Forecasting Currency Exchange Rates with an Artificial Bee Colony-Optimized Neural Network

Chukiat Worasucheep
Applied Computer Science, Department of Mathematics, Faculty of Science
King Mongkut's University of Technology Thonburi, Thailand
Email: chukiat.wor@kmutt.ac.th

***Abstract*—This paper applies a recent variant of Artificial Bee Colony (ABC) to optimize the weights of a three-layer feed-forward neural network for forecasting of currency exchange rates of USD/EUR and USD/Yen. The inputs to the network is built from historical prices and a set of well-known technical indicators, including Moving Average, Moving Average Convergence/Divergence and Relative Strength Index. The forecasting model becomes a complex minimization problem with fifty-decision variables, many of which are interdependent. The ABC variant in this work is ABCDE that is a hybrid algorithm of original ABC with two different mutation strategies of Differential Evolution (DE). The experimental results present a superior performance of ABCDE in terms of both training and testing errors against original ABC, Back Propagation and ODE [35], an efficient variant of DE.**

Keywords—Foreign exchange rate; Forecasting; Artificial Bee Colony; Neural Network

I. INTRODUCTION

Many studies on financial time series forecast have been conducted over the past few decades due to its highly potential profitability. These time series are known to be complex, non-stationary, non-linear and very noisy [1]. In the literature, different methods applied to forecast about the financial market can be grouped into two broad categories - fundamental analysis and technical analysis. The fundamental analysis examines market economic factors and information from financial statement of a corporation in order to forecast its market value of the stock. Technical analysis studies historical prices and volumes of a financial instrument and its trends. Believing in recurring patterns in market behavior, technical analysts use a number of technical indicators and charting patterns to identify price trend and forecast future prices [2].

However, financial time series forecast is still regarded as one of the most challenging applications of forecasting. Numerous forecasting models have been developed using many statistical and computing techniques as collected by a recent survey [3]. Artificial neural networks (ANNs) are probably the most widely used computing technique in the last two decades [4]–[7]. This is mainly because ANNs are non-linear in nature and are data-driven, requiring minimal assumptions about the model of the problem. It has been shown that ANNs have universal approximation ability for continuous functions. Among ANN algorithms, the Feed-Forward Neural Network (FFNN) with Back Propagation (BP) learning algorithm [8] is the most popular method in many applications such as classification, pattern recognition, and forecasting. However, FFNN requires the definition of several system parameters especially the weights of connections between its processing units, which are not easy to determine [7]. BP algorithm utilizes gradient descent technique which is favorable for local search [9]. Furthermore BP generates complex error surfaces with many local minima, and BP tends to converge into local optima instead of global minima, and thus dissatisfies generalization ability [10][11]. Several meta-heuristics including evolutionary algorithms, swarm intelligence techniques, and others have been applied to optimize the values of the weights instead of or in addition to BP [12]–[14]. These algorithms have been widely adopted by researchers for solving various complex optimization problems with difficult fitness landscape.

In this work, an improved Artificial Bee Colony (ABC) algorithm [15] is used for optimizing the weights of the ANN for forecasting a times series. ABC is a biological-inspired population-based stochastic algorithm, recently proposed by D. Karaboga, which mimics the foraging behavior of honey bee swarm. The ABC variant employed in this work is the *ABCDE* which is recently proposed [16]. It is a hybrid algorithm combining the strength of mutation operations of Differential Evolution (DE) [17] into the original ABC without additional algorithmic parameters. ABCDE is applied for optimizing the weights of an ANN to forecast of two pairs of currency exchange rates: USD/EUR and USD/Yen. Time series of currency rates are considered as very noisy, non-stationary, and chaotic, and thus very challenging [1]. Inputs to the ANN are historical prices and technical indicators widely used such as Moving Averages and Relative Strength Indexes. The performance of the ABCDE-optimized ANN is compared with that of BP and an efficient DE.

The remainder of the paper has the following structure. Section II presents background of ANN, ABC and the hybrid ABCDE. Section III describes the approach of optimizing ANN's weights with the proposed hybrid algorithm and evaluates the results. Section IV concludes the paper with some future works.

II. BACKGROUND

A. Artificial neural networks

Artificial neural networks (ANN) are computing techniques modelling of an interconnection of neurons of human's nervous system. In an ANN, artificial neurons are fundamental

978-1-4799-7492-4/15/$31.00 (c)2015 IEEE

computing units performing a weighted sum on all of their inputs and the results go through a transfer function to produce an output. An ANN is formed by a large number of neurons organized into layers. The most widely used architecture for time series forecast has three layers consisting of an input layer, an output layer and one hidden layer. Additional hidden layers make the training more difficult, time consuming and less generalization ability. Each layer has a number of neurons as nonlinear processing elements with transfer functions that are normally smooth (e.g. the logistic function or hyperbolic tangent function). All nodes of the adjacent layers are often fully interconnected in order to feed output forward from input layer through the hidden layer to the output layer. This category of ANNs is called a Feed-Forward Neural Network (FFNN).

An ANN must be trained in order to determine optimal values of the weights of these interconnections, which will produce the correct outputs. One of the most widely used method for training an ANN is Back Propagation (BP) [8]. BP is a supervised learning algorithm, i.e. learning from samples, with steepest descent technique for an FFNN [18]. The structure of an FFNN with one output node is represented as in Fig. 1. An FFNN is trained with the BP using a training data set that consists of inputs and a desired output. FFNN with BP processes input patterns one at a time and, given a set of weights and a transfer function, computes the associated output values of the hidden layer neurons. These output values of the hidden neurons then are used as inputs to a similar processing occurred between the hidden layer and the output layer. This final output value is then compared with the target output to compute the error for this record. A gradient steepest descent approach is used to propagate this error back through the network so as to adjust the weights to minimize the total accumulated error. The steps described above is repeated until the error is acceptable or a maximum number of iterations is reached. Details of BP algorithm can be reached at [8][18].

