ABSTRACT

Recently, Particle Swarm Optimization (PSO) has evolved as a promising alternative to the standard backpropagation (BP) algorithm for training Artificial Neural Networks (ANNs). PSO is advantageous due to its high search power, fast convergence rate and capability of providing global optimal solution. In this paper, the authors explore the improvements in forecasting accuracies of feedforward as well as recurrent neural networks through training with PSO. Three widely popular versions of the basic PSO algorithm, viz. Trelea-I, Trelea-II and Clerc-Type1 are used to train feedforward ANN (FANN) and Elman ANN (EANN) models. A novel nonlinear hybrid architecture is proposed to incorporate the training strengths of all these three PSO algorithms. Experiments are conducted on four real-world time series with the three forecasting models, viz. Box-Jenkins, FANN and EANN. Obtained results clearly demonstrate the superior forecasting performances of all three PSO algorithms over their BP counterpart for both FANN as well as EANN models. Both PSO and BP based neural networks also achieved notably better accuracies than the statistical Box-Jenkins methods. The forecasting performances of the neural network models are further improved through the proposed hybrid PSO framework.

Keywords: Artificial Neural Network (ANN), Box-Jenkins Model, Elman Network, Hybrid Model, Particle Swarm Optimization (PSO), Time Series Forecasting

1. INTRODUCTION

During the past two decades, Artificial Neural Networks (ANNs) have attracted overwhelming attentions in the domain of time series modeling and forecasting. ANNs are widely popular due to their non-linear, non-parametric, data-driven and self-adaptive nature (Zhang, 2003; Khashei & Bijari, 2010). Many traditional forecasting methods often suffer from one or more major limitations. For example, the well-known Box-Jenkins models (Box & Jenkins, 1970) solely require that the associated time series is linear in nature which is often rare for real-world data. In contrast, the usual nonlinear forecasting models have high mathematical complexity and they explicitly depend on the knowledge of the intrinsic data distribution process (Hamzaçoğlu,
et al., 2009). However, ANNs have the surprising ability of modeling both linear as well as nonlinear time series without requiring any preliminary information. ANNs adaptively learn from the successive training patterns, utilize the acquired knowledge to formulate an adequate model and then generalize the experience to forecast future observations. Additionally, ANNs are universal approximators, i.e. they can approximate any continuous function to any desired degree of precision (Hornik, et al., 1989). These distinctive features make them more general as well as flexible than many other traditional forecasting models. In spite of these unique strengths, the designing of an appropriate ANN model is in general quite tedious and needs a variety of challenging issues to resolve (Zhang, et al., 1998; Zhang, 2003). The most crucial among them is the selection of an appropriate training algorithm. The ANN training is an unconstrained nonlinear minimization problem and so far the Backpropagation (BP) is the most recognized method in this regard. However, the standard BP algorithm requires large computational time, has slow convergence rate and often gets stuck into local minima (Zhang, et al., 1998; Kamruzzaman, et al., 2006). These inherent drawbacks could not be entirely eliminated in spite of the development of several modifications or alterations of the BP algorithm in literature.

In recent years, the Particle Swarm Optimization (PSO) technique (Kennedy & Eberhart, 1995; Kennedy, et al., 2001) has gained notable popularity in the field of nonlinear optimization. PSO is a population based evolutionary computation method which is originally inspired from the social behavior in birds flock. The central aim of the PSO algorithm is to ultimately cluster all swarm particles in the vicinity of the desired global optimal solution. It is governed by the principle that the individual members in a social system are benefited from the intelligent information which iteratively emerges through the cooperation of all members (Jha, et al., 2009). Although, PSO and Genetic Algorithm (GA) have many similarities but they do differ on some fundamental aspects. GA depends on Darwin’s theory of survival of the fittest, whereas PSO is based on the principle of evolutionary computing (Leung, et al., 2003; Jha, et al., 2009). In GA, the population size successively decreases through eliminating the weakest solutions; however, the population size in PSO remains constant throughout. The two central operations of GA, viz. crossover and mutation do not exist in PSO. In a similar manner, the concept of personal and global best positions which is fundamental to PSO is irrelevant in GA. Over the past few years, PSO has found prolific applications for neural network training due to its many influential properties, e.g. high search power in the state space, fast convergence rate and ability of providing global optimal solution (Jha, et al., 2009; Chen, et al., 2011). Evidently, PSO can be a much better alternative to the standard BP training method. However, till now the existing literature on PSO based neural networks for time series forecasting is quite scarce and as such this topic surely needs more research attentions (Chen, et al., 2006; Jha, et al., 2009; Adhikari & Agrawal, 2011).

