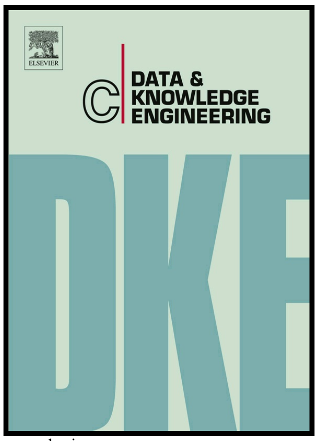
A novel evolutionary multi-objective ensemble learning approach for forecasting currency exchange rates

Lam Thu Bui, Vu Van Truong, Dinh Thi Thu Huong


www.elsevier.com

PII: S0169-023X(17)30054-X
DOI: http://dx.doi.org/10.1016/j.datak.2017.07.001
Reference: DATAK1598

To appear in: *Data & Knowledge Engineering*

Received date: 31 January 2017
Revised date: 7 July 2017
Accepted date: 10 July 2017

A novel evolutionary multi-objective ensemble learning approach for forecasting currency exchange rates

Lam Thu Bui[a], Vu Van Truong[a,*], Dinh Thi Thu Huong[b]

[a]*Le Quy Don Technical University, Hanoi, Vietnam*
[b]*Sai Gon University, Ho Chi Minh, Viet Nam*
∗Corresponding author. Truongvv@mta.edu.vn

Abstract: Due to the potential impact of the (currency) exchange rate risk in the financial market, forecasting exchange rate (FET) has become a hot topic in both academic and practical worlds. For many years, the various methods have been proposed and used for FET problems including the method of the artificial neural network (ANN). However, in many cases of FET, there is the limitation of using separate methods since they are not able to fully capture financial characteristics. Recently, more researchers have been beginning to pay attention to FET based on an ensemble of forecasting models (in other words, the combination of individual methods). Previous studies of ensemble methods have shown that the performance of an ensemble depends on two key elements *(1) The individual performance* and *(2) diversity degree of base learners*. The main idea behind this paper comes from these key elements, the authors use ANNs as the base method (or weak learners), and weights of these ANNs will be optimized by using the Non-Dominated Sorting Genetic Algorithms (NSGA). To assist NSGA, a number of diversity-preservation mechanisms are used to generate diverse sets of base classifiers and finally we propose to use modified Adaboost algorithms to combine the results of weak learners for overall forecasts. The results show that the proposed novel ensemble learning approach can achieve higher forecasting performance than those of individual ones.

Keywords: Currency exchange rates forecasting, ensemble learning, multi-objective evolutionary, non-dominated differential evolution.

1. INTRODUCTION

Nowadays, exchange rates play a vital role in controlling dynamics of the foreign exchange market. Economists and investors always tend to forecast the future exchange rates so that they can exploit the predictions to derive monetary values. For decades, FET has been a widely and continually studied topic in the financial field. There are a lot of computational methods being used for forecasting. These methods can be divided into 2 groups: *single (or base)* and *combination (or ensemble)* methods. Many base methods have been presented for FET, including traditional techniques (ARIMA, logistic regression, multiple regressions) and some nonlinear models (ANNs, SVM...). However, it has been indicated that individual forecasting methods still have a limited capability since each classifier has unstable results in many cases of datasets. An ensemble method is expected to reduce the variance of estimated errors and to improve the stability of overall prediction performance.

For ensemble methods, diversity and performance of members are key factors to generate a successful model. If the base methods are identical or give same results, then the ensemble brings no improvement except increasing the complexity. Further, if an ensemble member has very poor performance, the combination can't get the good result even though they are totally diverse [1]. So, the issue is how to design an ensemble of base methods that simultaneously ensure the above mentioned factors? This paper will tackle this issue.

In general, there are three approaches to generate diversity of base methods in ensemble learning [1]. One is to apply different base methods to a single dataset (e.g. using SVM, ANNs and ARIMA as base methods). Another is to apply one type of a base method with different parameters settings to a single dataset (e.g. using an ensemble of ANNs with different weights or topologies...). The last is an application of a single learning algorithm to different versions of a given dataset (i.e. an ensemble member is generated by applying a base learning algorithm to different distributions of the training dataset at each iteration).

In this paper, we select ANN as the base method and then apply each network to a sample data. The main contribution of this paper is to propose new methods for generating base learners. In which, we simultaneously consider diversity and performance. We aim at trying having different forms of handling the diversity. The key idea of the work is to use a combination method based on the multi-objective evolutionary algorithm (MOEA), the back propagation (BP) and the modified Adaboost. A base learner is actually an ANN. We encode each ANN or a group of ANNs as an individual in the framework of a multi-objective approach. As stated above, to ensure key factors in evaluating each individual, we choose diversity and performance (i.e. mean square error- MSE) as two objectives. After being trained for many generations by MOEA, the best individuals will be selected and continue improved by BP; finally, these best individuals will play as weak-learners of Adaboost. More detail will be described in the next sections.

The rest of the paper is divided into six parts. The literature review is introduced in Section 2. Section 3 presents the methodology. We describe our proposed ensemble learning approach in Section 4. Section 5 is about the empirical experiments, in which experimental data is described and analysis is made according to experimental results. Finally, we make a conclusion and discuss future research in Section 6.

2. BACKGROUND

Enhancing performance of forecasting exchange rate models has received considerable attention for many decades [2-5]. In early years, there were various traditional techniques being used to analyze and forecast time series problems. Most of them only paid attention to a certain single method. A typical example is reported in [6] using univariate approaches for predicting corporate bankruptcy. The logistic model describing the insolvency risk was applied in Laitinen and Laitinen [7]. In Taylor [8], authors described methods for choosing and assessing volatility forecasts using open, high, low and close prices and they then used a combination of the stationary and non-stationary forecasts for optimizing parameters of an ARMA model.

Soon after the introduction of back-propagation (BP) algorithm in Rumelhart, et al. [9], ANN was often used for forecasting. To our best of knowledge, Werbos [10] was the first study about applications of BP for forecasting over time with a model of natural gas markets. They found that ANNs, trained with BP, outperform the traditional statistical methods such as regression and Box-Jenkins. Hann and Steurer [11] indicated that ANNs seem better than the linear models on weekly data. Leung, et al. [12] compared performance of general regression neural networks (GRNN) with a variety of forecasting techniques, including multi-layered feed-forward network (MLFN), multivariate transfer function, and random walk models. The results show that GRNN not only has a higher degree of currency forecasting accuracy but also performs statistically better than that of other evaluated models for different data series.

Due to the limitations of using only single methods, some hybrid methods were used for FET. Zhang [13] proposed a hybrid methodology that combines both ARIMA and ANN models. Authors took advantages of the unique strength of ARIMA in linear and ANN in nonlinear modeling. Experimental results with real datasets indicated that the combined model could be an effective way to improve overall forecasting accuracy. Ince and Trafalis [14] proposed a two-stage forecasting model which incorporates parametric (ARIMA, VAR) and nonparametric techniques (Support Vector Regression - SVR, ANN). In particular, ARIMA and VAR are used in the first stage for determining the number of inputs then SVR and MLP networks are applied to exchange rate forecasting in the second stage. Sharma, et al. [15] applied a hybrid of ANN and fuzzy logic in an Adaptive Neuro-Fuzzy Inference System (ANFIS) that can be implemented successfully with non-linear data prediction. Authors used Mean Absolute

Error (MAE), Mean Absolute Percentage Error (MAPE), and Root Mean Square Error (RMSE) to compare the proposed method with ANNs. Qiu, et al. [16] proposed a new set of input variables for ANNs to improve the effectiveness of prediction algorithms. They used some global search techniques, i.e. a genetic algorithm (GA) and simulated annealing (SA) to improve the prediction accuracy of ANN and to overcome the local convergence problem of BP.

When combining different forecasting models, diversity and accuracy of the overall model are the most important factors to be considered when selecting single ones. Diversity can be achieved by using different models or the same models with different configurations. Fonseca and Gomez [17] proposed a new method for multiple-step prediction, based on a Self-Organizing Map (SOM) neural network and meta-features. Authors used a pruning technique for automatic adjustment of the required balance between diversity and accuracy in selection of forecasters. The results showed that, on average, this method obtained better forecasting results than that obtained by other state-of-the-art methods.

Pulido, et al. [18] used a hybrid ensemble approach between the neural networks and fuzzy logic. Authors used particle swarm optimization (PSO) with type-1 and type-2 fuzzy integration of responses for time series prediction. The simulated results show that the hybrid ensemble approach produces a good prediction of the dollar time series. Dinh and Bui [19] proposed a multi-objective method of ensemble learning techniques based on the non-dominated sorting differential evolution (NSDE) for training neural networks and application in Foreign Exchange forecasting problems. In that paper, authors used an ensemble of ANNs with the same topology but different weights. These ANNs ' weights were optimized by using NSDE. All members (ANNs) have the same role; this means that the results were averaged from this ensemble. Zhang, et al. [20] develop novel methods to form neural network ensembles for FET problems. They show several methods of creating an ensemble. These include neural networks trained with different initial random weights. They use 50 identical architecture networks with different initial random weights. The output results generated from these 50 trained networks are combined in the ensemble. Another method is to allow networks with different architectures (15 neural network structures with five levels of input nodes and three levels of hidden nodes are trained with the same training data nodes), and networks trained with different data. The results show that the ensemble models created with different neural network structures perform better than the other ones. In [21], Landassuri-Moreno, et al. use the evolutionary algorithm EPNet to create ensembles of ANNs. In which, EPNet algorithm is used to evolve ANN architectures and weights at the same time. In order to calculate the ensemble results, the authors use two different linear combination methods. These are the Average and the Rank-Based Linear Combination (RBLC) method. The results show that overall, RBLC ensembles are better than average ones.

In study [22], Lean Yu, et al use a three-stage nonlinear radial basis function (RBF) neural network ensemble forecasting model for FET. In the first stage, the authors use several techniques to create multiple single RBF neural network predictors. In the second stage, a collection of RBF neural network is chosen from candidates generated in the previous stage. In the final stage, another RBF neural network is used to combine the selected ensemble members. The experimental results show that the proposed RBF neural network ensemble is more suitable for FET than the other ensemble models as well the single method of ANN.

In the field of classification, bagging and boosting are the most popular techniques. Recently, they have been extended to time series regression. Adaboost algorithm is the best-known and most widely applied the boosting algorithm in both research and practice [23]. Allende and Valle [24] introduced algorithms for generating individual components of ensembles and various procedures for combining these components. Besides that, Bagging, Adaboost and Negative Correlation as well as combination rules and decision templates were introduced. Zhang, et al. [25] used multiple classifier fusion methods to predict the non-performing loans (NPL) of business banks. This study applied bagging and AdaBoost algorithms with several strong base-classifiers (decision tree, k nearest neighbors and SVM). The results illustrate that multiple classifier fusion algorithms outperform single base classifiers.

Furthermore, the AdaBoost method performs much better than its bagging counterpart in processing NPL data of business banks.

In summary, combination of individual methods always performs better than the worst individual and sometimes can outperform the best individual model. Specifically, to ensure the success of a combination model, individual methods in this model must have both good diversity and performance. As mentioned above, individual performance and diversity degree of base classifiers are known to be key elements for ensemble learning. To the best of authors' knowledge, most of previous studies used various powerful methods as base classifiers to ensure diversity and performance of the model. There are few studies that ensure simultaneously both accuracy and diversity for the ensemble [26, 27].

3. RELATED CONCEPTS

3.1 Time series

a) Definition

A time series is a sequential set of data points, measured typically over successive times [28]. It is mathematically defined as a set of vectors *x(t), t=0,1,2...*Where *t* represents the time elapsed and *x(t)* is treated as a random variable.

A time series contains records of a single variable (univariate) or more than one variables(multivariate). A time series can be continuous (e.g. temperature readings, the flow of a river, etc.) or discrete (e.g. population of a particular city, exchange rates between two different currencies. Usually, in a discrete time series, the consecutive observations are recorded at equally spaced time intervals such as hourly, daily, weekly, monthly or yearly time separations.

b) Components of a Time Series

A time series, in general, is supposed to be affected by four main components: Trend, Cycle, Seasonality and Irregularity. The trend is a long term movement in a time series (the general tendency of a time series to increase, decrease or stagnate). Cyclical variation in a time series describes the medium-term changes in the series, caused by circumstances, which repeat in cycles. Seasonal variations in a time series are fluctuations within a year during the season. Irregularity or random variations in a time series are caused by unpredictable influences (such as war, strike, earthquake, flood...).

3.2 Back-Propagation Neural Networks

Neural network is considered to be a powerful tool for solving the nonlinear forecasting problems; particularly in case the relationship between processes is not easy to be explicitly established [29]. Especially, the model of back-propagation neural network is always used as a benchmark for FET [30].

During the process of training a neural network, it is necessary to determine two components: a network architecture and a set of linked weight values. The network architecture (i.e. the number of input, output, layers and neurons in hidden layers) is often predetermined depending on the expert knowledge. Whereas, the weight values are often trained by using the Back-Propagation (BP) algorithm.

3.3 NSGA-II algorithm

Non-dominated Sorting Genetic Algorithm II (NSGA-II) is proposed in Deb, et al. [31]. It is an improved version of NSGA algorithm [32]. The idea of this algorithm is showed in Table 1. There are two important procedures in this algorithm: Non-dominated sorting and crowding distance sorting. The first one ensures the convergence whereas the second one ensures the diversity of each individual in the population.

Table 1. Pseudo code of NSGA-II Algorithm

```
NSGA-II Algorithm
P_t: Selected Parents at generation t
Q_t: the offspring that are generated from P_t
Step 1: Initialize
Create randomly a population ($Q_0$)
Step 2: Crossover
Apply crossover rate for each individual in a $P_t$, and select two parents.
Two parents perform crossover and generate two offspring.
Two offspring will be placed in the offspring population $Q_t$
Step 3: Non-dominated sorting
Apply non-dominated sorting to ($P_t$+$Q_t$) population.
All non-dominated fronts of ($P_t$+$Q_t$) are copied to the parent population rank by rank.
Step 4: Crowding distance sorting
Stop adding the individuals in the rank when the size of parent population is larger than the population size (N)
Individuals in the last accepted rank, that make the parent population size larger than N are sorted by crowding distance sorting.
```

Table 2. Pseudo code of DE algorithm with "DE/rand/1/bin" variant

```
Differential Evolution Algorithm
Input:
    N is number of individuals in a population
    Max_Gen: is the number of generations
    D is the dimensionality of the problem
    CR and F are user-defined parameters

G=0
Create a random initial population $x_{i,G}$ ∀i, i = 1, . . . , N
Evaluate $f(x_{i,G})$ ∀i, i = 1, . . . , N
FOR G=1 to Max_Gen
    FOR i=1 to N Do
        Select randomly r1≠ r2 ≠ r3.
        jrand = randint(1, D) // an random integer number between 1 and D
        FOR j=1 to D Do
            IF (randj (0, 1) < CR or j = jrand) THEN
                $u_{i,j,G+1} = x_{r3,j,G} + F (x_{r1,j,G} - x_{r2,j,G})$
            ELSE
                $u_{i,j,G+1} = x_{i,j,G}$
            END IF
        END FOR
        IF ($f(u_{i,G+1}) \le f(x_{i,G})$) THEN
            $x_{i,G+1} = u_{i,G+1}$
        ELSE
            $x_{i,G+1} = x_{i,G}$
        END IF
    END FOR
    G = G + 1
END FOR
```

3.4 Non-dominated Sorting Differential Evolution (NSDE) algorithm

This approach was first introduced in the study of Iorio and Li [33]. Since then, it has been widely studied due to its efficiency and simplicity [19, 34]. NSDE is a variant of NSGA-II with some features from Differential evolution (DE). The difference between this approach and NSGA-II is in the method for generating new individuals. NSGA-II uses a real-coded crossover and mutation operator, but in NSDE, these operators were replaced with the conventional operators of DE, a direction-based approach. The results of NSDE outperformed those produced by NSGA-II.

