

Ensemble of heterogeneous flexible neural trees using multiobjective genetic programming

Article

Accepted Version

Creative Commons: Attribution-Noncommercial-No Derivative Works 4.0

Ojha, V. K. ORCID: <https://orcid.org/0000-0002-9256-1192>, Abraham, A. and Snásel, V. (2017) Ensemble of heterogeneous flexible neural trees using multiobjective genetic programming. *Applied Soft Computing*, 52. pp. 909-924. ISSN 1568-4946 doi: 10.1016/j.asoc.2016.09.035 Available at <https://centaur.reading.ac.uk/82146/>

It is advisable to refer to the publisher's version if you intend to cite from the work. See [Guidance on citing](#).

To link to this article DOI: <http://dx.doi.org/10.1016/j.asoc.2016.09.035>

Publisher: Elsevier

All outputs in CentAUR are protected by Intellectual Property Rights law, including copyright law. Copyright and IPR is retained by the creators or other copyright holders. Terms and conditions for use of this material are defined in the [End User Agreement](#).

www.reading.ac.uk/centaur

CentAUR

Central Archive at the University of Reading

Reading's research outputs online

Ensemble of Heterogeneous Flexible Neural Trees Using Multiobjective Genetic Programming

Varun Kumar Ojha^{a,*}, Ajith Abraham^b, Václav Snášel^a

^a*IT4Innovations, VŠB-Technical University of Ostrava, Ostrava, Czech Republic*

^b*Machine Intelligence Research Labs (MIR Labs), Auburn, WA, USA*

Abstract

Machine learning algorithms are inherently multiobjective in nature, where approximation error minimization and model's complexity simplification are two conflicting objectives. We proposed a multiobjective genetic programming (MOGP) for creating a heterogeneous flexible neural tree (HFNT), tree-like flexible feedforward neural network model. The functional heterogeneity in neural tree nodes was introduced to capture a better insight of data during learning because each input in a dataset possess different features. MOGP guided an initial HFNT 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 the selected candidates. A comprehensive test over classification, regression, and time-series datasets proved the efficiency of the proposed algorithm over other available prediction methods. Moreover, the heterogeneous creation of HFNT proved to be efficient in making ensemble system from the final population.

Keywords: Pareto-based multiobjectives, flexible neural tree, ensemble, approximation, feature selection;

1. Introduction

Structure optimization of a feedforward neural network (FNN) and its impact on FNN's generalization ability inspired the flexible neural tree (FNT) [1]. FNN components such as weights, structure, and activation function are the potential candidates for the optimization, which improves FNN's generalization ability to a great extent [2]. These efforts are notable because of FNN's ability to solve a large range of real-world problems [3, 4, 5, 6]. Followings are the significance structure optimization methods: constructive and pruning algorithms [7, 8], EPNet [2], NeuroEvolution of Augmenting Topologies [9], sparse neural

*Varun Kumar Ojha

Email addresses: `varun.kumar.ojha@vsb.cz` (Varun Kumar Ojha), `ajith.abraham@ieee.org` (Ajith Abraham), `vaclav.snasel@vsb.cz` (Václav Snášel)

trees [10], Cooperative co-evolution approach [11], etc. Similarly, many efforts focus on the optimization of hybrid training of FNN such as [12, 13, 14]. FNT was an additional step into this series of efforts, which was proposed to evolve as a tree-like feed-forward neural network model, where the probabilistic incremental program evolution (PIPE) [15] was applied to optimize the tree structure [1]. The underlying parameter vector of the developed FNT (weights associated with the edges and arguments of the activation functions) was optimized by metaheuristic algorithms, which are nature-inspired parameter optimization algorithms [16]. The evolutionary process allowed FNT to select significant input features from an input feature set.

In the design of FNT, the non-leaf nodes are the computational nodes, which take an activation function. Hence, rather than relying on a fixed activation function, the selection of activation function at the computational nodes is allowed to be selected by the evolutionary process. Then, it produces heterogeneous FNTs (HFNT) with the heterogeneity in its structure, computational nodes, and input set. In addition, heterogeneous function allowed HFNT to capture different features of the datasets efficiently since each input in the datasets possesses different features. The evolutionary process provides adaptation in structure, weights, activation functions, and input features. Therefore, an optimum HFNT is the one that offers the lowest approximation error with the simplest tree structure and the smallest input feature set. However, approximation error minimization and structure simplification are two conflicting objectives [17]. Hence, a multiobjective evolutionary approach [18] may offer an optimal solution(s) by maintaining a balance between these objectives.

Moreover, in the proposed work, an evolutionary process guides a population of HFNTs towards Pareto-optimum solutions. Hence, the final population may contain several solutions that are close to the best solution. Therefore, an ensemble system was constructed by exploiting many candidates of the population (candidate, solution, and model are synonymous in this article). Such ensemble system takes advantage of many solutions including the best solution [19]. Diversity among the chosen candidates holds the key in making a good ensemble system [20]. Therefore, the solutions in a final population should fulfill the following objectives: low approximation error, structural simplicity, and high diversity. However, these objectives are conflicting to each other. A fast elitist nondominated sorting genetic algorithm (NSGA-II)-based multiobjective genetic programming (MOGP) was employed to guide a population of HFNTs [21]. The underlying parameters of selected models were further optimized by using differential evolution (DE) algorithm [22]. Therefore, we may summarize the key contributions of this work as follows:

- 1) A heterogeneous flexible neural tree (HFNT) for function approximation and feature selection was proposed.
- 2) HFNT was studied under an NSGA-II-based multiobjective genetic programming framework. Thus, it was termed HFNT^M.
- 3) Alongside *approximation error* and *tree size* (complexity), a *diversity index* was introduced to maintain

diversity among the candidates in the population.

- 4) HFNT^M 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.

A detailed literature review provides an overview of FNT usage over the past few years (Section 2). Conclusions derived from literature survey supports our HFNT^M approach, where a Pareto-based multiobjective genetic programming was used for HFNT optimization (Section 3.1). Section 3.2 provides a detailed discussion on the basics of HFNT: MOGP for HFNT structure optimization, and DE for HFNT parameter optimization. The efficiency of the above-mentioned hybrid and complex multiobjective FNT algorithm (HFNT^M) was tested over various prediction problems using a comprehensive experimental set-up (Section 4). The experimental results support the merits of proposed approach (Section 5). Finally, we provide a discussion of experimental outcomes in Section 6 followed by conclusions in Section 7.

2. Literature Review

The literature survey describes the following points: basics of FNT, approaches that improvised FNT, and FNTs successful application to various real-life problems. Subsequently, the shortcomings of basic FNT version are concluded that inspired us to propose HFNT^M.

FNT was first proposed by Chen et al. [1], where a tree-like-structure was optimized by using PIPE. Then, its approximation ability was tested for time-series forecasting [1] and intrusion detection [23], where a variant of simulated annealing (called degraded ceiling) [24], and particle swarm optimization (PSO) [25], respectively, were used for FNT parameter optimization. Since FNT is capable of input feature selection, in [26], FNT was applied for selecting input features in several classification tasks, in which FNT structure was optimized by using genetic programming (GP) [27], and the parameter optimization was accomplished by using memetic algorithm [28]. Additionally, they defined five different mutation operators, namely, changing one terminal node, all terminal nodes, growing a randomly selected sub-tree, pruning a randomly selected sub-tree, and pruning redundant terminals. Li et al. [29] proposed FNT-based construction of decision trees whose nodes were conditionally replaced by neural node (activation node) to deal with continuous attributes when solving classification tasks. In many other FNT based approaches, like in [30], GP was applied to evolve hierarchical radial-basis-function network model, and in [31] a multi-input-multi-output FNT model was evolved. Wu et al. [32] proposed to use grammar guided GP [33] for FNT structure optimization. Similarly, in [34], authors proposed to apply multi-expression programming (MEP) [35] for FNT structure optimization and immune programming algorithm [36] for the parameter vector optimization. To improve classification accuracy of FNT, Yang et al. [37] proposed a hybridization of FNT with a further-division-of-partition-space method. In [38], authors illustrated

crossover and mutation operators for evolving FNT using GP and optimized the tree parameters using PSO algorithm.

A model is considered efficient if it has generalization ability. We know that a consensus decision is better than an individual decision. Hence, an ensemble of FNTs may lead to a better-generalized performance than a single FNT. To address this, in [39], authors proposed to make an ensemble of FNTs to predict the chaotic behavior of stock market indices. Similarly, in [40], the proposed FNTs ensemble predicted the breast cancer and network traffic better than individual FNT. In [41], protein dissolution prediction was easier using ensemble than the individual FNT.

To improve the efficiency in terms of computation, Peng et al. [42] proposed a parallel evolving algorithm for FNT, where the parallelization took place in both tree-structure and parameter vector populations. In another parallel approach, Wang et al. [43] used gene expression programming (GEP) [44] for evolving FNT and used PSO for parameter optimization.

A multi-agent system [45] based FNT (MAS-FNT) algorithm was proposed in [46], which used GEP and PSO for the structure and parameter optimization, respectively. The MAS-FNT algorithm relied on the division of the main population into sub-population, where each sub-population offered local solutions and the best local solution was picked-up by analyzing tree complexity and accuracy.

Chen et al. [1, 26] referred the arbitrary choice of activation function at non-leaf nodes. However, they were restricted to use only Gaussian functions. A performance analysis of various activation function is available in [47]. Bouaziz et al. [48, 49] proposed to use beta-basis function at non-leaf nodes of an FNT. Since beta-basis function has several controlling parameters such as shape, size, and center, they claimed that the beta-basis function has advantages over other two parametric activation functions. Similarly, many other forms of neural tree formation such as balanced neural tree [50], generalized neural tree [51], and convex objective function neural tree [52], were focused on the tree improvement of neural nodes.

FNT was chosen over the conventional neural network based models for various real-world applications related to prediction modeling, pattern recognition, feature selection, etc. Some examples of such applications are cement-decomposing-furnace production-process modeling [53], time-series prediction from gene expression profiling [54], stock-index modeling [39], anomaly detection in peer-to-peer traffic [55], intrusion detection [56], face identification [57], gesture recognition [58], shareholder's management risk prediction [59], cancer classification [60], somatic mutation, risk prediction in grid computing [61], etc.

The following conclusions can be drawn from the literature survey. **First, FNT was successfully used in various real-world applications with better performance than other existing function approximation models. However, it was mostly used in time-series analysis. Second, the lowest approximation error obtained by an individual FNT during an evolutionary phase was considered as the best structure that propagated to the parameter optimization phase. Hence, there was no consideration as far as structural simplicity and generalization ability are concerned. Third, the computational nodes of the FNT were**

fixed initially, and little efforts were made to allow for its automatic adaptation. Fourth, little attention was paid to the statistical validation of FNT model, e.g., mostly the single best model was presented as the experimental outcome. However, the evolutionary process and the meta-heuristics being stochastic in nature, statistical validation is inevitably crucial for performance comparisons. Finally, to create a generalized model, an ensemble of FNTs were used. However, FNTs were created separately for making the ensemble. Due to stochastic nature of the evolutionary process, FNT can be structurally distinct when created at different instances. Therefore, no explicit attention was paid to create diverse FNTs within a population itself for making ensemble. In this article, a heterogeneous FNT called HFNT was proposed to improve the basic FNT model and its performance by addressing above mentioned shortcomings.

3. Multi-objectives and Flexible Neural Tree

In this section, first, Pareto-based multiobjective is discussed. Second, we offer a detailed discussion on FNT and its structure and parameter optimization using NSGA-II-based MOGP and DE, respectively. Followed by a discussion on making an evolutionary weighted ensemble of the candidates from the final population.

3.1. Pareto-Based Multi-objectives

Usually, learning algorithms owns a single objective, i.e., the approximation error minimization, which is often achieved by minimizing mean squared error (MSE) on the learning data. MSE E on a learning data is computed as:

$$E = \frac{1}{N} \sum_{i=1}^N (d_i - y_i)^2, \quad (1)$$

where d_i and y_i are the desired output and the model's output, respectively and N indicates total data pairs in the learning set. Additionally, a statistical goodness measure, called, correlation coefficient r that tells the relationship between two variables (i.e., between the desired output \mathbf{d} and the model's output \mathbf{y}) may also be used as an objective. Correlation coefficient r is computed as:

$$r = \frac{\sum_{i=1}^N (d_i - \bar{\mathbf{d}}_i) (y_i - \bar{\mathbf{y}}_i)}{\sqrt{\sum_{i=1}^N (d_i - \bar{\mathbf{d}}_i)^2 \sum_{i=1}^N (y_i - \bar{\mathbf{y}}_i)^2}}, \quad (2)$$

where $\bar{\mathbf{d}}$ and $\bar{\mathbf{y}}$ are means of the desired output \mathbf{d} and the model's output \mathbf{y} , respectively.