During the back propagation, there are two important parameters: a learning rate and a momentum. The learning rate, a positive constant between 0 and 1, controls the balance of accuracy, speed of convergence and stability. A lower value of learning rate often provides a more accurate and stable results but with a slow convergence, while a higher value gets opposite. The momentum term helps speed up the learning process while preventing the learning from being trapped into local minima, and is usually chosen in the interval [0, 1] [8].

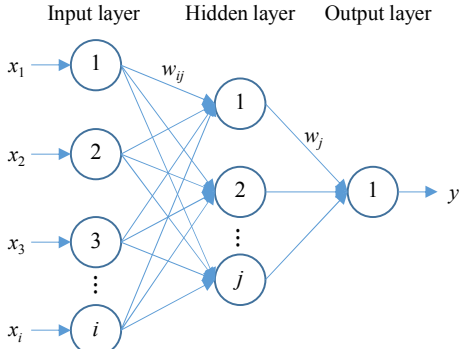

Fig. 1. Structure of an FFNN with one output node.

To avoid common drawbacks and to increase accuracy of the BP, researchers proposed several improvements including self-adaptation of BP parameters or using various search algorithms like genetic algorithm (GA), DE or Particle Swarm Optimization (PSO) to optimize network parameters like weights or the number of hidden nodes [19][20][12][14].

B. The hybrid ABCDE optimizer

Artificial Bee Colony (ABC) algorithm is a relatively new population-based algorithm for continuous problems [15]. Development of ABC is inspired from the foraging behavior of honey bee colony consisting of three groups of bees: employed bees, onlooker bees and scout bees. The employed bees search food sources and carry information about their positions to the hive. Onlooker bees are waiting in the hive for the food source information shared by the employed bees. An onlooker bee evaluates the information showed by the employed bees and choose a food source according to the probability proportional to the quality, the amount of nectar, of that food source. Once an onlooker is attracted to one food source information, it updates food information, keeps only the better one, and shares its information together. If a food source could not be improved in a certain time (called as abandon *limit*), the employed bee abandons this food source and becomes a scout bee exploring a new food source.

ABC algorithm has advantages of robustness and simplicity of having fewer control parameters [21]. Its performance is very competitive to other population-based algorithms [22], and ABC has gone through a continuous improvement resulting in a number of good variants [23]-[28]. So far, ABC has been applied for solving many practical problems successfully including surface roughness optimization [29], mobile robot path-planning [30], protein structure [31], power electric [32], etc.

The original ABC consists of a population of D-dimensional candidate solutions x_i (i = 1,…,N) representing N food source locations. The generic algorithmic structure of ABC is as follows:

Initialization
REPEAT
    Employed bees phase
    Onlooker bees phase
    Scout bees phase
    Memorize the best solution achieved so far
UNTIL any termination criteria is met

Both employed bees and onlooker bees use the same modification equation (1). This limits search behavior.

$$u_{i,j} = x_{i,j} + \phi_{i,j}(x_{i,j} - x_{k,j}), \tag{1}$$

where $\phi_{i,j}$ is a random number in the range [-1, 1]. A new food source is created depending on only one member chosen with a random, without directional information for improvement. In addition, only one dimension is updated at a time, limiting search diversity. The hybrid ABCDE used in this work [16] applies search equation of DE/current-to-rand [33][34] for the employed bees. The employed bees use the following equation (2) instead of equation (1).

***Algorithm 1*: Hybrid ABCDE algorithm**

Given objective function f, dimension D and MAXNFC
Create and randomly initialize all solutions x_i, $i = 1..PS$
Designate the bee with minimum $f(x_i)$ as *best*
Set $limit = 0.6 \cdot D \cdot PS$ and set $trial_i = 0$, $i = 1..PS$
$nfc = 0$
while Stop condition is not satisfied, i.e. nfc <= MAXNFC **do**
  //---------- the employed bee phase ----------
  Randomly create $F' \in [0, 1]$
  for $i = 1$ **to** PS **do**
    Create random integer numbers $r1, r2, r3 \in [0, PS] \wedge r1 \neq r2 \neq r3$
    Randomly create $k \in [0, 1]$
    Generate and evaluate a new food source u_i using equation (2)
    $nfc = nfc + 1$
    if $f(u_i) < f(x_i)$ **then**
      $x_i = u_i$, $trial_i = 0$
      Designate x_i as the *best* bee if $f(x_i) < f(x_{best})$
    else
      $trial_i = trial_i + 1$
    end if
  end for
  if nfc > MAXNFC **then**
    break
  end if
  //---------- the onlooker phase ----------
  Calculate the probability values p_i for each solutions p_i
  for $i = 1$ **to** PS **do**
    Create two random integer numbers $src, dest \in [0, PS]$
      $\wedge\ src \neq dest \wedge f(x_{dest}) < f(x_{src})$
    Randomly create $F \in [0, 1]$
    Generate and evaluate a new solution u_i according to equation (3)
    $nfc = nfc + 1$
    if $f(u_i) < f(x_i)$ **then**
      $x_i = u_i$, $trial_i = 0$
      Designate x_i as the *best* bee if $f(x_i) < f(x_{best})$
    else
      $trial_i = trial_i + 1$
    end if
  end for
  //---------- the scout bees ----------
  for $i = 1$ **to** PS **do**
    if $trial_i >$ limit **then**
      Randomly initialize the bee x_i with uniform random
      Evaluate the bee
      $nfc = nfc + 1$
  end for
end while