The aim of the present article is twofold. First, to train two different neural network structures, viz. the feedforward ANN (FANN) and the recurrent ANN of the Elman type (EANN) through the standard BP as well as three variants of the PSO algorithm, viz. PSO Trelea-I (Trelea, 2003; Jha, et al., 2009), PSO Trelea-II (Trelea, 2003; Jha, et al., 2009) and PSO Clerc-Type1 (Clerc & Kennedy, 2002) in order to forecast univariate time series. Second, to integrate the forecasting outputs from the three PSO based neural networks through an intelligent nonlinear hybrid mechanism. To the best of our knowledge, till now the PSO methodology has not been applied for training EANN models. Four real-world time series are considered in our empirical analysis. The Mean Absolute Error (MAE) and the Mean Squared Error (MSE) are used as the performance measures. The forecasting accuracies of the FANN and EANN models, trained with the PSO techniques are compared with those trained with the standard BP. We find it customary to further compare
the neural network results with those from the traditional Box-Jenkins models for all the four time series.

The remaining of the article is organized as follows. The three classes of time series forecasting models, viz. Box-Jenkins, FANN and EANN are described in section 2. The PSO algorithm with its three variants is discussed in section 3. Section 4 presents the details of our proposed nonlinear hybrid PSO architecture. The empirical results are reported in section 5 and finally the article is concluded in section 6.

2. TIME SERIES FORECASTING MODELS

In this section, we discuss about the three time series forecasting models, viz. Box-Jenkins, FANN and EANN which are used in this paper. Our primary focus is on concisely presenting the implementation related issues of these forecasting models rather than on a comprehensive study about them.

2.1. Box-Jenkins Models

The Box-Jenkins models are the outcome of the pioneering works of Box and Jenkins (1970) on time series modeling and forecasting. These are a class of most popular and widely used statistical forecasting methods. These models are based on the assumption that each future value of a time series is a linear function of several past values of the series together with random errors. A classical Box-Jenkins model is commonly known as the Autoregressive Integrated Moving Average (ARIMA) model and is denoted as ARIMA($p$, $d$, $q$). It is composed of three basic processes: autoregressive process of order $p$ or AR($p$), differencing process of order $d$ or I($d$) and moving average process of order $q$ or MA($q$). Mathematically, an ARIMA($p$, $d$, $q$) model is given by:

$$
\phi(L)(1-L)^d y_t = \theta(L) \varepsilon_t,
$$

where, $y_t$ and $\varepsilon_t$ are the actual observation and random error at time $t$ respectively and

$$
\phi(L) = 1 - \sum_{i=1}^{p} \phi_i L^i, \quad \theta(L) = 1 + \sum_{i=1}^{q} \theta_i L^i, \quad L y_t = \varepsilon_t.
$$

(2)

The pair ($p$, $q$) is commonly referred as the order of the ARIMA model. The random error terms $\varepsilon_t$ are independent and identically distributed (i.i.d.) normal variables with a zero mean and a constant variance. The ARIMA model transforms a nonstationary time series to a stationary one through a series of differencing processes. Usually, a differencing of more than unit order is seldom required. A special case is the ARIMA(0, 1, 0), i.e. $y_t - y_{t-1} = \varepsilon_t$ which is known as the Random Walk (RW) model and is often used in forecasting of financial data (Zhang, 2003).

For seasonal time series forecasting, Box and Jenkins (1970) generalized the basic ARIMA model to the Seasonal ARIMA (SARIMA) which adopts an additional seasonal differencing process in order to remove the effect of seasonality from the time series. This model is commonly denoted as SARIMA($p$, $d$, $q$)$_s$($P$, $D$, $Q$)$_s$ and is mathematically given by:

$$
\phi_p(B) \Phi_P(B^s) W_t = \theta_q(B) \Theta_Q(B^s) Z_t.
$$

(3)

Here, $s$ is the period of seasonality; $B$ is the lag or backshift operator, defined as $By_t = y_{t-s}$; $\phi_p, \Phi_P, \theta_q, \Theta_Q$ are the lagged polynomials in $B$ of orders $p$, $P$, $q$ and $Q$, respectively; $Z_t$ is a series of purely random errors and $W_t$ is the stationary nonseasonal series which is obtained after the differencing processes, i.e.

$$
W_t = (1 - B)^d \left(1 - B^s\right)^D y_t.
$$

(4)

The success of a Box-Jenkins model depends on the appropriate selection of the model order and degree of differencing. For
this purpose, Box and Jenkins proposed a robust methodology which consists of the three steps: (a) model identification, (b) parameter estimation, and (c) diagnostic checking (Box & Jenkins, 1970). These three steps attempt to rigorously select the most appropriate model by repeatedly testing and filtering a wide variety of relevant models (Zhang, 2003; Khashei & Bijari, 2010). This Box-Jenkins model building methodology is schematically depicted in Figure 1.

