying parameters of tree-structure) is optimized by using parameter optimization algorithms.

To create a near-optimum model, phase one starts with random tree-like structures (population of initial solutions), where parameters of each tree are fixed by a random guess. Once a near-optimum tree structure is obtained, *parameter-tuning* phase optimizes its parameter. The phases are repeated until a satisfactory solution is obtained. Fig. 3 is a lucid illustration of these two phases. Moreover, evolutionary algorithm allowed HFNT to select activation functions and input feature at the nodes from sets of activation functions and input features, respectively. Thus, HFNT possesses automatic feature selection ability.

3.1.1 Encoding: basic terminology

An HFNT G is a collection of function set F_G and instruction set T_G:

$$G = F_G \cup T_G = \left\{ +_2^{U(k)}, +_3^{U(k)}, \cdots, +_{tn}^{U(k)} \right\} \cup \{x_1, x_2, \ldots, x_p\} \tag{2}$$

where $+_j^k (j = 2, 3, \ldots, tn)$ denotes a non-leaf instruction (a computational node). It receives $2 \leq j \leq tn$ arguments and $U(k)$ is a function that randomly takes an activation function from a set of k activation functions. Maximum arguments tn to a computational node are predefined. A set of seven activation functions is shown in Table 1. Leaf node's instruction $x_1, x_2, \ldots, x_p$ denotes input variables. Fig. 4 is an illustration of a typical HFNT. Similarly, Fig. 5 is an illustration of a typical node in an HFNT.

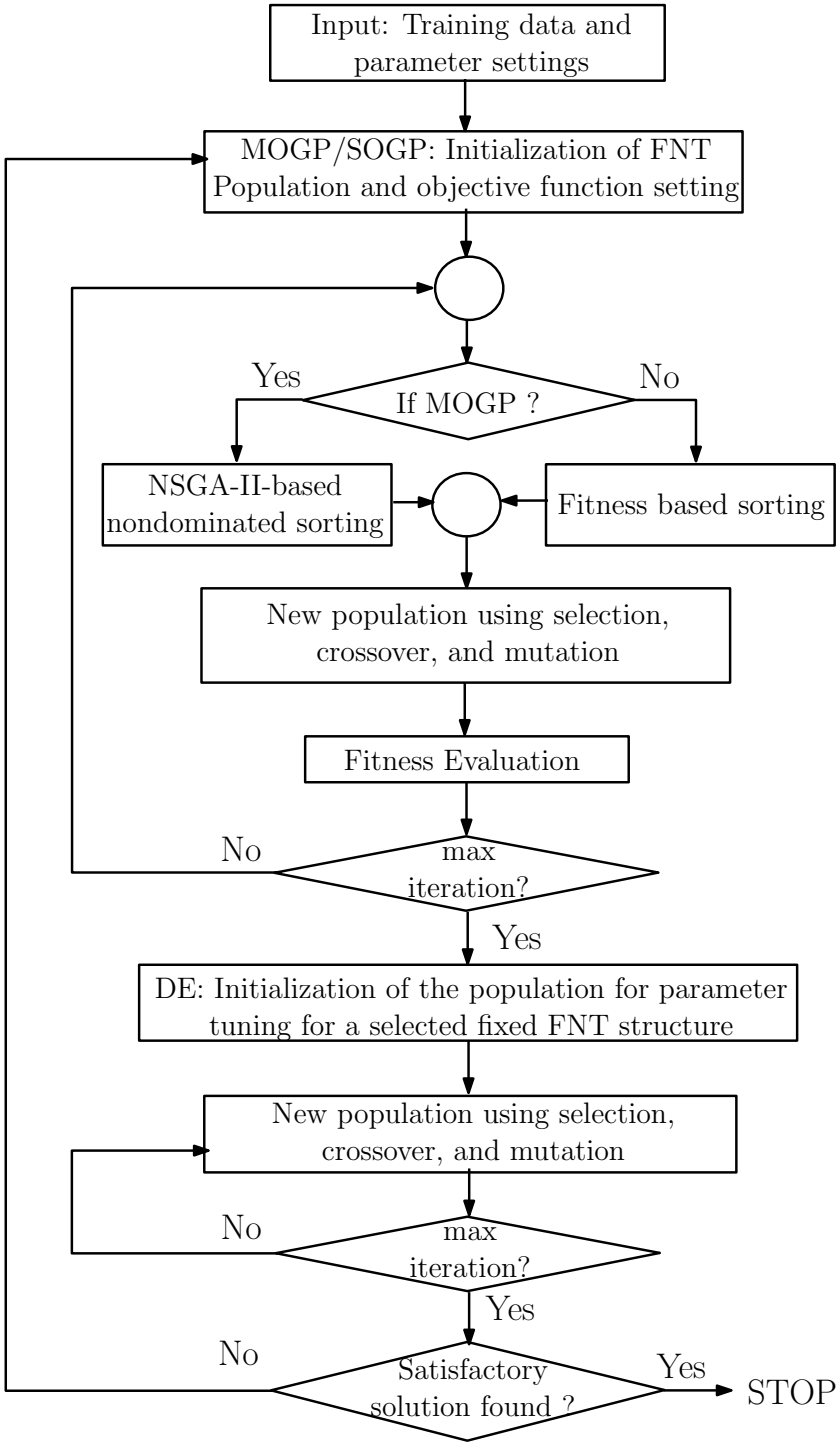


Figure 3: Two phase construction of the heterogeneous flexible neural tree.

Table 1: Set of activation function used in neural tree construction

Activation-function	k	Expression for $\varphi_i^k(a,b,x)$
Gaussian Function	1	$\varphi(x,a,b) = \exp\left(-((x-a)^2)/(b^2)\right)$
Tangent-Hyperbolic	2	$\varphi(x) = (e^x - e^{-x})/(e^x + e^{-x})$
Fermi Function	3	$\varphi(x) = 1/(1+e^{-x})$
Linear Fermi	4	$\varphi(x,a,b) = a \times 1/((1+e^{-x})) + b$
Linear Tangent-hyperbolic	5	$\varphi(x,a,b) = a \times (e^x - e^{-x})/(e^x + e^{-x}) + b$
Bipolar Sigmoid	6	$\varphi(x,a) = (1-e^{-2xa})/(a(1+e^{-2xa}))$
Unipolar Sigmoid	7	$\varphi(x,a) = (2\|a\|)/(1+e^{-2\|a\|x})$

The i-th computational node (Fig. 5) of a tree (say i-th node in Fig. 4) receives n^i inputs (denoted as z_j^i) through n^i connection-weights (denoted as w_j^i) and takes two adjustable parameters a^i and b^i that represents the arguments of the activation function $\varphi_i^k(.)$ at that node. The purpose of using an activation function at a computational node is to limit the output of the computational node within a certain range. For example, if the i-th node contains a Gaussian function $k = 1$ (Table 1). Then, its output y_i is computed as:

$$y_i = \varphi_i^k(a_i, b_i, o_i) = \exp\left(-\left(\frac{o_i - a_i}{b_i}\right)\right) \tag{3}$$

where o_i is the weighted summation of the inputs z_j^i and weights w_j^i ($j = 1$ to n^i) at the i-th computational node (Fig. 5), also known as excitation of the node. The net excitation o^i of the i-th node is computed as:

$$o_i = \sum_{j=1}^{n^i} w_j^i z_j^i \tag{4}$$

where $z_j^i \in \{x_1, x_2, \ldots, x_p\}$ or, $z_j^i \in \{y_1, y_2, \ldots, y_m\}$, i.e., z_j^i can be either an input feature (leaf node value) or the output of another node (a computational node output) in the tree. Weight w_j^i is the connection weight of real value in the range $[w_l, w_u]$. Similarly, the output of a tree y is computed from the root node of the tree, which is recursively computed by computing each node's output using (3) from right to left in a depth-first method.

The fitness of a tree depends on the problem. Usually, learning algorithm uses *approximation error*, i.e., MSE (1). Other fitness measures associated with the tree are *size* and *diversity index*. Tree size is the number of nodes (excluding root node) in a tree, e.g., the number of computational nodes and leaf nodes in the tree in Fig. 4 is 11 (three computational nodes and eight leaf-nodes). Hence, in this work, instead the number of free parameter count $k(\mathbf{w})$, the tree size was used as complexity indicator.

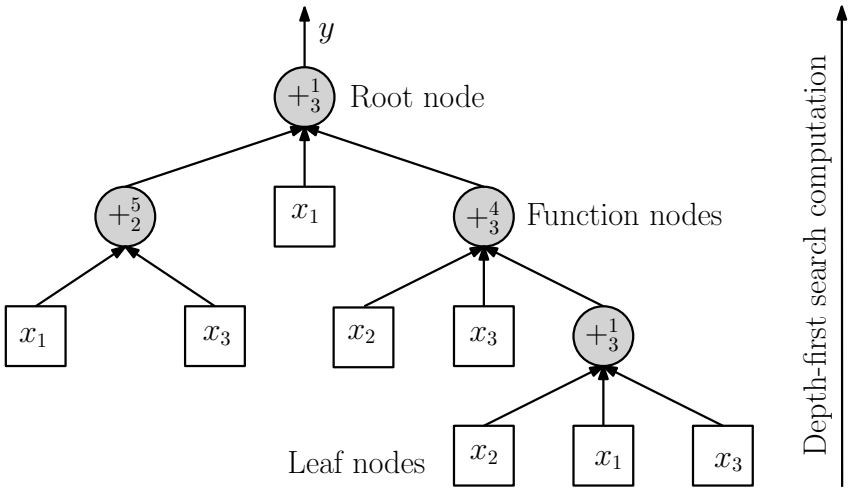


Figure 4: Typical representation of a neural tree $G = F_G \cup T_G$ whose function instruction set $F_G = \{+_3^1, +_2^4, +_3^5\}$ and terminal instruction set $T_G = \{x_1, x_2, x_3, x_4\}$.