$$\vec{u}_i = \vec{x}_i + k_i \cdot (\vec{x}_{r1} - \vec{x}_i) + F' \cdot (\vec{x}_{r2} - \vec{x}_{r3}) \quad (2)$$

where $r1$, $r2$ and $r3$ are indexes of food sources randomly chosen and $r1 \neq r2 \neq r3 \neq i$. $k_{i,j}$ is a random value from a uniform distribution between 0 and 1. F' is a random value from a uniform distribution between 0 and 1, created once for each iteration.

Onlooker bees use a combined DE/best/1 and DE/rand/2/dir so that the search is around the bee with the best fitness value and the objective function information is incorporated to guide the direction as in the following way:

$$u_{i,j} = x_{best,j} + F(x_{dest,j} - x_{src,j}) \quad (3)$$

where x_{best} denotes the bee with the currently best fitness value. x_{src} and x_{dest} are the bees chosen with random such that $f(x_{dest}) < f(x_{src})$, for a minimization problem. F is a random value from a uniform distribution between 0 and 1.

All the modifications described above do not introduce additional algorithmic parameters, and hence the strength of the ABC algorithm is retained. Algorithm ABCDE can be summarized as in *Algorithm 1*. The performance of ABCDE [16] was demonstrated using a set of twelve widely accepted benchmark of nonlinear minimization problems with different characteristics including multimodality, shift and rotation. The experimental results in [16] clearly demonstrated a superior performance of ABCDE against original ABC and Opposition-Based DE [35].

III. FORECASTING WITH ABCDE-OPTIMIZED ANN

A. Model construction

In this section, a Feed-Forward Neural Network (FFNN) is applied to construct a forecasting model to estimate the currency exchange rates. Many components of FFNN model must be carefully designed in order to avoid pitfalls [36], i.e. a selection of data set, input variables, the number of hidden nodes, etc. The data sets used in this experiment are time series from daily USD/EUR and USD/Yen exchange rates between 1/1/2010 and 31/12/2013 downloaded from www.oanda.com/currency/historical-rates. To avoid conclusion from too few sample tests, this study chooses 6 data sets out of the two time series as shown in Fig. 1. Each data set spans over two years and has either 730 or 731 records (due to a leap year effect). The size of training set is 585 (or 586) records (equivalent to 80%), whereas the size of test set 145 records (20%) for each data set. The training set is used for model fitting with different algorithms while the testing set measures forecasting performance of the model.

The forecasting FFNN has 9 input nodes (I) as listed in Fig. 2. Each record at day t is created from the exchange rate at day t (P_t) and some technical indicators which are most popularly used by researchers in forecasting of time series. The selected indicators are Moving Average (MA), Moving Average Convergence/Divergence (MACD) and Relative Strength Index (RSI).

MA is a simple arithmetic mean of prices over the most recent days and can be used to be interpreted as support in a rising market or resistance in a falling market. Many period lengths are often used such as 5, 10, 20, 40, 60, 80, 100, 200, etc. for different forecasting horizons. MACD is a momentum indicator that shows the difference between two moving averages – a short (or fast) and a long (or slow) price series. A signal line is then created by taking another moving average on the MACD. Thus MACD requires 3 parameters indicating the periods of calculating the moving averages. The most commonly used values are 12, 26, and 9. The difference between MACD and signal lines results in the divergence. The signal and the divergence lines together may reveal subtle shifts in strengths and direction in a price trend. RSI is a momentum oscillator that compares the magnitude of recent gains to the magnitude of recent losses, indicating current and historical strength and weakness of the closing prices of a recent trading period. RSI is mostly used on a 14 period

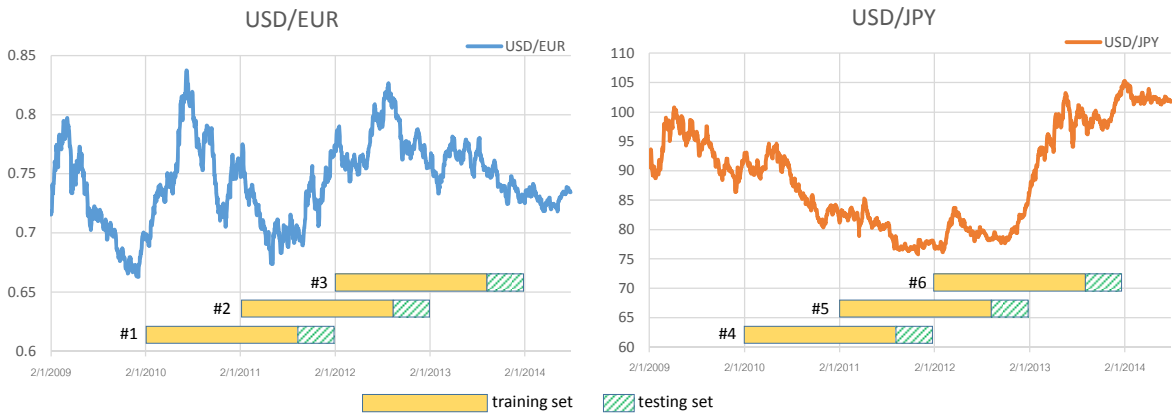

Fig. 2. Six datasets used in the experiment. Each is divided into 585 or 586 days of training set, and 145 days of testing set.