2.2. Feedforward Artificial Neural Network (FANN)

ANNs are a class of effective computational intelligence models which have been originally inspired from the biological structure of neural system in human brain. The most recognized ANN model for time series forecasting is the Multilayer Perceptron (MLP) (Zhang, 2003; Hamzaçebi, et al., 2009; Khashei & Bijari, 2010). A typical MLP consists of an input layer, one or more hidden layers and an output layer which together form a feedforward ANN (FANN) architecture. The nodes in each layer are connected to those in the immediate next layer through non-reverting links. It was proved by Cybenko (1989) that a single hidden layer is sufficient for approximating any continuous function to any given degree of accuracy. Due to this celebrated result, MLP structures with single hidden layer are used in most applications.

In the FANN with \( p \) input, \( h \) hidden and a single output node, the relationship between the inputs \( y_t(i=1, 2, \ldots, p) \) and the output \( y_t \) is given by the formula:

\[
y_t = G\left(\alpha_0 + \sum_{j=1}^{h} \alpha_j F\left(\sum_{i=1}^{p} \beta_{ij} y_{t-i}\right)\right).
\]

where, \( \alpha_j, \beta_{ij} (i=1, 2, \ldots, p; j=1, 2, \ldots, h) \) are the connection weights, \( \alpha_0, \beta_{0j} \) are the bias terms and \( F, G \) are the network activation functions.

---

**Figure 1. Flowchart of the three step iterative Box-Jenkins model building methodology**

- Postulate a general class of Box-Jenkins models
- **Model identification**
  - Identify the model which can be tentatively entertained
  - **Parameter estimation**
    - Estimate parameters of the tentatively entertained model
  - **Diagnostic checking**
    - Is the estimated model adequate for the problem?
      - **Yes**
        - Using this as the final model generate the required forecasts
      - **No**
        - Continue with model identification
International Journal of Applied Evolutionary Computation, 4(3), 75-90, July-September 2013

79

Usually, the logistic and identity functions, as defined in Equations (6) and (7) are used for $F$, $G$ respectively.

$$F(x) = \frac{1}{1 + \exp(-x)}$$  \hspace{1cm} (6)

$$G(x) = x$$ \hspace{1cm} (7)

The model, presented through Equation (5) is commonly referred as the $p \times h \times 1$ FANN model, a pictorial depiction of which is shown in Figure 2.

Selection of the appropriate network structure is crucial for the success of the FANN model. The number of input nodes corresponds to the number of lagged observations used to determine the inherent autocorrelation structure of a time series. On the other hand, the number of output nodes corresponds to the forecasting horizon of the associated problem and is relatively easy to specify (Zhang, et al., 1998). The selection of an appropriate number of hidden nodes is in general a complicated task. Too few may lead to poor estimation of the data generation process, whereas too much may cause the problem of overfitting and poor generalization (Kihoro, et al., 2004). For a prudent network structure with reasonably good forecasting ability, it is preferable to have a small number of hidden nodes. In the present article, we use the well-known Bayesian Information Criterion (BIC) (Faraway & Chatfield, 1998; Kihoro, et al., 2004) for determining the suitable network architecture. The BIC is mathematically defined as:

$$\text{BIC} = N_{p,h} \ln(n) + n \ln\left(\frac{S(W)}{n}\right)$$ \hspace{1cm} (8)

where, $N_{p,h} = h(p+2)+1$ is the number of total network parameters, $n = N_{\text{train}} - p$ is the number of effective observations, $N_{\text{train}}$ being the size of the training set and $S(W)$ is the network misfit function, $W$ being the space of all connection weights and biases. The function $S(W)$ is usually chosen to be the within-sample Sum of Squared Error (SSE) between the actual and forecasted observations. Out of several network structures with different input and hidden nodes, the one that minimizes the value of BIC is chosen to be the optimal one.

After selecting the network structure, the next phase in ANN modeling is training in which

Figure 2. A typical $p \times h \times 1$ FANN model with the slant arrows representing the input and hidden layer connection weights
the optimal weights and biases are iteratively determined with the goal of minimizing a misfit function, e.g. the SSE. So far, the standard BP is the most common training algorithm. In a time series forecasting problem with the available dataset \( \{y_1, y_2, ..., y_N\} \), a \( p\times h \times 1 \) ANN model consists of \((N-p)\) training patterns with the input vectors \( Y_i = [y_i, y_{i+1}, y_{i+2}, ..., y_{i+p-1}] \) and the targets \( y_{i+p} \) \((i=1, 2, ..., N-p)\). The backpropagation (BP) algorithm successively updates the network weights and biases in order to minimize the SSE (Zhang, et al., 1998; Jha, et al., 2009):

\[
E(W) = \frac{1}{2} \sum_{i=p+1}^{N} (y_i - \hat{y}_i)^2
\]

where, \( y_i \), \( \hat{y}_i \) are respectively the actual and target values. The factor \( \frac{1}{2} \) is included to simplify the mathematical computations during the training process.