The Differential Evolution Algorithm was proposed by Kenneth [35] to solve real-parameter optimization problems. DE uses a simple mutation operator based on differences between pairs of solutions (called vectors) with the aim of finding a search direction based on the distribution of solutions in the current population. There are some variants of the DE algorithm [33]: Variants with discrete recombination operator (DE/rand/1/bin); Variants with arithmetic recombination (DE/current-to-rand/1; DE/current-to-best/1) and Variants with combined arithmetic-discrete recombination (DE/current-to-rand/1/bin). The most popular variant is "DE/rand/1/bin". The corresponding algorithm of this variant is presented in Table 2. Note that in DE, "CR" and "F" are two important parameters. "CR" controls the influence of parents in generation of offspring. Higher values mean less influence. "F" scales the influence of pairs of solutions selected to calculate the mutation value.

3.5 Adaboost

AdaBoost (stand for "Adaptive Boosting") was first introduced by Freund and Schapire [37]. After that, there were several versions of this algorithm such as Adaboost.M1, Adaboost.M2 [38]. Adaboost combines multiple weak classifiers into a single strong one by assigning a weight to each training example. In order to do it, after finishing each training process, the weak classifiers are combined; and at the same time, the weights on the misclassified examples will be increased with a hope that other weak classifiers will perform better on them. The pseudo code of Adaboost is showed in Table 3.

Table 3. Pseudo code of Adaboost Algorithm (Freund and Schapire 1995)

```
Adaboost Algorithm
Input: Sequence of m examples $S_m$= $(x_1, y_1), (x_2, y_2), …, (x_m, y_m)$ where output y ϵ [1]
        Weak learning algorithm, denoted as Weaklearner
        Integer T specifying number of iteration (machines)
Initialize:
        Iteration t=1;
        Distribution $D_t(i)= 1/m$; i= [1, m];
        The ensemble F= Ø
        The Error rate $E_t=0$;
FOR t=1,2..., T
        1. Take a sample $R_t$ from $S_m$ using distribution $W_t$
        2. Build a classifier $f_t$ using $R_t$ as the training set
        3. Compute $\epsilon_t = \sum_{i:f_t \neq yi} D_t(i)$ and $\beta_t = \epsilon_t / (1- \epsilon_t)$
        4. Update the weight: $D_{t+1}(i) = normalize(D_t(i) * \beta^{lt(i)})$;
           with $lt(i) = \begin{cases} 1 & \text{if } ft(xi) = yi \\ -1 & otherwise \end{cases}$
END FOR
Output: The ensemble F= $\{f_1, f_2...f_T\}$ and A= $(\alpha_1, \alpha_2,…, \alpha_T)$
```

Based on success with the classification problem, AdaBoost has been extended to regression on time series data. Works in [39] introduced AdaBoost.M2 to boosting regression and called it AdaBoost.R. It solves regression problems by projecting the regression data into a classification dataset. Further, authors of [40] developed AdaBoost.R2 algorithm, which is a modification of AdaBoost.R. In [41], the Big Error Margin (BEM) was introduced based on an approach, called BEM, proposed in [42]. BEM method counts the number of correct and incorrect predictions by comparing prediction error with the preset threshold value. Authors in [43] developed a new boosting algorithm called AdaBoost.RT (R stands for Regression and T for Threshold). This algorithm uses the absolute relative error threshold to project training examples into two classes (poorly and well-predicted examples).

Adaboost.RT suffers a drawback that its performance is sensitive to the threshold. If this threshold is too low, then it is generally very difficult to get a sufficient number of correctly predicted examples. In order to estimate the value of the threshold efficiently, the work in [44] proposed a modified AdaBoost.RT (we call it as Adaboost.RT2) to overcome the limitation of original AdaBoost.RT by self-adaptively modifying the threshold value. By using this approach, users don't need to select the value of threshold by experiments anymore. The experiments showed this method can ensure the good performance of Adaboost.RT. In this study, we will use Adaboost.RT, Adaboost.RT2 and a modification of Adaboost.RT2 algorithm to create Adaboost.RT.FET1, Adaboost.RT.FET2 and Adaboost_FET, respectively, for FET problems.

4. METHODOLOGY

In previous work, we preliminarily investigated the usage of a multi-objective evolutionary method to define an ensemble for FET. Based on those results, we propose a new approach for forming an ensemble for FET incorporating the strength of Adaboost. For it, we introduce a new modification of Adaboost to fit FET, called Adaboost_FET. The motivation for our new proposal is that multi-objective evolutionary algorithm has a high capability to find simultaneously a set of weak learners, which satisfy accuracy criterion while being diverse as expected. This set of weak learners serves well as initial points for Adaboost algorithms. However, in order to hybridize with Adaboost, we need to redesign it, hence we have Adaboost_FET as our proposal.

4.1. A novel ensemble learning approach

The overall of the ensemble learning approach can be described in Fig. 1 with two main phases:
Phase 1: Using multi-objective evolutionary algorithms (NSGA-II or NSDE) to create weak learners and then the BP algorithm to improve their performances.
Phase 2: Using Adaboost to create a strong learner from the weak learners.

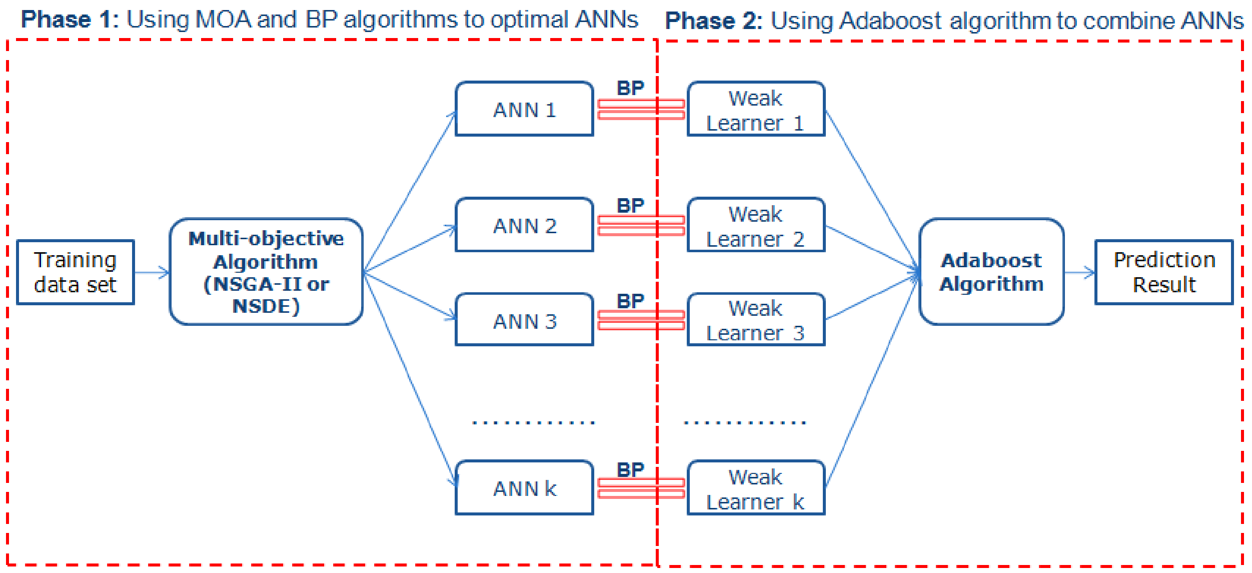

Phase 1: Using MOA and BP algorithms to optimal ANNs
Phase 2: Using Adaboost algorithm to combine ANNs
Training data set
Multi-objective Algorithm (NSGA-II or NSDE)
ANN 1
ANN 2
ANN 3
ANN k
BP
Weak Learner 1
Weak Learner 2
Weak Learner 3
Weak Learner k
Adaboost Algorithm
Prediction Result

Fig. 1. The generic architecture of ensemble learning approach. There are two phases, in the first phase, the MOAs are used with BP algorithm to create weak learners. In the second stage, these weak learners are inputs of Adaboost algorithm. The Adaboost will combine weak learners with different weights.

Note that ANN is selected as a base learner. It has been known that training ANN with BP might end up at local optima. This affects to performance of ANN. Meanwhile, evolutionary algorithms can fix this blemish. They are the population-based methods so they are able to find out the regions that contain global maxima points. That is why in Phase 1, we use evolutionary algorithms to optimize the ANNs first, then used BP to put them closer to the global extreme points. If all ANNs reach to the extreme points then the convergence (or performance, or accuracy) criterion will be met but the diversity will be decreased. To balance these criteria, in this study, we use diversity and performance as two objectives and then apply a multi-objective evolutionary algorithm (MOEA). The diversity objective not only prevents premature convergence but also ensure there is a certain difference between ANNs.

After Phase 1, we obtain a set of ANNs and they will serve as the weak learners of the Adaboost algorithm in Phase 2. These weak learners will be trained in the same dataset. For each iteration, the best learner will be selected and its impact coefficient to the final result will be increased. Distribution of training data is also recalculated after each iteration based on the best learner. After training process has finished, the prediction result will be calculated based on the weight values of each weak learner. Therefore, this ensemble learning approach promises to achieve better forecasting performance.

4.2. Using MOEA and BP algorithm to optimize ANNs (Phase 1)

4.2.1 Formulation of objective functions

a. Accuracy

In forecast, accuracy is the most important factor; it is to be measured performance of forecasters through error values. There are many error measures for time series prediction performance, but the more common ones are the Mean Squared Error (MSE) and the Root Mean Squared Error (RMSE). In this paper the authors choose MSE which is calculated according to the formula (1) as the first objective function.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(x_t - x_{t+1}\right)^2 \tag{1}$$

Where: n - the number of data value; x_t- real value; x_{t+1}- forecasting value.

b. Diversity

Diversity among ensemble members is the key to a successful classifier ensemble. In this research, we use three different measures to measure ensemble diversity (we call it as DIV), which are **Crowding distance (CD), Population distance (PD) and Sharing distance (SD).** Let N be the number of individuals in the population.

Definition 1: Crowding distance [28] of a particular individual in the population is the average distance of two points on either side of this point along each of the objectives (Fig.2).

$$CD_i = \frac{\left(MSE_{i+1} - MSE_{i-1}\right)}{\left(MSE_{\max} - MSE_{\min}\right)} \tag{2}$$

Where:

MSE_{i+1} and MSE_{i-1} corresponding, are the values of the first objective function of two individuals (i+1) and (i-1).

$MSE_{\max}$ and $MSE_{\min}$ corresponding, are the maximum and minimum values of the first objective function of the set of individuals on the same front layer with the considering i^{th} individual.

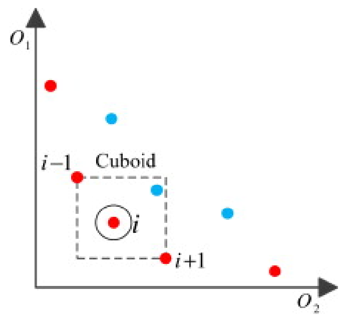


Fig. 2. The crowding distance of the i^{th} individual in its front (marked with red circles) is the average side length of the cuboid (shown with a dashed box).

Definition 2: Population distance of a particular individual in the population is the average total distance between this individual to the others in the population along each of the objectives.

$$PD_i = \frac{1}{(N-1)} \sum_{j=1, j \neq i}^{N} d(i, j) \tag{3}$$

Where

d (i, j)> 0 is the distance between the individual_i and the individual_j calculated as:

$$d(i,j) = |\, MSE_i - MSE_j| \tag{4}$$

Definition 3: Sharing distance of a particular individual in the population is the total sharing value (Sh) between this individual to the others in the population along both of the objectives.

$$SD_i = \sum_{j \in Pop} Sh\left[d\left[i,j\right]\right] \tag{5}$$

Sharing value is defined as: $Sh\left[d\right] = \begin{cases} 1 - d / \sigma share \ (d \leq \sigma share) \\ 0 \text{ otherwise} \end{cases}$ (6)

Where: $d\left[i,j\right]$ - the Euclidean distance between individuals i and individuals j;

σ_{share} - the radius of the neighborhood. $\sigma_{share} = Max\left|MSE_i - MSE_{i+1}\right|$ (7)

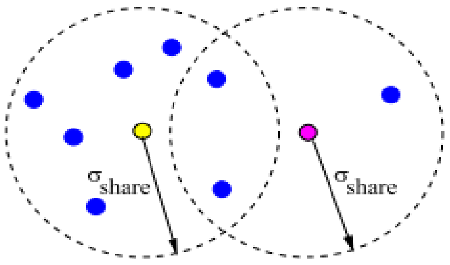


Fig. 3. The sharing distance with σ_{share} is the radius of the neighborhood.

In this study, we select σ_{share} as the maximum distance between two neighbors individuals in order to ensure there is always at least two individuals fall into a sharing distance.

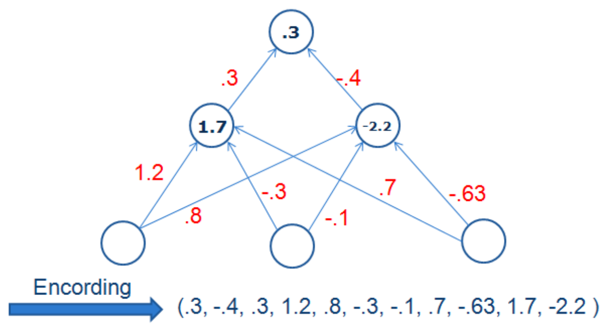


Fig. 4. Encording a Network on an Individual. Each individual is an array of the ANN's weight matrix.

4.2.2. The ANNs encodings

To optimize the performance of each ANNs instead of using randomly generated Weight matrix, we will use MOEA and BP to optimize them. In order to do that the weights (and biases) in the neural network are encoded as a list of real numbers (Fig. 4). In this study, the genes (nodes) are randomly initialized in the range (-1.5, +1.5).