P_t	exchange rate at day t.
$P_t - P_{t-1}$	P_{t-1} is the exchange rate at day $t-1$.
$P_t - P_{t-3}$	P_{t-3} is the exchange rate at day $t-3$.
$P_t - MA_5$	MA_5 is the 5-day Moving Average at day t.
$P_t - MA_{20}$	MA_{20} is the 20-day Moving Average at day t.
$P_t - MA_{40}$	MA_{40} is the 40-day Moving Average at day t.
$P_t - MA_{80}$	MA_{80} is the 80-day Moving Average at day t.
$P_t - MACD_{12,26,9}$	$MACD_{12,26,9}$ is the Moving Average Convergence/Divergence (12, 26, 9) at day t.
RSI_{14}	14-day Relative Strength Index at day t.

Fig. 3. Inputs to the FFNN.

timeframe and measured on a scale of 0 to 100. High and low levels are often marked at 70 and 30, respectively. The definitions or formulae of MA, MACD and RSI can be found in literature [37][38][2].

The activation function chosen is sigmoid function and given as:

$$Sigmoid(x) = 1/(1+e^{-x}) \tag{4}$$

All inputs to network are normalized to [0, 1] prior to the training for proper functions of the network [7], as follows.

$$x = (inp - inp_{\min})/(inp_{\max} - inp_{\min}), \tag{5}$$

where x is normalized value and inp is value to be normalized. $inp_{\min}$ and $inp_{\max}$ are the minimum value and maximum value, respectively, of the series to be normalized.

Output of FFNN is the one-day ahead exchange rate (P_{t+1}), and thus the number of output nodes (O) is 1. For the testing set, the performance is measured using a window shifting method that forecasts a point of $t+1$ using the data from $t-n$ to t, and moves forward the window (of size n) to the end of testing set [39].

The number of nodes in the hidden layer is very important. Too many or too few hidden nodes will cause overfitting or underfitting problem [36]. Unfortunately there is no definite formulae to determine this parameter. Commonly used guidelines suggest that the number of hidden nodes relate to the numbers of input and output nodes [40][41][42]. In this work, since $I = 9$ and $O = 1$, the number of hidden nodes (H) is set to 5.

B. *Algorithms and parameter setting*

Four learning algorithms: ABC, ABCDE, ODE and BP, are used independently to optimize the weight values of FFNN. Since ABC, ABCDE and ODE algorithms depend on fitness function to guide the search, the fitness function for minimization herein is Mean Square Error (MSE) as following:

$$f = MSE = \frac{1}{NS}\sum_{k=1}^{NS}[\hat{y}_k(t) - y_k(t)]^2, \tag{6}$$

where y_k is the model output and $\hat{y}_k$ is the corresponding desired value. NS is the number of samples over which the error is compared. A candidate solution (called a vector for ODE, and a food source for ABC and ABCDE) represents a 3-layer network discussed earlier. The network's weights are variables to be optimized by the algorithm for the lowest MSE. Dimensionality of the optimization problem is therefore equal to the number of weights, which is $I \cdot H + H \cdot O = 9 \times 5 + 5 \times 1 = 50$. MAXNFC is set to 200000 for all algorithms. The population size (PS) for ABC, ABCDE, and ODE is set to 40, 40, and 100 respectively, which is a bit larger than that is used in experiment in [16] to deal with more variables. PS for ODE is larger than that of ABC's because ODE is an evolutionary algorithm which generally requires more candidate solutions than a swarm intelligence technique such as ABC and PSO. PS for ODE is recommended widely from $3 \cdot D$ to $10 \cdot D$ [17][34].

For BP, the learning rate and the momentum used have two sets: (0.5, 0.1) and (0.01, 0.9), denoted as BP1 and BP2 respectively, in order to investigate the differences. Maximum epoch for BP1 and BP2 is set to 200000 which is equal to the MAXNFC. Every algorithm is executed for 20 independent runs. At the beginning, all weights are randomly initialized within [0, 1].

After training, the optimized model will be used for forecasting of the unseen testing set. To validate the forecasting accuracy, the mean absolute percentage error (MAPE) is used and given as:

$$MAPE = \frac{1}{NT}\sum_{k=1}^{NT}\left|\frac{y_k(t)-\hat{y}_k(t)}{y_k(t)}\right| \quad (7)$$

where NT is the size of testing set. The MAPE values can be multiplied with 100 to be shown in % value. While MSE is widely used during the training, MAPE is used here to validate the testing accuracy because managers understand percentages better than squared errors. Also the use of absolute value avoids the problem of positive and negative errors canceling each other. Note that the output is first denormalized to $[inp_{\min}, inp_{\max}]$ before MAPE is calculated.

C. Result Comparison and Discussions

Table 1 reports the forecasting performance of 5 algorithms on 6 data sets. Column *Training MSE* reports average and standard deviation of the MSE values obtained for the training set from 20 runs. The column *Testing MAPE* reports average and standard deviation of the MAPE values obtained for the testing set. Columns *Rank (Train)* and *(Test)* show the ranks of training MSE and testing MAPE, respectively, of each algorithm. This ranking employs Wilcoxon signed-rank test to compare each pair of algorithms at 0.05 significant level. An algorithm j is ranked better than algorithm k ($r_j < r_k$) if the Wilcoxon signed-rank test result of algorithms j against k gets a p-value below 0.05. Two algorithms are ranked equally if the Wilcoxon signed-rank test result is not significant. Table 2 summarizes the Average Aggregated Ranks (AAR's) of each algorithm for 6 different data sets tested.