The BP algorithm starts with an initial vector \( w_0 \) of weights and biases and then iteratively updates it at each step (epoch) \( i \) through the gradient descent rule:

\[
\Delta w_i = -\eta \nabla E(\mathbf{w}) \bigg|_{\mathbf{w}=w_i} + \alpha \Delta w_{i-1} \\
\mathbf{w}_i = \mathbf{w}_{i-1} + \Delta w_i
\]

where, \( \alpha \) and \( \eta \) are respectively the learning rate and momentum factor. The training stops when some predefined value of the minimum error or the maximum epoch is reached.

Despite its simplicity and immense popularity, the BP algorithm has a number of limitations which include large computational time, slow convergence rate, complex pattern of the error surface and getting stuck at local minima (Zhang, et al., 1998; Jha, et al., 2009; Khashei & Bijari, 2010). Although, several modifications of the standard BP algorithm have been suggested in literature but at present none of them can overcome all these drawbacks. To preserve impartiality in training algorithm selection, in this paper we train each neural network with five different versions of the BP algorithm and then average their forecasting outputs. These five training algorithms are: Levenberg-Marquardt (LM) (Hagan & Mentahj, 1994), Resilient Propagation (RPROP) (Reidmiller & Braun, 1993), Scaled Conjugate Gradient (SCG) (Moller, 1993), One Step Secant (OSS) (Battiti, 1992) and Broyden-Fletcher-Goldfarb-Shanno (BFGS) (Dennis & Schnabel, 1983).

### 2.3. Elman Artificial Neural Network (EANN)

The EANN is a recurrent neural network which is in fact a popular variation of the common feedforward architecture. An EANN differs from the FANN by including a new context layer and feedback connections, as can be seen in Figure 3. During the network related operations, the outputs of the hidden layer are again fed back to the context layer at each step. This recurrence provides dynamical properties to the network and makes it able to perform nonlinear time-varying mappings of the associated nodes (Lim & Goh, 2005).

There seems to be no general guidelines available in literature for selecting the appropriate EANN architecture. However, it is well-known that an EANN needs more hidden nodes than their FANN counterparts in order to properly model the temporal relationships (Lemke & Gabrys, 2010). In our empirical analysis, we use 15 hidden nodes for fitting all EANN models. Moreover, the algorithms which require large step sizes, e.g. LM or RPROP are not recommended for training recurrent networks as they cannot properly recognize the spatial and temporal patterns in the input data (Demuth, et al., 2010). In this paper we use \texttt{traingdx} as the BP training algorithm for EANN models.

### 3. PARTICLE SWARM OPTIMIZATION METHOD

The Particle Swarm Optimization (PSO) is a parallel evolutionary computation technique, developed by Kennedy and Eberhart (1995) for optimizing continuous multidimensional
functions. The PSO algorithm is originally motivated by the behavior of birds flock and it adopts an intelligent procedure to iteratively approach the desired optimal solution. The algorithm starts with a predefined number of particles in the population (also known as the swarm) which are initialized with randomized positions and velocities. Each particle evaluates a fitness function (which is commonly of SSE type) for its own. The particles are successively moved towards the location of their individual best fitness achieved so far as well as towards the global best fitness achieved across the whole swarm. In this manner, the whole swarm eventually clusters around the desired global optimal solution in the search space (Kennedy & Eberhart, 1995; Jha, et al., 2009; Adhikari & Agrawal, 2011).

Due to its various influential characteristics, PSO has been effectively applied in literature for neural network training. Suppose that we have a $p \times h \times 1$ neural network structure and $N_{\text{swarm}}$ be the swarm size. Then, each particle represents an individual ANN with dimension $D = h(p+2)+1$, which is the number of total network parameters. With this setting, the PSO algorithm updates the velocity and position of each particle as follows:

$$
\begin{align*}
    v_{id}(t+1) &= a v_{id}(t) + b_1 r_1 (p_{id} - x_{id}(t)) + b_2 r_2 (p_{gd} - x_{id}(t)), \\
    x_{id}(t+1) &= x_{id}(t) + v_{id}(t+1), \\
    \forall i = 1, 2, \ldots, N_{\text{swarm}}; \forall d = 1, 2, \ldots, D
\end{align*}
$$