However, single objective comes at the expense of model's complexity or generalization ability on unseen data, where generalization ability broadly depends on the model's complexity [62]. A common model complexity indicator is the number of free parameters in the model. The *approximation error* (1) and the number of free parameters minimization are two conflicting objectives. One approach is to combine these two objectives as:

$$f = \alpha E + (1 - \alpha)D, \quad (3)$$

where $0 \leq \alpha \leq 1$ is a constant, E is the MSE (1) and D is the total free parameter in a model. The scalarized objective f in (3), however, has two disadvantages. First, determining an appropriate α that controls the conflicting objectives. Hence, generalization ability of the produced model will be a mystery [63]. Second, the scalarized objective f in (3) leads to a single best model that tells nothing about how the conflicting objectives were achieved. In other words, no single solution exists that may satisfy both objectives, simultaneously.

We study a multiobjective optimization problem of the form:

$$\begin{aligned} &\text{minimize } \{f_1(\mathbf{w}), f_2(\mathbf{w}), \dots, f_m(\mathbf{w})\} \\ &\text{subject to } \mathbf{w} \in W \end{aligned}$$

where we have $m \geq 2$ objective functions $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$. We denote the vector of objective functions by $\mathbf{f}(\mathbf{w}) = \langle f_1(\mathbf{w}), f_2(\mathbf{w}), \dots, f_m(\mathbf{w}) \rangle$. The decision (variable) vectors $\mathbf{w} = \langle w_1, w_2, \dots, w_n \rangle$ belong to the set $W \subset \mathbb{R}^n$, which is a subset of the decision variable space \mathbb{R}^n . The word ‘minimize’ means that we want to minimize all the objective functions simultaneously.

A nondominated solution is one in which no one objective function can be improved without a simultaneous detriment to at least one of the other objectives of the solution [21]. The nondominated solution is also known as a Pareto-optimal solution.

Definition 1. *Pareto-dominance* - A solution \mathbf{w}_1 is said to dominate a solution \mathbf{w}_2 if $\forall i = 1, 2, \dots, m$, $f_i(\mathbf{w}_1) \leq f_i(\mathbf{w}_2)$, and there exists $j \in \{1, 2, \dots, m\}$ such that $f_j(\mathbf{w}_1) < f_j(\mathbf{w}_2)$ holds.

Definition 2. *Pareto-optimal* - A solution \mathbf{w}_1 is called Pareto-optimal if there does not exist any other solution that dominates it. A set Pareto-optimal solution is called Pareto-front.

Algorithm 1 is a basic framework of NSGA-II based MOGP, which was used for computing Pareto-optimal solutions from an initial HFNT population. The individuals in MOGP were sorted according to their dominance in population. Note that the function $size(\cdot)$ returns total number of rows (population size) for a 2-D matrix and returns total number of elements for a vector. The Moreover, individuals were sorted according to the rank/Pareto-front. MOGP is an elitist algorithm that allowed the best individuals to propagate into next generation. Diversity in the population was maintained by measuring crowding distance among the individuals [21].

Data: Problem and Objectives

Result: A bag M of solutions selected from Pareto-fronts

initialization: HFNT population P ;

evaluation: nondominated sorting of P ;

while *termination criteria not satisfied* **do**

 selection: binary tournament selection;

 generation: a new population Q ;

 recombination: $R = P + Q$;

 evaluation: nondominated sorting of R ;

 elitism: $P = size(P)$ best individuals from R ;

end

Algorithm 1: NSGA-II based multiobjective genetic programming

3.2. Heterogeneous Flexible Neural Tree

HFNT is analogous to a multi-layer feedforward neural network that has over-layer connections and activation function at the nodes. HFNT construction has two phases [1]: 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. Figure 1 is a lucid illustration of these two phases that work in some co-evolutionary manner. From Figure 1, it may be observed that two global search algorithms MOGP (for structure optimization) and DE (for parameter optimization) works in a nested manner to obtain a near optimum tree that may have less complex tree structure and better parameter. 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.

3.2.1. Basic Idea of HFNT

An HFNT S is a collection of function set F and instruction set T :

$$S = F \cup T = \{+_2^k, +_3^k, \dots, +_{tn}^k\} \cup \{x_1, x_2, \dots, x_d\} \quad (4)$$

where $+_j^k$ ($j = 2, 3, \dots, tn$) denotes a non-leaf instruction (a computational node). It receives $2 \leq j \leq tn$ arguments and takes an activation function k from a set of activation functions. Maximum arguments tn to a computational node are predefined. A set of seven activation functions is shown in Table 1. Leaf

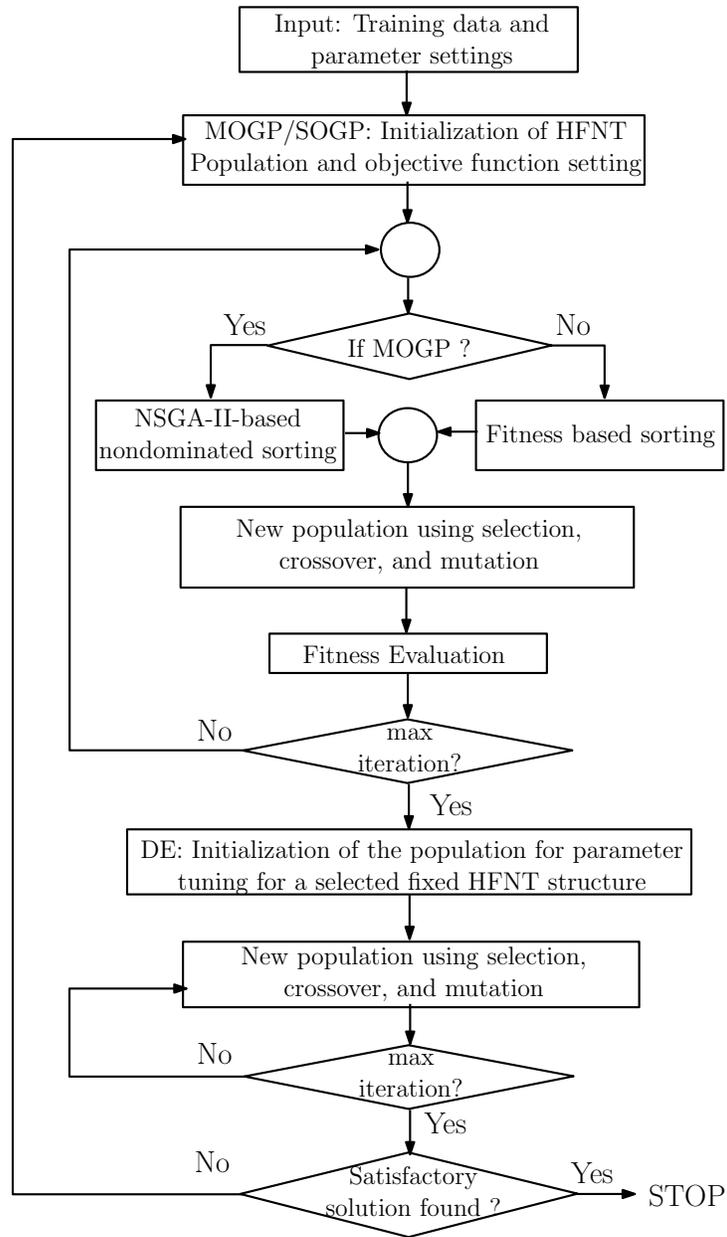


Figure 1: Co-evolutionary construction of the heterogeneous flexible neural tree.

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 | $f(x, a, b) = \exp(-((x - a)^2)/(b^2))$ |
| Tangent-Hyperbolic | 2 | $f(x) = (e^x - e^{-x})/(e^x + e^{-x})$ |
| Fermi Function | 3 | $f(x) = 1/(1 + e^{-x})$ |
| Linear Fermi | 4 | $f(x, a, b) = a \times 1/((1 + e^{-x})) + b$ |
| Linear Tangent-hyperbolic | 5 | $f(x, a, b) = a \times (e^x - e^{-x})/(e^x + e^{-x}) + b$ |
| Bipolar Sigmoid | 6 | $f(x, a) = (1 - e^{-2xa})/(a(1 + e^{-2xa}))$ |
| Unipolar Sigmoid | 7 | $f(x, a) = (2 a)/(1 + e^{-2 a x})$ |

node’s instruction x_1, x_2, \dots, x_d denotes input variables. Figure 2 is an illustration of a typical HFNT. Similarly, Figure 3 is an illustration of a typical node in an HFNT.

The i -th computational node (Figure 3) of a tree (say i -th node in Figure 2) 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 $\varphi_i^k(\cdot)$ 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:

$$y_i = \varphi_i^k(a_i, b_i, o_i) = \exp\left(-\left(\frac{o_i - a_i}{b_i}\right)^2\right) \quad (5)$$

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 (Figure 3), also known as excitation of the node. The net excitation o^i of the i -th node is computed as:

$$o_i = \sum_{j=1}^{n^i} w_j^i z_j^i \quad (6)$$

where $z_j^i \in \{x_1, x_2, \dots, x_d\}$ or, $z_j^i \in \{y_1, y_2, \dots, y_m\}$, 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 (5) 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 *tree size* and *diversity index*. The *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 Figure 2 is 11 (three computational nodes and eight leaf-nodes). 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 Figure 2 is three ($+\frac{1}{3}$, $+\frac{4}{3}$, and $+\frac{5}{3}$). Hence, its *diversity index* is three.

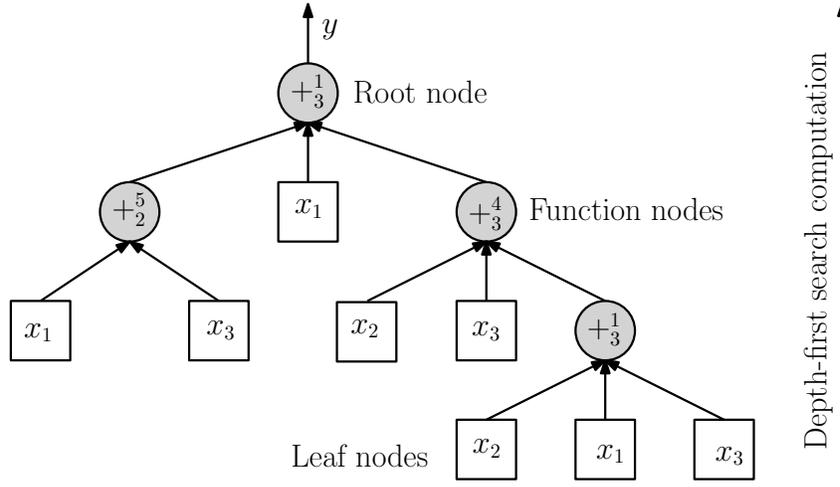


Figure 2: Typical representation of a neural tree $S = F \cup T$ whose function instruction set $F = \{+_3^1, +_2^5, +_3^4\}$ and terminal instruction set $T = \{x_1, x_2, x_3, x_4\}$.

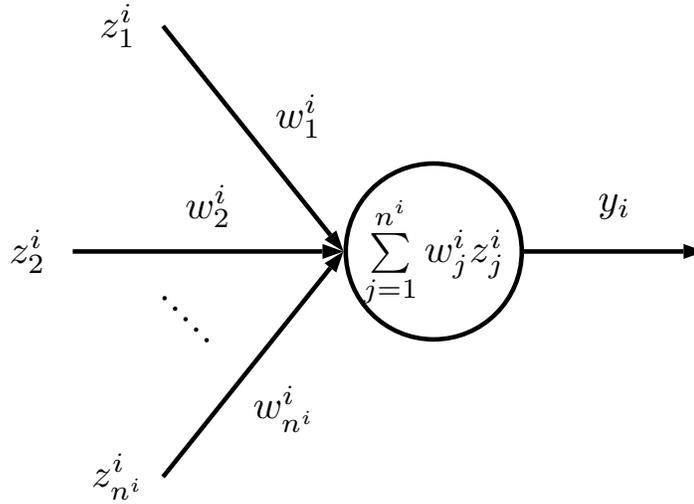


Figure 3: Illustration of a computational node. The variable n^i indicates the number of inputs z_j^i and weights w_j^i received at the i -th node and the variable y^i is the output of the i -th node.