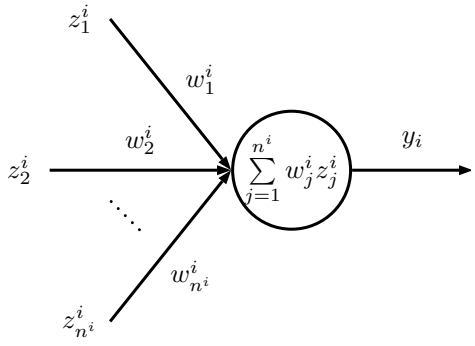


Figure 5: Illustration of a computational node. The variable n^i indicates the number of inputs z_j^i and weights w_j^i received at the i-th node and the variable y^i is the output of the i-th node.

The number of distinct activation functions (including root node function) randomly selected from a set of activation functions gives the diversity index of a tree. Total activation functions (denoted as k in $+_j^k$) selected by the tree in Fig. 4 is three ($+_3^1, +_3^4,$ and $+_3^5$). Hence, its diversity index is three.

3.2 Near optimal tree: structure and parameter learning

A tree that offers the lowest approximation error and the simplest structure is a near optimal tree, which can be obtained by using an evolutionary algorithm such as GP [9], PIPE [10], GEP [11], MEP [12], and so on. To optimize tree parameters, algorithms such as genetic algorithm [13], evolution strategy [13], artificial bee colony [14], PSO [15], DE [16], and so on can be used.

3.2.1 Structure Optimization

Initial Population Two fitness measure was considered: *approximation error* E minimization and number of free parameter count $k(\mathbf{w})$ minimization or tree size minimization (in this chapter tree size minimization, instead $k(\mathbf{w})$, was considered). These two objectives cannot be achieved simultaneously. Hence, during the *structure-tuning* phase, an initial population W_g^0 of random tree was constructed and sorted according to non-dominance described in [17].

Selection In selection operation, a *mating pool* W_g^p of size $size(W_g^0)/2$ was obtained using *binary tournament selection* that selects two candidates randomly at a time from a population W_g^t and the best solution (according to its rank and crowding distance) is copied into the mating pool W_g^p. This process is continued until the mating pool becomes full.

Generation An offspring population W_g^c is generated by using the individuals of the mating pool W_g^p. Two distinct individuals (parents) are randomly selected from the mating pool to create new individuals using genetic operators crossover and mutation for offspring population W_g^c.

Crossover In crossover operation, randomly selected sub-trees of two parent trees are swapped. The swapping includes the exchange of nodes. A detailed description of the crossover operation in genetic programming is available in [13,18]. The crossover operation is selected with a crossover probability pc.

Mutation The mutation operators used in tree are as follows [13, 18]:

a) Replacing a randomly selected terminal $x_i \in T$ with a newly generated terminal $x_j \in T$ for $j \neq i$.

b) Replacing all terminal nodes of a tree with a new set of terminal nodes derived from T.

c) Replacing a randomly selected node $N_i \in F$ with a newly generated node $N_j \in F$ for $j \neq i$.

d) Replacing a randomly selected terminal node $x_i \in T$ with a newly generated node $N_i \in F$.

e) Deleting a randomly selected terminal node $x_i \in T$ or deleting a randomly selected node $N_i \in F$.

The mutation operation is selected with a probability pm and the type of mutation operator (a, or b, or c, or d, or e) is selected randomly during the mutation operation.

Recombination The offspring population W_g^c and the main population W_g^t are combined together making a combined population W_g^g.

Elitism In this step, $size(W_g^c)$ worst individuals are weeded out from the combined population W_g^g. In other words, $size(W_g^t)$ best individuals are propagated to new generation $t+1$ as the main population W_g^{t+1}.

3.2.2 Parameter-tuning

In *parameter-tuning* phase, a single objective, i.e., approximation error was used in optimization of HFNT parameter by DE. The tree parameters such as weights of tree edges and arguments of activation functions were encoded into a vector $\mathbf{w} = \langle w_1, w_2, \ldots, w_n \rangle$ for the optimization. In addition, a cross-validation (CV) phase was used for statistical validation of HFNTs.

For parameter tuning, DE version "DE/rand-to-best/1/bin" was used [16]. The basic principle of the DE is as follows. First, an initial population matrix $W_h^t = (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m)$ at the iteration $t = 0$ is randomly initialized. The population W_h^t contains m solution vectors. A solution vector $\mathbf{w}$ in the population is an n-dimensional vector representing the free parameters of a model. Secondly, the population W_h^{t+1} is created using binomial trials. Hence, to create a new solution vector for the population W_h^{t+1}, three distinct solution vectors $\mathbf{w}^a$, $\mathbf{w}^b$, and $\mathbf{w}^c$ and the best solution vector $\mathbf{w}^g$ are selected from the

population W_h^t. Then, for a random index $k \in [1, n]$ and for the selected trail vector $\mathbf{w}^a = \langle w_1^a, w_2^a, \ldots, w_n^a \rangle$, the j-th variable of the modified trial vector $\mathbf{w}^{a'}$ is created as:

$$w_j^{a'} = \begin{cases} w_j^a + \delta(w_j^g - w_j^a) + \delta(w_j^b - w_j^c) & \text{if } r_j < cr \parallel j = k \\ w_j^a & \text{if } r_j \geq cr \end{cases} \tag{5}$$

where $r_j \in [0, 1]$ is a uniform random sample, $cr \in [0, 1]$ is the crossover rate, and $\delta \in [0, 2]$ is the differential weight factor. Similarly, all the variables $j = 1$ to n of the trail vector $\mathbf{w}^{a'}$ is created using (5). After creation of the modified trail vector $\mathbf{w}^{a'}$, it is *recombined* as:

$$\mathbf{w}^a = \begin{cases} \mathbf{w}^{a'} & \text{if } c_f(\mathbf{w}^{a'}) < c_f(\mathbf{w}^a) \\ \mathbf{w}^a & \text{if } c_f(\mathbf{w}^{a'}) \geq c_f(\mathbf{w}^a) \end{cases} \tag{6}$$

where c_f is the function that returns the fitness of a solution vector using (1). The application of the DE operators *selection*, *crossover*, and *recombination* are repeated until an optimal solution vector $\mathbf{w}^*$ is found.

3.3 Input feature analysis

Feature analysis was conducted to understand the significance of the input features. For this purpose, at first, M many HFNT models were created. Second, two performance dimensions were used: feature selection rate R and predictability score P. Feature selection rate R describes the total number of times a particular input feature set was appeared in the list that was prepared out all M models. The feature selection rate is defined as:

$$R_j = \frac{1}{M} \sum_{i=1}^{M} \mathbb{I}(Z_j) \tag{7}$$

where R_j is the selection rate of j-th input feature set $Z_j \in \mathbf{Z}$, which is a power set $\mathbf{Z} = \mathcal{P}(Z) - \emptyset$ and $Z = \{z_1, z_2, \ldots, z_p\}$ is an input feature set; M is the total number of models in the list; and function $\mathbb{I}(Z_j)$ is an identity function that returned "1" if j-th input-feature set Z_j is selected by the i-th model, otherwise, returned "0." Feature selection rate R equal to one is the highest (100% selection rate) and R equal to zero is the lowest (0% selection rate). In other words, the value of selection rate R equal to one means an input feature set was selected by all the models in the prepared list, and the value of R equal to zero means an input feature set was selected by none of the models in the prepared list.

Since the models in the list may not be equal in their performances, the predictability score P based on the MSEs/RMSEs of the models was computed. The predictability score P describes the predictability of an input feature set. To compute predictability score P

of j-th input-feature set Z_j, at first, the fitness F_j of the corresponding input-feature set Z_j was computed as:

$$F_j = \begin{cases} \sum_{i=1}^{M} E_i \cdot \mathbb{I}(Z_j), & \text{if } |Z_j| = 1 \\ \sum_{i=1}^{M} E_i \cdot \mathbb{I}(Z_j) / \sum_{i=1}^{M} \mathbb{I}(Z_j), & \text{if } |Z_j| > 1 \end{cases} \tag{8}$$

where E_i is the MSEs/RMSEs of i-th model. For $|Z_j|$ equal to one, fitness F_j is the sum of MSEs/RMSEs, and for $|Z_j|$ greater than one, fitness F_j is the average of MSEs/RMSEs of all models that selects subset Z_j. Then, the predictability score P_j corresponding to an input-feature set Z_j was computed by normalizing the fitness as:

$$P_j = \frac{F_j}{\max\limits_{j=1 \text{ to } |\mathbf{Z}^s|}(F_j)} \tag{9}$$

where function $\max(\cdot)$ determines the maximum fitness value from all F_j and $\mathbf{Z}^s$ is the set of selected feature sets. Similar to the selection rate R, the predictability score P equal to one indicates that the feature set has the highest impact on the predictability of the model and predictability score P equal to zero has the lowest.

3.4 Results

3.4.1 HFNT performance on classification datasets

In this work, Friedman test was conducted to examine the significance of the algorithms. For this purpose, the classification accuracy (test results) was considered. The average ranks obtained by each method in the Friedman test is shown in Table 2. The Friedman statistic at $\alpha = 0.05$ (distributed according to chi-square with 2 degrees of freedom) is 5.991, i.e., $\chi^2_{(\alpha,2)} = 5.991$. The obtained test value Q according to Friedman statistic is 6. Since $Q > \chi^2_{(\alpha,2)}$, then the *null hypothesis* that "there is no difference between the algorithms" is *rejected*. In other words, the computed p-value by Friedman test is 0.049787 which is less than or equal to 0.05, i.e., p-value $\leq \alpha$-value. Hence, we reject the null hypothesis.

Table 2 describes the significance of differences between the algorithms. To compare the differences between the best rank algorithm in Friedman test, i.e., between the proposed algorithm HFNT$^{\mathrm{M}}$ and the other two algorithms, Holm's method [19] was used. Holm's method rejects the hypothesis of equality between the best algorithm (HFNT$^{\mathrm{M}}$) and other algorithms if the p-value is less than α/i, where i is the position of an algorithm in a list sorted in ascending order of z-value (Table 3). From the post hoc analysis, it was observed that the proposed algorithm HFNT$^{\mathrm{M}}$ outperformed both HDT [20] and FNT [20] algorithms.