Figure 3. The structure of an Elman ANN model; the solid arrows represent the feedforward connections, whereas the dashed arrows represent the feedback connections

where, $x_{id}$ and $v_{id}$ are respectively the position and velocity of the $i^{th}$ particle at the $d^{th}$ dimension; $p_{id}$ and $p_{gd}$ are respectively the personal and global best positions, achieved so far at the $d^{th}$ dimension; $b_1$ and $b_2$ are the acceleration coefficients which control the movements of the particles; $r_1$ and $r_2$ are two uniform random variables in the $[0, 1]$ interval and $a$ is the inertia weight. The position of each particle is calculated through a particular fitness function which is the SSE in our case. The process of updating positions
and velocities in the PSO algorithm continues until some stopping criterion, e.g. the maximum number of iterations or the maximum increase in validation error is reached (Jha, et al., 2009; Adhikari & Agrawal, 2011). The flowchart of the PSO algorithm is shown in Figure 4.

Numerous variations of the basic PSO algorithm have been developed in literature. In this article, we consider the versions suggested by Trelea (2003) and Clerc (Clerc & Kennedy, 2002).

### 3.1. Trelea’s Modified PSO Algorithm

Trelea (2003) provided a modified deterministic version of the basic PSO algorithm and carried out a comprehensive study for the appropriate selection of the associated tuning parameters. The formulae for updating positions and velocities in this version of PSO are given by:

\[
\begin{align*}
    v_{id}(t + 1) &= av_{id}(t) + b(p_d - x_{id}(t)) \\
    x_{id}(t + 1) &= x_{id}(t) + v_{id}(t + 1)
\end{align*}
\]  \hspace{1cm} (12)

where,

\[
    r_1 = r_2 = \frac{1}{2}, \quad b = \frac{b_1 + b_2}{2}; \quad p_d = \frac{b_1}{b_1 + b_2} p_{id} + \frac{b_2}{b_1 + b_2} p_{gd}.
\]

After a lot of empirical analysis, Trelea suggested two sets of values for the parameters \(a\) and \(b\) which are: \(a=0.6, b=1.7\) and \(a=0.729, b=1.494\). The PSO algorithms, corresponding to these two sets of parameters are commonly referred as PSO Trelea-I and PSO Trelea-II respectively.

*Figure 4. The flowchart of the basic PSO algorithm*
3.2. Clerc's Constricted PSO Algorithm

The basic PSO algorithm, presented through Equations (11) is unstable and the particles sometimes scatter haphazardly from the global optimal location. To prevent this undesirable behavior, Clerc and Kennedy (2002) suggested the use of a constriction factor which is given by:

\[
\chi = \begin{cases} 
\frac{2\kappa}{\phi - 2 + \sqrt{\phi^2 - 4\phi}} & \text{if } \phi \geq 4 \\
\kappa & \text{if } 0 < \phi < 4
\end{cases}
\] (13)

where, \(0 < \kappa < 1\) and \(\phi = b_1 + b_2\). Using this constriction factor, the velocity updating becomes:

\[
v_{id}^{\text{new}}(t + 1) = \chi[v_{id}(t + 1)]
\] (14)

where, \(v_{id}(t+1)\) is the updated velocity according to the original PSO algorithm.

The above formulation is known as the PSO with Clerc-Type1 constriction. The constriction factor \(\kappa\) in Equation (13) controls the velocities of the particles. In practical applications, often the values \(\phi = 4\) and \(\kappa = 0.729\) are used.

4. THE PROPOSED HYBRID MECHANISM

As mentioned in the outset, the second objective of this paper is to derive a hybrid mechanism for integrating the outputs from the three PSO based neural networks. A combination of forecasts from different models remarkably improves the overall accuracy and also often outperforms the constituent models (Lemke & Gabrys, 2010). Here, we extend the usual linear combination framework in order to consider the possible interactions between two forecasts. The related mathematical description follows next.

Let, \(Y = [y_1, y_2, \ldots, y_N]^T\) be the actual testing dataset of the given time series and \(\hat{Y}^{(i)} = [\hat{y}_1^{(i)}, \hat{y}_2^{(i)}, \ldots, \hat{y}_N^{(i)}]^T\) for \(i = 1, 2, 3\) be the forecasts obtained through the three PSO based neural network models. Without loss of generality, we suppose that \(i=1, 2, 3\) respectively refers to the ANN model trained with PSO Trelea-I, PSO Trelea-II, and PSO Clerc-Type1. Then our proposed hybrid mechanism evaluates the combined forecasts as:

\[
\hat{y}_k^{(c)} = w_m \bar{y}_k^{(m)} + \theta_1 |\hat{y}_k^{(1)} - \hat{y}_k^{(2)}| + \theta_2 |\hat{y}_k^{(2)} - \hat{y}_k^{(3)}| + \theta_3 |\hat{y}_k^{(3)} - \hat{y}_k^{(1)}| \\
\forall k = 1, 2, \ldots, N
\] (15)

where, \(\bar{y}_k^{(m)}\) is the mean of \(\hat{y}_k^{(i)}\) for \(i = 1, 2, 3\). Equivalently, we can write Equation (15) as:

\[
\hat{Y}^{(c)} = Fw + G\theta
\] (16)

where, we have as presented in Box 1.