3.3. Structure and Parameter Learning (Near optimal Tree)

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 [27], PIPE [15], GEP [44], MEP [35], and so on. To optimize tree parameters, algorithms such as genetic algorithm [64], evolution strategy [64], artificial bee colony [65], PSO [25, 66], DE [22], and any hybrid algorithm such as GA and PSO [67] can be used.

3.3.1. Tree-construction

The proposed multiobjective optimization of FNT has three fitness measures: *approximation error* (1) minimization, *tree size* minimization, and *diversity index* maximization. These objectives are simultaneously optimized during the *tree construction* phase using MOGP, which guides an initial population P of random tree-structures according to Algorithm 1. The detailed description of the components of Algorithm 1 are as follows:

Selection. In selection operation, a *mating pool* of size $size(P)r$ is created using *binary tournament selection*, where two candidates are randomly selected from a population and the best (according to rank and crowding distance) among them is placed into the mating pool. This process is continued until the mating pool is full. An offspring population Q is generated by using the individuals of mating pool. Two distinct individuals (parents) are randomly selected from the mating pool to create new individuals using genetic operators: crossover and mutation. The crossover and mutation operators are applied with probabilities pc and pm , respectively.

Crossover. In crossover operation, randomly selected sub-trees of two parent trees were swapped. The swapping includes the exchange of activation-nodes, weights, and inputs as it is described in [38, 64, 68].

Mutation. The mutation of a selected individual from mating pool took place in the following manner [38, 64, 68]:

- 1) A randomly selected terminal node is replaced by a newly generated terminal node.
- 2) All terminal nodes of the selected tree were replaced by randomly generated new terminal nodes.
- 3) A randomly selected terminal node or a computational node is replaced by a randomly generated sub-tree.
- 4) A randomly selected terminal node is replaced by a randomly generated computational node.

In the proposed MOGP, during the each mutation operation event, one of the above-mentioned four mutation operators was randomly selected for mutation of the tree.

Recombination. The offspring population Q and the main population P , are merged to make a combined population R .

Elitism. In this step, $size(Q)$ worst individuals are weeded out. In other words, $size(P)$ best individuals are propagated to a new generation as main population P .

3.3.2. 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 $\mathbf{w} = \langle w_1, w_2, \dots, w_n \rangle$ for the optimization. In addition, a cross-validation (CV) phase was used for statistical validation of HFNTs.

The basics of DE is as follows. For an initial population H of parameter vectors \mathbf{w}_i for $i = 1$ to $size(H)$, DE repeats its steps mutation, recombination, and selection until an optimum parameter vector \mathbf{w}^* is obtained. DE updates each parameter vector $\mathbf{w}_i \in H$ by selecting the best vector \mathbf{w}_i^g and three random vectors $\mathbf{r}_i^0, \mathbf{r}_i^1$, and \mathbf{r}_i^2 from H such that $\mathbf{r}_i^0 \neq \mathbf{r}_i^1 \neq \mathbf{r}_i^2$ holds. The random vector \mathbf{r}^0 is considered as a trial vector \mathbf{w}_i^t . Hence, for all $i = 1, 2, \dots, size(H)$, and $j = 1, 2, \dots, n$, the j -th variable w_{ij}^t of i -th trail-vectors \mathbf{w}_i^t is generated by using *crossover*, *mutation*, and *recombination* as:

$$w_{ij}^t = \begin{cases} r_{ij}^{(0)} + F(w_{ij}^g - r_{ij}^0) + F(r_{ij}^1 - r_{ij}^2) & u_{ij} < cr \ \parallel \ j = k \\ r_{ij}^{(0)} & u_{ij} \geq cr \ \& \ j \neq k \end{cases} \quad (7)$$

where k is a random index in $[1, n]$, u_{ij} is within $[0, 1]$, k is in $\{1, 2, \dots, n\}$, cr is crossover probability, and $F \in [0, 2]$ is mutation factor. The trail vector \mathbf{w}_i^t is selected if

$$\mathbf{w}_i = \begin{cases} \mathbf{w}_i^t & f(\mathbf{w}_i^t) < f(\mathbf{w}_i) \\ \mathbf{w}_i & f(\mathbf{w}_i^t) \geq f(\mathbf{w}_i) \end{cases} \quad (8)$$

where $f(\cdot)$ returns fitness of a vector as per (1). Hence, the process of *crossover*, *mutation*, *recombination*, and *selection* are repeated until an optimal parameter vector solution \mathbf{w}^* is found.

3.4. Ensemble: Making use of MOGP Final Population

In *tree construction* phase, MOGP provides a population from which we can select tree models for making the ensemble. Three conflicting objectives such as *approximation error*, *tree size*, and *diversity index* allows the creation of Pareto-optimal solutions, where solutions are distributed on various Pareto-optimal fronts according to their rank in population. Ensemble candidates can be selected from the first line of solutions (Front 1), or they can be chosen by examining the three objectives depending on the user's need and preference. Accuracy and diversity among the ensemble candidate are important [20]. Hence, in this work, *approximation error*, and diversity among the candidates were given preference over *tree size*. Not to confuse "*diversity index*" with "diversity". The *diversity index* is an objective in MOGP, and the diversity is the number of distinct candidates in an ensemble. A collection M of the diverse candidate is called a bag of candidates [69]. In this work, any two trees were considered diverse (distinct) if the followings hold: 1) Two trees were of different size. 2) The number of function nodes/or leaf nodes in two

trees were dissimilar. 3) Two models used a different set of input features. 4) Two models used a different set of activation functions. Hence, diversity div of ensemble M (a bag of solutions) was computed as:

$$div = \frac{distinct(M)}{size(M)}, \quad (9)$$

where $distinct(M)$ is a function that returns total distinct models in an ensemble M and $size(M)$ is a total number of models in the bag.

Now, for a classification problem, to compute combined vote of respective candidate's outputs $m_1, m_2, \dots, m_{size(M)}$ of bag M and classes $\omega_1, \omega_2, \dots, \omega_C$, we used an indicator function $\mathbb{I}(\cdot)$ which takes 1 if '.' is true, and takes 0 if '.' is false. Thus, ensemble decisions by weighted majority voting is computed as [70, 71]:

$$y = \arg \max_{j=1}^C \sum_{t=1}^{size(M)} w_t \mathbb{I}(m_t = \omega_j), \quad (10)$$

where w_t is weight associated with the t -th candidate m_t in an ensemble M and y is set to class ω_j if the total weighted vote received by ω_j is higher than the total vote received by any other class. Similarly, the ensemble of regression methods was computed by weighted arithmetic mean as [70]:

$$y = \sum_{t=1}^{size(M)} w_t m_t, \quad (11)$$

where w_t and m_t are weight and output of t -th candidate in a bag M , respectively, and y is the ensemble output, which is then used for computing MSE (1) and correlation coefficient (2). The weights may be computed according to fitness of the models, or by using a metaheuristic algorithm. In this work, DE was applied to compute the ensemble weights w_t , where population size was set to 100 and number of function evaluation was set to 300,000.

3.5. Multiobjective: A General Optimization Strategy

A summary of general HFNT learning algorithm is as follows:

- Step 1. Initializing HFNT training parameters.
- Step 2. Apply *tree construction* phase to guide initial HFNT population towards Pareto-optimal solutions.
- Step 3. Select tree-model(s) from MOGP final population according to their *approximation error*, *tree size*, and *diversity index* from the Pareto front.
- Step 4. Apply *parameter-tuning* phase to optimize the selected tree-model(s).
- Step 5. Go to Step 2, if no satisfactory solution found. Else go to Step 6.
- Step 6. Using a cross-validation (CV) method to validate the chosen model(s).
- Step 7. Use the chosen tree-model(s) for making ensemble (recommended).
- Step 8. Compute ensemble results of the ensemble model (recommended).

Table 2: Multiobjective flexible neural tree parameter set-up for the experiments

| Parameter | Definition | Default Rang | Value |
|-----------------|---|--|--------|
| Scaling | Input-features scaling range. | $[dl, du], dl \in \mathbb{R}, du \in \mathbb{R}$ | [0,1] |
| Tree height | Maximum depth (layers) of a tree model. | $\{td \in \mathbb{Z}^+ td > 1\}$ | 4 |
| Tree arity | Maximum arguments of a node $+_{tn}^k$. | $\{tn \in \mathbb{Z}^+ n \geq 2\}$ | 5 |
| Node range | Search space of functions arguments. | $[nl, nu], nl \in \mathbb{R}, nu \in \mathbb{R}$ | [0,1] |
| Edge range | Search space for edges (weights) of tree. | $[wl, wu], wl \in \mathbb{R}, wu \in \mathbb{R}$ | [-1,1] |
| P | MOGP population. | $size(P) > 20$ | 30 |
| Mutation | Mutation probability | pm | 0.3 |
| Crossover | Crossover probability | $pc = 1 - pm$ | 0.7 |
| Mating pool | Size of the pool of selected candidates. | $size(P)r, 0 \leq r \leq 1$ | 0.5 |
| Tournament | Tournament selection size. | $2 \leq bt \leq size(P)$ | 2 |
| H | DE population. | $size(H) \geq 50$ | 50 |
| General i_g | Maximum number of trails. | $\{i_g \in \mathbb{Z}^+ i_g > 1\}$ | 3 |
| Structure i_s | MOGP iterations | $\{i_s \in \mathbb{Z}^+ i_s \geq 50\}$ | 30 |
| Parameter i_p | DE iterations | $\{i_p \in \mathbb{Z}^+ i_p \geq 100\}$ | 1000 |

4. Experimental Set-Up

Several experiments were designed for evaluating the proposed HFNT^M. A careful parameter-setting was used for testing its efficiency. A detailed description of the parameter-setting is given in Table 2, which includes: definitions, default range, and selected value. The phases of the algorithm were repeated until the stopping criteria met, i.e., either the lowest predefined *approximation error* was achieved, or the maximum function evaluations were reached. The repetition holds the key to obtaining a good solution. A carefully designed repetition of these two phases may offer a good solution in fewer of function evaluations.

In this experiment, three general repetitions i_g were used with 30 *tree construction* iterations i_s , and 1000 *parameter-tuning* iterations i_p (Figure 1). Hence, the maximum function evaluation¹ [$size(P) + i_g\{i_s(size(P) + size(P)r) + i_p size(H)\}$] was 154,080. The DE version *DE/rand – to – best/1/bin* [22] with cr equal to 0.9 and F equal to 0.7 was used in the *parameter-tuning* phase.

The experiments were conducted over classification, regression, and time-series datasets. [A detailed description of the chosen dataset from the UCI machine learning \[72\] and KEEL \[73\] repository is available in Table A.17.](#) The parameter-setting mentioned in Table 2 was used for the experiments over each dataset. Since the stochastic algorithms depend on random initialization, a pseudorandom number generator called, Mersenne Twister algorithm that draws random values using probability distribution in a pseudo-random manner was used for initialization of HFNTs [74]. Hence, each run of the experiment was conducted with a random seed drawn from the system. We compared HFNT^M performance with various other

¹Initial GP population + three repetition ((GP population + mating pool size) \times MOGP iterations + MH population \times MH iterations) = $30 + 3 \times [(30 + 15) \times 30 + 50 \times 1000] = 154,080$.