Table 2: Average rankings of the algorithms

Algorithm	Ranking
HFNT$^{\mathbf{M}}$	1.0
HDT	2.5
FNT	2.5

Table 3: Post Hoc comparison between HFNT$^{\mathrm{M}}$ and other algorithms for $\alpha = 0.1$

i	algorithm	z	p	α/i	Hypothesis
2	HDT	2.12132	0.033895	0.05	rejected
1	FNT	2.12132	0.033895	0.1	rejected

3.4.2 HFNT performance on regression datasets

For regression datasets, Friedman test was conducted to examine the significance of the algorithms. For this purpose, the best test MSE was considered of the algorithms MLP, ANFIS-SUB, TSK-IRL, LINEAR-LMS, LEL-TSK, and METSK-HD^{e} and the best test MSE of algorithm HFNT$^{\mathrm{M}}$ was considered. The average ranks obtained by each method in the Friedman test is shown in Table 4. The Friedman statistic at $\alpha = 0.05$ (distributed according to chi-square with 5 degrees of freedom) is 11, i.e., $\chi^2_{(\alpha,5)} = 11$. The obtained test value Q according to Friedman statistic is 11. Since $Q > \chi^2_{(\alpha,5)}$, then the *null hypothesis* that "there is no difference between the algorithms" is *rejected*. In other words, the computed p-value by Friedman test is 0.05 which is less than or equal to 0.05, i.e., p-value $\leq \alpha$-value. Hence, we reject the null hypothesis.

Table 4: Average rankings of the algorithms

Algorithm	Ranking
HFNT$^{\mathbf{M}}$	1.5
METSK-HD^{e}	2.75
LEL-TSK	3.25
LINEAR-LSM	3.5
MLP	4.5
ANFIS-SUB	5.5

From the Friedman test, it is clear that the proposed algorithm HFNT$^{\mathrm{M}}$ performed best among all the other algorithms. However, in the post-hoc analysis presented in Table 5 describes the significance of difference between the algorithms. For this purpose,

we apply Holm's method [19], which rejects the hypothesis of equality between the best algorithm (HFNT^{M}) and other algorithms if the p-value is less than α/i, where i is the position of an algorithm in a list sorted ascending order of z-value (Table 5).

In the obtained result, the equality between ANFIS-SUB, MLP and HFNT^{M} was rejected, whereas the HFNT^{M} equality with other algorithms can not be rejected with $\alpha = 0.1$, i.e., with 90% confidence. However, the p-value shown in Table 5 indicates the quality of their performance and the statistical closeness to the algorithm HFNT^{M}. It can be observed that the algorithm METSK-HD^{e} performed closer to algorithm HFNT^{M}, followed by LEL-TSK, and LINEAR-LSM.

Table 5: Post Hoc comparison between HFNT^{M} and other algorithms for $\alpha = 0.1$.

i	algorithm	z	p	α/i	Hypothesis
5	ANFIS-SUB	3.023716	0.002497	0.02	rejected
4	MLP	2.267787	0.023342	0.025	rejected
3	LINEAR-LSM	1.511858	0.13057	0.033	
2	LEL-TSK	1.322876	0.185877	0.05	
1	METSK-HD^{e}	0.944911	0.344704	0.1	

3.4.3 HFNT performance on real-world application

In this chapter, HFNT was also applied to predict die filling performance of pharmaceutical granules and to identify significant die filling process variables. The performance of the HFNT was compared with other CI techniques: multilayered perceptron (MLP), Gaussian process regression (GPR), and reduced error pruning tree (REP-Tree). The accuracy of the CI model was evaluated experimentally using die filling as a case study. The die filling experiments were performed using a model shoe system and three different grades of microcrystalline cellulose (MCC) powders (MCC PH 101, MCC PH 102, and MCC DG).

Table 6 describes the performance of the best models created by using HFNT, MLP, GPR, and REP-Tree. In Table 6, the generated models are arranged in descending order (the highest accuracy to the lowest accuracy) of their test correlation values. It may be observed that the performance of the models created using HFNT (when compared the test accuracies, i.e., correlation values) was better than the other CI techniques. Hence, the detail observation of the model Nos. 1, 2, and 3, which were created using HFNT are provided.

Table 7 describes the feature analysis results performed for the individual input features. The significance of individual features true density, d50, granule size, and shoe speed was examined. The features true density and d50 represents the powder proper-

Table 6: Performance of the prediction models and validation over 10-FCV.

Model No.	Model Type	Mean of RMSEs		Mean of r		Std over r		Model Complexity[1]	Selected Features[2]
		Train	Test	Train	Test	Train	Test		
1	HFNT	2.0206	2.0571	0.93	0.95	0.0087	0.0383	43	1, 2, 3, 4
2		2.3891	2.3934	0.91	0.91	0.0083	0.0617	34	2, 3, 4
3		2.5491	2.2618	0.88	0.91	0.0078	0.0563	32	3, 4
4	REP-Tree	2.5751	3.1637	0.88	0.82	-	-	99	1, 2, 3, 4
5	GPR	2.9632	3.4023	0.86	0.79	-	-	-	1, 2, 3, 4
6	MLP	3.3687	3.4427	0.81	0.79	-	-	-	1, 2, 3, 4

Note: [1]Complexity is the sum of total nodes in the created tree-model.

Table 7: Significance of individual input features.

#	Input Features set	Selection Rate (R)	Predictability Score (P)
1	Z_1= True density	0.55173	0.541356
2	Z_2= d50	0.62069	0.586262
3	Z_3= Granule size	1	1
4	Z_4= Shoe speed	0.86207	0.92563

ties. Whereas, the granule size and shoe speed represents the die filling process variables. It can be observed that the selection rate and predictability score of d50 (0.62069 and 0.58626) were higher than that of the selection rate and predictability score of true density (0.55173 and 0.54136). Therefore, d50 possess comparatively higher importance as a powder property than that of the true density.

4 Metaheuristic design of fuzzy inference system

Fuzzy inference system (FIS) has the ability to model uncertain, incomplete, and noisy data more efficiently than the neural network. In the past decades, it has been used for modeling complex real-world problems. Metaheuristic design of fuzzy, particularly the use of evolutionary algorithms, has given a significant contribution in the tuning of FISs: Mamdani-type and Takagi-Sugano-Kang (TSK)-type. Moreover, neuro-fuzzy paradigm, which combines both FNN and FIS allowed us to improve the approximation ability. One basic issue with FIS is in handling input dimensionality, where hierarchical design has played a crucial role in addressing these problems to a great extent. Additionally, the multiobjective optimization of FIS has addressed the interpretation and accuracy trade-offs. This chapter summarizes various FIS paradigms.

4.1 Fuzzy inference system

FRBSs are composed of the following components: 1) A *knowledge base* (KB), which contains fuzzy rules of the form IF-THEN, i.e.,

IF a set of conditions is satisfied
THEN a set of consequent can be inferred

2) An *inference engine*, which includes *fuzzification* module that fuzzify crisp input data into fuzzy sets, and the inference engine also does the reasoning to infer knowledge from KB. 3) A *defuzzification* part that translate inferred knowledge from KB by inference engine into a rule action (crisp output). An illustration of such discussion is shown in Fig. 6.

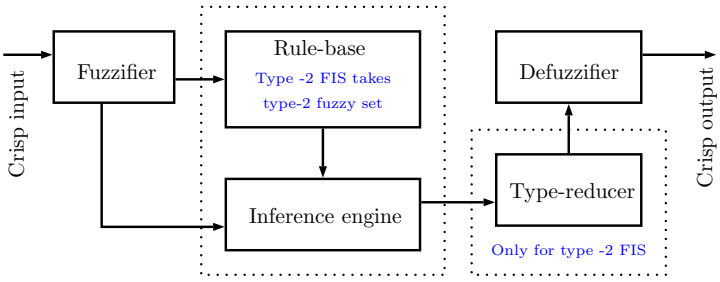


Figure 6: Typical fuzzy inference system.

4.1.1 Type-1 TSK-fuzzy inference system

A TSK-type FIS is governed by IF–THEN rule of the form [21]:

$$R^i : \text{IF } x_1 \text{ is } A_1^i \text{ and } \ldots \text{ and } x_{p^i} \text{is } A_{p^i}^i \text{ THEN } y^i \text{ is } B^i \tag{10}$$

where R^i is the i-th rule in a FIS, A^i is a T1FS, B^i is a function of an input vector $\mathbf{x} = \langle x_1, x_2, \ldots, x_{p^i} \rangle$ that returns a crisp output y^i, and p^i is the total number of inputs presented at the i-th rule. Note that the number of inputs may vary from rule-to-rule. Hence, the dimension of inputs at a rule is denoted as p^i. In TSK, function B^i is usually expressed as:

$$B^i = c_0^i + \sum_{j=1}^{p^i} c_j^i x_j, \tag{11}$$

where c_j^i for $j = 0$ to p^i is the free parameters at the consequent part of a rule. The basic building blocks of a FIS is shown in Fig. 6 whose defuzzified crisp output is computed as follows. First, the inference engine fires the rules from rule-base. The firing strength f^i of the i-th rule is computed as:

$$f^i = \prod_{j=1}^{p^i} \mu_{A_j^i}(x_j) \tag{12}$$

where $\mu_{A_j^i}$ is the value of j-th T1FS MF at the i-th rule. Then, the defuzzified output $\hat{y}$ of FIS is computed as:

$$\hat{y} = \frac{\sum_{i=1}^{M} B^i f^i}{\sum_{i=1}^{M} f^i} \tag{13}$$

where M is the number of rules in the rule-base.

An example of T1FS A is illustrated in Fig. 7(a), which has the following form:

$$\mu_A(x) = \frac{1}{1 + \left(\frac{x-m}{\sigma}\right)} \tag{14}$$

where m and σ are the center and the width respectively of MF $\mu_A(x)$.