4.2.3 Using MOEA and BP to create and optimize Weak learners

The problem of forecasting exchange rates can be formulated as the following bi-objective optimization problem:

$$\begin{cases} f_1 = MSE \\ f_2 = DIV \end{cases}$$
$$\min\{f_1\};\max\{f_2\} \tag{8}$$

Where DIV denotes any of the diversity measures proposed in Section 4.2.1.

In this work, we use two MOEA (NSGA-II and NSDE). Basically, the steps implemented in two algorithms are the same. The only difference is in using mutation and crossover operator to create the new individuals. Thus here we will only show the detail steps of NSDE algorithm.

There are two models used in this paper. The difference between these two models is the way an individual encoded. While in model 1 we set each individual is an ANN, in model 2 it is a group of ANNs. Each model will use different selection mechanism to create Weak learners for Adaboost Algorithm.

a. Model 1: Each individual is an ANNs

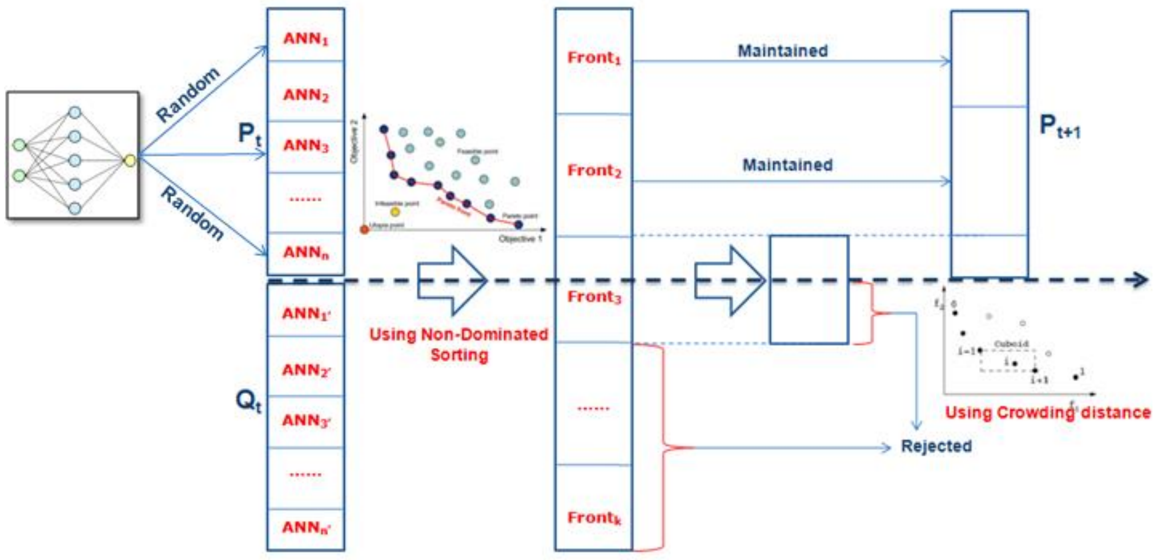


Fig. 5. Model1 with each individual is an ANNs. Firstly, these individuals are randomly initialized, after that, by using NSGA or NSDE algorithm, they will be optimized and selected as *weak learners* of adaboost algorithm.

We call the hybrid algorithm between NSDE and BP is EL.NSDE1. This algorithm has two main steps:
+ Step 1: Find out the region that contains the global extreme by using NSDE.
+ Step 2: Using BP to reach to extreme points.
The detail steps of this algorithm is presented in Table 4 and Fig. 5.

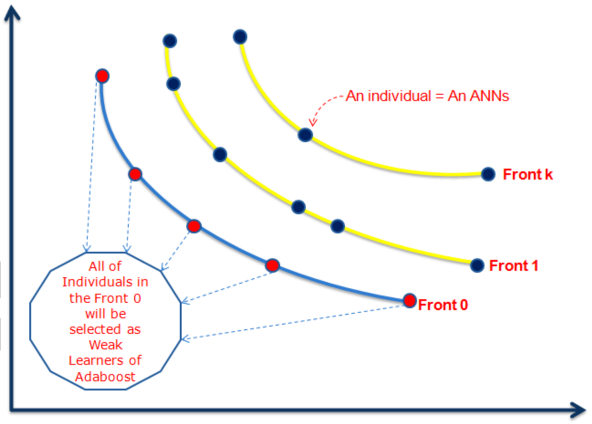

An individual = An ANNs
Front k
Front 1
Front 0
All of Individuals in the Front 0 will be selected as Weak Learners of Adaboost

Fig. 6. Selection mechanism in model 1. All of individuals in the first front of the NSGA-II (or NSDE) will be selected as weak learners of Adaboost.

Initially, a population of ANNs is randomly generated. These ANNs have the same configuration (input number, output number, layers count and the hidden neurons number). In this model, due to DIV values are calculated based on MSE values of all of the individuals in the population, therefore we have to calculate MSE for all individuals fist, after that we can determine DIV values.

Table 4. EL. NSDE1 Algorithm

EL.NSDE1 Algorithm

```
    Input:
        All parameters of ANNs, NSGA-II and DE.
        Training dataset.
1   % Step 1: Using NSDE to get best individuals
```

```
Create a random initial population $x_{i,G}$ $\forall$i, i = 1, . . . , N.
Calculate the first objective(MSE) for each $x_{i,G}$ $\forall$i, i = 1, . . . , N.
Calculate the second objective (DIV) for each $x_{i,G}$ $\forall$i, i = 1, . . . , N.
**DO**
    **FOR EACH** Individual$_i$ in population
        **FOR EACH** gene in anindividual$_j$
            Using **Mutate and Crossover of DE** to create new genes
        **END FOR**
        ***Calculate the first objective (MSE) for new Individual***
        Add new individual to population.
    **END FOR**
    ***Calculate the second objective (DIV) for all individuals in combined population***
    **% Using Non-dominated sorting and crowding distance method**
    Ranking base on 2 objectives and create Fronts
    Select N best individuals from the Fronts into the new population.
**WHILE**(Condition)

***% Step 2:  Refine using BP algorithm***
Get all individuals in Front (0)
Sort in descending order base on MSE value.
Select individual which has lowest MSE value (or best individual).
**FOR EACH** Individual$_i$ in Front (0)
    Using BP to train individual$_i$
**END FOR**

**Output:**  All individuals (ANNs) in Front(0)  (or first front)
```

In the first step, once new individual is obtained using DE operators, the new population is combined with the existing parent's population and then the best members of the combined population (parents plus offspring) are chosen based on the fast non-dominated sorting approach of NSGA-II. The non-dominated sorting mechanism ranks the individuals of the population in different levels (Front) and Individuals with lower rank are always preferred for selection. This means the first front (or Front$_0$) will contain the optimal individuals.

Note that, the mutation and crossover operators are performed node-by-node (it means each node at the same position of ANN will be performed together).

In the second step, all individuals in the Front$_0$ (Fig. 6) will be chosen to continue refining. We consider these individuals as the best individuals because they satisfy both criteria: having the small forecasting error and good diverse level.

Table 5. ELNSDE2 Algorithm

```
**ELNSDE2 Algorithm**
**Input**:
        All parameters of ANNs, NSGA-II and DE.
        Training dataset
***% Step 1: Using NSDE to get best individuals***
Create a random initial population $x_{i,G}$  $\forall$i, i = 1, . . . , N
Calculate 2 objectives: MSE and DIV for each $x_{i,G}$  $\forall$i, i = 1, . . . , N
**DO**
    **FOR EACH** Individual$_i$ in population
        **FOR EACH** *Sub-Individual$_j$* in *Individual$_i$*
            **FOR EACH** *gene* in a *sub-Individual$_j$*
                Using **Mutate and Crossover of DE** to create new genes
                END FOR
            END FOR
            Calculate 2 objectives: MSE and DIV for new Individual
            Add new individual to population.
        END FOR
% Using Non-dominated sorting and crowding distance method
        Ranking base on 2 objectives and create Fronts
        Select N best individuals from the Fronts into new population.
WHILE(Condition)

% Step 2:  Refine using BP algorithm
Get individuals in Front (0)
Sort in descending order base on DIV value.
Select individual which has maximum DIV value (or best individual).
FOR EACH $Sub\text{-}Individual_i$ in Best individual.
        Using BP to train $Sub\text{-}Individual_i$
END FOR

Output:  All sub-individuals ( ANNs) in the best Individual
```

b. Model 2: Each individual is a group of ANNs

We call the hybrid algorithm for this model is EL. NSDE2. This algorithm has two main steps same as EL. NSDE1 but having some minor changes. The detail steps of this algorithm is presented in Table 5 and Fig. 7. In this model, each individual is a group of ANNs (We call an ANNs as a sub-individual). Unlike Model 1, in this model, DIV values are made based on distances between sub-individuals in the same individuals. So that, whenever new individual is created, its objectives can be calculated immediately without waiting for calculation of other individuals.

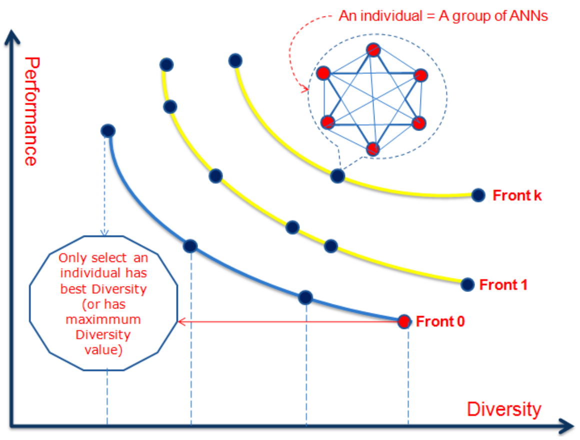

An individual = A group of ANNs
Performance
Front k
Front 1
Front 0
Only select an individual has best Diversity (or has maximmum Diversity value)
Diversity

Fig. 7. Selection mechanism in model 2. Due to each individual is a group of ANNs so only selecting an individual has best diversity. This group plays as weak learners of Adaboost algorithm.

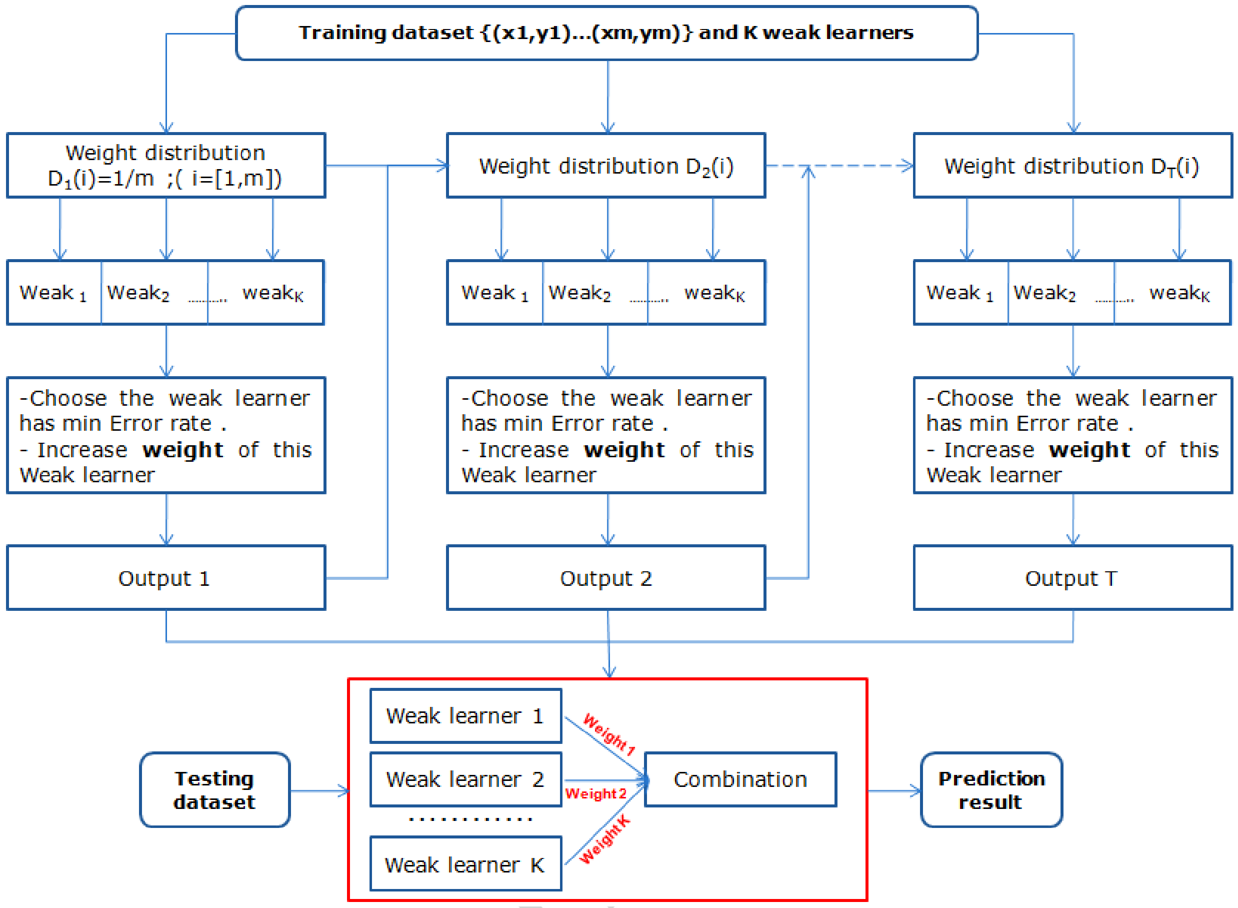

Training dataset {(x1,y1)...(xm,ym)} and K weak learners
Weight distribution D1(i)=1/m ;(i=[1,m])
Weight distribution D2(i)
Weight distribution DT(i)
Weak 1
Weak2
weakK
-Choose the weak learner has min Error rate .
- Increase weight of this Weak learner
Output 1
Output 2
Output T
Testing dataset
Weak learner 1
Weak learner 2
Weak learner K
Weight1
Weight2
WeightK
Combination
Prediction result

Fig. 8. The framework of Adaboost ensemble method for FET

Another major difference between the two models is the selection mechanism that good individuals will be selected to become weak learners. In this model, instead of choosing all individuals in the first front as Model 1, only one individual (i.e. a group of sub-individuals) is selected. This individual is the one having a maximum DIV value compared with others in the first front. The reason why we choose the individuals with the greatest DIV value instead of one with smallest MSE value is that after using NSDE and BP most of the individuals reached to the region containing the global extreme. This will make the diversity of individuals dropped (in the case of these individuals reached global extremes). Therefore, we choose the individual having maximum DIV value to preserve the diversity of sub-individuals in that individual.