It can be observed from Equations (15) and (16) that in our proposed combination scheme:

Combined forecast = bias \\
+weight \times \{mean of individual forecasts\} \\
+weights \times \{absolute difference between forecasts pairs\}

A similar kind of nonlinear model combination framework has been recently suggested by Adhikari and Agrawal (2012) in which the combined forecast is an aggregation of individual forecasts as well as the correlation between pairs of forecasts. The optimal combination weights can be determined by minimizing the forecast SSE, given by:
Evaluating the partial derivatives \( \frac{\partial}{\partial w} (\text{SSE}) = 0 \) and \( \frac{\partial}{\partial \theta} (\text{SSE}) = 0 \), we can reach at the following system of linear equations:

\[
\begin{bmatrix}
    F^T F & F^T G \\
    G^T F & G^T G
\end{bmatrix}
\begin{bmatrix}
    w \\
    \theta
\end{bmatrix}
= 
\begin{bmatrix}
    F^T Y \\
    G^T Y
\end{bmatrix}
\] (18)

A solution of the linear system in Equation (18) (provided it is solvable) gives the optimal values of the weight vectors \( w \) and \( \theta \). From this matrix equation, we also note that the optimal weights depend on the original testing dataset \( Y \) which is actually unknown in advance. In order to resolve this issue, we adopt a successive validation mechanism which determines the optimal weights solely from the available observations of the time series. The proposed mechanism is as follows:

We divide the available in-sample dataset into a predefined number of pairs of training and validation sets. For each pair, the neural network model is trained on the training set with the three PSO variants. Using these three ANNs, the corresponding validation set is forecasted and the matrices \( F, G \) are calculated. Then from Equation (18), the weight vectors \( w \) and \( \theta \) are determined by replacing the original test dataset \( Y \) with the validation dataset. This process is continued until the end of the total validation experiments is reached. The final optimal weights are then calculated as the arithmetic means of the corresponding optimal weights of all pairs. The necessary details of our proposed hybrid scheme are concisely presented below.

### Table 1. Information regarding the four time series datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Size</th>
<th>Box-Jenkins model</th>
<th>FANN model</th>
<th>EANN model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunspots</td>
<td>Total: 288 Testing: 67</td>
<td>ARIMA (9,0,0)</td>
<td>11×9×1</td>
<td>11×15×1</td>
</tr>
<tr>
<td>Exchange</td>
<td>Total: 681 Testing: 116</td>
<td>Random Walk</td>
<td>6×6×1</td>
<td>6×15×1</td>
</tr>
<tr>
<td>Earthquake</td>
<td>Total: 99 Testing: 29</td>
<td>Random Walk</td>
<td>11×7×1</td>
<td>11×15×1</td>
</tr>
<tr>
<td>Airline</td>
<td>Total: 144 Testing: 36</td>
<td>SARIMA (0, 1, 1)×(0, 1, 1)(^2)</td>
<td>12×3×1</td>
<td>12×15×1</td>
</tr>
</tbody>
</table>
Let, \( Y = \begin{bmatrix} y_1; y_2; \ldots; y_{N_{\text{train}}} \end{bmatrix}^T \) be the available in-sample training dataset of size \( N_{\text{train}} \) be the size of the validation set and \( no_{\text{val}} \) be the total number of validations. Also, let \( W \) and \( \Theta \) be the weight matrices of size \( 2 \times N_{\text{val}} \) and \( 3 \times N_{\text{val}} \) respectively. Then the following steps are carried out:

**Step 1:** Set \( W = [] \) and \( \Theta = [] \) as the empty matrices, and the number of iterations \( i = 1 \).

**Step 2:** Calculate \( \alpha = N_{\text{train}} - N_{\text{val}} - no_{\text{val}} + i \). Divide the available in-sample dataset into a pair of training and validation subsets \( Y_{\text{train}} \) and \( Y_{\text{val}} \) respectively.

\[
Y_{\text{train}} = \begin{bmatrix} y_1; y_2; \ldots; y_{\alpha} \end{bmatrix}^T \\
Y_{\text{val}} = \begin{bmatrix} y_{\alpha+1}; y_{\alpha+2}; \ldots; y_{\alpha+N_{\text{val}}} \end{bmatrix}^T
\]

**Step 3:** Train the neural network model on \( Y_{\text{train}} \) with the three variants of the PSO algorithm, viz. Trelea-I, Trelea-II and Clerc-Type1.