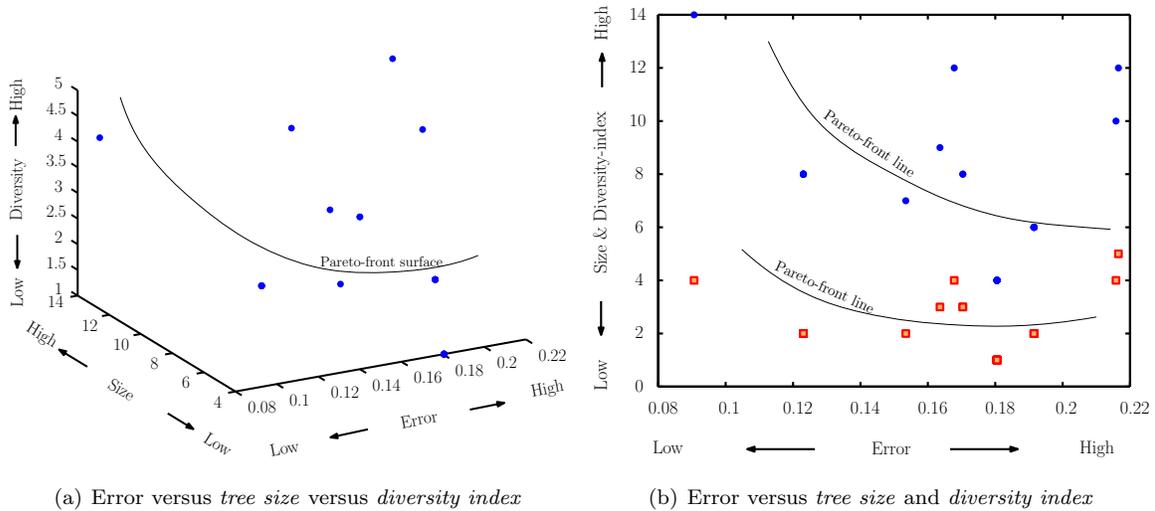


Figure 4: Pareto-front of a final population of 50 individuals generated from the training dataset of time-series problem MGS. (a) 3-D plot of solutions and a Pareto-front is a surface. (b) 2-D plot of Error versus complexity (in blue dots) and Error versus diversity (in red squares).

approximation models collected from literature. A list of such models is provided in Table B.18. A developed software tool based on the proposed HFNT^M algorithm for predictive modeling is available in [75].

To construct good ensemble systems, highly diverse and accurate candidates were selected in the ensemble bag M . To increase diversity (9) among the candidates, the Pareto-optimal solutions were examined by giving preference to the candidates with low *approximation error*, small *tree size* and distinct from others selected candidates. Hence, $size(M)$ candidates were selected from a population P . An illustration of such selection method is shown in Figure 4, which represents an MOGP final population of 50 candidate solutions computed over dataset MGS.

MOGP simultaneously optimized three objectives. Hence, the solutions were arranged on the three-dimensional map (Figure 4(a)), in which along the x-axis, error was plotted; along the y-axis, *tree size* was plotted; and along z-axis, *diversity index* (diversity) was plotted. However, for the simplicity, we have arranged solutions also in 2-D plots (Figure 4(b)), in which along the x-axis, computed error was plotted; and along the y-axis, *tree size* (indicated by blue dots) and *diversity index* (indicated by red squares) were plotted. From Figure 4(b), it is evident that a clear choice is difficult since decreasing *approximation error* increases models *tree size* (blue dots in Figure 4(b)). Similarly, decreasing *approximation error* increases models *tree size* and diversity (red squares in Figure 4(b)). Hence, solutions along the Pareto-front (rank-1), i.e., Pareto surface indicated in the 3-D map of the solutions in Figure 4(a) were chosen for the ensemble. For all datasets, ensemble candidates were selected by examining Pareto-fronts in a similar fashion as described for the dataset MGS in Figure 4.

The purpose of our experiment was to obtain sufficiently good prediction models by enhancing pre-

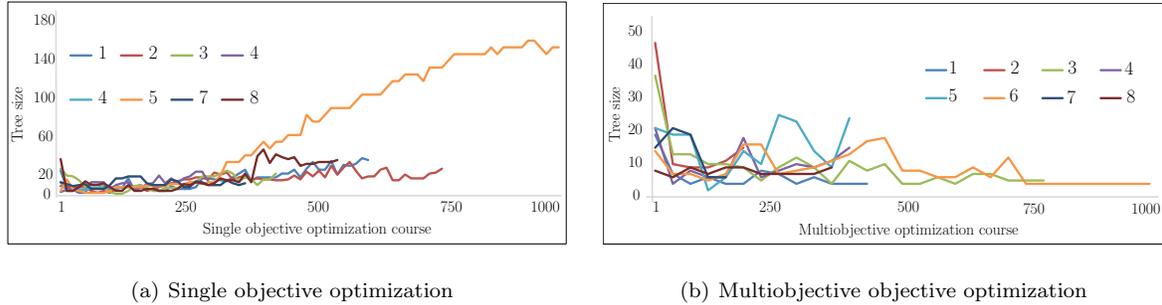


Figure 5: Comparison of single and multiobjective optimization course.

dictability and lowering complexity. We used MOGP for optimization of HFNTs. Hence, we were compromising fitness by lowering models complexity. In single objective optimization, we only looked for models fitness. Therefore, we did not possess control over model’s complexity. Figure 5 illustrates eight runs of both single and multiobjective optimization course of HFNT, where models *tree size* (complexity) is indicated along y-axis and x-axis indicates fitness value of the HFNT models. The results shown in Figure 5 was conducted over MGS dataset. For each single objective GP and multiobjective GP, optimization course was noted, i.e., successive fitness reduction and *tree size* were noted for 1000 iterations.

It is evident from Figure 5 that the HFNT^M approach leads HFNT optimization by lowering model’s complexity. Whereas, in the single objective, model’s complexity was unbounded and was abruptly increased. The average *tree size* of eight runs of single and eight runs of multiobjective were 39.265 and 10.25, respectively; whereas, the average fitness were 0.1423 and 0.1393, respectively. However, in single objective optimization, given the fact that the *tree size* is unbounded, the fitness of a model may improve at the expense of model’s complexity. Hence, the experiments were set-up for multiobjective optimization that provides a balance between both objectives as described in Figure 4.

5. Results

Experimental results were classified into three categories: classification, regression, and time-series. Each category has two parts: 1) First part describes the best and average results obtained from the experiments; 2) Second part describes ensemble results using tabular and graphical form.

5.1. Classification dataset

We chose five classification datasets for evaluating HFNT^M, and the classification accuracy was computed as:

$$f_a = \frac{tp + tn}{tp + fn + fp + tn}, \quad (12)$$

where tp is the total positive samples correctly classified as positive samples, tn is the total negative samples correctly classified as negative samples, fp is the total negative samples incorrectly classified as

positive samples, and fn is the total positive samples incorrectly classified as negative samples. Here, for a binary class classification problem, the positive sample indicates the class labeled with ‘1’ and negative sample indicates class labeled with ‘0’. Similarly, for a three-class (ω_1, ω_2 , and ω_3) classification problem, the samples which are labeled as a class ω_1 are set to 1, 0, 0, i.e., set to positive for class ω_1 and negative for ω_2 , and ω_3 . The samples which are labeled as a class ω_2 are set to 0, 1, 0, and the samples which are labeled as a class ω_3 are set to 0, 0, 1.

5.1.1. 10-Fold CV

The experiments on classification dataset were conducted in three batches that produced 30 models, and each model was cross-validated using 10-fold CV, in which a dataset is equally divided into 10 sets and the training of a model was repeated 10 times. Each time a distinct set was picked for the testing the models, and the rest of nine set was picked for the training of the model. Accordingly, the obtained results are summarized in Table 3. Each batch of experiment produced an ensemble system of 10 models whose results are shown in Table 7.

The obtained results presented in Table 3 describes the best and mean results of 30 models. We present a comparative study of the best 10-fold CV models results of HFNT^M and the results reported in the literature in Table 4. In Table 4, the results of HDT and FNT [29] were of 10 fold CV results on the test dataset. Whereas, the results of FNT [76] was the best test accuracy and not the CV results. The results summarized in Table 4 suggests a comparatively better performance of the proposed HFNT^M over the previous approaches. For the illustration of a model created by HFNT^M approach, we chose the best model of dataset WDB that has a test accuracy of 97.02% (shown in Table 3). A pictorial representation of the WDB model is shown in Figure 6, where the model’s *tree size* is 7, total input features are 5, (x_3, x_4, x_{12}, x_{17} , and x_{22}) and the selected activation function is tangent hyperbolic ($k = 2$) at both the non-leaf nodes. Similarly, we may represent models of all other datasets.

Table 3: Best and mean results of 30 10-fold CV models (300 runs) of HFNT^M

| Data | Best of 30 models | | | | Mean of 30 models | | | |
|------|-------------------|------------|------------------|----------|-------------------|------------|-----------------------|-----------|
| | train f_a | test f_a | <i>tree size</i> | Features | train f_a | test f_a | avg. <i>tree size</i> | diversity |
| AUS | 87.41% | 87.39% | 4 | 3 | 86.59% | 85.73% | 5.07 | 0.73 |
| HRT | 87.41% | 87.04% | 8 | 5 | 82.40% | 80.28% | 7.50 | 0.70 |
| ION | 90.92% | 90.29% | 5 | 3 | 87.54% | 86.14% | 6.70 | 0.83 |
| PIM | 78.67% | 78.03% | 10 | 5 | 71.12% | 70.30% | 6.33 | 8.67 |
| WDB | 97.02% | 96.96% | 6 | 5 | 94.51% | 93.67% | 7.97 | 0.73 |

In this work, Friedman test was conducted to examine the significance of the algorithms. For this purpose, the classification accuracy (test results) was considered (Table 4). The average ranks obtained by each method in the Friedman test is shown in Table 5. 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

Table 4: Comparative results: 10-fold CV test accuracy f_a and variance σ of algorithms

| Algorithms | AUS | | HRT | | ION | | PIM | | WDB | |
|-------------------------|------------|----------|------------|----------|------------|----------|------------|----------|------------|----------|
| | test f_a | σ |
| HDT [29] | 86.96% | 2.058 | 76.86% | 2.086 | 89.65% | 1.624 | 73.95% | 2.374 | | |
| FNT [29] | 83.88% | 4.083 | 83.82% | 3.934 | 88.03% | 0.953 | 77.05% | 2.747 | | |
| FNT [76] | | | | | | | | | 93.66% | n/a |
| HFNT^M | 87.39% | 0.029 | 87.04% | 0.053 | 90.29% | 0.044 | 78.03% | 0.013 | 96.96% | 0.005 |

according to Friedman statistic is 6. Since $Q > \chi_{(\alpha,2)}^2$, 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 5 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^M and the other two algorithms, Holm’s method [77] was used. Holm’s method 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 in ascending order of z -value (Table 6). From the post hoc analysis, it was observed that the proposed algorithm HFNT^M outperformed both HDT [29] and FNT [29] algorithms.

Table 5: Average rankings of the algorithms

| Algorithm | Ranking |
|-------------------------|---------|
| HFNT^M | 1.0 |
| HDT | 2.5 |
| FNT | 2.5 |

Table 6: Post Hoc comparison between HFNT^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 |

5.1.2. Ensembles

The best accuracy and the average accuracy of 30 models presented in Table 3 are the evidence of HFNT^M efficiency. However, as mentioned earlier, a generalized solution may be obtained by using an ensemble. All 30 models were created in three batches. Hence, three ensemble systems were obtained. The results of those ensemble systems are presented in Table 7, where ensemble results are the accuracies f_a obtained by weighted majority voting (10). In Table 7, the classification accuracies f_a were computed over

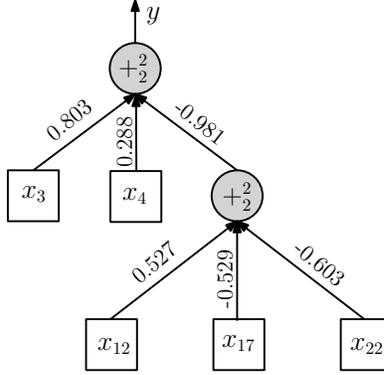


Figure 6: HFNT model of classification dataset WDB (test $f_a = 97.02\%$).

CV test dataset. From Table 7, it may be observed that high diversity among the ensemble candidates offered comparatively higher accuracy. Hence, an ensemble model may be adopted by examining the performance of an ensemble system, i.e., average *tree size* (complexity) of the candidates within the ensemble and the selected input features.