4.2. An Adaboost ensemble method for FET (Phase 2)

Table 6. Adaboost ensemble method for FET

```
Adaboost_FET ( Adaboost ensemble method for FET)
Input: Sequence of m examples $S_m$= $(x_1, y_1), (x_2, y_2) ..., (x_m, y_m)$ where output $y \in R$
K Weak learners ($WeakLearner_1$, $WeakLearner_2$...$WeakLearner_k$)
        Integer T specifying number of iteration (machines)
Initialize:
        Iteration t=1;
        Distribution $D_t(i)$= 1/m; for all i= [1, m];
        The array Weights contains the weight of each WeakLearner (this value will increase if this WeakLearner
        has minimum Error)
FOR t=1,2...., T
```

```
    1. Loop all Weak Learners
    FOR p=1,2....,K
        1.1 Pass all training set through $weaklearner_p$ to obtain a prediction $y_i^p(x_i)$; i=1,2...m.
        1.2 Calculate a loss $L_i$ for each training sample; ($L_i$ may be any functional form as long as $L_i \in [0,1]$).
                $L_i = \frac{|y_i^p(xi)-yi|}{Max\,(|y_i^p(xi)-yi|)}$ (i=1,2..,m)
        1.3. Compute the error value of weak leaner_p: $Error_p = \sum_{i=1}^{m} L_i * D_t(i)$;
    END FOR
    2.Find $WeakLearner_{min}$ has minimum Error value ($Error_{min}$).
    3. If ($Error_{min}$> 0.5) Exits;
        4. Calculate $\beta = \frac{Error_{min}}{1-Error_{min}}$;
    5. Update Weights[min]+=log($\frac{1}{\beta}$);
        6. Update the Distribution: $D_{t+1}(i) = normalize(D_t(i) * \beta^{(1-L_{min})})$ (i=1,..m);
END FOR
Output: the prediction value of each training data i (i=1 …, m):
                $f_i = \frac{\sum_{t=1}^{K} weights[t]*y_i^t}{\sum_{t=1}^{K}(weights[t])}$
```

In this phase, we will use the Adaboost algorithm to create a strong learner from the weak ones that were optimally found at Phase 1. The Adaboost ensemble method is designed as depicted in Fig. 8 and the pseudo code is presented in Table 6**Error! Reference source not found.**. Basically, the idea as well as implementation of method's steps is based on the original version of AdaBoost. It firstly samples a training set from the initial dataset according to a uniform distribution. It means all initial weight distributions are given a value of 1/m (m is the number of training data). These weight distributions will be updated adaptively at each iteration. Depending on the forecasting results, the weight distribution on each example will be differently updated. The samples that are correctly forecasting by the weak learner will get lower weights (they are considered as the easy sample); on the other hand, the difficult samples will get higher weights. In the next iteration, these weight distributions will be used to sample the training dataset. Hopefully, there will be another weak learner, which can give better performance on difficult samples. The final result is made through a weighted combination of the weak learners' outputs. We call it as Adaboost_FET.

Compared with the original version, this approach has some differences. In particular, the original Adaboost calls only one weak learner at each iteration. In theory, a weak learner can be any one as long as it can give result better than random guessing [38]. After each iteration Adaboost will compose a weak classifier and its outputs will be used to produce the final prediction result. For our approach, K weak learners were preselected and at each iteration, we have to perform all K weak learners. We then select one has the best performance. The combination weight (*i.e. the coefficients used to calculate the output of the ensemble, the greater this coefficient is, the more influence it affects the overall results*) of the best one will be increased. This means when T iterations are processed, the ensemble still has K weak learners and each of them has different combination weights. Whereas in the original version, the ensemble, at this time, will have T weak classifiers (each of them gets from each iteration).

Table 7. Adaboost.RT.FET1 Algorithm

```
Adaboost.RT.FET1 Algorithm
Input: Sequence of m examples $S_m$= ($x_1$, $y_1$), ($x_2$, $y_2$) .... ($x_m$, $y_m$) where output $y \in R$
        Weak learning algorithm denoted as Weaklearner
        Integer T specifying number of iteration (machines)
        Threshold Ø (0< Ø<1) for demarcating correct and incorrect predictions
Initialize:
        Iteration t=1;
        Distribution $D_t$ (i)= 1/m; for all i= [1, m];
        The array Weights contains the weight of each WeakLearner
FOR t=1,2...., T
```

```
    1. Loop all Weak Learners
    FOR p=1,2....,K
        1.1 Pass all training set through $weaklearner_p$ to obtain a prediction $y_i^p(x_i)$; i=1,2...m.
        1.2 Calculate Absolute relative error for each training example as
                $ARE_i = \left| \frac{y_i^p(xi)-yi}{yi} \right|$ (i=1,2..,m)
        1.3. Compute the error value of $weak\ leaner_p$: $Error_p = \sum_{i:ARE(i)>\emptyset} D_t(i)$
    END FOR
    2.Find $WeakLearner_{min}$ has minimum Error value ($Error_{min}$).
    3. If ($Error_{min}$> 0.5) Exits;
    4. Calculate $\beta = \frac{Error_{min}}{1-Error_{min}}$;
    5. Update Weights[min]+=$\log(\frac{1}{\beta})$;
    6. Update the Distribution: $D_{t+1}(i) = normalize(D_t(i) * \beta^{(1-ARE_{min})})$ (i=1,..m);
END FOR
Output: the final hypothesis:
        $f_{fmin}(x) = \frac{\sum_{t=1}^{T}\{\left(\log\frac{1}{\beta t}\right)*ft(x)\}}{\sum_{t=1}^{T}\left(\log\frac{1}{\beta t}\right)}$
```

Adaboost_FET algorithm proposed in this study can be considered as a variant of Adaboost.R2. Besides, we also apply the Adaboost.RT algorithm [44] to FET, hence it creates two variants (named Adaboost.RT.FET1 and Adaboost.RT.FET2). The pseudo code of the Adaboost.RT.FET1 is presented in Table 7. The algorithm uses a threshold Ø to project the regression problem into the binary classification problem. The absolute relative error (ARE) is used to divide examples as well or poorly predicted. If ARE is greater than Ø then the predicted value is considered to be incorrect and otherwise. The pseudo code of Adaboost.RT.FET2 is presented in Table 8**.** In this algorithm, a self-adaptive mechanism is proposed to modify Ø. In other words, Ø is adjusted during the process. This method can overcome the limitation of Adaboost.RT.FET1. As indicated in[44], the value of Ø at the beginning of Adaboost.RT should be in (0, 0.4) and the authors choose 0.2 as the default initial value of Ø.

5. EMPIRICAL STUDIES

In this section, we report experimentswith these variants of AdaBoost algorithm (Adaboost_FET, Adaboost.RT.FET1, Adaboost.RT.FET2) to check the most suitable variant for FET problem. Based on the obtained results, we analyze the performance of our proposal against its counterparts.

Table 8. The Adaboost.RT.FET2 Algorithm

```
Adaboost.RT.FET2 Algorithm
Input: Sequence of m examples $S_m$= ($x_1$, $y_1$), ($x_2$, $y_2$), ..., ($x_m$, $y_m$) where output y ϵ R
        Weak learning algorithm, denoted as Weaklearner
        Integer T specifying number of iteration (machines)
        Threshold Ø (0< Ø<1) for demarcating correct and incorrect predictions
Initialize:
        Iteration t=1;
        $Ø_t$= 0.2 (0< Ø<0.4)
        Distribution $D_t$ (i)= 1/m; for all i= [1, m];
        The array Weights contains the weight of each WeakLearner
For t=1,2, ..., T
    1. Loop all Weak Learners
    For p=1,2....,K
        1.1 Pass all training set through $weaklearner_p$ to obtain a prediction $y_i^p(x_i)$; i=1,2...m.
        1.2 Calculate Absolute relative error for each training example as
          $ARE_i = \left| \frac{y_i^p(xi)-yi}{yi} \right|$ (i=1,2..,m)
        1.3. Compute the error value of weak leaner_p: Error_p= $\sum_{i:ARE(i)>Øt} D_t(i)$
    End For
    2.Find WeakLearner_min has minimum Error value (Error_min).
    3. If (Error_min> 0.5) Exits;
    4. Calculate the root mean square error (RMSE):
```

$$e_t = \sqrt{\frac{1}{n}\sum_{i=1}^{m}(y^{\min}{}_i - y_i)}$$

```
    5. Refine Ø
```

$$Ø_{t+1} = \begin{cases} Ø_t * (1-\lambda) & while \quad et < et-1 \\ Ø_t * (1+\lambda) & while \quad et > et-1 \end{cases}$$

```
     Where $\lambda$ is relative to the change rate of RMSE
```

$$\lambda = 0.5 * | \frac{e_t - e_{t-1}}{e_t} |$$

```
    6. Calculate β= $\frac{Error_{min}}{1-Error_{min}}$;
    7. Update Weights[min]+=log($\frac{1}{\beta}$);
    8. Update the Distribution: $D_{t+1}(i) = normalize(D_t(i) * \beta^{(1-ARE_{min})})$ (i=1,..m);
End For
Output: the final hypothesis:
```

$$f_{fmin}(x) = \frac{\sum_{t=1}^{T}\{\left(\log\frac{1}{\beta t}\right) * ft(x)\}}{\sum_{t=1}^{T}\left(\log\frac{1}{\beta t}\right)}$$

5.1. Data description

In these studies, we used 4 datasets (HKD, JPY, EURO and USD) and the exchange rates are converted into Australian dollars. These data are taken online on the website http://fx-rate.net/. In order to retrieve the data from this site, we use the query string:

*http://fx-rate.net/historical/?c_input={**0**}&cp_input={**1**}&date_to_input={**2**}&range_input={**3**}&csv=true*

In which, parameter {0} is the source currency that we want to convert, parameter {1} is destination currency; parameter {2} is the last day to retrieve data; parameter {3} is the amount of data to be taken.

In this research, we used 600 data during the period from 1st May 2015 to 20th December 2016. The time series of these exchange rates are presented in Fig. 9. There was no particular reason for selecting this period. We just followed the common trend of preparing data in this area of research. In order to evaluate the effectiveness of proposed algorithms, these data points were separated sequentially into two parts: training data (70%) and test data (30%).

5.2. Experimental design

a. Scenarios

This paper focuses on computational methods for FET. Therefore, there is a need to understand their behaviors and dynamics, not just the final prediction outcomes. For our proposal, we concern the following issues: (1) for Model 1, the solution for forming an ensemble is to select all individuals on the first front. The question is whether this solution is better than selecting only the best individual for forecasting? (2) How much impact does the diversity objective has on the final result? In which cases, it gives better results and vice versa?; (3) For three diversity measurements (CD, PD, SD), which one is most suitable for the ensemble learning model?; (4) How the proposed method perform under the effect of different MOEA (i.e. NSDE or NSGA-II)?; (5) for variants of DE, which one gives the best result when being used for NSDE?; (6) when applying ensemble learning to FET, how does the combination weight method behave in comparison to the averaged method?; (7) In Model 2, there is a parameter, called K, which is used to control the ensemble size. There is a need to analyze the effect of K to the overall performance; (8) It is a need to analyze the performance of Model 1 and 2 in comparison with each other? (9) for the variants of AdaBoost algorithm, which variant will give a better result? (10) How do ensemble learning models perform in comparing with single models?

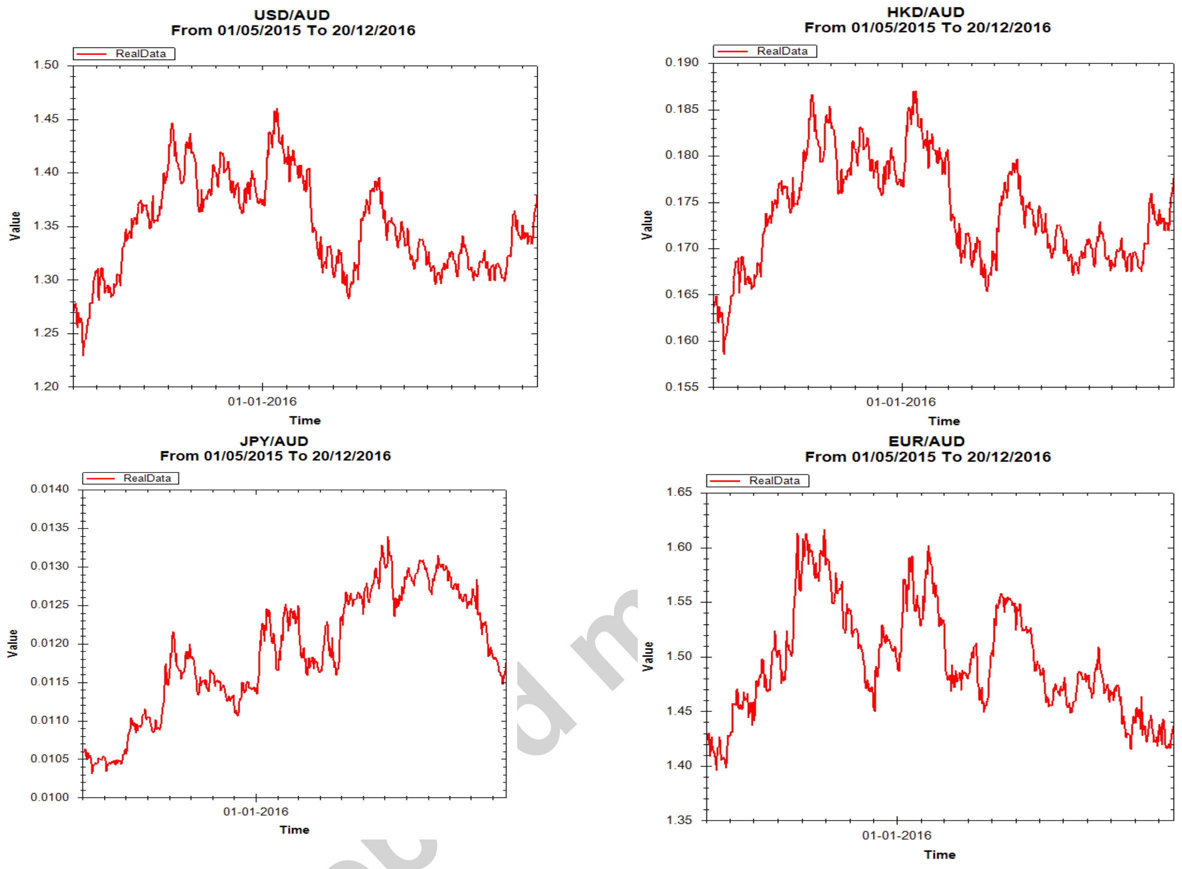


Fig. 9. The time series of 4 exchange rates data (USD/AUD, HKD/AUD, JPY/AUD. EUR/AUD).

From these questions we conducted the experiments as follows:

Firstly, a single model using ANN will be tested to get baseline results. After that, the experiment with a hybrid algorithm between NSGA-II and BP is performed. There are three variants of this hybrid algorithm including:

- NSGAII_K1_ENSEMBLE: In this variant, an individual presents an ANN, all individuals in the first front will be selected and the result is averaged from these individuals.
- NSGAII_K1_BEST: each individual is an ANN, only get the best individual in the first front and use it to get the result.
- NSGAII_ENSEMBLE: An individual presents a group of *K* ANNs. Only best individual in the first front is chosen and then *K* ANNs in this best individual will be used. The overall outcome is averaged from results obtained from these ANNs.