From Table 1, it can be observed that ABCDE achieves the best (lowest) training MSE and testing MAPE for 5 out of 6 data sets. On average, the first runner up is ODE while ABC gets the third rank. All these three stochastic algorithms clearly outperform BP1 and BP2. The complex landscape of fitness function pulls the BP to be trapped in some local optima before getting a better solution. BP1 and BP2 perform equally except only in data sets #5 and #6. Fig. 4–6 present the forecasting results with actual exchange rates and the MAPE's of ABCDE and ODE for six data sets. Analysis of results leads to the following summary.

1) Stochastic search algorithms, such as ABC or DE and their variants, are less prone to local optima, as being compared to BP that uses gradient descent technique.

2) A comparison of different learning algorithms shows the superior forecasting performance of ABCDE both in terms of learning and generalization ability.

3) In addition, the graphs of forecasting results of all algorithms swing on the actual graph. The reason is probably that fitness function during training herein uses solely MSE. MSE concerns with only error size and ignores direction (up/down). Inclusion of other metric such as the Prediction of Correctness in Direction (POCID) could be helpful. POCID measures the ability of a method to forecast if the future series value (prediction target) will increase or decrease with respect to the previous value. A proper combining of POCID and MSE to make a composite fitness function during training could anticipate a better forecasting result.

IV. Conclusion

This paper applies a recently proposed hybrid Artificial Bee Colony algorithm with Differential Evolution, called ABCDE [16], for optimizing weights in a Feed-Forward Neural Network for forecasting of currency exchange time series. Inputs are constructed from some widely used technical indicators such as MA, MACD and RSI. With an architecture of 9-5-1, the network model becomes a 50-dimensional optimization function with complex interdependency of variables. The results reveal that ABCDE has demonstrated its superior performance to ODE, original ABC and BP algorithms in both optimization of network's weights during training and generalization during testing.

Future research may include investigation of other technical indicators and tuning of the parameters of neural network model such as the number of hidden nodes. Having a validation set in data separation could help better avoid overfitting. Other accuracy metrics like POCID or those in [7] or a composite of them should be considered for inclusion in the fitness for training. Finally, the more advanced learning algorithms than BP and the network optimization using other algorithms like GA or PSO should be used for a comparison.

References

[1] C. Giles, S. Lawrence, S. Tsoi, "Noisy time series prediction using a recurrent ANN," Journal of Machine Learning, vol. 44, pp. 161–183, 2001.

[2] R. Majhi, G. Panda, and G. Sahoo, "Development and performance evaluation of FLANN based model for forecasting of stock markets," Expert Systems with Applications, vol. 36, pp. 6800–6808, 2009.

[3] G.S. Atsalakis, K.P. Valavanis, "Surveying stock market forecasting techniques – part II: Soft computing methods," Expert Systems with Applications, vol. 36, pp. 5932–5941, 2009.

[4] G. Zhang, B. E. Patuwo, M. Y. Hu, "Forecasting with artificial neural networks: the state of the art," International Journal of Forecasting, vol. 14, pp. 35–62, 1998.

[5] R. Sitte, J. Sitte, "Neural networks approach to the random walk dilemma of financial time series," Applied Intelligence, vol. 16, no. 3, pp. 163–171, 2002.

[6] G.P. Zhang, D. Kline, "Quarterly time-series forecasting with neural networks," IEEE Transactions on Neural Networks, vol. 18, no. 6, pp. 1800–1814, 2007.

[7] F.A. de Oliveira, C.N. Nobre, L.E. Zárate, "Applying Artificial Neural Networks to prediction of stock price and improvement of the directional prediction index – Case study of PETR4," Petrobras, Brazil, Expert Systems with Applications, vol. 40, no. 18, pp. 7596-7606, 2013.

[8] D.E. Rumelhart, E.H. Geoffey, J.W. Ronald, "Learning representations by back-propagating errors," Nature, vol. 323, no. 6088, pp. 533–536, 1986.

[9] R. Sexton, R. Dorsey, J. Johnson, "Optimization of neural networks: a comparative analysis of the genetic algorithm and simulated annealing," Eur. J. Operation Research, vol. 114, pp. 589–601, 1999.

[10] Engoziner, S., Tomesn, E., "An accelerated learning algorithm for multiplayer perception: Optimization layer by layer," IEEE Transactions on Neural Networks, vol. 6, pp. 31–42, 1995.

[11] Gupta JND., Sexton R.S., "Comparing backpropagation with a genetic algorithm for neural network training," Omega Int J Manag Sci, vol. 27, pp. 679–684, 1999.

[12] R. Bisoi, P.K. Dash, "A hybrid evolutionary dynamic neural network for stock market trend analysis and prediction using unscented Kalman filter," Applied Soft Computing, vol. 19, pp. 41–56, 2014.

[13] J.-R. Zhang, J. Zhang, T.-M. Lok, M.R. Lyu, "A hybrid particle swarm optimization–back–propagation algorithm for feedforward neural

network training," Applied Mathematics and Computation, vol. 185, no. 2, pp. 1026–1037, 2007.

[14] C. Worasucheep, "Training a single multiplicative neuron with a harmony search algorithm for prediction of S&P500 index-An extensive performance evaluation," In Knowledge and Smart Technology, 2012 4th International Conference on, pp. 1–5, 2012.

[15] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Erciyes University, Kayseri, Turkey, Technical Report-TR06, 2005.

