# Effectiveness of PSO Based Neural Network for Seasonal Time Series Forecasting

**Conference Paper** · January 2011

CITATIONS
32

READS
889

**2 authors:**

Ratnadip Adhikari
Fractal Analytics, Bengaluru, India
**21** PUBLICATIONS **1,101** CITATIONS
SEE PROFILE

R. K. Agrawal
Jawaharlal Nehru University
**138** PUBLICATIONS **2,214** CITATIONS
SEE PROFILE

# Effectiveness of PSO Based Neural Network for Seasonal Time Series Forecasting

R. Adhikari and R.K. Agrawal

School of Computer and Systems Sciences,
Jawaharlal Nehru University, New Delhi-110067, India
{adhikari.ratan,rkajnu}@gmail.com

**Abstract.** Recently, the Particle Swarm Optimization (PSO) technique has gained much attention in the field of time series forecasting. Although PSO trained Artificial Neural Networks (ANNs) performed reasonably well in stationary time series forecasting, their effectiveness in tracking the structure of non-stationary data (especially those which contain trends or seasonal patterns) is yet to be justified. In this paper, we have trained neural networks with two types of PSO (Trelea1 and Trelea2) for forecasting seasonal time series data. To assess their performances, experiments are conducted on three well-known real world seasonal time series. Obtained forecast errors in terms of three common performance measures, viz. MSE, MAE and MAPE for each dataset are compared with those obtained by the Seasonal ANN (SANN) model, trained with a standard backpropagation algorithm. Comparisons demonstrate that training with PSO-Trelea1 and PSO-Trelea2 produced significantly better results than the standard backpropagation rule.

**Keywords:** Particle Swarm Optimization, Evolutionary Computation, Time Series Forecasting, ANN, Seasonal ANN.

## 1 Introduction

Time series forecasting is a dynamic research area which has important applications in various practical decision making situations. In time series forecasting, the past observations are carefully collected and rigorously studied to develop an appropriate model which in turn is used to generate future values for the series. Thus time series forecasting can be termed as the act of predicting the future by understanding the past. The most popular statistical techniques used for forecasting are the Box-Jenkins or ARIMA models [1]. Although these models can represent several varieties of time series with ease and simplicity, they make the presumption that the associated series is linear in nature. This imposed restriction makes them inadequate in many practical situations, as most of the real world time series are generated from non-linear processes [2]. To overcome this drawback, various non-linear statistical models have been proposed in literature. Some of them are: the Autoregressive Conditional Heteroskedasticity (ARCH) model [2, 3], the Generalized ARCH (GARCH) [2, 4] and the Exponential Generalized ARCH (EGARCH) [4], the Threshold Autoregressive (TAR) model [5],

the Non-linear Autoregressive (NAR) model [6], the Non-linear Moving Average (NMA) model [2], etc. However, these models have a high mathematical complexity and more importantly they depend on the explicit knowledge of the underlying data generation process of the concerned time series, which is often much difficult to predict. During the past few years, Artificial Neural Networks (ANNs) have been extensively used in time series forecasting problems. The salient feature of ANNs is their non-linear, data-driven and self-adaptive nature [7]. ANNs can perform non-linear modeling of a wide variety of time series without any prior knowledge about the inherent pattern. An excellent review on the applications of neural networks for time series forecasting can be found in the paper of Zhang et al. [8] and also of Adya and Collopy [9]. Although ANNs have provided better estimates of unknown observations in numerous forecasting problems, their accuracy heavily relies on the correct selection of the model structure and related parameters. For expecting a reasonably well forecast using neural networks, one must have to determine beforehand the appropriate network architecture, the number of intermediate layers, the number of processing elements in each layer, the significant time lags, the proper network training algorithm, etc. [7, 8]. Unfortunately, at present there is no systematic theoretical procedure to determine these optimal parameters and usually experiments are conducted for this purpose. Another critical issue is that the standard backpropagation network training algorithm shows slow convergence rate and more dangerously it often converges to a local minimum instead of the desired global one [10]. Due to these facts several improved versions of the backpropagation algorithm, such as the Levenberg-Marquardt (LM) [11], Resilient propagation(RPROP) [12], scaled conjugate gradient (SCG) [13], one step secant (OSS) [14], etc. have been proposed in the literature.