4.1.2 Type-2 TSK-fuzzy inference system

A T2FS $\tilde{A}$ is characterized by a 3-dimensional membership function [22]. The three axes of T2FS are defined as follows. The x-axis is called primary variable, the y-axis is called secondary variable (or primary MF, which is denoted by u), and the z-axis is called the MF value (or secondary MF value, which is denoted by μ). Hence, in a universal set X, a T2FS $\tilde{A}$ has the form:

$$\tilde{A} = \left\{ \left((x,u), \mu_{\tilde{A}}(x,u) \right) | \, \forall x \in X, \forall u \in [0,1] \right\}. \tag{15}$$

The MF value μ has a 2-dimensional support called the *footprint of uncertainty* of $\tilde{A}$, which is bounded a lower membership function (LMF) $\underline{\mu}_{\tilde{A}}(x)$ and an upper membership function (UMF) $\bar{\mu}_{\tilde{A}}(x)$. Such form of T2FS that is bounded by lower and upper MFs is called interval type-2 FS (IT2FS). The *footprint of uncertainty* is the area enclosed within the LMF and the UMF (Fig. 7(b)). A Gaussian function with uncertain mean within $[m_1, m_2]$ and standard deviation σ is called an interval type-2 MF (Fig. 7(b)), which is expressed as:

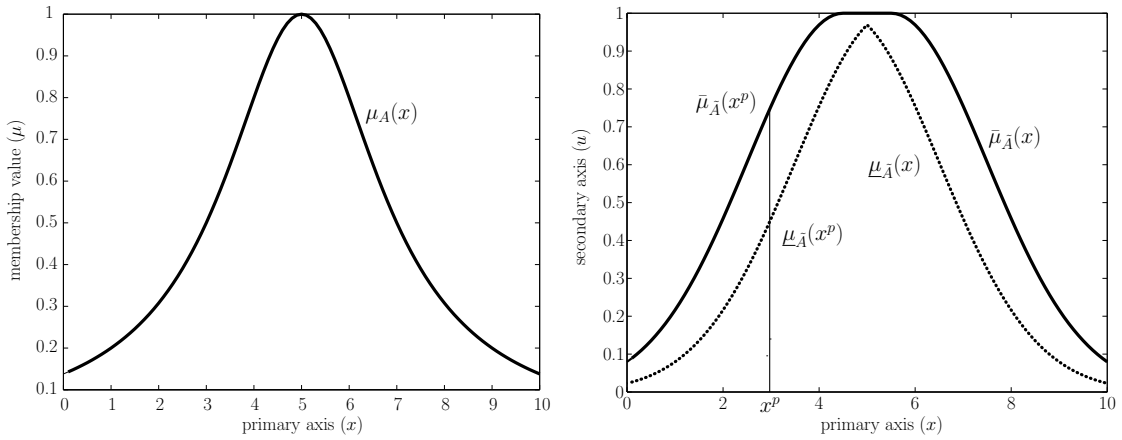
$$\mu_{\tilde{A}}(x, m, \sigma) = \exp\left(-\frac{1}{2} \left(\frac{x-m}{\sigma} \right) \right), \quad m \in [m_1, m_2]. \tag{16}$$

The LMF of a IT2FS can be defined as [23]:

$$\underline{\mu}_{\tilde{A}}(x) = \begin{cases} \mu_{\tilde{A}}(x, m_2, \sigma), & x \leq (m_1 + m_2)/2 \\ \mu_{\tilde{A}}(x, m_1, \sigma), & x > (m_1 + m_2)/2 \end{cases} \tag{17}$$

and the UMF can be defined as [23]:

$$\bar{\mu}_{\tilde{A}}(x) = \begin{cases} \mu_{\tilde{A}}(x, m_1, \sigma), & x < m_1 \\ 1, & m_1 \leq x \leq m_2 \\ \mu_{\tilde{A}}(x, m_2, \sigma), & x > m_2 \end{cases} \tag{18}$$



(a) (b)

Figure 7: Fuzzy membership functions. (a) Type-1 MF (14) with mean $m = 5.0$ and $\sigma = 2.0$. (b)Type-2 Fuzzy MF with fixed $\sigma = 2.0$ and means $m_1 = 4.5$ and $m_2 = 5.5$. UMF $\bar{\mu}_{\tilde{A}}(x)$ as per (18) is in solid line and LMF $\underline{\mu}_{\tilde{A}}(x)$ as per (17) is in dotted line.

In Fig. 7(b), a point x^p along the x-axis of 3-dimensional T2FS MFs cuts the UMF and LMF along the y-axis, and the value of the type-2 MF is considered to be along the z-axis (not shown in Fig. 7(b)) are $\bar{\mu}_{\tilde{A}}(x^p)$ and $\underline{\mu}_{\tilde{A}}(x^p)$. Considering T2FS MFs, i-th IF–THEN rule of type-2 TSK-FIS for an input vector $\mathbf{x} = \langle x_1, x_2, \ldots, x_{p^i} \rangle$ takes the following form:

$$R^i : \text{IF } x_1 \text{ is } \tilde{A}_1^i \text{ and } \ldots \text{ and } x_{p^i} \text{ is } \tilde{A}_{p^i}^i \text{ THEN } y^i \text{ is } \tilde{B}^i \tag{19}$$

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

$$\tilde{B}^i = [c_0^i - s_0^i, c_0^i + s_0^i] + \sum_{j=1}^{p^i} [c_j^i - s_j^i, c_j^i + s_j^i] x_j, \tag{20}$$

where c_j^i for $j = 0$ to p^i is the free parameter at the consequent part of a rule and s_j^i for $j = 0$ to p^i is the deviation factor of the free parameter. The firing strength of IT2FS $F^i = [\underline{f}^i, \bar{f}^i]$ is computed as:

$$\underline{f}^i = \prod_j^{p^i} \underline{\mu}_{\tilde{A}_j^i} \quad \text{and} \quad \bar{f}^i = \prod_j^{p^i} \bar{\mu}_{\tilde{A}_j^i} \tag{21}$$

At this stage, inference engine fires the rule and the type-reducer reduces the T2FS to T1FS. Here, center of set type-reducer prescribed by Karnik [23] can be used for type reduction purpose. The center of set y_{cos} is computed as:

$$y_{cos} = \bigcup_{f^i \in F^i,\ b^i \in \tilde{B}^i} \frac{\sum_{i=1}^M f^i b^i}{\sum_{i=1}^M f^i} = [y_l, y_r], \tag{22}$$

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

$$y_l = \frac{\sum_{i=1}^{L} \bar{f}^i \underline{b}^i + \sum_{i=L+1}^{M} \underline{f}^i \underline{b}^i}{\sum_{i=1}^{L} \bar{f}^i + \sum_{i=L+1}^{M} \underline{f}^i}, \tag{23}$$

$$y_r = \frac{\sum_{i=1}^{R} \underline{f}^i \bar{b}^i + \sum_{i=R+1}^{M} \bar{f}^i \bar{b}^i}{\sum_{i=1}^{R} \underline{f}^i + \sum_{i=R+1}^{M} \bar{f}^i}, \tag{24}$$

where L and R are the switch point, determined by

$$\underline{b}^L \leq y_l \leq \underline{b}^{L+1} \text{ and } \bar{b}^R \leq y_r \leq \bar{b}^{R+1},$$

respectively. The defuzzified crisp output $\hat{y}$ is computed as:

$$\hat{y} = \frac{y_l + y_r}{2}. \tag{25}$$

5 Multiobjective hierarchical fuzzy inference trees

Introduction of the fuzzy-set (type-1) enabled the modeling of uncertain and noisy information and type-2 fuzzy set took this further ahead by allowing fuzzy membership function to be fuzzy itself. Therefore, the design of a fuzzy inference system (FIS) that can offer a viable trade-off between accuracy and complexity is the desirable design. To meet this objective, in this chapter, a multiobjective genetic programming (MOGP) for designing a hierarchical fuzzy inference tree (HFIT), which produces an optimum tree-like structure that accommodates simplicity by combining several low-dimensional fuzzy subsystems was proposed. Such design produces highly accurate FIS models. The constructions of HFIT took place in a two-phase construction manner.

First, a nondominated sorting based MOGP was applied to find an optimum tree structure. Then, differential evolution (DE), a metaheuristics algorithm, was applied for tuning the parameters of the tree structure, which are the membership function parameters and the free parameters at the consequent part of the rules. Such process of structure optimization using MOGP and parameter tuning using DE are repeated until an optimal HFIT was obtained. The HFIT offered an automatic feature selection because it uses MOGP for the self-organization of tree-like structures whose nodes are low-dimensional FISs that uses subsets derived from given input set.

The HFIT was studied in the context of both type-1 and type-2 FIS, and its performance was evaluated over six application problems. Moreover, the proposed multiobjective HFIT was compared with recently proposed FIS algorithms in literature such as McIT2FIS, TSCIT2FNN, SIT2FNN, RIT2FNS-WB, eT2FIS, MRIT2NFS, IT2FNN-SVR, etc. From the obtained results, it is evident that the proposed HFIT offers an

efficient and competitive alternative to the other data mining algorithm for the function approximation and the feature selection.

Hence, this chapter aims to address the followings:

1. Construction of a hierarchical tree-like arrangement of low-dimensional fuzzy systems using a coevolutionary approach, which involves multiobjective genetic programming for evolving a tree-like structure and metaheuristic for tuning parameters of the evolved tree.

2. Feature selection and the simplification of the rule-base in fuzzy subsystems by exploiting the dynamics of the genetic programming.

3. Implementation of the HFIT for both T1FIS and T2FIS and the comparative evaluation of their single objective and multiobjective orientations with recently proposed FIS algorithms from the literature.