[16] C. Worasucheep, "A Hybrid Artificial Bee Colony with Differential Evolution," International Journal of Machine Learning and Computing vol. 5, no. 3, pp. 179–186, 2015.

[17] R. Storn, K. Price, "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces," Journal of Global Optimization, vol. 11, pp. 341–359, 1997.

[18] S. Haykin, "Neural Networks, a Comprehensive Foundation," Upper Saddle River, NJ, USA: Prentice Hall; 1999.

[19] W. Shen , X. G. C. Wub, D. Wu, "Forecasting stock indices using radial basis function neural networks optimized by artificial fish swarm algorithm," Knowledge Based Systems, 24, pp. 378-385, 2011.

[20] S. Oh, W. Kim, W. Pedrycz, S. Joo, "Design of K-means clustering-based polynomial radial basis function neural networks (PRBF NNs) realized with the aid of particle swarm optimization and differential evolution," Neuro computing, vol. 78, pp. 121-132, 2012.

[21] D. Karaboga, B. Basturk, "On the performance of artificial bee colony (ABC) algorithm," Applied Soft Computing, vol. 8, pp. 687–697, 2008.

[22] D. Karaboga, B. Akay, "A comparative study of artificial bee colony algorithm," Applied Mathematics and Computation, vol. 214, pp. 108–132, 2009.

[23] B. Alatas, "Chaotic bee colony algorithms for global numerical optimization," Expert Systems with Applications, vol. 37, no. 8, pp. 5682–5687, 2010.

[24] M.S. Kıran, M. Gündüz, "A recombination-based hybridization of particle swarm optimization and artificial bee colony algorithm for continuous optimization problems," Applied Soft Computing, vol. 13, pp. 2188–2203, 2013.

[25] W.F. Gao, S.Y. Liu, L.L. Huang, "A novel artificial bee colony algorithm based on modified search equation and orthogonal learning," IEEE Transactions on Systems, Man, and Cybernetics - Part B 43, pp. 1011–1024, 2013.

[26] W. Xiang, S. Ma, M. An, "hABCDE: A hybrid evolutionary algorithm based on artificial bee colony algorithm and differential evolution," Applied Mathematics and Computation, vol. 238, pp. 370–386, 2014.

[27] H. Wang, Z. Wu, S. Rahnamayan, H. Sun, Y. Liu, J.-S. Pan, "Multi-strategy ensemble artificial bee colony algorithm," Information Sciences, 279, pp. 587-603, 2014.

[28] D. Bose, S. Biswas, A.V. Vasilakos, S. Laha, Optimal filter design using an improved artificial bee colony algorithm, Information Sciences, 281, pp. 443–461, 2014.

[29] M.K. Das, K. Kumar, T. Kr. Barman, P. Sahoo, "Application of Artificial Bee Colony Algorithm for Optimization of MRR and Surface Roughness in EDM of EN31 Tool Steel," Procedia Materials Science, vol. 6, 2014, pp. 741-751.

[30] J.-H. Liang, C.-H. Lee, "Efficient collision-free path-planning of multiple mobile robots system using efficient artificial bee colony algorithm," Advances in Engineering Software, vol. 79, pp. 47-56, Jan. 2015.

[31] B. Li, R. Chiong, M. Lin, "A balance-evolution artificial bee colony algorithm for protein structure optimization based on a three-dimensional AB off-lattice model," Computational Biology and Chemistry, vol. 54, pp. 1-12, Feb. 2015.

[32] P. Lu, J. Zhou, H. Zhang, R. Zhang, C. Wang, "Chaotic differential bee colony optimization algorithm for dynamic economic dispatch problem with valve-point effects", Electrical Power and Energy Systems 62, pp. 130–143, 2014.

[33] K.V. Price, An introduction to differential evolution, in New Ideas in Optimization, D. Corne, M. Dorigo, and F. Glover, Eds. London, U.K.: McGraw-Hill, 1999, pp. 79–108.

[34] S. Das, P.N. Suganthan, "Differential evolution—A survey of the state-of-the-art," IEEE Trans. Evolutionary Computation, vol. 15, no. 1, pp. 4–31, 2011.

[35] S. Rahnamayan, H.R. Tizhoosh, M.M.A. Salama, "Opposition-Based Differential Evolution", IEEE Transactions on Evolutionary Computation, vol. 12, pp. 64–79, 2008.

[36] G.P. Zhang, "Avoiding pitfalls in neural network research," IEEE Trans. Syst., Man, Cybern., Part C: Appl. Rev., vol. 37, pp. 3–16, 2007.

[37] C.M. Hsu, "A hybrid procedure for stock price prediction by integrating self-organizing map and genetic programming," Expert Systems with Applications, vol. 38, no. 11, pp. 14026-14036, 2011.

[38] K.-J. Kim, "Financial time series forecasting using support vector machines," Neurocomputing, vol. 55, 307–319, 2003.

[39] M. O'Connor, W. Remus, K. Griggs, "Does updating judgmental forecasts improve forecast accuracy?" International Journal of Forecasting, 16, pp. 101-109, 2000.

[40] T. Masters, Practical Neural Network Recipes in C++, Academic Press, Boston, MA., 1994.

[41] M.N. Jadid, D.R. Fairbairn, "Predicting moment curvature parameters from experimental data," Eng. Appl. Artif. lntell. vol. 9, pp. 309-319, 1996.

[42] G.B. Huang, "Learning capability and storage capacity of two-hidden-layer feedforward networks," IEEE Transactions on Neural Networks, vol. 14, 274–281, 2003.