An ensemble system created from a genetic evolution and adaptation is crucial for feature selection and analysis. Summarized ensemble results in Table 7 gives the following useful information about the HFNT^M feature selection ability: 1) TSF - total selected features; 2) MSF - most significant (frequently selected) features; and 3) MIF - most infrequently selected features. Table 7 illustrates feature selection results.

| Data | Batch | test f_a | avg. D | div (9) | TSF | MSF | MIF |
|------|-------|---------------|----------|-----------|-----|---------------------------|---------------------------|
| AUS | 1 | 86.96% | 5 | 0.7 | 4 | | |
| | 2 | 85.51% | 6 | 0.7 | 5 | $x_6, x_8, x_{10},$ | $x_1, x_2, x_3,$ |
| | 3 | 86.81% | 4.2 | 0.8 | 5 | x_{12} | x_{11}, x_{14} |
| HRT | 1 | 77.41% | 6.8 | 0.5 | 6 | | |
| | 2 | 70.37% | 7.6 | 0.6 | 9 | $x_3, x_4, x_{12},$ | x_6 |
| | 3 | 87.04% | 8.1 | 1 | 10 | x_{13} | |
| ION | 1 | 82.86% | 7.2 | 0.9 | 15 | | $x_{15}, x_{16}, x_{18},$ |
| | 2 | 90.29% | 7.3 | 1 | 16 | x_2, x_4, x_5, x_{27} | $x_{19}, x_{21}, x_{23},$ |
| | 3 | 86.57% | 5.6 | 0.6 | 6 | | x_{25}, x_{30}, x_{32} |
| PIM | 1 | 76.32% | 6.9 | 1 | 8 | | |
| | 2 | 64.74% | 5.6 | 0.7 | 7 | $x_1, x_3, x_4, x_5,$ | x_2 |
| | 3 | 64.21% | 7.4 | 0.9 | 8 | x_6, x_7 | |
| WDB | 1 | 94.29% | 8.2 | 0.7 | 15 | | |
| | 2 | 93.75% | 5 | 1 | 15 | $x_{21}, x_{22}, x_{24},$ | $x_1, x_5, x_6, x_8,$ |
| | 3 | 94.29% | 10.7 | 0.5 | 15 | x_{25} | x_{14}, x_{20}, x_{30} |

5.2. Regression dataset

5.2.1. 5-Fold CV

For regression dataset, the performance of HFNT^M was examined by using 5-fold CV method, in which the dataset was divided into 5 sets, each was 20% in size, and the process was repeated five times. Each time, four set was used to training and one set for testing. Hence, a total 5 runs were used for each model. As described in [78], MSE $E = 0.5 \times E$ was used for evaluating HFNT^M, where E was computed as per (1). The training MSE is represented as E_n and test MSE is represented as E_t . Such setting of MSE computation and cross-validation was taken for comparing the results collected from [78]. Table 8 presents results of 5-fold CV of each dataset for 30 models. Hence, each presented result is averaged over a total 150 runs of experiments. Similarly, in Table 9, a comparison between HFNT^M and other collected algorithms from literature is shown. It is evident from comparative results that HFNT^M performs very competitive to other algorithms. The literature results were averaged over 30 runs of experiments; whereas, HFNT^M results were averaged of 150 runs of experiments. Hence, a competitive result of HFNT^M is evidence of its efficiency.

Moreover, HFNT^M is distinct from the other algorithm mentioned in Table 9 because it performs feature selection and models complexity minimization, simultaneously. On the other hand, the other algorithms used entire available features. Therefore, the result’s comparisons were limited to assessing average MSE, where HFNT^M, which gives simple models in comparison to others, stands firmly competitive with the others. An illustration of the best model of regression dataset DEE is provided in Figure 7, where the model offered a test MSE E_t of 0.077, *tree size* equal to 10, and four selected input features (x_1, x_3, x_4 , and x_5). The selected activation functions were unipolar sigmoid ($+\frac{7}{2}$), bipolar sigmoid ($+\frac{6}{2}$), tangent hyperbolic ($+\frac{2}{2}$), and Gaussian ($+\frac{1}{2}$). Note that while creating HFNT models, the datasets were normalized as described in Table 2 and the output of models were denormalized accordingly. Therefore, normalized inputs should be presented to the tree (Figure 7), and the output y of the tree (Figure 7) should be denormalized.

Table 8: Best and mean results of 30 5-fold CV models (150 runs) of HFNT^M.

| Data | Best of 30 models | | | | Mean of 30 models | | | |
|------|-------------------|------------|------------------|-----------|-------------------|------------|------------------|-----------|
| | train E_n | test E_t | <i>tree size</i> | #Features | train E_n | test E_t | <i>tree size</i> | diversity |
| ABL | 2.228 | 2.256 | 14 | 5 | 2.578 | 2.511 | 11.23 | 0.7 |
| BAS | 198250 | 209582 | 11 | 5 | 261811 | 288688.6 | 7.69 | 0.6 |
| DEE | 0.076 | 0.077 | 10 | 4 | 0.0807 | 0.086 | 11.7 | 0.7 |
| ELV* | 8.33 | 8.36 | 11 | 7 | 1.35 | 1.35 | 7.63 | 0.5 |
| FRD | 2.342 | 2.425 | 6 | 5 | 3.218 | 3.293 | 6.98 | 0.34 |

Note: *Results of ELV should be multiplied with 10^{-5}

Table 9: Comparative results: 5-fold CV training MSE E_n and test MSE E_t of algorithms.

| Algorithms | ABL | | BAS | | DEE | | ELV* | | FRD | |
|----------------------------|-------|--------------|--------|---------------|--------|--------------|--------|-------------|-------|-------------|
| | E_n | E_t | E_n | E_t | E_n | E_t | E_n | E_t | E_n | E_t |
| MLP | - | 2.694 | - | 540302 | - | 0.101 | - | 2.04 | | 3.194 |
| ANFIS-SUB | 2.008 | 2.733 | 119561 | 1089824 | 3087 | 2083 | 61.417 | 61.35 | 0.085 | 3.158 |
| TSK-IRL | 2.581 | 2.642 | | | 0.545 | 882.016 | | | 0.433 | 1.419 |
| LINEAR-LMS | 2.413 | 2.472 | 224684 | 269123 | 0.081 | 0.085 | 4.254 | 4.288 | 3.612 | 3.653 |
| LEL-TSK | 2.04 | 2.412 | 9607 | 461402 | 0.662 | 0.682 | | | 0.322 | 1.07 |
| METSK-HD ^e | 2.205 | 2.392 | 47900 | 368820 | 0.03 | 0.103 | 6.75 | 7.02 | 1.075 | 1.887 |
| HFNT^M ** | 2.578 | 2.511 | 261811 | 288688.6 | 0.0807 | 0.086 | 1.35 | 1.35 | 3.218 | 3.293 |

Note: *ELV results should be multiplied with 10^{-5} , **HFNT^M results were averaged over 150 runs compared to MLP, ANFIS-SUB, TSK-IRL, LINEAR-LMS, LEL-TSK, and METSK-HD^e, which were averaged over 30 runs.

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 from Table 9 and the best test MSE of algorithm HFNT^M was considered from Table 8. The average ranks obtained by each method in the Friedman test is shown in Table 10. 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 10: Average rankings of the algorithms

| Algorithm | Ranking |
|-------------------------|---------|
| HFNT^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^M performed best among all the other algorithms. However, in the post-hoc analysis presented in Table 11 describes the significance of difference between the algorithms. For this purpose, we apply Holm’s method [77], 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 11).

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 11 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 11: 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 | |

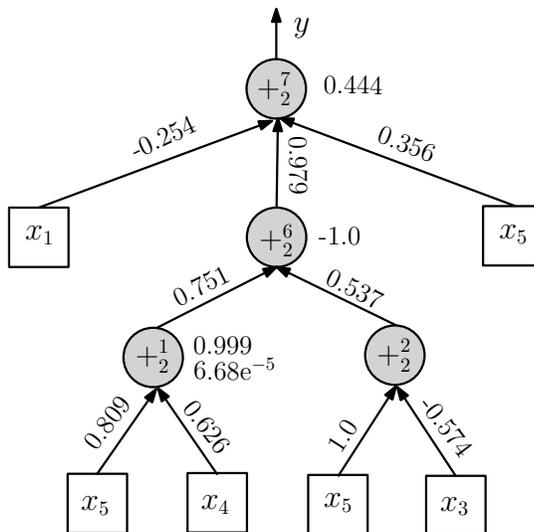


Figure 7: HFNT model of regression dataset DEE (test MSE $E_t = 0.077$).

5.2.2. Ensembles

For each dataset, we constructed five ensemble systems by using 10 models in each batch. In each batch, 10 models were created and cross-validated using 5×2 -fold CV. In 5×2 -fold CV, a dataset is randomly divided into two equal sets: A and B. Such partition of the dataset was repeated five times and each time when the set A was presented for training, the set B was presented for testing, and vice versa. Hence, total 10 runs of experiments for each model was performed. The collected ensemble results are presented in Table 12, where ensemble outputs were obtained by using weighted arithmetic mean as mentioned in (11).

The weights of models were computed by using DE algorithm, where the parameter setting was similar to the one mentioned in classification dataset. Ensemble results shown in Table 12 are MSE

Table 12: Ensemble test MSE E_t computed for 5×2 -fold CV of 10 model in each batch

| Data | batch | MSE E_t | r_t | avg. D | div (9) | TSF | MSF | MIF |
|-------|----------|-----------|-------------|----------|---------|-----|-----------------------|--------------------|
| ABL | 1 | 3.004 | 0.65 | 5 | 0.1 | 3 | | |
| | 2 | 2.537 | 0.72 | 8.3 | 1 | 7 | | |
| | 3 | 3.042 | 0.65 | 8.5 | 0.5 | 5 | $x_2, x_3, x_5,$ | x_1 |
| | 4 | 2.294 | 0.75 | 10.7 | 1 | 7 | x_6 | |
| | 5 | 2.412 | 0.73 | 11.2 | 0.7 | 7 | | |
| BAS* | 1 | 2.932 | 0.79 | 5.6 | 0.3 | 5 | | |
| | 2 | 3.275 | 0.76 | 8.2 | 0.3 | 6 | | |
| | 3 | 3.178 | 0.77 | 5 | 0.2 | 7 | $x_3, x_7, x_8,$ | $x_1, x_2, x_5,$ |
| | 4 | 3.051 | 0.78 | 5.7 | 0.3 | 5 | x_9, x_{11}, x_{13} | x_6, x_{10} |
| | 5 | 2.707 | 0.81 | 7.3 | 0.7 | 9 | | |
| DEE | 1 | 0.112 | 0.88 | 4.3 | 0.2 | 4 | | |
| | 2 | 0.115 | 0.88 | 8.9 | 0.6 | 6 | | |
| | 3 | 0.108 | 0.88 | 5.4 | 0.5 | 3 | $x_1, x_3, x_4,$ | x_2 |
| | 4 | 0.123 | 0.87 | 10.8 | 0.9 | 5 | x_5, x_6 | |
| | 5 | 0.111 | 0.88 | 5.2 | 0.6 | 4 | | |
| EVL** | 1 | 1.126 | 0.71 | 9.3 | 0.1 | 12 | | |
| | 2 | 1.265 | 0.67 | 9.6 | 0.1 | 12 | | |
| | 3 | 1.124 | 0.71 | 10.4 | 0.1 | 15 | $x_1, x_3, x_4,$ | x_7, x_8, x_{12} |
| | 4 | 1.097 | 0.72 | 9.2 | 0.2 | 10 | x_6, x_{17} | |
| | 5 | 2.047 | 0.31 | 3.8 | 0.4 | 3 | | |
| FRD | 1 | 3.987 | 0.86 | 6.2 | 0.2 | 4 | | |
| | 2 | 4.154 | 0.83 | 8 | 0.2 | 4 | | |
| | 3 | 4.306 | 0.83 | 5.2 | 0.4 | 5 | $x_1, x_2, x_4,$ | x_3 |
| | 4 | 3.809 | 0.86 | 7.8 | 0.5 | 4 | x_5 | |
| | 5 | 2.395 | 0.91 | 7.7 | 0.4 | 5 | | |

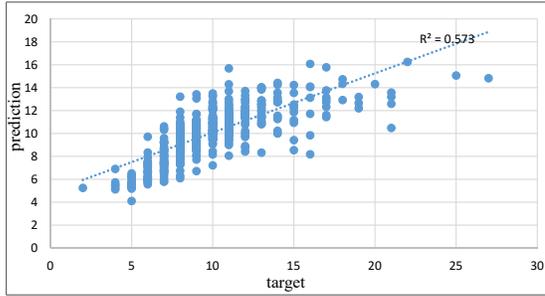
Note: *BAS results should be multiplied with 10^5 , **ELV results should be multiplied with 10^{-5} .

and correlation coefficient computed on CV test dataset. From ensemble results, it can be said that the ensemble with higher diversity offered better results than the ensemble with lower diversity. The models of the ensemble were examined to evaluate MSF and MIF presented in Table 12. A graphical illustration of ensemble results is shown in Figure 8 using scattered (regression) plots, where a scatter plots show how much one variable is affected by another (in this case model's and desired outputs). Moreover, it tells the relationship between two variables, i.e., their correlation. Plots shown in Figure 8 represents the best ensemble batch (numbers indicated bold in Table 12) four, five, three, four and five where MSEs are 2.2938, 270706, 0.1085, $1.10E-05$ and 2.3956, respectively. The values of r^2 in plots tell about the regression curve fitting over CV test datasets. In other words, it can be said that the ensemble models were obtained with generalization ability.