We do experiment with different K values, K= {3, 5, 7, 9, 11, 13, and 15} with NSGAII_ENSEMBLE and test NSGAII_K1_ENSEMBLE, NSGAII_K1_BEST with three diversity types {Crowding distance - CD; Population distance- PD; Sharing distance- SD}.

Secondly, we do the test with NSDE. Actually, we have three variants of this algorithm:

- NSDE-Rand1 *('Rand 1' short for DE/rand/1/bin)*
- NSDE-Current *('Current' short for DE/current-to-rand/1)*
- NSDE-Best *('Best' short for DE/current-to-best/1)*

All variants will be run with K= {3, 5, 7, 9, 11, 13, and 15}.

Thirdly, we conduct the experiments with the combination method between NSDE and AdaBoost algorithm. This Adaboost used here is our new proposal **Adaboost_FET.** We also have three variants of this algorithm same as NSDE case. We named them as

- Adaboost_Rand1
- Adaboost_Current
- Adaboost_Best

All variants will be run with K= {3, 5, 7, 9, 11, 13, and 15}.

Finally, we do the test to compare **Adaboost_FET** with the other versions of the Adaboost algorithm, which are

- Adaboost.RT.FET1
- Adaboost.RT.FET2

b. Parameter settings

Note that parameters of algorithms used in our studies are presented in the Table 9. The learning rate of ANNs was set as 0.03, with 5 input nodes, 10 hidden nodes and 1 output node. These parameters were determined through experiments. Here, we use Spice-MLP tool (http://download.cnet.com/Spice-MLP/3000-2054_4-75363004.html?tag=mncol;1) to choose ANNs parameters that are most suitable for FET data.

In NSGA-II, the mutation probability was selected depending on the number of genes in a chromosome as suggested in the related literature. In the DE algorithm, there are two important parameters: *upper* and *lower bound*. We conducted a lot of experiments and found that the bounds of [-1.5; 1.5] are the best range for FET problems.
In Adaboost, T is a number of iterations, this parameter is used for all Adaboost's variants, here we set it to 200. Ø is the threshold, which is used to project the regression problem into the binary classification problem. It is used in two variants (**Adaboost.RT.FET1 and Adaboost.RT.FET2**).

Table 9. Parameter setting of algorithms.

Method	Parameters	Value
ANN	Learning rate	0.03
	Number inputs	5
	Number hidden nodes	10
	Number outputs	1
	The iterations of BP	1000
	Alpha value	2
	Testing rate (%)	30
NSGA-II	Population size	50 (100)
	Crossover probability	0.9
	Mutation probability	1/GensNumber

	Generations	1000
NSDE	CR	0.9
	F	0.5
	Upper bound	1.5
	Lower bound	-1.5
Adaboost	T	200
	Ø	0.4

5.3. Results and analysis

In this research, we did total 54 experiments on 4 exchange rate datasets. For each experiment, we ran 20 times and got the mean values. We use 4 measures to evaluate the accuracy of forecasting. They are MSE (Mean Squared Error) Train, MSE test, MAE (Mean Absolute Error) [45] train and MAE test.

5.3.1 Results on JPY data

The experimental results on JPY data are listed in appendix Table 12-16 and Fig.10. Results from using training data are given in Table 13, and Table 14. Meanwhile, the ones from testing data are reported in Table 15 and Table 16. The comparison of diversity measures is given in Table 12.

It is observed from Table 12 that the base method of ANNs gives the worse results in all accuracy measures. This supports our claim of using multi-objectivity to create a new way of forecasting FET. Exploitation of multi-objectivity gives us more flexibility in choosing learners. Further, NSGAII_K1_Ensemble is better than NSGAII_K1_Best in all diversity measures. Once again, the usage of ensemble learning makes a huge boost for forecasting that will be further later in this section. For effects of diversity types on performance of NSGAII_K1_Ensemble, *Sharing distance* (SD) option outperforms all others. This indicates that sharing distance exhibits more local information, so that MOEAs have more chances to exploit local information for the evolutionary process.

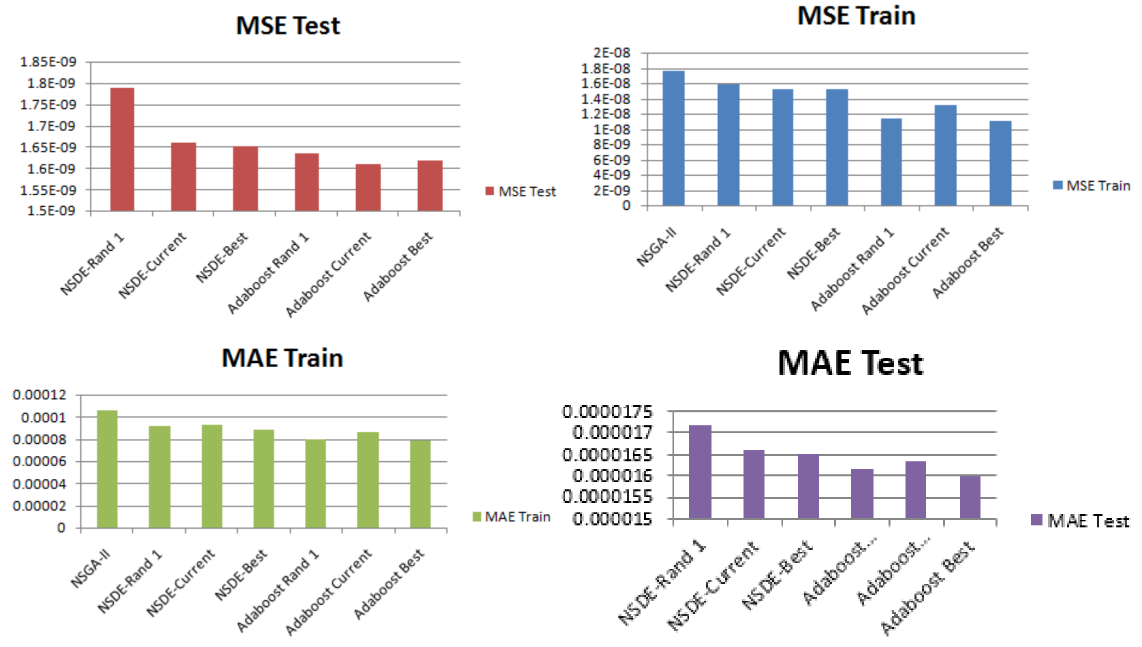


Fig. 10. Performance comparisons between three Adaboost's variants and the other four algorithms using MSE and MAE metrics on JPY data.

Regarding performance of ensemble methods, NSGAII_K1_Ensemble uses the simplest ensemble technique, in which weights of all individuals in the ensemble are identical. Our hypothesis is that we can get better results on FET with a more adaptive way of defining weights. That is why we propose to experiment on three variants of **Adaboost_FET** (including *Adaboost_Rand1; Adaboost_Current and Adaboost_Best)*.
As can be seen in Tables 13, 14, 15, 16, three variants of **Adaboost_FET** outperform all other algorithms in all accuracy measures. Comparingbetween them, Adaboost_Best is the best option (3 out of 4 accuracy measures). It is followed by Adaboost_Rand1 and Adaboost_Current. Especially, while Adaboost_Rand1 gives better results with train data, Adaboost_Current is better with testing data. Adaboost_Best always gives the stable good results (14 out of 28). NSDE-Best really helps in cooperating with Adaboost.Besides, all variants of Adaboost_FET give the best results with K=15 whereas the others give best results with K=3.Overall, with JPY data, Adaboost_FET is better than NSDE and NSGA-II is the worst.

5.3.2 Results on HKD data

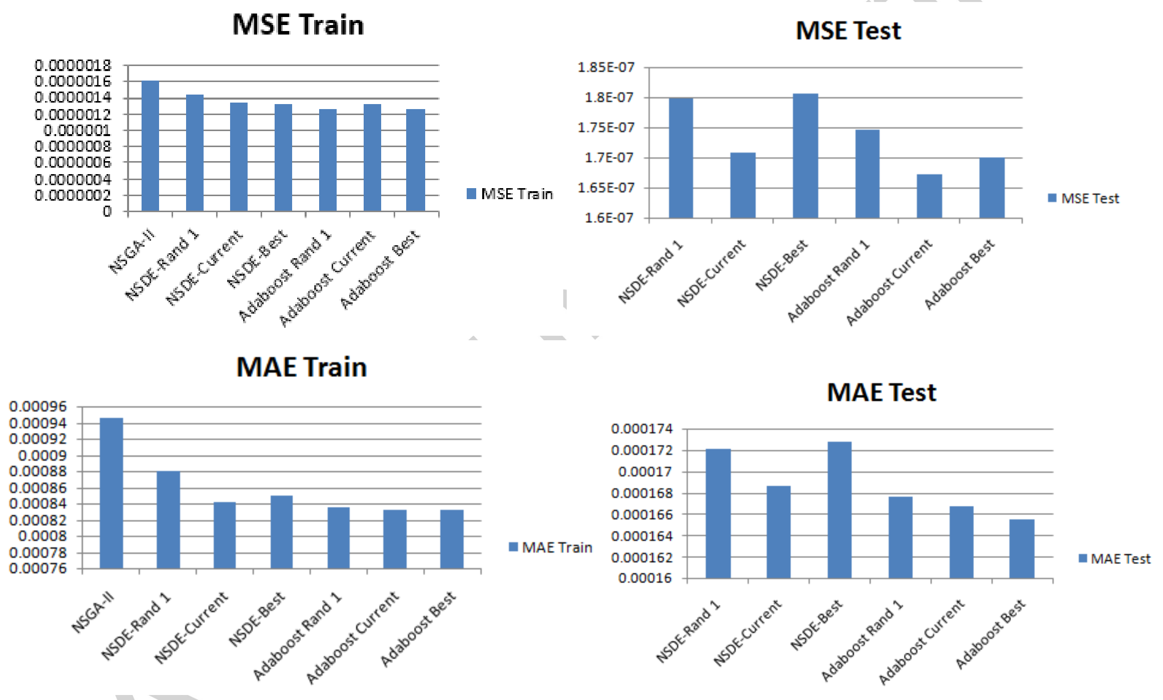


Fig. 11. Performance comparisons between three Adaboost's variants and the other four algorithms using MSE and MAE metrics on HKD data.

The experimental results on HKD data are listed in appendix Table 17-21 and Fig.11. Similar to the case of JPY data, trom these results, we found that Adaboost_FET (with 3 variants Adaboost_Rand1; Adaboost_Current and Adaboost_Best) shows better metric values than others on all comparisons. Adaboost_Best continues is the best one. This observation demonstrates that using K=13 and K=15 gives exceptionally good results in training dataset. Once again it supports our finding in the case of JPY data.

5.3.3 Results on USD data

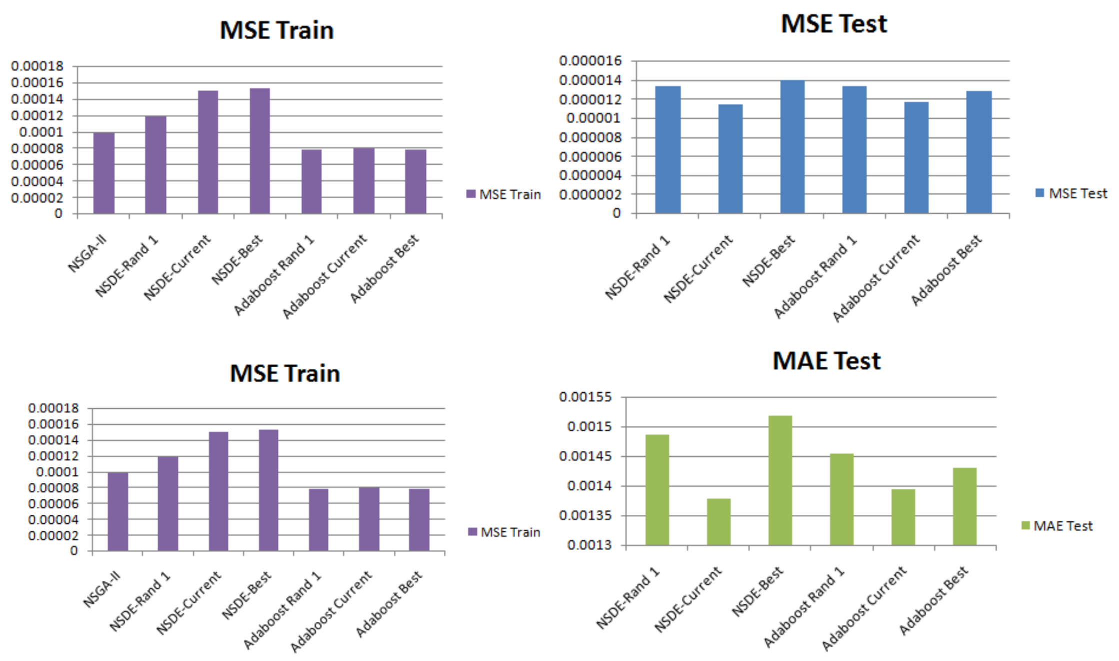


Fig. 12. Performance comparisons between three Adaboost's variants and the other four algorithms using MSE and MAE metrics on USD data.

The experimental results on USD data are summarized in appendix Table 22-26. From the plots shown in Fig.12 we find that while NSDE-Current variant gives the best result on the test dataset. Adaboost-Best gives the best results on the training dataset. From the results of NSDE-Current, we find that this DE variant is very suitable with USD dataset. Overall, Adaboost Best still gives the most stable results. Besides, Adaboost Best often gets better results with great value K (11, 13, 15) meanwhile NSDE get better results with smaller K (3,5).