TABLE 1 FORECASTING PERFORMANCE COMPARISONS OF 5 ALGORITHMS FOR SIX DATA SETS.

#	Algo.	Training MSE		Testing MAPE		Rank	
		Mean	S.D.	Mean	S.D.	Train	Test
1	ABCDE	0.0156	0.0130	0.0825	0.0670	2	2
	ABC	0.0204	0.0128	0.0704	0.0582	3	2
	ODE	0.0119	0.0103	0.0410	0.0568	1	1
	BP1	0.0501	0.0215	0.1492	0.0948	4	4
	BP2	0.0384	0.0256	0.1833	0.0624	4	4
2	ABCDE	0.0109	0.0074	0.0498	0.0328	1	1
	ABC	0.0298	0.0182	0.1026	0.0680	3	3
	ODE	0.0127	0.0111	0.0514	0.0629	1	1
	BP1	0.0803	0.0354	0.2183	0.1213	4	4
	BP2	0.0790	0.0486	0.2264	0.1305	4	4
3	ABCDE	0.0056	0.0063	0.0514	0.0448	1	1
	ABC	0.0187	0.0141	0.1182	0.0704	3	3
	ODE	0.0146	0.0109	0.0928	0.0956	1	3
	BP1	0.0413	0.0384	0.4592	0.1748	4	4
	BP2	0.0405	0.0355	0.3343	0.2728	4	4

#	Algo.	Training MSE		Testing MAPE		Rank	
		Mean	S.D.	Mean	S.D.	Test	Train
4	ABCDE	0.0357	0.2600	0.0536	0.0360	1	1
	ABC	0.0723	0.4675	0.1094	0.0575	2	2
	ODE	0.0842	0.0115	0.1147	0.0840	2	2
	BP1	0.1705	0.1080	0.4021	0.2380	4	4
	BP2	0.1408	0.1170	0.3969	0.1720	4	4
5	ABCDE	0.0304	0.0285	0.0789	0.0480	1	1
	ABC	0.0604	0.0495	0.1316	0.0535	2	2
	ODE	0.0613	0.0530	0.1167	0.0565	2	2
	BP1	0.2460	0.1875	0.4320	0.1885	4	4
	BP2	0.3915	0.1305	0.5600	0.1575	5	5
6	ABCDE	0.0970	0.0605	0.1500	0.0665	1	1
	ABC	0.1170	0.0830	0.1681	0.0531	1	1
	ODE	0.0940	0.0520	0.1319	0.0719	1	1
	BP1	0.3966	0.1415	0.4165	0.1462	5	5
	BP2	0.2720	0.1160	0.3308	0.0985	4	4

TABLE 2 AVERAGE RANKS OF THE FORECASTING ALGORITHMS FOR SIX DATA SETS (LOWER IS BETTER).

Algorithm	*ABCDE*	*ABC*	*ODE*	*BP1*	*BP2*
Rank (Train)	1.17	2.33	1.33	4.17	4.17
Rank (Test)	1.17	2.17	1.67	4.33	4.17

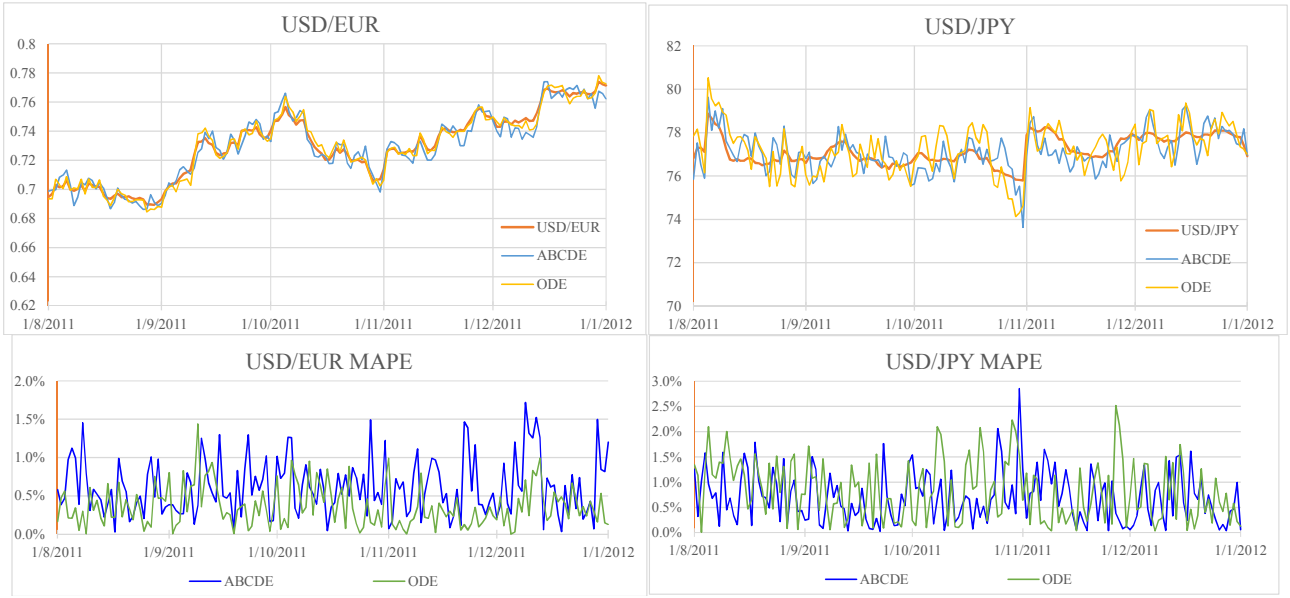

Fig. 4. Comparison of test results with actual prices (up) and errors (down) for data sets 1 (left) and 3 (right) using ABCDE and ODE.