5.1 Hierarchical fuzzy inference tree formation

A hierarchical fuzzy inference tree (HFIT), denoted as H, is a collection of FIS node set F_H and terminal node set T_H:

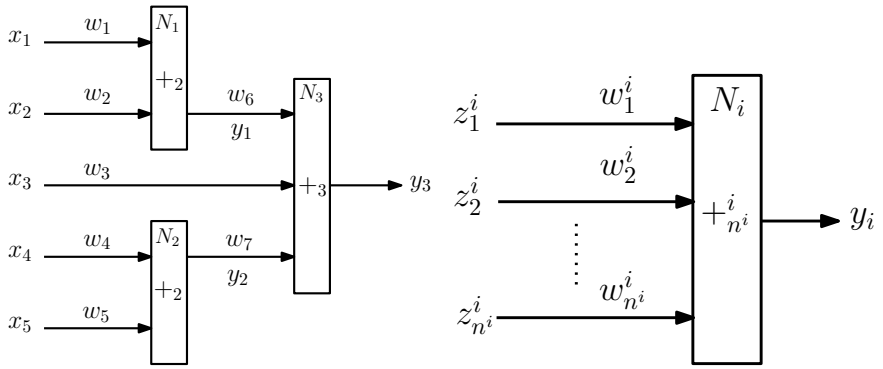
$$H = F_H \cup T_H = \{+_2, +_3, \cdots, +_{tn}\} \cup \{x_1, x_2, \ldots, x_p\} \tag{26}$$

where $+_j\,(j = 2, 3, \ldots, tn)$ denotes non-leaf instruction and has $2 \leq j \leq tn$ arguments. Leaf node's instruction $x_1, x_2, \ldots, x_p$ takes no argument and represents input variable/instruction. A typical HFIT is shown in Fig. 8(a); whereas, Fig. 8(b) is an illustration of i-th node N_i in an HFIT that takes n^i inputs. The inputs $z_j^i \in \{x_1, x_2, \ldots, x_p\}$ for $j = 1$ to n^i to the node N_i is either from the input layer or from another node in HFIT. Each node in HFIT receives a weighted input $x_i w_i$, where w_i is the weight. In this work, the weights in HFIT were set to 1.0. Hence, the inputs were not influenced by the weights of the tree edges. Thus, setting weights fixed to 1.0 led to the reduction in the total parameter count during the *parameter-tuning* phase.

5.2 Rule formation

Each node in HFIT is a FIS of either type-1 or type-2. Hence, the rules at a node were created as follows. Considering a reference to the node N_1 from Fig. 8(a) that has two arguments/inputs x_1 and x_2 and assuming that each input x_1 and x_2 has two T1FSs A_{11}^1, A_{12}^1 and A_{21}^1, A_{22}^1, respectively, the rules for T1FIS are generated as:

$$R_{ij}^1 : \text{IF } x_1 \text{ is } A_{1i}^1 \text{ and } x_2 \text{ is } A_{2j}^1 \text{ THEN}$$
$$y_{ij}^1 = c_{ij}^0 + c_{ij}^1 x_1 + c_{ij}^2 x_2, \text{ for } i = 1, 2 \text{ and } j = 1, 2.$$



(a) Two stage hierarchical tree (b) Structure of a node

Figure 8: Hierarchical fuzzy inference tree. (a) Complete tree with three nodes N_1, N_2, and N_3 and with inputs x_1, x_2, x_3, x_4, and x_5. (b) Illustration of the i-th node N_i that has n^i inputs $z_j^i \in [x_1, \ldots, x_n]$ for $j = 1$ to n^i and output y_i.

The output y^1 of node N_1 is computed as:

$$y^1 = \frac{\sum_{i=1}^{2}\sum_{j=1}^{2}\sigma_{ij}^1 y_{ij}^1}{\sum_{i=1}^{2}\sum_{j=1}^{2}\sigma_{ij}^1} \tag{27}$$

where

$$\sigma_{ij}^1 = \mu_{A_{1i}}(x_1)\mu_{A_{2j}}(x_2) \text{ for } i = 1, 2 \text{ and } j = 1, 2. \tag{28}$$

Similarly, the output of the node N_2 is computed. The output y^3 of the HFIT shown in Fig. 8(a) is computed from the node N_3 that revives inputs y^1 and y^2 and x_3, where y^1 and y^2 are the outputs of the nodes N_1 and N_2, respectively.

If the nodes of HFIT in Fig. 8(a) are type-2 nodes; then, assuming that the node N_1 has two T2FSs $\tilde{A}_{11}^1, \tilde{A}_{12}^1$ and $\tilde{A}_{21}^1, \tilde{A}_{22}^1$, respectively, the rules for T2FIS are generated as:

$$\begin{aligned}&R_{ij}^1 : \text{IF } x_1 \text{ is } \tilde{A}_{1i}^1 \text{ and } x_2 \text{ is } \tilde{A}_{2j}^1 \text{ THEN}\\ &y_{ij}^1 = [c_{ij}^0 - s_{ij}^0] + [c_{ij}^1 - s_{ij}^1]x_1 + [c_{ij}^2 - s_{ij}^2]x_2,\\ &\text{for } i = 1, 2 \text{ and } j = 1, 2\end{aligned}$$

and the firing strength of the rule at the node N_1 is computed as described in (21). Moreover, the type-reduction of the node is performed by using (22), where left and right intervals are computed as per (23) and (24). Finally, the output of the node N_1 is computed as per (25). Subsequently, the output of the type-2 HFIT shown in Fig. 8(a) is computed from the node N_3.

5.2.1 Structure tuning

An HFIT that offers the lowest approximation error and simplest structure is the desirable solution. To obtain such set of Pareto-optimal solution nondominated sorting algorithm was applied (Section 3.2.1).

5.2.2 Parameter tuning

In structure tuning phase, an optimum phenotype (HFIT) is derived with the parameters being initially fixed by random guess. Hence, the obtained phenotype is further tuned in the parameter tuning phase by using a parameter optimization algorithm. To tune the parameters of the obtained phenotype, its parameters are mapped onto a genotype, i.e., onto a real vector, called solution vector.

The selection of the best phenotype in a single objective training is solely based on the comparison of the RMSEs, but selecting a solution in a multiobjective training is a difficult choice. In this work, after the multiobjective training of HFIT, the best solution for parameter tuning was picked from the Pareto-front. Strictly, the solution that gave the best RMSE among the solutions in the Pareto-optimal set of rank one was chosen. Fig. 9 is an illustration of the solutions that belong to Pareto-front of rank one. The genotype mapping of the T1FIS and the T2FIS differ only with the respect to their number of parameters.

The T1FIS uses the MF mentioned in (14), which has two arguments m and σ, and each rule in T1FIS has p^i+1 variables at the consequent part as mentioned in (11), where p^i is the number of inputs to the i-th rule. On the other hand, since interval type-2 MF is bounded by a LMF and an UMF (Fig. 7(b)), it has two Gaussian means m_1 and m_2 and a variance σ that need to be optimized. The Gaussian means m_1 and m_2 for type-2 Gaussian MF (16) were defined as:

$$m_1 = m + \gamma\sigma \tag{29}$$

and

$$m_2 = m - \gamma\sigma, \tag{30}$$

where $\gamma \in [0, 1]$ was a random variable taken from uniform distribution and m was the center of Gaussian means m_1 and m_2 taken from [0, 1]. Similarly, σ of type-2 Gaussian MF (16) was taken from [0, 1]. The consequent part of T2FIS rule was computed as mentioned in (20), which led to $2 \times (p^i + 1)$ variables.

Assume that an HFIT (a tree like Fig. 8(a)) has k nodes, and each node in the phenotype takes $2 \leq p^i \leq tn$ inputs, where each input is partitioned into two fuzzy sets (membership functions). Then, the number of the fuzzy sets at a node is $2 \times p^i$. Since the number of inputs at a node is p^i and each input is partitioned into two fuzzy sets, the number of rules at a node is 2^{p^i}. Hence, the number of parameters at a T1FIS node is $[2^{p^i} \times (2 \times 2 \times (p^i + p^i + 1))]$ and the number of parameters at a T2FIS node is $[2^{p^i} \times (3 \times 2 \times p^i + 2 \times (p^i + 1))]$. Therefore, the total number of parameters in an HFIT is the summation of the number of parameters at all nodes in the tree. Assuming n is the

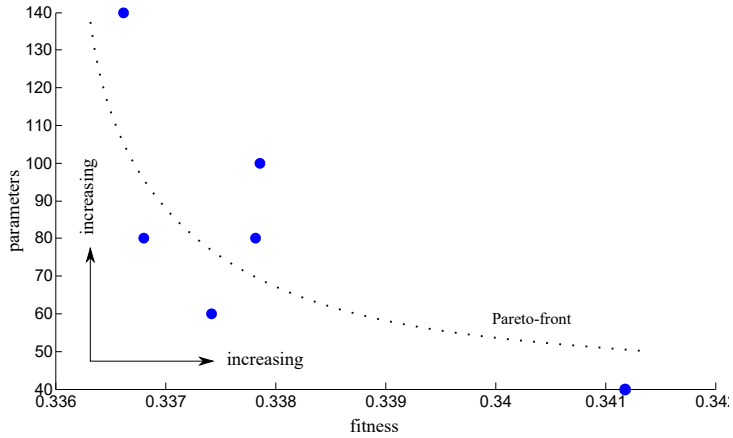


Figure 9: FIS fitness versus FIS parameters mapping across the Pareto-front.

total number of parameters in tree, the genotype or the solution vector $\mathbf{w}$ is expressed as:

$$\mathbf{w} = \langle w_1, w_2, \ldots, w_n \rangle. \tag{31}$$

where elements of the vector are mapped from the phenotype, i.e., from the tree. Therefore, to optimize parameter vector $\mathbf{w}$, meta-heuristic algorithms can be used. In this work, differential evolution (DE) [16] was used, which is discussed elaborately in Section 3.2.2.