5.3.4 Results on EUR data

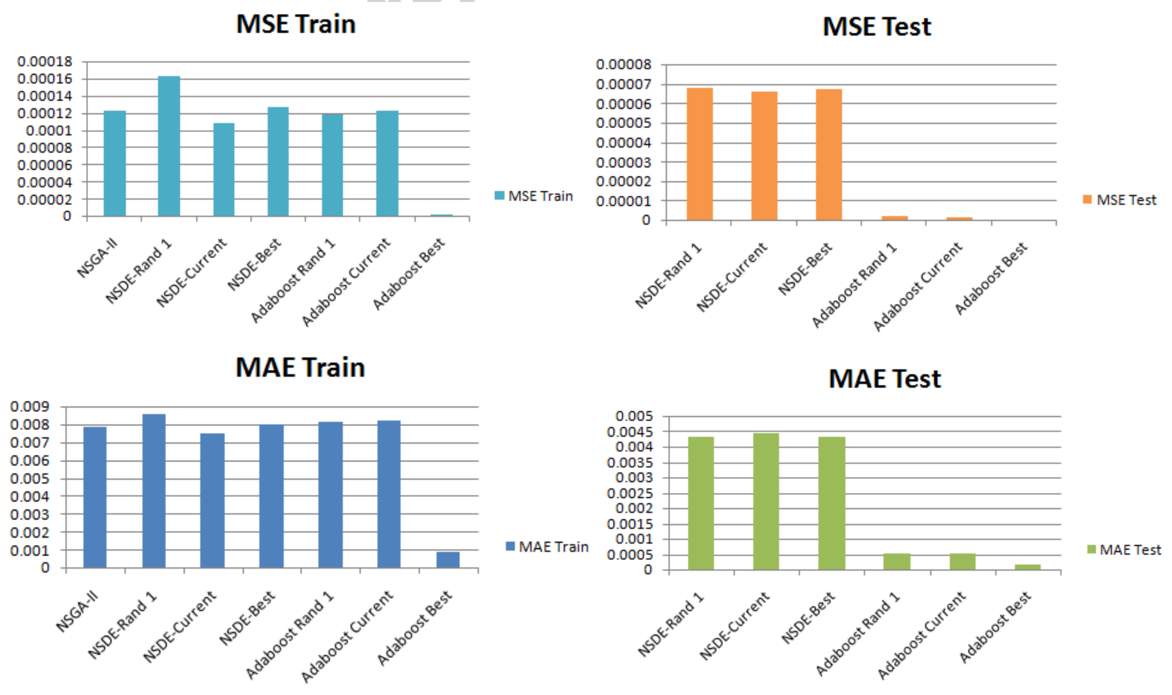


Fig. 13. Performance comparisons between three Adaboost's variants and the other four algorithms using MSE and MAE metrics on EUR data.

Overall the trend in the case of EUR is still similar to that of the previous ones. However, Adaboost_FET's variants clearly show their strength of generalization for EUR data. From Fig.13 and appendix Table 27-31, we can clearly see the superiority of the algorithm Adaboost_FET especially on the testing dataset. From the values in appendix Table 27-31, Adaboost-Best creates a surge in the results. This algorithm is really suitable for this type of FET.

5.3.5 Performance analysis of Adaboost variants

Table 10. Experimental result on the variants of Adaboost. The best metric value in each column is highlighted in bold with gray background. The second one is also highlighted with light gray background. It is clear that the proposed algorithm (Adaboost_FET) outperforms the others in almost all test instances and the Adaboost.RT.FET2 is better than Adaboost.RT.FET1 in 11 out of 16 comparisons.

Methods	**EUR**				**HKD**			
	MSE		**MAE**		**MSE**		**MAE**	
	Train	Test	Train	Test	Train	Test	Train	Test
Adaboost_FET	1.27E-06	1.70E-07	8.32E-04	1.65E-04	1.27E-06	1.70E-07	8.32E-04	1.65E-04
Adaboost.RT.FET1	1.33E-04	2.10E-06	8.78E-03	5.66E-04	1.32E-06	2.51E-07	8.56E-04	2.34E-04
Adaboost.RT.FET2	1.30E-04	2.33E-06	8.62E-03	5.58E-04	1.33E-06	2.50E-07	8.61E-04	2.35E-04
Methods	**USD**				**JPY**			
	MSE		**MAE**		**MSE**		**MAE**	
	Train	Test	Train	Test	Train	Test	Train	Test
Adaboost_FET	7.78E-05	1.27E-05	6.47E-03	1.43E-03	1.11E-08	1.62E-09	7.87E-05	1.60E-05
Adaboost.RT.FET1	8.18E-05	1.68E-05	6.70E-03	1.92E-03	1.16E-08	1.56E-09	8.11E-05	1.58E-05
Adaboost.RT.FET2	8.17E-05	1.65E-05	6.68E-03	1.90E-03	1.15E-08	1.57E-09	8.07E-05	1.59E-05

In previous section, we have shown that our proposal Adaboost_FET works well on FET problem. In order to verify its performance, there is a need to analyze its performance against other Adaboost variants. That is why we designed two variants of Adaboost.RT for this purpose. Table 10 shows the experimental results of thee Adaboost's variants on 4 datasets. The best mean metric value is highlighted in bold. From this Table, we can find that the Adaboost_FET is the best variant compared to the others in most cases (15 out of 16). The Adaboost.RT.FET2 is the modification of the Adaboost.RT.FET1 so that its result is better than the original version; it outperforms 11 out of 16 comparisons. From the experimental results, we selected Adaboost_FET as the main version of our Ensemble framework.

5.3.6. Comparison to similar studies

As indicated early in the paper, there have been several works using neural-based ensembles for FET. To evaluate the capability of our algorithm, we made a comparison with the method proposed by Yu [22] (RBF-based ensemble). The motivation for our comparison is that Yu's work has been the most recent and related to our proposal and it was published in a well-known journal in neural computing.

In order to make a fair comparison, we prepared experiments on the same datasets that they did in [22]. The foreign exchange data are monthly obtained from Pacific Exchange Rates Services (http://fx.sauder.ubc.ca/). Yu [22] use four currencies British pounds (GBP), euros (EUR), German marks (DEM) and Japanese yen (JPY). These currencies are taken from January 1971 to December 2006. In which, 360 observations (from 1971 to 2000) are used for training and 71 remaining observations (from 2001 to 2006) are used for testing. The normalized mean squared error (NMSE) indicator is used for comparison.

Table 11. The NMSE comparison with different forecasting models for different currency rates. In this Table, 6 first models are introduced and compared by Yu [22]. The best metric value in each column is highlighted in bold. It is clear that the proposed algorithm (Adaboost_FET) outperforms the others in all test instances.

Models	**GBP**		**EUR**		**DEM**		**JPY**	
	NMSE	**Rank**	**NMSE**	**Rank**	**NMSE**	**Rank**	**NMSE**	**Rank**
Single RBF model	0.0614	5	0.0862	6	0.0895	7	0.0942	6
Simple averaging	0.0686	6	0.0956	7	0.0812	6	0.0978	7
Simple MSE	0.0789	7	0.0724	5	0.0733	5	0.0825	5
Stacked regression	0.0484	4	0.0668	4	0.0598	3	0.0767	4
Variance-based model	0.0467	3	0.0545	3	0.0654	4	0.0596	3
RBF-based ensemble	0.0388	2	0.0451	2	0.0462	2	0.0511	2
Adaboost_FET	**0.000437**	**1**	**0.00104**	**1**	**0.00052**	**1**	**0.00048**	**1**

Several single base methods were also selected to make the ground truth for ensemble learning methods. The results obtained are showed in Table 11. Generally speaking, our proposed Adaboost_FET performs the best in all the cases. Focusing on GBP, DEM and JPY testing case, Adaboost_FET outperforms RBF-based ensemble (nearly 100 times) and better more than 10 times with EUR testing case. This indicates that the proposed Adaboost_FET are more suitable for FET problem than the other ensemble models and single ANN model.

6. CONCLUSION

In this research, we have developed a novel ensemble learning approach based on the multi-objective evolutionary algorithms for forecasting currency exchange rates. We used MOEAs to create weak learners having both of diversity and accuracy. We proposed to design a number of diversity-preservation mechanisms and hybrid algorithms with different variants. The results suggested that although accuracy is the most important factor but the result will be worse if only select one has the best accuracy instead of choosing an ensemble. Experimental results showed that the proposed framework Adaboost_FET outperformed the other methods (including ensemble methods without using Adaboost and single methods). Hence, it can greatly improve the prediction accuracy and prediction stability for FET problem.

For future works, we intend to do further analysis on the performance of our approach related to the problem domain. In such way, we will have more understandings on the way to apply this fundamental research to practice.

Acknowledgments

This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.01-2015.12.

APPENDIX TABLES

Table 12. Comparison results between NSGAII_K1_Ensemble, NSGAII_K1_Best and ANN on JPY/AUD data. The best mean metric value in each column is highlighted in bold with gray background. It is clear that NSGAII_K1_Ensemble using sharing distance gives the best results in all metrics.

K	Method	Diversity	MSE Train	MSE Test	MAE Train	MAE Test
1	**NSGAII_K1_Ensemble**	CD	1.53E-08	1.31E-08	9.84E-05	8.57E-05
		PD	1.43E-08	1.24E-08	9.38E-05	8.20E-05
		SD	1.39E-08	1.21E-08	9.23E-05	8.10E-05
	NSGAII_K1_Best	CD	1.70E-08	1.49E-08	1.04E-04	9.32E-05
		PD	1.77E-08	1.55E-08	1.06E-04	9.57E-05
		SD	1.79E-08	1.55E-08	1.06E-04	9.59E-05
	ANN		1.86E-08	1.59E-08	1.09E-04	9.74E-05

Table 13. Comparison results between NSGA-II, NSDE (with 3 variants) and Adaboost (with 3 variants of NSDE) using MSE TRAIN metric on JPY/AUD data. The best mean metric value in each row (corresponding to each different case of the individual group size.) is highlighted in bold with gray background. In which, Adaboost Rand 1 and Adaboost_ Best are better than the others. Adaboost _Best has the best mean metric value.

K	NSGA-II	NSDE-Rand 1	NSDE-Current	NSDE-Best	Adaboost Rand 1	Adaboost Current	Adaboost Best
3	1.70E-08	1.17E-08	1.44E-08	2.71E-08	1.13E-08	1.36E-08	1.11E-08
5	1.81E-08	2.58E-08	1.51E-08	1.18E-08	1.35E-08	1.42E-08	1.10E-08
7	1.74E-08	1.57E-08	1.52E-08	1.15E-08	1.09E-08	1.32E-08	1.15E-08
9	1.79E-08	1.18E-08	1.50E-08	1.18E-08	1.08E-08	1.30E-08	1.15E-08
11	1.74E-08	1.46E-08	1.51E-08	1.34E-08	1.18E-08	1.25E-08	1.07E-08
13	1.75E-08	1.69E-08	1.69E-08	1.56E-08	1.11E-08	1.28E-08	1.13E-08
15	1.78E-08	1.49E-08	1.50E-08	1.50E-08	1.04E-08	1.24E-08	1.05E-08
Mean	**1.76E-08**	**1.59E-08**	**1.53E-08**	**1.52E-08**	**1.14E-08**	**1.31E-08**	**1.11E-08**

Table 14. Comparison results between NSGA-II, NSDE (with 3 variants) and Adaboost (with 3 variants of NSDE) using MSE TEST metric on JPY/AUD data. The best mean metric value in each row (corresponding to each different case of the individual group size.) is highlighted in bold with gray background. In general, Adaboost are better than NSDE and NSGA-II. Adaboost _Current has the best mean metric value.

K	NSGA-II	NSDE-Rand 1	NSDE-Current	NSDE-Best	Adaboost Rand 1	Adaboost Current	Adaboost Best
3	6.15E-07	1.77E-09	1.66E-09	1.44E-09	1.63E-09	1.67E-09	1.58E-09
5	4.54E-07	1.86E-09	1.65E-09	1.77E-09	1.62E-09	1.57E-09	1.43E-09
7	7.27E-07	1.79E-09	1.64E-09	1.60E-09	1.78E-09	1.63E-09	1.66E-09

9	6.76E-07	1.66E-09	1.62E-09	1.64E-09	1.79E-09	1.66E-09	1.67E-09
11	6.56E-07	1.83E-09	1.62E-09	1.62E-09	1.60E-09	1.59E-09	1.60E-09
13	5.57E-07	1.78E-09	1.78E-09	1.72E-09	1.63E-09	1.58E-09	1.47E-09
15	5.56E-07	1.83E-09	1.64E-09	1.76E-09	1.41E-09	1.55E-09	1.90E-09
Mean	**6.06E-07**	**1.79E-09**	**1.66E-09**	**1.65E-09**	**1.64E-09**	**1.61E-09**	**1.62E-09**

Table 15. Comparison results between NSGA-II, NSDE (with 3 variants) and Adaboost (with 3 variants of NSDE) using MAE TRAIN metric on JPY/AUD data. The best mean metric value in each row (corresponding to each different case of the individual group size.) is highlighted in bold with gray background. In which, Adaboost Rand 1 and Adaboost_ Best are better than the others. Adaboost _Best has the best mean metric value.

K	**NSGA-II**	**NSDE-Rand 1**	**NSDE-Current**	**NSDE-Best**	**Adaboost Rand 1**	**Adaboost Current**	**Adaboost Best**
3	1.04E-04	8.13E-05	9.02E-05	1.01E-04	7.95E-05	8.76E-05	7.89E-05
5	1.07E-04	1.07E-04	9.26E-05	8.18E-05	8.69E-05	8.98E-05	7.87E-05
7	1.05E-04	8.99E-05	9.29E-05	8.07E-05	7.85E-05	8.61E-05	8.00E-05
9	1.07E-04	8.19E-05	9.21E-05	8.19E-05	7.77E-05	8.56E-05	8.02E-05
11	1.06E-04	8.98E-05	9.23E-05	8.57E-05	8.15E-05	8.37E-05	7.72E-05
13	1.06E-04	9.78E-05	9.78E-05	9.19E-05	7.88E-05	8.46E-05	7.96E-05
15	1.07E-04	9.04E-05	9.22E-05	9.21E-05	7.56E-05	8.32E-05	7.64E-05
Mean	**1.06E-04**	**9.11E-05**	**9.28E-05**	**8.79E-05**	**7.98E-05**	**8.58E-05**	**7.87E-05**

Table 16. Comparison results between NSGA-II, NSDE (with 3 variants) and Adaboost (with 3 variants of NSDE) using MAE TEST metric on JPY/AUD data. The best mean metric value in each row (corresponding to each different case of the individual group size.) is highlighted in bold with gray background. In which, Adaboost_ Best are better than the others in most comparisons (5/7 test cases).

K	**NSGA-II**	**NSDE-Rand 1**	**NSDE-Current**	**NSDE-Best**	**Adaboost Rand 1**	**Adaboost Current**	**Adaboost Best**
3	6.35E-04	1.70E-05	1.66E-05	1.53E-05	1.60E-05	1.66E-05	1.58E-05
5	5.48E-04	1.73E-05	1.66E-05	1.71E-05	1.64E-05	1.61E-05	1.49E-05
7	7.15E-04	1.72E-05	1.65E-05	1.63E-05	1.68E-05	1.64E-05	1.63E-05
9	6.65E-04	1.66E-05	1.64E-05	1.65E-05	1.71E-05	1.66E-05	1.64E-05
11	7.31E-04	1.74E-05	1.64E-05	1.64E-05	1.59E-05	1.62E-05	1.57E-05
13	5.86E-04	1.72E-05	1.72E-05	1.69E-05	1.61E-05	1.62E-05	1.55E-05
15	5.94E-	1.74E-05	1.65E-05	1.71E-05	1.47E-05	1.61E-05	1.74E-05

	04						
Mean	**6.39E-04**	**1.72E-05**	**1.66E-05**	**1.65E-05**	**1.61E-05**	**1.63E-05**	**1.60E-05**

Table 17. Comparison results between NSGAII_K1_Ensemble, NSGAII_K1_Best and ANN on HKD/AUD data. The best mean metric value in each column is highlighted in bold with gray background. It is clear that NSGAII_K1_Ensemble using sharing distance gives the best results in all metrics.