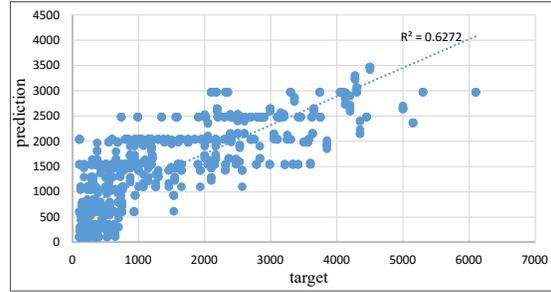
5.3. Time-series dataset

5.3.1. 2-Fold CV

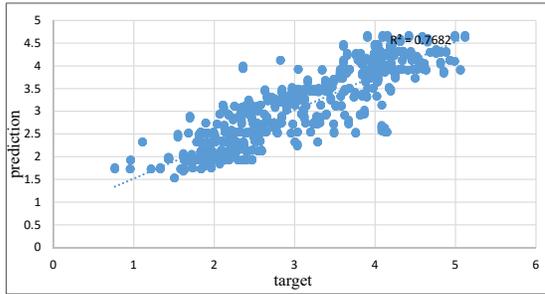
In literature survey, it was found that efficiency of most of the FNT-based models was evaluated over time-series dataset. Mostly, Macky-Glass (MGS) dataset was used for this purpose. However, only the best-obtained results were reported. For time-series prediction problems, the performances were computed



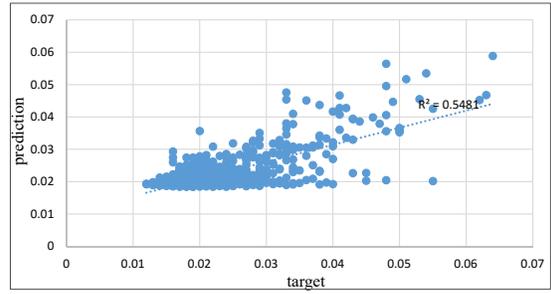
(a) Dataset ABL. $r_t = 0.75$



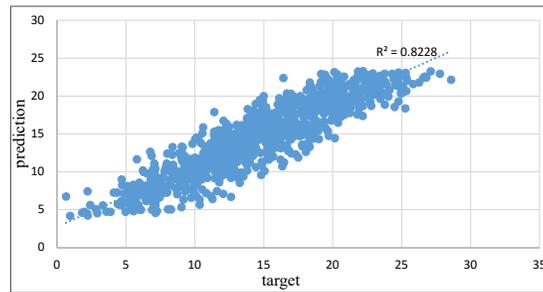
(b) Dataset BAS. $E_t = 0.81$



(c) Dataset DEE. $r_t = 0.88$



(d) Dataset EVL. $r_t = 0.72$



(e) Dataset FRD. $r_t = 0.91$

Figure 8: Regression plots of the best ensemble batches on datasets R1, R2, R3, R4, and R5.

using the root of mean squared error (RMSE), i.e., we took the square root of E given in (1). Additionally, correlation coefficient (2) was also used for evaluating algorithms performance.

For the experiments, first 50% of the dataset was taken for training and the rest of 50% was used for testing. Table 13 describes the results obtained by HFNT^M, where E_n is RMSE for training set and E_t is RMSE for test-set. The best test RMSE obtained by HFNT^M was $E_t = 0.00859$ and $E_t = 0.06349$ on datasets MGS and WWR, respectively. HFNT^M results are competitive with most of the algorithms listed in Table 14. Only a few algorithms such as LNF and FWNN-M reported better results than the one obtained by HFNT^M. FNT based algorithms such as FNT [1] and FBBFNT-EGP&PSO reported RMSEs close to the results obtained by HFNT^M. The average RMSEs and its variance over test-set of 70 models were 0.10568 and 0.00283, and 0.097783 and 0.00015 on dataset MGS and WWR, respectively. The low variance indicates that most models were able to produce results around the average RMSE value. The results reported by other function approximation algorithms (Table 13) were merely the best RMSEs. Hence, the robustness of other reported algorithm cannot be compared with the HFNT^M. However, the advantage of using HFNT^M over other algorithms is evident from the fact that the average complexity of the predictive models were 8.15 and 8.05 for datasets MGA and WWR, respectively.

The best model obtained for dataset WWR is shown in Figure 9, where the *tree size* is equal to 17 and followings are the selected activation functions: tangent hyperbolic, Gaussian, unipolar sigmoid, bipolar sigmoid and linear tangent hyperbolic. The selected input features in the tree (Figure 9) are x_1 , x_2 , x_3 and x_4 . Since in time series category experiment, we have only two datasets and for each dataset HFNT^M was compared with different models from literature. Hence, the statistical test was not conducted in this category because differences between algorithms are easy to determine from Table 14.

Table 13: Best and mean results 2-fold CV training RMSE E_n and test RMSE E_t .

| Data | Best of 70 models | | | | Mean of 70 models | | |
|------|-------------------|---------|-----|----------|-------------------|---------|------|
| | E_n | E_t | D | Features | E_n | E_t | D |
| MGS | 0.00859 | 0.00798 | 21 | 4 | 0.10385 | 0.10568 | 8.15 |
| WWR | 0.06437 | 0.06349 | 17 | 4 | 0.10246 | 0.09778 | 8.05 |

5.3.2. Ensembles

The ensemble results of time-series datasets are presented in Table 15, where the best ensemble system of dataset MGS (marked bold in Table 15) offered a test RMSE $E_t = 0.018151$ with a test correlation coefficient $r_t = 0.99$. Similarly, the best ensemble system of dataset WWR (marked bold in Table 15) offered a test RMSE $E_t = 0.063286$ with a test correlation coefficient $r_t = 0.953$. However, apart from the best results, most of the ensemble produced low RMSEs, i.e., high correlation coefficients. The best ensemble batches (marked bold in Table 15) of dataset MGS and WWR were used for graphical plots in Figure 10. A one-to-one fitting of target and prediction values is the evidence of a high correlation between model’s output and desired output, which is a significant indicator of model’s efficient performance.

Table 15: Ensemble results computed for 50% test samples of time-series datasets

| Data | batch | E_t | r_t | avg. tree size | div (9) | TSF | MSF | MIF |
|------|-------|-------|-------------|----------------|---------|-----|-----------------|-----|
| MGS | 1 | 0.018 | 0.99 | 9.4 | 0.6 | 4 | x_1, x_3, x_4 | - |
| | 2 | 0.045 | 0.98 | 5.8 | 0.2 | 3 | | |
| | 3 | 0.026 | 0.99 | 15.2 | 0.5 | 3 | | |
| | 4 | 0.109 | 0.92 | 5.1 | 0.4 | 3 | | |
| | 5 | 0.156 | 0.89 | 7 | 0.2 | 3 | | |
| | 6 | 0.059 | 0.97 | 8.2 | 0.5 | 3 | | |
| | 7 | 0.054 | 0.98 | 6.4 | 0.4 | 4 | | |
| WWR | 1 | 0.073 | 0.94 | 5 | 0.1 | 3 | x_1, x_2 | - |
| | 2 | 0.112 | 0.85 | 6 | 0.2 | 2 | | |
| | 3 | 0.097 | 0.91 | 10.6 | 0.3 | 4 | | |
| | 4 | 0.113 | 0.84 | 5 | 0.1 | 2 | | |
| | 5 | 0.063 | 0.96 | 14.4 | 0.9 | 4 | | |
| | 6 | 0.099 | 0.89 | 8.5 | 0.7 | 3 | | |
| | 7 | 0.101 | 0.88 | 6.9 | 0.4 | 3 | | |

Note: E_t , r_t , and div indicate test RMSE, test correlation coefficient, and diversity, respectively

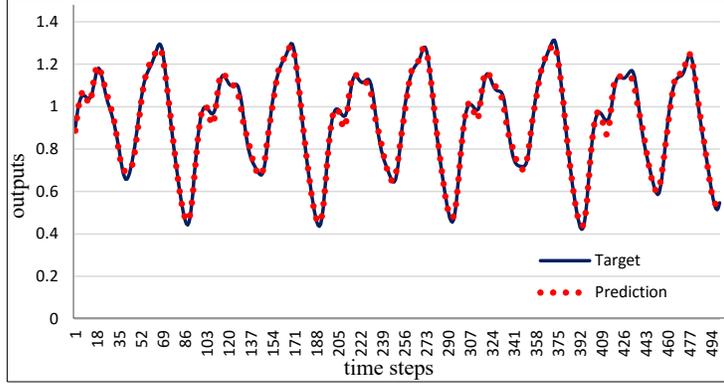
population were diverse according to structure, parameters, activation function, and input feature. Hence, the model’s selection from Pareto-fronts, as indicated in Section 4, led to a good ensemble system.

Table 16: Performance of activation functions during the best performing ensembles

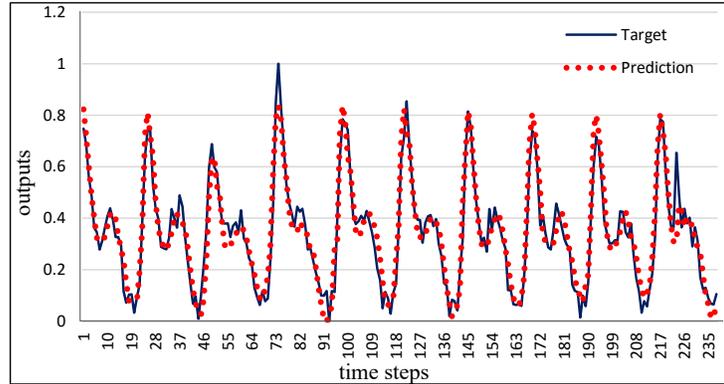
| Data | activation function (k) | | | | | | |
|-------|-----------------------------|----|----|----|----|----|----|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| AUS | 10 | - | - | 2 | - | - | - |
| HRT | 10 | - | 9 | 4 | - | 5 | 3 |
| ION | 6 | 5 | - | - | 2 | 4 | 4 |
| PIM | 3 | 8 | 2 | 5 | 2 | 1 | - |
| WDB | - | 3 | - | 7 | 8 | 10 | 8 |
| ABL | 2 | 10 | - | - | - | 10 | - |
| BAS | 2 | 5 | - | - | 2 | 10 | - |
| DEE | - | 6 | 6 | 4 | 4 | 10 | - |
| EVL | 10 | 5 | - | 3 | - | - | 6 |
| FRD | 10 | 10 | - | - | - | - | - |
| MGS | 4 | 1 | - | 2 | 1 | 10 | 10 |
| WWR | 10 | - | 4 | - | 4 | 7 | - |
| Total | 67 | 53 | 21 | 27 | 23 | 67 | 31 |

Note: 67 is the best and 21 is the worst

HFNT^M was applied to solve classification, regression, and time-series problems. Since HFNT^M is stochastic in nature, its performance was affected by several factors: random generator algorithm, random seed, the efficiency of the meta-heuristic algorithm used in *parameter-tuning* phase, the activation function



(a) Dataset MGS $E_t = 01815$



(b) Dataset WWR $E_t = 0.06328$

Figure 10: Target versus prediction plot obtained for time-series datasets MGS and WWR.

selected at the nodes, etc. Therefore, to examine the performance of HFNT^M, several HFNT-models were created using different random seeds and the best and average *approximation error* of all created models were examined. In Section 5, as far as the best model is concerned, the performance of HFNT^M surpass other approximation models mentioned from literature. Additionally, in the case of each dataset, a very low average value (high accuracy in the case of classification and low approximation errors in case of regression and time-series) were obtained, which significantly suggests that HFNT^M often led to good solutions. Similarly, in the case of the ensembles, it is clear from the result that combined output of diverse and accurate candidates offered high quality (in terms of generalization ability and accuracy) approximation/prediction model. From the results, it is clear that the final population of HFNT^M offered the best ensemble when the models were carefully examined based on *approximation error*, average complexity (*tree size*), and selected features.