5.3 Results

Benchmark data

Example 1—system identification: Online identification of the nonlinear system is a widely studied problem. The significance of this problem is evident from its usage in literature for the validation of the approximation algorithms [24–27]. An interesting paper describing several plant identification is available in [28]. The nonlinear system identification of the plant is described by the following nonlinear difference equation:

$$y_p(k+1) = \frac{y_p(k)}{1 + y_p(k)^2} + u^3(k), \tag{32}$$

where $[u(k), y_p(k)]$ is the input–output pair of the single input and the single output plant at the time k and $y_p(k+1)$ is the one step ahead prediction. Hence, the objective is to predict $y_p(k+1)$ of the system based on the sinusoidal input $u(k) = \sin(2\pi k/100)$ and the current output $y_p(k)$. Let us assign the input $x_1 = u(k)$ and the input $x_2 = y(k)$.

Example 2—mackey-Glass chaotic time series: A chaotic time series dataset, the Mackey-Glass chaotic time series, was used in this example, which was generated using the following delay differential equation:

$$\frac{dx(k)}{dk} = \frac{0.2x(k-\tau)}{1 + x^{10}(k-\tau)} - 0.1x(k), \tag{33}$$

where $\tau > 17$. In this example, the objective was to predict $x(k)$ using the past outputs of the time series as mentioned in [27, 29]. Hence, input–output pattern was of the form:

$$[x(k-24), x(k-18), x(k-12), x(k-6); x(k)]\,.$$

Let us say the inputs are $x_1 = x(k-24)$, $x_2 = x(k-18)$, $x_3 = x(k-12)$, and $x_4 = x(k-6)$. For the training of the proposed algorithms, a total of 1000 patterns were generated from $k = 124$ to 1123.

Example 3—abalone age prediction: In this example, a prediction problem was taken, in which the age of a person was predicted based on the physical measurements. The abalone dataset was collected from the UCI machine learning repository [30].

Example 4—box-Jenkins gas furnace problem: In this example, the Box and Jenkins gas furnace dataset that was taken from [31], which has 296 data samples. The objective of this example was to predict the CO_2 concentration from the gas-flow rate.

5.4 HFIT performance on benchmark data

Table 9 and Table 10 shows type-1 and type-2 HFIT performance comparison with algorithms from literature listed in Table 8.

5.5 HFIT performance on PLGA data

This example illustrates a pharmaceutical industrial problem related to PLGA dissolution profile prediction, which is a complex problem since a huge number of factors governs its dissolution-rate profile. As per the dataset provided in [50–52], this problem has a total of 300 potential factors that influences the PLGA protein particle's dissolution-rate [53], The input features are categorized into four groups (protein descriptor, plasticizer, formulation characteristics, and emulsifier), which has 85, 17, 98, 99, and 1 features, respectively. This problem has a very high noise and redundancy because data were obtained from various experimental measurements and instruments.

Table 11 shows a comparison of the results of the proposed T1HFIT$^{\mathrm{M}}$ and T2HFIT$^{\mathrm{M}}$ with the results of the algorithms such as multilayer preceptron (MLP), reduced error pruning tree (REP Tree), heterogeneous flexible neural tree (HFIT), and Gaussian process regression (GPR). It is evident from the results that the proposed algorithm predicted the PLGA dissolution profile with a lower number of features, and its approximation error was very competitive with the performance of other algorithms.

Table 8: Descriptions of the existing FIS algorithms adopted for the performance comparisons.

FIS	Algorithm	Ref.	Description	Type	Parameter tuning
Type–1	DyEFuNN	[32]	Dynamic evolving neural-fuzzy inference system	TSK	Least-square estimator
	D-FNN	[33]	Dynamic fuzzy neural networks	TSK	Backpropagation
	EFuNN	[34]	Evolving fuzzy neural networks	Mamdani	Widrow–Hoff least square
	FALCON	[35]	ART-based fuzzy adaptive learning control network	−−	Backpropagation
	GNN	[36]	Granular neural networks	−−	Genetic algorithm
	H-TS-FS	[37]	Hierarchical Tukagi–Sugno fuzzy system	TSK	Evolutionary programming
	HyFIS	[38]	Hybrid neural fuzzy inference system	−−	Gradient descent learning
	IFRS and AFRS	[39]	Incremental and aggregated fuzzy relational systems	Mamdani	Backpropagation
	RBF-AFA	[40]	Radial basis function based adaptive fuzzy systems	TSK	Gradient descent learning
	SaFIN	[41]	Self-adaptive fuzzy inference network	Mamdani	Levenberg-Marquardt
	SONFIN	[42]	Self-constructing neural fuzzy inference network	TSK	Backpropagation
	SuPFuNIS	[43]	Subsethood-product fuzzy neural inference system	−−	Gradient descent
	SVR-FM	[44]	Support-vector regression fuzzy model	TSK	Support vector regression
Type–2	eT2FIS	[45]	Evolving type-2 neural fuzzy inference system	Mamdani	Gradient descent learning
	IT2FNN-SVR-N/F	[27]	IT2FNN-support-vector regression-fuzzy and numeric	TSK	Support vector regression
	McIT2FIS-UM/US	[26]	Metacognitive interval type-2 neuro-FIS	TSK	Gradient descent learning
	NNT2FW	[46]	Type-1 and type-2 fuzzy BP neural networks	TSK	Backpropagation
	RIT2FNS-WB	[29]	Reduced IT2NFS-weighted bound-set	TSK	Gradient descent learning
	MRIT2NFS	[29]	Reduced IT2NFS-weighted bound-set	Mamdani	Gradient descent learning
	SEIT2FNN	[24]	Self-evolving IT2FIS	TSK	Kalman filter algorithm
	SIT2FNN	[47]	Simplified Interval Type-2 Fuzzy Neural Networks	TSK	gradient descent learning
	T2FLS	[48]	Interval type-2 fuzzy logic system (TSK and singleton)	TSK	−−
	T2FLS-G	[49]	Gradient-descent based IT2FIS tuning	TSK	Derivation-based learning
	TSCIT2FNN	[25]	Compensatory interval type-2 fuzzy neural network	TSK	Kalman filter algorithm

Table 9: Performance comparison of type-1 algorithm with Type-1 HFIT based on test RMSE

Example 1		**Example 2**		**Example 3**		**Example 4**	
Algorithm	RMSE	Algorithm	RMSE	Algorithm	RMSE	Algorithm	RMSE
SaFIN	0.012	NNT1FW	0.055	HS	3.16	T1-NFS	0.4074
SONFIN	0.0085	AFRS	0.0256	General	3.15	GNN-1	0.3114
T1HFIT$^{\mathbf{S}}$	0.0043	IFRS	0.0253	CCL	2.65	GNN-2	0.2983
T1HFIT$^{\mathbf{M}}$	0.0041	HTS-FS1	0.0129	Chen	2.59	**T1HFIT$^{\mathbf{S}}$**	0.2455
		HTS-FS2	0.0151	**T1HFIT$^{\mathbf{S}}$**	2.126	**T1HFIT$^{\mathbf{M}}$**	0.2838
		RBF-AFA	0.0128	**T1HFIT$^{\mathbf{M}}$**	2.348		
		HyFIS	0.01				
		D-FNN	0.008				
		SuPFuNIS	0.0057				
		T1HFIT$^{\mathbf{S}}$	0.0122				
		T1HFIT$^{\mathbf{M}}$	0.0119				

Table 10: Performance comparison of type-2 algorithms with Type-2 HFIT based on test RMSE

Example 1		**Example 2**		**Example 3**		**Example 4**	
Algorithm	RMSE	Algorithm	RMSE	Algorithm	RMSE	Algorithm	RMSE
T2TSKFNS	0.0324	T2FLS	0.043	RIT2NFS-WB	2.135	SEIT2FNN	0.269
T2FNN	0.0281	T2FLS (TSK)	0.043	McIT2FIS-UM	1.874	RIT2NFS-WB	0.353
SIT2FNN	0.0241	NNT2FW	0.039	SEIT2FNN	2.433	McIT2FIS-UM	0.314
RIT2NFS-WB	0.0151	SEIT2FNN1	0.003	McIT2FIS-US	1.839	McIT2FIS-US	0.318
MRI2NFS	0.0051	SEIT2FNN2	0.005	**T2HFIT$^{\mathbf{S}}$**	2.182	**T2HFIT$^{\mathbf{S}}$**	0.277
T2FLS-G	0.0379	**T2HFIT$^{\mathbf{S}}$**	0.009	**T2HFIT$^{\mathbf{M}}$**	2.143	**T2HFIT$^{\mathbf{M}}$**	0.284
SEIT2FNN	0.0022	**T2HFIT$^{\mathbf{M}}$**	0.006				
T2HFIT$^{\mathbf{S}}$	0.0034						
T2HFIT$^{\mathbf{M}}$	0.0028						

Table 11: Performance comparison.

Algorithm	Ref.	RMSE E_t	No. of Features
MLP	[50]	14.3	17
HFIT	[54]	13.2	15
REP Tree	[55]	13.3	15
GPR	[55]	14.9	15
MLP	[55]	15.2	15
MLP	[50]	15.4	11
T1HFIT$^{\mathbf{M}}$	This Chapter	18.6	7
T2HFIT$^{\mathbf{M}}$	This Chapter	15.2	**4**

6 Conclusions and future research

6.1 Conclusions

For the development of the proposed algorithms, feedforward neural network (FNN) and fuzzy inference system (FIS) paradigms were investigated. FNN and FIS faces challenges in predictive modeling for data that are insufficient, imbalanced, high-dimensional, and abundant. Thus, in the literature, several FNN, and FIS based approximation methods has been proposed. Moreover, the components of FNN (weights, structure, activation function, and learning algorithm) and FIS (rules base and membership function) are responsible for design of various algorithms (Chapters 2 and 4). Hence, this thesis proposed to optimize these components of FNN and FIS using an adaptive tree-like model and by using a two-phase construction approach for optimizing the tree. The proposed algorithms were the heterogeneous flexible neural tree (HFNT) and the hierarchical fuzzy inference tree (HFIT). The proposed algorithms were adaptive the following sense: the tree structure were determined automatically (by using the principle of natural selection), nodes of the tree were adapted automatically, and tree parameters were tuned.