Method	Diversity	**MSE Train**	**MSE Test**	**MAE Train**	**MAE Test**
NSGAII_K1_Ensemble	CD	1.56E-06	6.80E-07	9.29E-04	6.22E-04
	PD	1.54E-06	6.74E-07	9.24E-04	6.20E-04
	SD	1.48E-06	6.70E-07	9.02E-04	6.20E-04
NSGAII_K1_Best	CD	1.64E-06	6.96E-07	9.56E-04	6.27E-04
	PD	1.61E-06	6.98E-07	9.45E-04	6.31E-04
	SD	1.60E-06	6.94E-07	9.43E-04	6.28E-04
ANN		1.67E-06	7.01E-07	9.70E-04	6.31E-04

Table 18. Comparison results between NSGA-II, NSDE (with 3 variants) and Adaboost (with 3 variants of NSDE) using MSE TRAIN metric on HKD/AUD data. The best mean metric value in each row (corresponding to each different case of the individual group size.) is highlighted in bold with gray background. In which, Adaboost Rand 1 and Adaboost_ Best are better than the others.

K	**NSGA-II**	**NSDE-Rand 1**	**NSDE-Current**	**NSDE-Best**	**Adaboost Rand 1**	**Adaboost Current**	**Adaboost Best**
3	1.62E-06	1.30E-06	1.34E-06	1.30E-06	1.33E-06	1.34E-06	1.30E-06
5	1.61E-06	1.70E-06	1.34E-06	1.29E-06	1.28E-06	1.31E-06	1.28E-06
7	1.60E-06	1.48E-06	1.35E-06	1.29E-06	1.27E-06	1.32E-06	1.26E-06
9	1.61E-06	1.41E-06	1.35E-06	1.50E-06	1.26E-06	1.32E-06	1.26E-06
11	1.60E-06	1.37E-06	1.34E-06	1.29E-06	1.27E-06	1.31E-06	1.26E-06
13	1.60E-06	1.35E-06	1.34E-06	1.29E-06	1.24E-06	1.30E-06	1.25E-06
15	1.62E-06	1.42E-06	1.34E-06	1.34E-06	1.24E-06	1.30E-06	1.25E-06
Mean	**1.61E-06**	**1.43E-06**	**1.34E-06**	**1.33E-06**	**1.27E-06**	**1.31E-06**	**1.27E-06**

Table 19. Comparison results between NSGA-II, NSDE (with 3 variants) and Adaboost (with 3 variants of NSDE) using MSE TEST metric on HKD/AUD data. The best mean metric value in each row (corresponding to each different case of the individual group size.) is highlighted in bold with gray background. In general, Adaboosts are better than NSDEs and NSGA-II. Adaboost _Current has the best mean metric value.

K	**NSGA-II**	**NSDE-Rand 1**	**NSDE-Current**	**NSDE-Best**	**Adaboost Rand 1**	**Adaboost Current**	**Adaboost Best**
3	2.19E-05	1.71E-07	1.79E-07	1.98E-07	1.92E-07	1.65E-07	1.96E-07

5	1.06E-05	1.79E-07	1.73E-07	1.78E-07	1.86E-07	1.62E-07	1.85E-07
7	1.36E-05	1.82E-07	1.68E-07	1.71E-07	1.88E-07	1.65E-07	1.48E-07
9	1.26E-05	1.86E-07	1.69E-07	1.81E-07	1.71E-07	1.65E-07	1.58E-07
11	1.88E-05	1.82E-07	1.69E-07	1.79E-07	1.49E-07	1.69E-07	1.56E-07
13	1.66E-05	1.77E-07	1.70E-07	1.77E-07	1.88E-07	1.66E-07	1.80E-07
15	9.53E-06	1.81E-07	1.68E-07	1.79E-07	1.49E-07	1.79E-07	1.66E-07
Mean	**1.48E-05**	**1.80E-07**	**1.71E-07**	**1.81E-07**	**1.75E-07**	**1.67E-07**	**1.70E-07**

Table 20. Comparison results between NSGA-II, NSDE (with 3 variants) and Adaboost (with 3 variants of NSDE) using MAE TRAIN metric on HKD/AUD data. The best mean metric value in each row (corresponding to each different case of the individual group size.) is highlighted in bold with gray background. In general, Adaboosts are better than NSDEs and NSGA-II. Adaboost _Current and Adaboost_Best have the best mean metric value.

K	**NSGA-II**	**NSDE-Rand 1**	**NSDE-Current**	**NSDE-Best**	**Adaboost Rand 1**	**Adaboost Current**	**Adaboost Best**
3	9.48E-04	8.41E-04	8.41E-04	8.42E-04	8.53E-04	8.42E-04	8.38E-04
5	9.46E-04	9.33E-04	8.40E-04	8.38E-04	8.35E-04	8.33E-04	8.38E-04
7	9.43E-04	8.93E-04	8.43E-04	8.40E-04	8.37E-04	8.34E-04	8.32E-04
9	9.45E-04	8.77E-04	8.42E-04	8.93E-04	8.33E-04	8.32E-04	8.32E-04
11	9.44E-04	8.65E-04	8.42E-04	8.36E-04	8.36E-04	8.31E-04	8.28E-04
13	9.44E-04	8.62E-04	8.41E-04	8.37E-04	8.23E-04	8.27E-04	8.27E-04
15	9.49E-04	8.89E-04	8.41E-04	8.56E-04	8.24E-04	8.26E-04	8.28E-04
Mean	**9.46E-04**	**8.80E-04**	**8.41E-04**	**8.49E-04**	**8.34E-04**	**8.32E-04**	**8.32E-04**

Table 21. Comparison results between NSGA-II, NSDE (with 3 variants) and Adaboost (with 3 variants of NSDE) using MAE TEST metric on HKD/AUD data. The best mean metric value in each row (corresponding to each different case of the individual group size.) is highlighted in bold with gray background. In general, Adaboosts are better than NSDEs and NSGA-II. Adaboost_Best has the best mean metric value.

K	**NSGA-II**	**NSDE-Rand 1**	**NSDE-Current**	**NSDE-Best**	**Adaboost Rand 1**	**Adaboost Current**	**Adaboost Best**
3	3.86E-03	1.67E-04	1.72E-04	1.80E-04	1.75E-04	1.65E-04	1.78E-04
5	2.53E-03	1.71E-04	1.70E-04	1.72E-04	1.75E-04	1.64E-04	1.72E-04
7	2.85E-03	1.73E-04	1.67E-04	1.69E-04	1.74E-04	1.65E-04	1.55E-04

9	2.76E-03	1.75E-04	1.68E-04	1.73E-04	1.66E-04	1.66E-04	1.59E-04
11	3.60E-03	1.74E-04	1.68E-04	1.73E-04	1.56E-04	1.68E-04	1.59E-04
13	3.41E-03	1.71E-04	1.68E-04	1.71E-04	1.74E-04	1.66E-04	1.70E-04
15	2.50E-03	1.73E-04	1.67E-04	1.73E-04	1.55E-04	1.72E-04	1.65E-04
Mean	**3.07E-03**	**1.72E-04**	**1.69E-04**	**1.73E-04**	**1.68E-04**	**1.67E-04**	**1.65E-04**

Table 22. Comparison results between NSGAII_K1_Ensemble, NSGAII_K1_Best and ANN on USD/AUD data. The best mean metric value in each column is highlighted in bold with gray background. It is clear that NSGAII_K1_Ensemble using sharing distance gives the best results in all metrics.

K	**Method**	Diversity	**MSE Train**	**MSE Test**	**MAE Train**	**MAE Test**
1	**NSGAII_K1_Ensemble**	CD	9.49E-05	4.16E-05	7.20E-03	4.86E-03
		PD	9.23E-05	4.06E-05	7.12E-03	4.82E-03
		SD	9.27E-05	4.10E-05	7.12E-03	4.83E-03
	NSGAII_K1_Best	CD	9.82E-05	4.20E-05	7.34E-03	4.87E-03
		PD	9.93E-05	4.25E-05	7.39E-03	4.90E-03
		SD	9.85E-05	4.23E-05	7.36E-03	4.90E-03
	ANN		1.00E-04	4.29E-05	7.44E-03	4.92E-03

Table 23. Comparison results between NSGA-II, NSDE (with 3 variants) and Adaboost (with 3 variants of NSDE) using MSE TRAIN metric on USD/AUD data. The best mean metric value in each row (corresponding to each different case of the individual group size.) is highlighted in bold with gray background. In which, Adaboost Rand 1 and Adaboost_ Best are better than the others.

K	**NSGA-II**	**NSDE-Rand 1**	**NSDE-Current**	**NSDE-Best**	**Adaboost Rand 1**	**Adaboost Current**	**Adaboost Best**
3	9.79E-05	9.10E-05	1.51E-04	8.73E-05	7.94E-05	8.09E-05	7.99E-05
5	9.73E-05	1.93E-04	1.46E-04	8.79E-05	7.84E-05	8.02E-05	7.77E-05
7	9.85E-05	8.82E-05	1.49E-04	1.40E-04	7.83E-05	7.92E-05	7.84E-05
9	9.86E-05	9.22E-05	1.48E-04	1.85E-04	7.83E-05	7.94E-05	7.74E-05
11	9.84E-05	8.79E-05	1.53E-04	2.15E-04	7.74E-05	7.92E-05	7.70E-05
13	9.91E-05	1.82E-04	1.53E-04	1.94E-04	7.74E-05	7.88E-05	7.66E-05
15	9.78E-05	8.80E-05	1.52E-04	1.61E-04	7.68E-05	7.91E-05	7.74E-05
Mean	**9.82E-05**	**1.17E-04**	**1.50E-04**	**1.53E-04**	**7.80E-05**	**7.96E-05**	**7.78E-05**

Table 24. Comparison results between NSGA-II, NSDE (with 3 variants) and Adaboost (with 3 variants of NSDE) using MSE TEST metric on USD/AUD data. The best mean metric value in each row (corresponding to each different case of the individual group size.) is highlighted in bold with gray background. In general, NSDE_Current is better than the others (4/7 cases).

K	NSGA-II	NSDE-Rand 1	NSDE-Current	NSDE-Best	Adaboost Rand 1	Adaboost Current	Adaboost Best
3	1.01E-03	1.33E-05	1.09E-05	1.41E-05	1.41E-05	1.13E-05	1.19E-05
5	9.86E-04	1.31E-05	1.19E-05	1.27E-05	1.24E-05	1.13E-05	1.36E-05
7	7.88E-04	1.32E-05	1.14E-05	1.51E-05	1.58E-05	1.17E-05	1.35E-05
9	1.21E-03	1.28E-05	1.14E-05	1.45E-05	1.10E-05	1.18E-05	1.31E-05
11	1.10E-03	1.40E-05	1.12E-05	1.43E-05	1.29E-05	1.16E-05	1.16E-05
13	1.10E-03	1.44E-05	1.16E-05	1.31E-05	1.25E-05	1.26E-05	1.39E-05
15	6.84E-04	1.27E-05	1.15E-05	1.39E-05	1.43E-05	1.16E-05	1.17E-05
Mean	**9.81E-04**	**1.34E-05**	**1.14E-05**	**1.40E-05**	**1.33E-05**	**1.17E-05**	**1.27E-05**

Table 25. Comparison results between NSGA-II, NSDE (with 3 variants) and Adaboost (with 3 variants of NSDE) using MAE TRAIN metric on USD/AUD data. The best mean metric value in each row (corresponding to each different case of the individual group size.) is highlighted in bold with gray background. In which, Adaboost Current and Adaboost_ Best are better than the others. Adaboost Current gives the best mean metric value.

K	NSGA-II	NSDE-Rand 1	NSDE-Current	NSDE-Best	Adaboost Rand 1	Adaboost Current	Adaboost Best
3	7.32E-03	7.41E-03	1.01E-02	7.21E-03	6.56E-03	6.48E-03	6.56E-03
5	7.31E-03	9.04E-03	9.93E-03	7.24E-03	6.50E-03	6.45E-03	6.44E-03
7	7.35E-03	7.27E-03	1.00E-02	8.28E-03	6.52E-03	6.43E-03	6.50E-03
9	7.36E-03	7.47E-03	1.00E-02	9.58E-03	6.51E-03	6.41E-03	6.45E-03
11	7.36E-03	7.25E-03	1.02E-02	9.82E-03	6.46E-03	6.41E-03	6.43E-03
13	7.39E-03	9.49E-03	1.02E-02	1.04E-02	6.44E-03	6.40E-03	6.43E-03
15	7.33E-03	7.26E-03	1.02E-02	9.68E-03	6.44E-03	6.41E-03	6.47E-03
Mean	**7.35E-03**	**7.89E-03**	**1.01E-02**	**8.89E-03**	**6.49E-03**	**6.43E-03**	**6.47E-03**

Table 26. Comparison results between NSGA-II, NSDE (with 3 variants) and Adaboost (with 3 variants of NSDE) using MAE TEST metric on EUR/AUD data. The best mean metric value in each row (corresponding to each different case of the individual group size.) is highlighted in bold with gray background. In general, NSDE_Current is better than the others (4/7 test cases) and it has the best mean metric value.

K	NSGA-II	NSDE-Rand 1	NSDE-Current	NSDE-Best	Adaboost Rand 1	Adaboost Current	Adaboost Best
3	2.46E-02	1.47E-03	1.34E-03	1.52E-03	1.50E-03	1.37E-03	1.38E-03
5	2.47E-02	1.47E-03	1.40E-03	1.45E-03	1.39E-03	1.37E-03	1.48E-03

7	2.20E-02	1.48E-03	1.38E-03	1.58E-03	1.56E-03	1.39E-03	1.48E-03
9	2.84E-02	1.46E-03	1.38E-03	1.55E-03	1.32E-03	1.40E-03	1.44E-03
11	2.64E-02	1.53E-03	1.36E-03	1.54E-03	1.45E-03	1.39E-03	1.38E-03
13	2.53E-02	1.55E-03	1.39E-03	1.47E-03	1.43E-03	1.44E-03	1.48E-03
15	2.16E-02	1.45E-03	1.38E-03	1.52E-03	1.53E-03	1.39E-03	1.37E-03
Mean	**2.47E-02**	**1.49E-03**	**1.38E-03**	**1.52E-03**	**1.45E-03**	**1.39E-03**	**1.43E-03**

Table 27. Comparison results between NSGAII_K1_Ensemble, NSGAII_K1_Best and ANN on EUR/AUD data. The best mean metric value in each column is highlighted in bold with gray background. It is clear that NSGAII_K1_Ensemble gives the best result (in which, using sharing distance gives the best results in test data, while, using population distance gives the best results in train data.