Moreover, the performances of the best performing activation functions were examined. For this purpose, the best ensemble system obtained for each dataset were considered. Accordingly, the performance of activation functions was evaluated as follows. The best ensemble system of each dataset had 10 models;

therefore, in how many models (among 10) an activation function k appeared, was counted. Hence, for a dataset, if an activation function appeared in all models of an ensemble system, then the total count was 10. Subsequently, counting was performed for all the activation functions for the best ensemble systems of all the datasets. Table 16, shows the performance of the activation functions. It can be observed that the activation function Gaussian ($k = 1$) and Bipolar Sigmoid ($k = 6$) performed the best among all the other activation functions followed by Tangent-hyperbolic ($k = 2$) function. Hence, no one activation function performed exceptionally well. Therefore, the efforts of selecting activation function, adaptively, by MOGP was essential in HFNTs performance.

In this work, we were limited to examine the performance of our approach to only benchmark problems. Therefore, in presences of *no free lunch theorem* [79, 80] and the algorithm’s dependencies on random number generator, which are platforms, programming language, and implementation sensitive [81], it is clear that performance of the mentioned approach is subjected to careful choice of training condition and parameter-setting when it comes to deal with other real-world problems.

7. Conclusion

Effective use of the final population of the heterogeneous flexible neural trees (HFNTs) evolved using Pareto-based multiobjective genetic programming (MOGP) and the subsequent parameter tuning by differential evolution led to the formation of high-quality ensemble systems. The simultaneous optimization of accuracy, complexity, and diversity solved the problem of structural complexity that was inevitably imposed when a single objective was used. MOGP used in the *tree construction* phase often guided an initial HFNT population towards a population in which the candidates were highly accurate, structurally simple, and diverse. Therefore, the selected candidates helped in the formation of a good ensemble system. The result obtained by HFNT^M approach supports its superior performance over the algorithms collected for the comparison. In addition, HFNT^M provides adaptation in structure, computational nodes, and input feature space. Hence, HFNT is an effective algorithm for automatic feature selection, data analysis, and modeling.

Acknowledgment

This work was supported by the IPROCUM Marie Curie Initial Training Network, funded through the People Programme (Marie Curie Actions) of the European Unions Seventh Framework Programme FP7/20072013/, under REA grant agreement number 316555.

References

- [1] Y. Chen, B. Yang, J. Dong, A. Abraham, Time-series forecasting using flexible neural tree model, Information Sciences 174 (3) (2005) 219–235.

- [2] X. Yao, Y. Liu, A new evolutionary system for evolving artificial neural networks, *IEEE Transactions on Neural Networks* 8 (3) (1997) 694–713.
- [3] I. Basheer, M. Hajmeer, Artificial neural networks: Fundamentals, computing, design, and application, *Journal of Microbiological Methods* 43 (1) (2000) 3–31.
- [4] A. J. Maren, C. T. Harston, R. M. Pap, *Handbook of neural computing applications*, Academic Press, 2014.
- [5] I. K. Sethi, A. K. Jain, *Artificial neural networks and statistical pattern recognition: Old and new connections*, Vol. 1, Elsevier, 2014.
- [6] M. Tkáč, R. Verner, Artificial neural networks in business: Two decades of research, *Applied Soft Computing* 38 (2016) 788–804.
- [7] S. E. Fahlman, C. Lebière, The cascade-correlation learning architecture, in: D. S. Touretzky (Ed.), *Advances in Neural Information Processing Systems 2*, Morgan Kaufmann Publishers Inc., 1990, pp. 524–532.
- [8] J.-P. Nadal, Study of a growth algorithm for a feedforward network, *International Journal of Neural Systems* 1 (1) (1989) 55–59.
- [9] K. O. Stanley, R. Miikkulainen, Evolving neural networks through augmenting topologies, *Evolutionary Computation* 10 (2) (2002) 99–127.
- [10] B.-T. Zhang, P. Ohm, H. Mühlenbein, Evolutionary induction of sparse neural trees, *Evolutionary Computation* 5 (2) (1997) 213–236.
- [11] M. A. Potter, K. A. De Jong, Cooperative coevolution: An architecture for evolving coadapted subcomponents, *Evolutionary computation* 8 (1) (2000) 1–29.
- [12] M. Yaghini, M. M. Khoshraftar, M. Fallahi, A hybrid algorithm for artificial neural network training, *Engineering Applications of Artificial Intelligence* 26 (1) (2013) 293–301.
- [13] S. Wang, Y. Zhang, Z. Dong, S. Du, G. Ji, J. Yan, J. Yang, Q. Wang, C. Feng, P. Phillips, Feed-forward neural network optimized by hybridization of PSO and ABC for abnormal brain detection, *International Journal of Imaging Systems and Technology* 25 (2) (2015) 153–164.
- [14] S. Wang, Y. Zhang, G. Ji, J. Yang, J. Wu, L. Wei, Fruit classification by wavelet-entropy and feed-forward neural network trained by fitness-scaled chaotic abc and biogeography-based optimization, *Entropy* 17 (8) (2015) 5711–5728.
- [15] R. Salustowicz, J. Schmidhuber, Probabilistic incremental program evolution, *Evolutionary Computation* 5 (2) (1997) 123–141.

- [16] A. K. Kar, Bio inspired computing—a review of algorithms and scope of applications, *Expert Systems with Applications* 59 (2016) 20–32.
- [17] Y. Jin, B. Sendhoff, Pareto-based multiobjective machine learning: An overview and case studies, *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* 38 (3) (2008) 397–415.
- [18] K. Deb, *Multi-objective optimization using evolutionary algorithms*, Vol. 16, John Wiley & Sons, 2001.
- [19] X. Yao, Y. Liu, Making use of population information in evolutionary artificial neural networks, *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 28 (3) (1998) 417–425.
- [20] L. I. Kuncheva, C. J. Whitaker, Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy, *Machine Learning* 51 (2) (2003) 181–207.
- [21] K. Deb, S. Agrawal, A. Pratap, T. Meyarivan, A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II, in: *Parallel Problem Solving from Nature PPSN VI*, Vol. 1917 of *Lecture Notes in Computer Science*, Springer, 2000, pp. 849–858.
- [22] S. Das, S. S. Mullick, P. Suganthan, Recent advances in differential evolution—an updated survey, *Swarm and Evolutionary Computation* 27 (2016) 1–30.
- [23] Y. Chen, A. Abraham, J. Yang, Feature selection and intrusion detection using hybrid flexible neural tree, in: *Advances in Neural Networks—ISNN*, Vol. 3498 of *Lecture Notes in Computer Science*, Springer, 2005, pp. 439–444.
- [24] L. Sánchez, I. Couso, J. A. Corrales, Combining GP operators with SA search to evolve fuzzy rule based classifiers, *Information Sciences* 136 (1) (2001) 175–191.
- [25] J. Kennedy, R. C. Eberhart, Y. Shi, *Swarm Intelligence*, Morgan Kaufmann, 2001.
- [26] Y. Chen, A. Abraham, B. Yang, Feature selection and classification using flexible neural tree, *Neurocomputing* 70 (1) (2006) 305–313.
- [27] R. Riolo, J. H. Moore, M. Kotanchek, *Genetic programming theory and practice XI*, Springer, 2014.
- [28] X. Chen, Y.-S. Ong, M.-H. Lim, K. C. Tan, A multi-facet survey on memetic computation, *IEEE Transactions on Evolutionary Computation* 15 (5) (2011) 591–607.
- [29] H.-J. Li, Z.-X. Wang, L.-M. Wang, S.-M. Yuan, Flexible neural tree for pattern recognition, in: *Advances in Neural Networks—ISNN*, Vol. 3971 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2006, pp. 903–908.

- [30] Y. Chen, Y. Wang, B. Yang, Evolving hierarchical RBF neural networks for breast cancer detection, in: *Neural Information Processing*, Vol. 4234 of *Lecture Notes in Computer Science*, Springer, 2006, pp. 137–144.
- [31] Y. Chen, F. Chen, J. Yang, Evolving MIMO flexible neural trees for nonlinear system identification, in: *International Conference on Artificial Intelligence*, Vol. 1, 2007, pp. 373–377.
- [32] P. Wu, Y. Chen, Grammar guided genetic programming for flexible neural trees optimization, in: *Advances in Knowledge Discovery and Data Mining*, Springer, 2007, pp. 964–971.
- [33] Y. Shan, R. McKay, R. Baxter, H. Abbass, D. Essam, H. Nguyen, Grammar model-based program evolution, in: *Congress on Evolutionary Computation*, Vol. 1, 2004, pp. 478–485.
- [34] G. Jia, Y. Chen, Q. Wu, A MEP and IP based flexible neural tree model for exchange rate forecasting, in: *Fourth International Conference on Natural Computation*, Vol. 5, IEEE, 2008, pp. 299–303.
- [35] M. Oltean, C. Groşan, Evolving evolutionary algorithms using multi expression programming, in: *Advances in Artificial Life*, Springer, 2003, pp. 651–658.
- [36] P. Musilek, A. Lau, M. Reformat, L. Wyard-Scott, Immune programming, *Information Sciences* 176 (8) (2006) 972–1002.
- [37] B. Yang, L. Wang, Z. Chen, Y. Chen, R. Sun, A novel classification method using the combination of FDPS and flexible neural tree, *Neurocomputing* 73 (46) (2010) 690 – 699.
- [38] S. Bouaziz, H. Dhahri, A. M. Alimi, A. Abraham, Evolving flexible beta basis function neural tree using extended genetic programming & hybrid artificial bee colony, *Applied Soft Computing*.
- [39] Y. Chen, B. Yang, A. Abraham, Flexible neural trees ensemble for stock index modeling, *Neurocomputing* 70 (46) (2007) 697 – 703.
- [40] B. Yang, M. Jiang, Y. Chen, Q. Meng, A. Abraham, Ensemble of flexible neural tree and ordinary differential equations for small-time scale network traffic prediction, *Journal of Computers* 8 (12) (2013) 3039–3046.
- [41] V. K. Ojha, A. Abraham, V. Snasel, 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 Second International Afro-European Conference for Industrial Advancement AECIA 2015*, Springer, 2016, pp. 155–165.
- [42] L. Peng, B. Yang, L. Zhang, Y. Chen, A parallel evolving algorithm for flexible neural tree, *Parallel Computing* 37 (10–11) (2011) 653–666.

- [43] L. Wang, B. Yang, Y. Chen, X. Zhao, J. Chang, H. Wang, Modeling early-age hydration kinetics of portland cement using flexible neural tree, *Neural Computing and Applications* 21 (5) (2012) 877–889.
- [44] C. Ferreira, *Gene expression programming: mathematical modeling by an artificial intelligence*, Vol. 21, Springer, 2006.
- [45] G. Weiss, *Multiagent systems: A modern approach to distributed artificial intelligence*, MIT Press, 1999.
- [46] M. Ammar, S. Bouaziz, A. M. Alimi, A. Abraham, Negotiation process for bi-objective multi-agent flexible neural tree model, in: *International Joint Conference on Neural Networks (IJCNN)*, 2015, IEEE, 2015, pp. 1–9.
- [47] T. Burianek, S. Basterrech, Performance analysis of the activation neuron function in the flexible neural tree model, in: *Proceedings of the Dateso 2014 Annual International Workshop on Databases, TExts, Specifications and Objects*, 2014, pp. 35–46.
- [48] S. Bouaziz, H. Dhahri, A. M. Alimi, A. Abraham, A hybrid learning algorithm for evolving flexible beta basis function neural tree model, *Neurocomputing* 117 (2013) 107–117.
- [49] S. Bouaziz, A. M. Alimi, A. Abraham, Universal approximation propriety of flexible beta basis function neural tree, in: *International Joint Conference on Neural Networks*, IEEE, 2014, pp. 573–580.
- [50] C. Micheloni, A. Rani, S. Kumar, G. L. Foresti, A balanced neural tree for pattern classification, *Neural Networks* 27 (2012) 81–90.
- [51] G. L. Foresti, C. Micheloni, Generalized neural trees for pattern classification, *IEEE Transactions on Neural Networks* 13 (6) (2002) 1540–1547.
- [52] A. Rani, G. L. Foresti, C. Micheloni, A neural tree for classification using convex objective function, *Pattern Recognition Letters* 68 (2015) 41–47.
- [53] Q. Shou-ning, L. Zhao-lian, C. Guang-qiang, Z. Bing, W. Su-juan, Modeling of cement decomposing furnace production process based on flexible neural tree, in: *Information Management, Innovation Management and Industrial Engineering*, Vol. 3, IEEE, 2008, pp. 128–133.
- [54] B. Yang, Y. Chen, M. Jiang, Reverse engineering of gene regulatory networks using flexible neural tree models, *Neurocomputing* 99 (2013) 458–466.
- [55] Z. Chen, B. Yang, Y. Chen, A. Abraham, C. Grosan, L. Peng, Online hybrid traffic classifier for peer-to-peer systems based on network processors, *Applied Soft Computing* 9 (2) (2009) 685–694.