Moreover, structure optimization and generalization ability are strongly correlated with each others. Therefore, FNN architecture optimization and for that matter various model designs were of particular interest to the research community. Additionally, FNN node optimization, learning parameter optimization, hybrid metaheuristic training to FNN, and multiobjective treatment to FNN optimization were of particular focus in the past. The proposed HFNT algorithm is an adaptive algorithm that accommodates all these forms of optimization strategies into a single entity (function approximation algorithm).

Similarly, for a connection-based FIS model (e.g., neuro-fuzzy and hierarchical fuzzy systems), the design of the structure holds the key to its approximation ability. Moreover,

parameter tuning of FIS rules and the interpretability–accuracy trade-off is utmost important issues in constructing a FIS model. The proposed HFIT algorithm is an adaptive algorithm that addresses multiobjective optimization of FIS structure by evolving a tree -like model, and parameter tuning by using metaheuristic in a two-phase construction manner.

HFNT was a FNN based model, which was a tree-like model whose nodes were neural nodes (Chapter 3). On the other hand, HFIT was a connection-based hierarchical FIS model (tree-like model) whose nodes were low-dimensional FIS of type-1 or type-2 (Chapter 5). The multiobjective evolutionary algorithm minimized the approximation error and the model's complexity simultaneously. Thus, the obtained tree structure of HFNT and HFIT were simple (small in the parameter count). Finally, the parameters were tuned by using differential evolution algorithm that further improved the model's approximation ability.

Both the algorithms were compared with the algorithms from the literature. The obtained results were the evidence of their (the proposed algorithms) superior performance over other algorithms. Moreover, the HFNT was used for modeling a real-world problem of pharmaceutical industry that had data that represented the dynamic environment of the pharmaceutical die filling process. It was found the HFNT offered significantly better result than other computational intelligence models such as multilayer perceptron and reduced error pruning tree. Similarly, the performance of HFIT (its version type-1 HFIT and type-2 HFIT) when tested over benchmark and real-world problems, it surpasses its competitor algorithms in the literature. Additionally, HFIT was used for modeling a real-world pharmaceutical problem that had a high-dimensional data that governed the prediction of drug desolation rate. It was found that the HFIT had a high prediction ability with a small set of features than other algorithms.

FNN and FIS are known as a universal approximator and it is known that for any problem (X, Y), a FIS model can be designed, which can approximate (X, Y) to equal degree that of a FNN model can, and vice versa [56]. Such equivalence of their approximation ability is evident from the results obtained in the experiments conducted in this thesis. The approximation ability of the HFNT and HFIT was fond very similar when they were used for predicting two problems: Mackey-Glass chaotic time series prediction and PLGA drug dissolution prediction. For the former, the approximation quality of the HFIT was found slightly better than the HFNT. On the other hand, the approximation quality of the HFNT was found slightly better than the HFIT for the latter. However, a significant improvement in feature selection was observed during HIT modeling because it was able to predicate to an equal degree of HFNT by using fewer number of features.

The proposed algorithms are capable of feature selection using the dynamics of the natural section and are able to approximate to a very high degree. In addition to that,

they offer very simple models. These are the desirable significant attributes to an efficient, effective, and robust algorithm. Therefore, the proposed algorithm stands as a viable alternative to the research community to date.

6.2 Future research directions

The proposed algorithms are efficient tools, which can be used for solving several high-dimensional and noisy real-world problems.

In this thesis, the tree structure was tuning using multiobjective genetic programming and the parameters were optimized by differential evolution. However, hybrid metaheuristic approach may be advantageous to use. Similarly, the model's structure, in this thesis, was represented as a tree-like. However, structure representation is an open research problem.

The proposed algorithms may be enhanced, modified to suit a particular class of problem, i.e., the nodes of the HFNT and HFIT may be replaced or designed according to the specification of the possible problems. Moreover, the fuzzy rules at the nodes of HFIT can be further optimized by using iterative learning methods or by using Pittsburgh approach.

Evolving fuzzy paradigm indicates an opportunity to extend HFNT and HFIT algorithms towards dynamic learning system, in which a dynamic mechanism may be employed to expand and contract a tree such that it can learn the knowledge contained in the incoming (on-line stream) data. Therefore, it can efficiently solve a broad range of dynamic problems (problems that has non-stationary/stream data).

References

[1] Y. Chen, B. Yang, and J. Dong, "Nonlinear system modelling via optimal design of neural trees," *Int. J. Neural Syst.*, vol. 14, no. 2, pp. 125–137, 2004.

[2] Y. Chen, B. Yang, J. Dong, and A. Abraham, "Time-series forecasting using flexible neural tree model," *Inform. Sci.*, vol. 174, no. 3, pp. 219–235, 2005.

[3] L. Zadeh, "Fuzzy sets," *Inform. Control*, vol. 8, no. 3, pp. 338 – 353, 1965.

[4] H. A. Hagras, "A hierarchical type-2 fuzzy logic control architecture for autonomous mobile robots," *IEEE Trans. Fuzzy Syst.*, vol. 12, no. 4, pp. 524–539, 2004.

[5] W. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bull. Math. Biol.*, vol. 5, no. 4, pp. 115–133, 1943.

[6] A. Jain, R. Duin, and J. Mao, "Statistical pattern recognition: a review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 1, pp. 4–37, 2000.

[7] G. Zhang, "Neural networks for classification: a survey," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 30, no. 4, pp. 451–462, 2000.

[8] R. Selmic and F. Lewis, "Neural-network approximation of piecewise continuous functions: application to friction compensation," *IEEE Trans. Neural Netw.*, vol. 13, no. 3, pp. 745–751, 2002.

[9] R. Riolo, J. H. Moore, and M. Kotanchek, *Genetic programming theory and practice XI.* Springer, 2014.

[10] R. Salustowicz and J. Schmidhuber, "Probabilistic incremental program evolution," *Evol. Comput.*, vol. 5, no. 2, pp. 123–141, 1997.

[11] C. Ferreira, *Gene expression programming: mathematical modeling by an artificial intelligence.* Springer, 2006, vol. 21.

[12] M. Oltean and C. Groşan, "Evolving evolutionary algorithms using multi expression programming," in *Advances in Artificial Life.* Springer, 2003, pp. 651–658.

[13] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing.* Springer, 2015.

[14] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm," *J. Global Optim.*, vol. 39, no. 3, pp. 459–471, 2007.

[15] J. Kennedy, R. C. Eberhart, and Y. Shi, *Swarm Intelligence.* Morgan Kaufmann, 2001.

[16] S. Das, S. S. Mullick, and P. Suganthan, "Recent advances in differential evolution–an updated survey," *Swarm Evol. Comput.*, vol. 27, pp. 1–30, 2016.

[17] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II," in *Parallel Problem Solving from Nature PPSN VI*, ser. Lecture Notes in Computer Science, M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. Merelo, and H.-P. Schwefel, Eds. Springer, 2000, vol. 1917, pp. 849–858.

[18] S. Bouaziz, H. Dhahri, A. M. Alimi, and A. Abraham, "Evolving flexible beta basis function neural tree using extended genetic programming & hybrid artificial bee colony," *Appl. Soft Comput.*, 2016.

[19] S. Holm, "A simple sequentially rejective multiple test procedure," *Scandinavian Journal of Statistics*, pp. 65–70, 1979.

[20] H.-J. Li, Z.-X. Wang, L.-M. Wang, and S.-M. Yuan, "Flexible neural tree for pattern recognition," in *Advances in Neural Networks–ISNN*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, vol. 3971, pp. 903–908.

[21] 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.

[22] J. M. Mendel, "On km algorithms for solving type-2 fuzzy set problems," *IEEE Trans. Fuzzy Syst.*, vol. 21, no. 3, pp. 426–446, 2013.

[23] N. N. Karnik, J. M. Mendel, and Q. Liang, "Type-2 fuzzy logic systems," *IEEE Trans. Fuzzy Syst.*, vol. 7, no. 6, pp. 643–658, 1999.

[24] C.-F. Juang and Y.-W. Tsao, "A self-evolving interval type-2 fuzzy neural network with online structure and parameter learning," *IEEE Trans. Fuzzy Syst.*, vol. 16, no. 6, pp. 1411–1424, 2008.

[25] Y.-Y. Lin, J.-Y. Chang, and C.-T. Lin, "A TSK-type-based self-evolving compensatory interval type-2 fuzzy neural network (TSCIT2FNN) and its applications," *IEEE Trans. Ind. Electron.*, vol. 61, no. 1, pp. 447–459, 2014.

[26] A. K. Das, K. Subramanian, and S. Sundaram, "An evolving interval type-2 neurofuzzy inference system and its metacognitive sequential learning algorithm," *IEEE Trans. Fuzzy Syst.*, vol. 23, no. 6, pp. 2080–2093, 2015.

[27] C.-F. Juang, R.-B. Huang, and W.-Y. Cheng, "An interval type-2 fuzzy-neural network with support-vector regression for noisy regression problems," *IEEE Trans. Fuzzy Syst.*, vol. 18, no. 4, pp. 686–699, 2010.

[28] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Netw*, vol. 1, no. 1, pp. 4–27, 1990.

[29] C.-F. Juang and K.-J. Juang, "Reduced interval type-2 neural fuzzy system using weighted bound-set boundary operation for computation speedup and chip implementation," *IEEE Trans. Fuzzy Syst.*, vol. 21, no. 3, pp. 477–491, 2013.

[30] M. Lichman, "UCI machine learning repository," 2013, http://archive.ics.uci.edu/ml Accessed on: 01.05.2016.

[31] G. E. P. Box and G. M. Jenkins, *Time Series Analysis, Forecasting and Control*. San Francisco, CA, USA: Holden–Day, 1976.

[32] 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.

[33] S. Wu and M. J. Er, "Dynamic fuzzy neural networks-a novel approach to function approximation," *IEEE Trans. Syst. Man Cybern. Part B Cybern.*, vol. 30, no. 2, pp. 358–364, 2000.

[34] N. Kasabov, "Evolving fuzzy neural networks for supervised/unsupervised online knowledge-based learning," *IEEE Trans. Syst. Man Cybern. Part B Cybern.*, vol. 31, no. 6, pp. 902–918, 2001.