**Step 4:** Use these three trained neural network models to forecast the dataset \( Y_{\text{val}} \) and then calculate the matrices \( F \) and \( G \).

**Step 5:** Replace the original testing dataset in Equation (18) with \( Y_{\text{val}} \) and determine the \( i^{\text{th}} \) optimal weight vectors \( w_i \) and \( \theta_i \). Augment these weights to the corresponding weight matrices as \( W = [W \mid w_i] \) and \( \Theta = [\Theta \mid \theta_i] \). Then make a unit increment to the iteration number \( i \).

**Step 6:** If \( i = no_{\text{val}} \) then stop the process, else repeat all the steps from step 2 to step 5.

**Step 7:** Calculate the desired weight vectors as:

\[
w_{\text{comb}} = \text{mean}(W, \text{row-wise}) \\
\theta_{\text{comb}} = \text{mean}(\Theta, \text{row-wise})
\]

**Step 8:** Use \( w_{\text{comb}} \) and \( \theta_{\text{comb}} \) in Equation (16) to find the final combined forecast vector.

The above algorithm provides a robust method for finding the optimal combination weight vectors. In our empirical analysis, we set \( N_{\text{val}} \) to be equal to the size of the testing dataset and \( no_{\text{val}} \) to be equal to 10 for all four time series.

### 5. EMPIRICAL RESULTS

Four real-world time series of different natures are selected from the well-known Time Series Data Library (TSDL) repository (Hyndman, 2011) for our empirical study. These four series are: (1) Sunspots: contains annual number of observed sunspots from 1700 to 1987, (2) Exchange rate: contains the daily exchange rate between US dollars and Indian rupees from 1 July, 2009 to 16 September, 2011, (3) Earthquake: contains the number of earthquakes per year (as per the report of National earthquake information centre) of magnitude 7.0 or greater from 1900 to 1998, (4) Airline: contains the number of monthly international airline passengers (in thousands) from January, 1949 to December, 1960. The time plots of these four time series are shown in Figure 5.

The experiments are carried out in MATLAB. The ANN toolbox (Demuth, Beale, & Hagan, 2010) is used for modeling the BP based neural networks, whereas the PSO toolbox of Birge (2003) is used for implementing the PSO algorithms. Suitable data preprocessing, e.g. data normalization in [0, 1], logarithmic transformation, etc. are applied to each dataset before fitting the forecasting methods. Following past works (Jha, et al., 2009; Adhikari & Agrawal, 2011), the PSO swarm sizes are selected from the range of 24 to 30. For each dataset, the swarm size is varied within this range and subsequent PSO performances are noted on the validation set. The appropriate swarm size is then chosen to be the one for which the validation error is minimized. The forecasting performances are evaluated through MAE and MSE, defined as:

\[
\text{MAE} = \frac{1}{N} \sum_{t=1}^{N} |y_t - \hat{y}_t|
\]
\[ \text{MSE} = \frac{1}{N} \sum_{t=1}^{N} (y_t - \hat{y}_t)^2 \]  

(20)

where, \( y_t \), \( \hat{y}_t \) are respectively the actual and forecasted observations and \( N \) is the size of the testing set. For an efficient forecasting performance, the values of both these error measures are desired to be as small as possible.

The information regarding the appropriate models which are found suitable for each dataset is given in Table 1. The comparisons of the forecasting accuracies in terms of MAE and MSE are presented in Table 2.

After analyzing Table 2, the following important observations are noticed:

- The MAE and MSE values, obtained through both FANN and EANN models are notably less than those obtained through the traditional Box-Jenkins models.
- For both types of neural networks, each of the three PSO based training algorithms produced considerably less MAE and MSE values than the standard BP algorithm.
- Clearly, the best forecasting accuracies are achieved by combining the outputs from the three PSO based neural network models through our proposed nonlinear hybrid mechanism. The proposed scheme also beat the simple average combination method in terms of both the MAE and MSE values.

Figure 6 depicts the actual observations (in solid line) and their corresponding FANN forecasts (in dotted line) through our proposed hybrid framework for all four time series. The remarkable closeness between actual and forecasted series can be observed in each subplot of Figure 6.