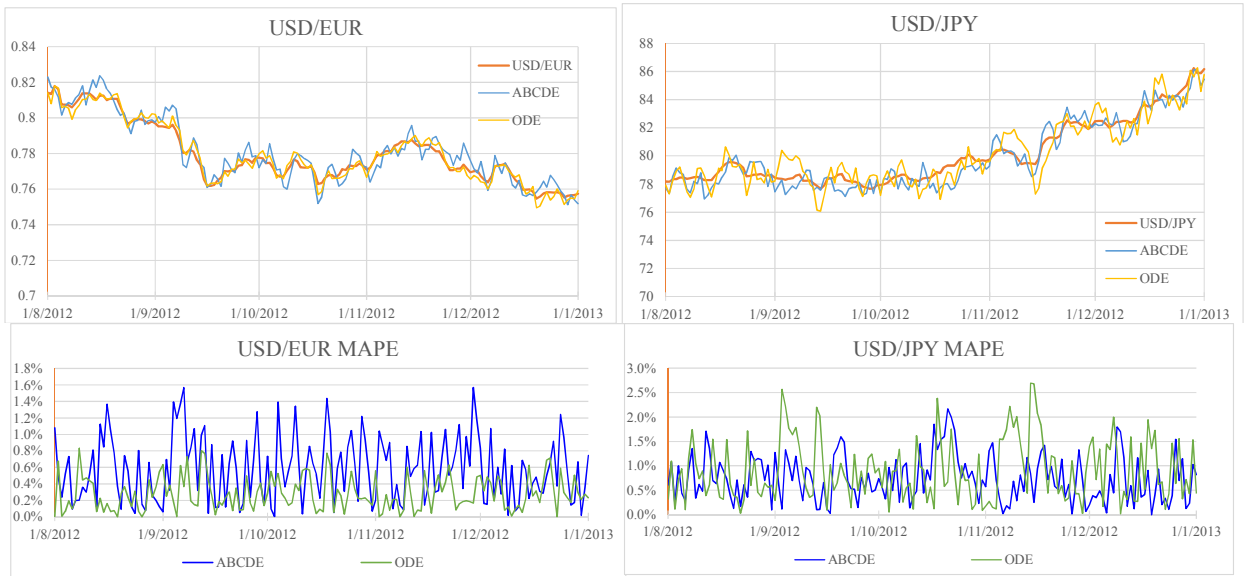

Fig. 5. Comparison of test results with actual prices (up) and errors (down) for data sets 2 (left) and 4 (right) using ABCDE and ODE.

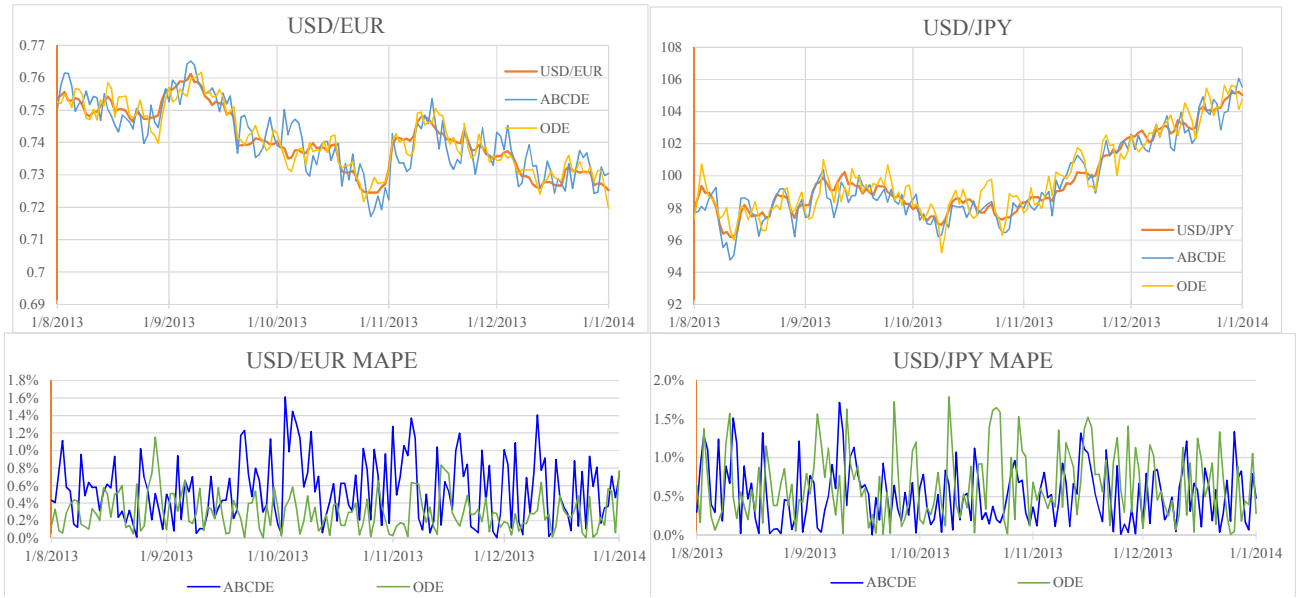

Fig. 6. Comparison of test results with actual prices (up) and errors (down) for data sets 3 (left) and 6 (right) using ABCDE and ODE.