[35] C.-J. Lin and C.-T. Lin, "An ART-based fuzzy adaptive learning control network," *IEEE Trans. Fuzzy Syst.*, vol. 5, no. 4, pp. 477–496, 1997.

[36] Y.-Q. Zhang, B. Jin, and Y. Tang, "Granular neural networks with evolutionary interval learning," *IEEE Trans. Fuzzy Syst.*, vol. 16, no. 2, pp. 309–319, 2008.

[37] Y. Chen, B. Yang, A. Abraham, and L. Peng, "Automatic design of hierarchical takagi–sugeno type fuzzy systems using evolutionary algorithms," *IEEE Trans. Fuzzy Syst.*, vol. 15, no. 3, pp. 385–397, 2007.

[38] J. Kim and N. Kasabov, "HyFIS: adaptive neuro-fuzzy inference systems and their application to nonlinear dynamical systems," *Neural Netw.*, vol. 12, no. 9, pp. 1301–1319, 1999.

[39] J.-C. Duan and F.-L. Chung, "Multilevel fuzzy relational systems: structure and identification," *Soft Comput.*, vol. 6, no. 2, pp. 71–86, 2002.

[40] 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.

[41] S. W. Tung, C. Quek, and C. Guan, "SaFIN: A self-adaptive fuzzy inference network," *IEEE Trans. Neural Netw*, vol. 22, no. 12, pp. 1928–1940, 2011.

[42] C.-F. Juang and C.-T. Lin, "An online self-constructing neural fuzzy inference network and its applications," *IEEE Trans. Fuzzy Syst.*, vol. 6, no. 1, pp. 12–32, 1998.

[43] S. Paul and S. Kumar, "Subsethood-product fuzzy neural inference system (SuPFuNIS)," *IEEE Trans. Neural Netw*, vol. 13, no. 3, pp. 578–599, 2002.

[44] 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.

[45] S. W. Tung, C. Quek, and C. Guan, "eT2FIS: an evolving type-2 neural fuzzy inference system," *Inform. Sci.*, vol. 220, pp. 124–148, 2013.

[46] P. P. Angelov and D. P. 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.

[47] Y.-Y. Lin, S.-H. Liao, J.-Y. Chang, and C.-T. Lin, "Simplified interval type-2 fuzzy neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 5, pp. 959–969, 2014.

[48] J. M. Mendel, *Uncertain Rule-Based Fuzzy Logic Systems: Introduction and New Directions.* Upper Saddle River, NJ: Prentice-Hall, 2001.

[49] ——, "Computing derivatives in interval type-2 fuzzy logic systems," *IEEE Trans. Fuzzy Syst.*, vol. 12, no. 1, pp. 84–98, 2004.

[50] J. Szlek, A. Pacławski, R. Lau, R. Jachowicz, and A. Mendyk, "Heuristic modeling of macromolecule release from PLGA microspheres," *Int. J. Nanomed.*, vol. 8, no. 1, pp. 4601–4611, 2013.

[51] V. K. Ojha, K. Jackowski, V. Snášel, and A. Abraham, "Dimensionality reduction and prediction of the protein macromolecule dissolution profile," in *Proc. 5th Int. Conf. Innovations in Bio-Inspired Comput. Appl. IBICA 2014.* Springer, 2014, pp. 301–310.

[52] V. K. Ojha, K. Jackowski, A. Abraham, and V. Snášel, "Feature selection and ensemble of regression models for predicting the protein macromolecule dissolution profile," in *2014 6th World Congr. on Nature and Biologically Inspired Comput. (NaBIC),.* IEEE, 2014, pp. 121–126.

[53] C. E. Astete and C. M. Sabliov, "Synthesis and characterization of PLGA nanoparticles," *J. Biomater. Sci., Polym. Ed.*, vol. 17, no. 3, pp. 247–289, 2006.

[54] V. K. Ojha, A. Abraham, and V. Snášel, "Ensemble of heterogeneous flexible neural tree for the approximation and feature-selection of Poly (Lactic-co-glycolic Acid) micro-and nanoparticle," in *Proc. 2nd Int. Afro-European Conf. Ind. Adv. AECIA 2015*. Springer, 2016, pp. 155–165.

[55] V. K. Ojha, K. Jackowski, A. Abraham, and V. Snášel, "Dimensionality reduction, and function approximation of poly (lactic-co-glycolic acid) micro-and nanoparticle dissolution rate," *Int. J. Nanomed.*, vol. 10, p. 1119, 2015.

[56] Y. Hayashi and J. J. Buckley, "Approximations between fuzzy expert systems and neural networks," *Int. J. Approximate Reasoning*, vol. 10, no. 1, pp. 63–73, 1994.

Publications list

Articles/Journals

Published

J1. **Ojha, V. K.**, Jackowski, K., Abraham, A., and Snášel, V. (2015). Dimensionality reduction, and function approximation of poly (lactic-co-glycolic acid) micro-and nanoparticle dissolution rate. *International Journal of Nanomedicine*, 10, 1119.
(**IF:** 4.38, Quartile: Q1).

J2. **Ojha, V. K.**, Schiano, S., Wu, C.Y., Snášel, V., and Abraham, A. (2016) Predictive Modeling of the die filling process of the pharmaceutical granules using Flexible Neural Tree. *Neural Computing Application.* (Accepted) (**IF:** 1.57)

Under review

J3. **Ojha, V. K.**, Abraham, A., and Snášel, V. (2016). Metaheuristic Design of Feedforward Neural Networks: A Review of Two Decades of Research, *Engineering Applications in Artificial Intelligence.* (**IF:** 2.21)

J4. **Ojha, V. K.**, Abraham, A., and Snášel, V. (2016). Ensemble of Heterogeneous Flexible Neural Trees Using Multiobjective Genetic Programming, *Applied Soft Computing.* (One round revision completed) (**IF:** 2.81)

J5. **Ojha, V. K.**, Abraham, A., and Snášel, V. (2016). Multiobjective Programming for Type II Hierarchical Fuzzy Trees, *IEEE Transaction on Fuzzy Systems.* (**IF:** 8.75)

Conference proceedings

C1. **Ojha, V. K.**, Abraham, A., and Snášel, V. (2016) Metaheuristic Tuning of Type-II Fuzzy Inference Systems for Data Mining. IEEE World Congress on Computational Intelligence, IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2016). (Accepted) (Indexing: SCOPUS)

C2. Zjavka, L., Snášel, V., Pedrycz, W., and **Ojha, V.K.**, A Substitution of the General Partial Differential Equation with Extended Polynomial Networks, IEEE World Congress on Computational Intelligence, International Joint Conference on Neural Networks, 2016, (IJCNN, IEEE). (Accepted) (Indexing: SCOPUS)

C3. **Ojha, V. K.**, Abraham, A., and Snášel, V. (2016). Ensemble of Heterogeneous Flexible Neural Tree for the Approximation and Feature-Selection of Poly (Lactic-co-glycolic Acid) Micro-and Nanoparticle. In Proceedings of the 2nd International Afro-European Conference for Industrial Advancement AECIA 2015 (pp. 155-165). Springer. (Indexing: SCOPUS)

C4. Abdelwahab, S., **Ojha, V. K.**, and Abraham, A. (2016). Ensemble of Flexible Neural Trees for Predicting Risk in Grid Computing Environment. In Proceedings of the 6th International Conference on Innovations in Bio-Inspired Computing and Applications IBICA 2015 (pp. 151-161). Springer. (Indexing: SCOPUS)

C5. Abdelwahab, S., **Ojha, V. K.**, and Abraham, A. (2016). Neuro-Fuzzy Risk Prediction Model for Computational Grids. In Proceedings of the 2nd International Afro-European Conference for Industrial Advancement AECIA 2015 (pp. 127-136). Springer. (Indexing: SCOPUS)

C6. Ahmed, N., **Ojha, V. K.**, and Abraham, A. (2016). An Ensemble of Neuro-Fuzzy Model for Assessing Risk in Cloud Computing Environment. In Advances in Nature and Biologically Inspired Computing (pp. 27-36). Springer. (Indexing: SCOPUS)

C7. **Ojha, V. K.**, Schiano, S., Wu, C.Y., Abraham, A., and Snášel, V. (2016) The development of a machine learning tool for modelling die filling of pharmaceutical granules. International Congress on Particle Technology, 2016, PARTEC. (German National Library).

C8. **Ojha, V. K.**, Abraham, A., and Snášel, V. (2014). ACO for continuous function optimization: A performance analysis. In 14th International Conference on Intelligent Systems Design and Applications (ISDA), 2014 (pp. 145-150). IEEE. (Indexing: SCOPUS)

C9. **Ojha, V. K.**, Abraham, A., and Snášel, V. (2015). Simultaneous optimization of neural network weights and active nodes using metaheuristics. In 14th International Conference on Hybrid Intelligent Systems (HIS), 2014 (pp. 248-253). IEEE. (Indexing: SCOPUS)

C10. Gabralla, L. A., Wahby, T. M., **Ojha, V. K.**, and Abraham, A. (2014). Ensemble of adaptive neuro-fuzzy inference system using particle swarm optimization for prediction of crude oil prices. In 14th International Conference on Hybrid Intelligent Systems (HIS), 2014 (pp. 141-146). IEEE. (Indexing: SCOPUS)

C11. **Ojha, V. K.**, Jackowski, K., Snášel, V., and Abraham, A. (2014). Dimensionality Reduction and Prediction of the Protein Macromolecule Dissolution Profile. In Proceedings of the 5th International Conference on Innovations in Bio-Inspired Computing and Applications IBICA 2014 (pp. 301-310). Springer. (Indexing: SCOPUS)

C12. **Ojha, V. K.**, Jackowski, K., Abraham, A., and Snášel, V. (2014). Feature selection and ensemble of regression models for predicting the protein macromolecule dissolution profile. In 6th World Congress on Nature and Biologically Inspired Computing (NaBIC), 2014 (pp. 121-126). IEEE. (Indexing: SCOPUS)