Figure 5. Time plots of the four time series datasets
### Table 2. Comparisons of the forecasting accuracies in terms of obtained MAE and MSE values through the Box-Jenkins, FANN and EANN models; the least error measures for each dataset are shown in bold letters

<table>
<thead>
<tr>
<th>Time series</th>
<th>Sunspots</th>
<th>Exchange rate</th>
<th>Earthquake</th>
<th>Airline</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MAE</td>
<td>MSE</td>
<td>MAE</td>
<td>MSE</td>
</tr>
<tr>
<td>Box-Jenkins</td>
<td>22.96</td>
<td>911.68</td>
<td>0.372</td>
<td>0.209</td>
</tr>
<tr>
<td>BP</td>
<td>19.97</td>
<td>812.62</td>
<td>0.357</td>
<td>0.195</td>
</tr>
<tr>
<td>PSO Trelea-I</td>
<td>17.34</td>
<td>770.45</td>
<td>0.298</td>
<td>0.139</td>
</tr>
<tr>
<td>PSO Trelea-II</td>
<td>19.14</td>
<td>764.71</td>
<td>0.280</td>
<td>0.124</td>
</tr>
<tr>
<td>FANN</td>
<td>17.42</td>
<td>667.00</td>
<td>0.284</td>
<td>0.122</td>
</tr>
<tr>
<td>PSO average</td>
<td>17.70</td>
<td>640.89</td>
<td>0.227</td>
<td>0.077</td>
</tr>
<tr>
<td>PSO combined</td>
<td><strong>16.54</strong></td>
<td><strong>569.37</strong></td>
<td><strong>0.224</strong></td>
<td><strong>0.076</strong></td>
</tr>
<tr>
<td>BP</td>
<td>18.61</td>
<td>899.88</td>
<td>0.340</td>
<td>0.159</td>
</tr>
<tr>
<td>PSO Trelea-I</td>
<td>18.40</td>
<td>793.37</td>
<td>0.256</td>
<td>0.105</td>
</tr>
<tr>
<td>PSO Trelea-II</td>
<td>18.52</td>
<td>755.38</td>
<td>0.270</td>
<td>0.124</td>
</tr>
<tr>
<td>EANN</td>
<td>19.86</td>
<td>773.25</td>
<td>0.270</td>
<td>0.130</td>
</tr>
<tr>
<td>PSO average</td>
<td>17.83</td>
<td>724.71</td>
<td>0.213</td>
<td>0.070</td>
</tr>
<tr>
<td>PSO combined</td>
<td><strong>16.87</strong></td>
<td><strong>594.48</strong></td>
<td><strong>0.205</strong></td>
<td><strong>0.066</strong></td>
</tr>
</tbody>
</table>

**Figure 6.** Actual observations and their FANN forecasts through the proposed hybrid PSO framework; the solid and dotted line respectively represents the original and predicted dataset
6. CONCLUSION

ANNs are a class of powerful methods for time series forecasting. Recently, PSO has evolved as a promising technique for ANN training. In this paper, three popular variants of the basic PSO algorithm, viz. Trelea-I, Trelea-II and Clerc-Type 1 are used for training two types of ANN models: feedforward ANN (FANN) and Elman ANN (EANN) which are then used to forecast future observations. A robust nonlinear hybrid mechanism is proposed to efficiently combine the obtained outputs from these three PSO based ANN models. The empirical results, conducted on four real-world time series clearly demonstrate that each of the three PSO based training algorithms achieved reasonably better forecasting accuracies than their BP counterparts for both FANN and EANN models. The neural networks, trained with BP as well as PSO algorithms also performed notably better than the traditional Box-Jenkins models for all four time series. It is further observed that the best forecasting accuracies in all cases are obtained through our proposed PSO hybridization mechanism.

ACKNOWLEDGMENT

Both the authors are indebted to the anonymous reviewers for their timely as well as encouraging reviews. The first author also expresses his sincere gratitude to the Council of Scientific and Industrial Research (CSIR), India for the obtained financial assistance which provided indispensable help to carry out the present research work.

REFERENCES


Ratnadip Adhikari received his B.Sc degree with Mathematics Honours from Assam University, India in 2004 and M.Sc in applied mathematics from Indian Institute of Technology, Roorkee in 2006. After that he obtained M.Tech in computer science and technology from Jawaharlal Nehru University, New Delhi, India in 2009. Currently he is pursuing PhD from School of Computer and Systems Sciences, Jawaharlal Nehru University. His primary research interests include time series modeling and forecasting, data stream classification and ensemble of multiple models.
R. K. Agrawal received his M. Tech. degree in computer application from Indian Institute of Technology, Delhi. He has done his Ph.D. in computational physics from University of Delhi. Presently, he is working as an Associate professor in School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi. His current areas of research are classification, regression, clustering, feature extraction and selection for pattern recognition problems in domains of image processing, security, medical imaging and bioinformatics.