- [56] T. Novosad, J. Platos, V. Snásel, A. Abraham, Fast intrusion detection system based on flexible neural tree, in: International Conference on Information Assurance and Security, IEEE, 2010, pp. 106–111.
- [57] Y.-Q. Pan, Y. Liu, Y.-W. Zheng, Face recognition using kernel PCA and hybrid flexible neural tree, in: International Conference on Wavelet Analysis and Pattern Recognition, 2007. ICWAPR'07, Vol. 3, IEEE, 2007, pp. 1361–1366.
- [58] Y. Guo, Q. Wang, S. Huang, A. Abraham, Flexible neural trees for online hand gesture recognition using surface electromyography, *Journal of Computers* 7 (5) (2012) 1099–1103.
- [59] S. Qu, A. Fu, W. Xu, Controlling shareholders management risk warning based on flexible neural tree, *Journal of Computers* 6 (11) (2011) 2440–2445.
- [60] A. Rajini, V. K. David, Swarm optimization and flexible neural tree for microarray data classification, in: International Conference on Computational Science, Engineering and Information Technology, ACM, 2012, pp. 261–268.
- [61] S. Abdelwahab, V. K. Ojha, A. Abraham, Ensemble of flexible neural trees for predicting risk in grid computing environment, in: Innovations in Bio-Inspired Computing and Applications, Springer, 2016, pp. 151–161.
- [62] Y. Jin, B. Sendhoff, E. Körner, Evolutionary multi-objective optimization for simultaneous generation of signal-type and symbol-type representations, in: Evolutionary Multi-Criterion Optimization, Vol. 3410 of Lecture Notes in Computer Science, Springer, 2005, pp. 752–766.
- [63] I. Das, J. E. Dennis, A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems, *Structural optimization* 14 (1) (1997) 63–69.
- [64] A. E. Eiben, J. E. Smith, Introduction to Evolutionary Computing, Springer, 2015.
- [65] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm, *Journal of Global Optimization* 39 (3) (2007) 459–471.
- [66] Y. Zhang, S. Wang, G. Ji, A comprehensive survey on particle swarm optimization algorithm and its applications, *Mathematical Problems in Engineering* 2015 (2015) 1–38.
- [67] C.-F. Juang, A hybrid of genetic algorithm and particle swarm optimization for recurrent network design, *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 34 (2) (2004) 997–1006.

- [68] W. Wongseree, N. Chaiyaratana, K. Vichittumaros, P. Winichagoon, S. Fucharoen, Thalassaemia classification by neural networks and genetic programming, *Information Sciences* 177 (3) (2007) 771–786.
- [69] T. Hastie, R. Tibshirani, J. Friedman, T. Hastie, J. Friedman, R. Tibshirani, *The elements of statistical learning*, Vol. 2, Springer, 2009.
- [70] R. Polikar, Ensemble based systems in decision making, *IEEE Circuits and Systems Magazine* 6 (3) (2006) 21–45.
- [71] Z.-H. Zhou, *Ensemble methods: Foundations and algorithms*, CRC Press, 2012.
- [72] M. Lichman, UCI machine learning repository, <http://archive.ics.uci.edu/ml> Accessed on: 01.05.2016 (2013).
- [73] J. Alcalá-Fdez, L. Sanchez, S. Garcia, M. J. del Jesus, S. Ventura, J. Garrell, J. Otero, C. Romero, J. Bacardit, V. M. Rivas, et al., Keel: a software tool to assess evolutionary algorithms for data mining problems, *Soft Computing* 13 (3) (2009) 307–318.
- [74] M. Matsumoto, T. Nishimura, Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator, *ACM Transactions on Modeling and Computer Simulation* 8 (1) (1998) 3–30.
- [75] V. K. Ojha, MOGP-FNT multiobjective flexible neural tree tool, <http://dap.vsb.cz/aat/> Accessed on: 01.05.2016 (May 2016).
- [76] Y. Chen, A. Abraham, Y. Zhang, et al., Ensemble of flexible neural trees for breast cancer detection, *The International Journal of Information Technology and Intelligent Computing* 1 (1) (2006) 187–201.
- [77] S. Holm, A simple sequentially rejective multiple test procedure, *Scandinavian Journal of Statistics* (1979) 65–70.
- [78] M. J. Gacto, M. Galende, R. Alcalá, F. Herrera, METSK-HD^e: A multiobjective evolutionary algorithm to learn accurate TSK-fuzzy systems in high-dimensional and large-scale regression problems, *Information Sciences* 276 (2014) 63–79.
- [79] D. H. Wolpert, W. G. Macready, No free lunch theorems for optimization, *IEEE Transactions on Evolutionary Computation* 1 (1) (1997) 67–82.
- [80] M. Koppen, D. H. Wolpert, W. G. Macready, Remarks on a recent paper on the” no free lunch” theorems, *IEEE Transactions on Evolutionary Computation* 5 (3) (2001) 295–296.

- [81] P. L'Ecuyer, F. Panneton, Fast random number generators based on linear recurrences modulo 2: Overview and comparison, in: *Proceedings of the 2005 Winter Simulation Conference*, IEEE, 2005, pp. 10–pp.
- [82] S. Haykin, *Neural networks and learning machines*, Vol. 3, Pearson Education Upper Saddle River, 2009.
- [83] Z.-H. Zhou, Z.-Q. Chen, Hybrid decision tree, *Knowledge-Based Systems* 15 (8) (2002) 515–528.
- [84] J.-S. R. Jang, ANFIS: adaptive-network-based fuzzy inference system, *IEEE Transactions on Systems, Man and Cybernetics* 23 (3) (1993) 665–685.
- [85] O. Cordón, F. Herrera, A two-stage evolutionary process for designing TSK fuzzy rule-based systems, *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 29 (6) (1999) 703–715.
- [86] J. S. Rustagi, *Optimization techniques in statistics*, Academic Press, 1994.
- [87] R. Alcalá, J. Alcalá-Fdez, J. Casillas, O. Cordón, F. Herrera, Local identification of prototypes for genetic learning of accurate tsk fuzzy rule-based systems, *International Journal of Intelligent Systems* 22 (9) (2007) 909–941.
- [88] K. B. Cho, B. H. Wang, Radial basis function based adaptive fuzzy systems and their applications to system identification and prediction, *Fuzzy Sets and Systems* 83 (3) (1996) 325–339.
- [89] F. Van den Bergh, A. P. Engelbrecht, A cooperative approach to particle swarm optimization, *IEEE Transactions on Evolutionary Computation* 8 (3) (2004) 225–239.
- [90] A. M. A. H. Dhahri, F. Karray, Designing beta basis function neural network for optimization using particle swarm optimization, in: *IEEE Joint Conference on Neural Network*, 2008, pp. 2564–2571.
- [91] C. Aouiti, A. M. Alimi, A. Maalej, A genetic designed beta basis function neural network for approximating multi-variables functions, in: *International Conference on Artificial Neural Nets and Genetic Algorithms*, Springer, 2001, pp. 383–386.
- [92] C.-F. Juang, C.-M. Hsiao, C.-H. Hsu, Hierarchical cluster-based multispecies particle-swarm optimization for fuzzy-system optimization, *IEEE Transactions on Fuzzy Systems* 18 (1) (2010) 14–26.
- [93] S. Yilmaz, Y. Oysal, Fuzzy wavelet neural network models for prediction and identification of dynamical systems, *IEEE Transactions on Neural Networks* 21 (10) (2010) 1599–1609.
- [94] H. Dhahri, A. M. Alimi, A. Abraham, Hierarchical multi-dimensional differential evolution for the design of beta basis function neural network, *Neurocomputing* 97 (2012) 131–140.

- [95] A. Miranian, M. Abdollahzade, Developing a local least-squares support vector machines-based neuro-fuzzy model for nonlinear and chaotic time series prediction, *IEEE Transactions on Neural Networks and Learning Systems* 24 (2) (2013) 207–218.
- [96] N. K. Kasabov, *Foundations of neural networks, fuzzy systems, and knowledge engineering*, Marcel Alencar, 1996.
- [97] N. Kasabov, *Evolving fuzzy neural networks for adaptive, on-line intelligent agents and systems*, in: *Recent Advances in Mechatronics*, Springer, Berlin, 1999.
- [98] S. Bouaziz, A. M. Alimi, A. Abraham, Extended immune programming and opposite-based PSO for evolving flexible beta basis function neural tree, in: *IEEE International Conference on Cybernetics*, IEEE, 2013, pp. 13–18.

Appendix A. Dataset Description

Table A.17: Collected datasets for testing HFNT^M

| Index | Name | Features | Samples | Output | Type |
|-------|--------------|----------|---------|--------|----------------|
| AUS | Australia | 14 | 691 | 2 | |
| HRT | Heart | 13 | 270 | 2 | |
| ION | Ionshpere | 33 | 351 | 2 | Classification |
| PIM | Pima | 8 | 768 | 2 | |
| WDB | Wdbc | 30 | 569 | 2 | |
| ABL | Abalone | 8 | 4177 | 1 | |
| BAS | Baseball | 16 | 337 | 1 | |
| DEE | DEE | 6 | 365 | 1 | Regression |
| EVL | Elevators | 18 | 16599 | 1 | |
| FRD | Fridman | 5 | 1200 | 1 | |
| MGS | Mackey-Glass | 4 | 1000 | 1 | |
| WWR | Waste Water | 4 | 475 | 1 | Time-series |

Appendix B. Algorithms from literature

Table B.18: Algorithms from literature for the comparative study with HFNT^M

| Ref. | Algorithms | Definition |
|------|-----------------------|---|
| [82] | MLP | Multi-layer Perceptron |
| [83] | HDT | Hybrid Decision Tree |
| [76] | FNT | Flexible Neural Tree |
| [84] | ANFIS-SUB | Adaptive Neuro-Fuzzy Inference System Using Subtractive Clustering |
| [85] | TSK-IRL | Genetic Learning of TSK-rules Under Iterative Rule Learning |
| [86] | LINEAR-LMS | Least Mean Squares Linear Regression |
| [87] | LEL-TSK | Local Evolutionary Learning of TSK-rules |
| [88] | RBF | Classical Radial Basis Function |
| [89] | CPSO | Cooperative Particle Swarm Optimization (PSO) |
| [90] | PSO-BBFN | PSO-based Beta Basis Function Neural Network |
| [91] | G-BBFNN | GA-based BBFNN |
| [92] | HCMSPSO | Hierarchical Cluster-Based Multispecies PSO |
| [93] | FWNN-M | Fuzzy Wavelet Neural Network Models |
| [94] | HMDDE-BBFNN | Hierarchical Multidimensional DE-Based BBFNN |
| [95] | LNF | Local Least-Squares Support Vector Machines-Based Neuro-Fuzzy Mode |
| [96] | BPNN | Back-propagation Neural Network |
| [97] | EFuNNs | Evolving Fuzzy Neural Networks |
| [98] | FBFNT-EGP&PSO | Extended Immune Programming and Opposite-PSO for Flexible BBFNN |
| [78] | METSK-HD ^e | Multiobjective Evolutionary Learning of TSK-rules for High-Dimensional Problems |