K	**Method**	**Diversity**	**MSE Train**	**MSE Test**	**MAE Train**	**MAE Test**
1	**NSGAII_K1_Ensemble**	CD	1.10E-04	7.31E-05	7.75E-03	6.10E-03
		PD	9.05E-05	7.19E-05	6.97E-03	6.16E-03
		SD	1.11E-04	5.32E-05	7.58E-03	5.36E-03
	NSGAII_K1_Best	CD	1.18E-04	5.33E-05	7.79E-03	5.37E-03
		PD	1.22E-04	5.33E-05	7.90E-03	5.36E-03
		SD	1.22E-04	5.33E-05	7.90E-03	5.37E-03
	ANN		1.27E-04	7.55E-05	8.40E-03	6.21E-03

Table 28. Comparison results between NSGA-II, NSDE (with 3 variants) and Adaboost (with 3 variants of NSDE) using MSE TRAIN metric on EUR/AUD data. The best mean metric value in each row (corresponding to each different case of the individual group size.) is highlighted in bold with gray background. It is clear that Adaboost_Best outperforms in all test instances.

K	**NSGA-II**	**NSDE-Rand 1**	**NSDE-Current**	**NSDE-Best**	**Adaboost Rand 1**	**Adaboost Current**	**Adaboost Best**
3	1.24E-04	2.74E-04	1.07E-04	8.79E-05	1.30E-04	1.27E-04	1.30E-06
5	1.21E-04	1.57E-04	1.11E-04	2.00E-04	1.21E-04	1.26E-04	1.28E-06
7	1.22E-04	1.47E-04	1.06E-04	1.21E-04	1.19E-04	1.23E-04	1.26E-06
9	1.22E-04	1.27E-04	1.09E-04	1.27E-04	1.16E-04	1.21E-04	1.26E-06
11	1.22E-04	1.71E-04	1.08E-04	1.28E-04	1.18E-04	1.21E-04	1.26E-06
13	1.22E-04	1.31E-04	1.07E-04	1.10E-04	1.12E-04	1.20E-04	1.25E-06
15	1.22E-04	1.38E-04	1.05E-04	1.17E-04	1.11E-04	1.20E-04	1.25E-06
Mean	**1.22E-04**	**1.63E-04**	**1.08E-04**	**1.27E-04**	**1.18E-04**	**1.23E-04**	**1.27E-06**

Table 29. Comparison results between NSGA-II, NSDE (with 3 variants) and Adaboost (with 3 variants of NSDE) using MSE TEST metric on EUR/AUD data. The best mean metric value in each row (corresponding to each different case of the individual group size.) is highlighted in bold with gray background. It is clear that Adaboost_Best outperforms in all test instances.

K	NSGA-II	NSDE-Rand 1	NSDE-Current	NSDE-Best	Adaboost Rand 1	Adaboost Current	Adaboost Best
3	1.53E-03	6.83E-05	6.55E-05	6.83E-05	2.22E-06	1.65E-06	1.96E-07
5	1.23E-03	6.51E-05	6.66E-05	6.75E-05	3.04E-06	1.73E-06	1.85E-07
7	1.15E-03	6.77E-05	6.58E-05	6.81E-05	1.73E-06	1.66E-06	1.48E-07
9	1.66E-03	6.88E-05	6.65E-05	6.64E-05	1.61E-06	1.61E-06	1.58E-07
11	1.70E-03	7.03E-05	6.70E-05	6.53E-05	2.11E-06	1.83E-06	1.56E-07
13	9.60E-04	6.77E-05	6.61E-05	6.55E-05	1.33E-06	1.50E-06	1.80E-07
15	1.16E-03	6.59E-05	6.64E-05	6.91E-05	2.02E-06	1.57E-06	1.66E-07
Mean	**1.34E-03**	**6.77E-05**	**6.63E-05**	**6.72E-05**	**2.01E-06**	**1.65E-06**	**1.70E-07**

Table 30. Comparison results between NSGA-II, NSDE (with 3 variants) and Adaboost (with 3 variants of NSDE) using MAE TRAIN metric on EUR/AUD data. The best mean metric value in each row (corresponding to each different case of the individual group size.) is highlighted in bold with gray background. It is clear that Adaboost_Best outperforms in all test instances.

K	NSGA-II	NSDE-Rand 1	NSDE-Current	NSDE-Best	Adaboost Rand 1	Adaboost Current	Adaboost Best
3	7.93E-03	9.22E-03	7.47E-03	7.04E-03	8.69E-03	8.41E-03	8.38E-04
5	7.84E-03	8.36E-03	7.61E-03	8.96E-03	8.25E-03	8.39E-03	8.38E-04
7	7.86E-03	8.25E-03	7.43E-03	7.86E-03	8.19E-03	8.27E-03	8.32E-04
9	7.86E-03	7.95E-03	7.54E-03	8.03E-03	8.05E-03	8.15E-03	8.32E-04
11	7.85E-03	9.25E-03	7.51E-03	8.22E-03	8.14E-03	8.15E-03	8.28E-04
13	7.87E-03	8.25E-03	7.47E-03	7.70E-03	7.90E-03	8.10E-03	8.27E-04
15	7.87E-03	8.43E-03	7.38E-03	8.02E-03	7.83E-03	8.11E-03	8.28E-04
Mean	**7.87E-03**	**8.53E-03**	**7.49E-03**	**7.97E-03**	**8.15E-03**	**8.23E-03**	**8.32E-04**

Table 31. Comparison results between NSGA-II, NSDE (with 3 variants) and Adaboost (with 3 variants of NSDE) using MAE TEST metric on EUR/AUD data. The best mean metric value in each row (corresponding to each different case of the individual group size.) is highlighted in bold with gray background. It is clear that Adaboost_Best outperforms in all test instances.

K	NSGA-II	NSDE-Rand 1	NSDE-Current	NSDE-Best	Adaboost Rand 1	Adaboost Current	Adaboost Best

3	3.19E-02	4.41E-03	4.42E-03	4.45E-03	5.65E-04	5.21E-04	1.78E-04
5	2.79E-02	4.17E-03	4.47E-03	4.30E-03	6.40E-04	5.33E-04	1.72E-04
7	2.69E-02	4.38E-03	4.42E-03	4.32E-03	5.13E-04	5.21E-04	1.55E-04
9	3.37E-02	4.39E-03	4.46E-03	4.23E-03	4.90E-04	5.16E-04	1.59E-04
11	3.50E-02	4.38E-03	4.47E-03	4.25E-03	5.36E-04	5.48E-04	1.59E-04
13	2.46E-02	4.34E-03	4.43E-03	4.22E-03	4.45E-04	4.95E-04	1.70E-04
15	2.81E-02	4.24E-03	4.44E-03	4.38E-03	5.02E-04	5.08E-04	1.65E-04
Mean	**2.97E-02**	**4.33E-03**	**4.44E-03**	**4.31E-03**	**5.27E-04**	**5.20E-04**	**1.65E-04**

References

[1] J. Sun and H. Li, "Listed companies' financial distress prediction based on weighted majority voting combination of multiple classifiers," Expert Systems with Applications, vol. 35, pp. 818-827, 2008.

[2] C. Hsiao, "Autoregressive modeling of Canadian money and income data," Journal of the American Statistical Association, vol. 74, pp. 553-560, 1979.

[3] R. Meese and K. Rogoff, "The out-of-sample failure of empirical exchange rate models: sampling error or misspecification?," in Exchange rates and international macroeconomics, ed: University of Chicago Press, 1983, pp. 67-112.

[4] W. T. Woo, "The monetary approach to exchange rate determination under rational expectations: The dollar-deutschmark rate," Journal of International Economics, vol. 18, pp. 1-16, 1985.

[5] M. G. Finn, "Forecasting the exchange rate: A monetary or random walk phenomenon?," Journal of International Money and Finance, vol. 5, pp. 181-193, 1986.

[6] E. I. Altman, "Financial ratios, discriminant analysis and the prediction of corporate bankruptcy," The journal of finance, vol. 23, pp. 589-609, 1968.

[7] E. K. Laitinen and T. Laitinen, "Bankruptcy prediction: Application of the Taylor's expansion in logistic regression," International review of financial analysis, vol. 9, pp. 327-349, 2001.

[8] S. J. Taylor, "Forecasting the volatility of currency exchange rates," International Journal of Forecasting, vol. 3, pp. 159-170, 1987.

[9] D. Rumelhart, G. Hinton, and R. Williams, "Learning internal representation by backpropagating errors. DE Rumelhart, & JL McCleland Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 1," ed: MIT Press, Boston, 1986.

[10] P. J. Werbos, "Generalization of backpropagation with application to a recurrent gas market model," Neural Networks, vol. 1, pp. 339-356, 1988.

[11] T. H. Hann and E. Steurer, "Much ado about nothing? Exchange rate forecasting: Neural networks vs. linear models using monthly and weekly data," Neurocomputing, vol. 10, pp. 323-339, 1996.

[12] M. T. Leung, A.-S. Chen, and H. Daouk, "Forecasting exchange rates using general regression neural networks," Computers & Operations Research, vol. 27, pp. 1093-1110, 2000.

[13] G. P. Zhang, "Time series forecasting using a hybrid ARIMA and neural network model," Neurocomputing, vol. 50, pp. 159-175, 2003.

[14] H. Ince and T. B. Trafalis, "A hybrid model for exchange rate prediction," Decision Support Systems, vol. 42, pp. 1054-1062, 2006.

[15] H. Sharma, D. K. Sharma, and H. S. Hota, "A HYBRID NEURO-FUZZY MODEL FOR FOREIGN EXCHANGE RATE PREDICTION," Academy of Accounting & Financial Studies Journal, vol. 20, 2016.

[16] M. Qiu, Y. Song, and F. Akagi, "Application of artificial neural network for the prediction of stock market returns: The case of the Japanese stock market," Chaos, Solitons & Fractals, vol. 85, pp. 1-7, 2016.

[17] R. Fonseca and P. Gomez, "Automatic Model Selection in Ensembles for Time Series Forecasting," IEEE Latin America Transactions, vol. 14, pp. 3811-3819, 2016.

[18] M. Pulido, P. Melin, and O. Castillo, "Design of Ensemble Neural Networks for Predicting the US Dollar/MX Time Series with Particle Swarm Optimization," in Recent Developments and New Direction in Soft-Computing Foundations and Applications, ed: Springer, 2016, pp. 317-329.

[19] T. T. H. Dinh and T. L. Bui, "A multi-objective ensemble learning approach based on the non-dominated sorting differential evolution for forecasting currency exchange rates," in Knowledge and Systems Engineering (KSE), 2016 Eighth International Conference on, 2016, pp. 96-102.

[20] Zhang, G. Peter, and V. L. Berardi. "Time series forecasting with neural network ensembles: an application for exchange rate prediction." Journal of the Operational Research Society 52.6 (2001): 652-664.

[21] Landassuri-Moreno, Victor M., and John A. Bullinaria. "Neural network ensembles for time series forecasting." Proceedings of the 11th Annual conference on Genetic and evolutionary computation. ACM, 2009.

[22] Yu, Lean, Kin Keung Lai, and Shouyang Wang. "Multistage RBF neural network ensemble learning for exchange rates forecasting." Neurocomputing 71.16 (2008): 3295-3302.

[23] R. E. Schapire, "The boosting approach to machine learning: An overview," in Nonlinear estimation and classification, ed: Springer, 2003, pp. 149-171.

[24] H. Allende and C. Valle, "Ensemble Methods for Time Series Forecasting," in Claudio Moraga: A Passion for Multi-Valued Logic and Soft Computing, ed: Springer, 2017, pp. 217-232.

[25] Y. Zhang, G. Yu, and D. Yang, "Predicting non-performing loan of business bank by multiple classifier fusion algorithms," Journal of Interdisciplinary Mathematics, vol. 19, pp. 657-667, 2016.

[26] S. Gu and Y. Jin, "Generating diverse and accurate classifier ensembles using multi-objective optimization," in Computational Intelligence in Multi-Criteria Decision-Making (MCDM), 2014 IEEE Symposium on, 2014, pp. 9-15.

[27] J. W. Shavlik, "Generating accurate and diverse members of a neural-network ensemble," 1996.

[28] R. Adhikari and R. Agrawal, "An introductory study on time series modeling and forecasting," arXiv preprint arXiv:1302.6613, 2013.

[29] E. Alpaydin, Introduction to machine learning: MIT press, 2014.

[30] P. G. Harrald and M. Kamstra, "Evolving artificial neural networks to combine financial forecasts," IEEE Transactions on Evolutionary Computation, vol. 1, pp. 40-52, 1997.
[31] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," IEEE Transactions on Evolutionary Computation, vol. 6, pp. 182-197, 2002.
[32] N. Srinvas and K. Deb, "Multi-objective function optimization using non-dominated sorting genetic algorithms," Evolutionary Computation, vol. 2, pp. 221-248, 1994.
[33] A. W. Iorio and X. Li, "Solving rotated multi-objective optimization problems using differential evolution," in Australasian Joint Conference on Artificial Intelligence, 2004, pp. 861-872.
[34] P. Xiaoying, Z. Jing, C. Hao, C. Xuejing, and H. Kaikai, "A differential evolution-based hybrid NSGA-II for multi-objective optimization," in 2015 IEEE 7th International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM), 2015, pp. 81-86.
[35] V. Kenneth, "Price, An introduction to differential evolution, New ideas in optimization," ed: McGraw-Hill Ltd., UK, Maidenhead, UK, 1999.
[36] E. Mezura-Montes, M. Reyes-Sierra, and C. A. C. Coello, "Multi-objective optimization using differential evolution: a survey of the state-of-the-art," in Advances in differential evolution, ed: Springer, 2008, pp. 173-196.
[37] Y. Freund and R. E. Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting," in European conference on computational learning theory, 1995, pp. 23-37.
[38] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in Icml, 1996, pp. 148-156.
[39] Y. Freund and R. E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," Journal of Computer and System Sciences, vol. 55, pp. 119-139, 1997.
[40] H. Drucker, "Improving regressors using boosting techniques," in ICML, 1997, pp. 107-115.
[41] R. Feely, "Predicting stock market volatility using neural networks," BA (Mod) Dissertation, Department of Computer Science, Trinity College Dublin, 2000.
[42] R. Avnimelech and N. Intrator, "Boosting regression estimators," Neural computation, vol. 11, pp. 499-520, 1999.
[43] D. L. Shrestha and D. P. Solomatine, "Experiments with AdaBoost. RT, an improved boosting scheme for regression," Neural computation, vol. 18, pp. 1678-1710, 2006.
[44] H.-X. Tian and Z.-Z. Mao, "An ensemble ELM based on modified AdaBoost. RT algorithm for predicting the temperature of molten steel in ladle furnace," IEEE Transactions on Automation Science and Engineering, vol. 7, pp. 73-80, 2010.
[45] S. Makridakis and M. Hibon, "Evaluating accuracy (or error) measures," 1995.