Particle Swarm Optimization (PSO) [15] is a parallel evolutionary computation technique based on the social behavior in a bird flock. The basic aim of the PSO algorithm is to guide the whole swarm towards the global optimal location by iteratively updating the two parameters, viz. position and velocity associated with each particle in the swarm [15, 16]. It is based on the principle that the individual members in a social system benefit from the intelligent information, collectively emerging from the cooperation of all the members in that system [17]. PSO shares many common properties with the Genetic Algorithm (GA) [18]. But the remarkable differences between them is that GA depends on the Darwin's theory of survival of the fittest, whereas PSO is based on evolutionary computation [10, 15, 16, 18].

Due to its high search power for the optimal location in the state space, PSO can efficiently train neural networks. PSO based neural networks have been widely used for optimization problems in many diverse fields [16, 19, 20]. However, till now this technique has not been applied much for time series forecasting. Some recognizable works in this area can be found in Jha et al. [10], Paulo et al. [21] and Chen et al. [22]. Jha et al. used a neural network model, trained by the PSO algorithm to forecast a financial time series [10]. They applied the method after making the associated series stationary. The PSO trained neural networks

produced good forecast results in their case. But, their works were silent about its efficiency in tracking non-stationarity (like trend and seasonal effect) in a time series, without removing it.

Seasonal variations in a time series are fluctuations within a year during the season. Most of the economic and business time series consist of seasonal effects. Normally, seasonality in a time series is either additive or multiplicative. The most popular statistical technique for modeling seasonal time series is SARIMA$(p, d, q) \times (P, D, Q)^s$ developed by Box and Jenkins as a generalization of the ARIMA$(p, d, q)$ model [1, 23, 24]. It applies an ordinary as well as a seasonal differencing to the time series observations in order to remove the inherent seasonality from the series, so that the new series becomes stationary. Like ARIMA$(p, d, q)$, the parameters $P$, $D$, $Q$ in the SARIMA model represent the autoregressive, degree of differencing and moving average processes, respectively and $s$ is the period of seasonality [23, 24]. Usually the degree of differencing terms $d$ and $D$ need not exceed one.

ANN has been considerably used for modeling seasonal time series; however opinions differ regarding its efficiency in recognizing seasonal effect in the data structure. According to some analysts, ANN cannot adequately track the seasonal variation in a time series without removing it [25–27]; whereas some others reported the just opposite conclusion [28–30]. Faraway and Chatfield [31] carried out an extensive study in this regard by fitting a wide variety of ANN models to the well-known monthly airline passenger dataset. Their work compared the results with those obtained from the Box-Jenkins and Holt-Winters method and from the analysis they concluded that ANN is inappropriate for seasonal time series. However, Kihoro and Otieno [32] used the same airline data together with two other datasets to show that with proper network structure and optimal model parameters ANN is quite capable in finding the seasonal pattern in a time series. Thus, the effectiveness of ANN in modeling seasonal time series is still a matter of debate.

In this paper, two variants of PSO (Trelea1 and Trelea2) are used to train multi-layer feed forward neural networks for forecasting seasonal time series. The input and output neurons are determined using the seasonal parameter of the associated series. The out-of-sample forecasting ability of the two PSO based ANN models are tested on three real world seasonal time series. The performance measures, viz. Mean Absolute Error (MAE), Mean Squared Error (MSE) and Mean Absolute Percentage Error (MAPE) are considered for evaluating the models. The prediction efficiencies of the two PSO based ANNs are compared with the Seasonal ANN (SANN) model [23], trained by a standard backpropagation algorithm.

The rest of the paper is organized as follows. Section 2 describes the ANN and SANN models. In Sect. 3 we introduce the PSO algorithm and its use for training neural network for seasonal time series forecasting. Experimental results are reported in Sect. 4 and finally in Sect. 5 we conclude the paper.

## 2 ANN for Time Series Forecasting

Artificial neural networks are a class of generalized non-linear, non-parametric, self-adaptive statistical models [33], which are originally inspired by the intelligent working paradigm of the human brain. They try to recognize regularities and patterns, present in the input data, learn from experience and then provide generalized results based on their known previous knowledge. The most widely used ANN for time series forecasting problems is the multi-layer perceptron (MLP), which uses a feed forward architecture [7, 8]. The model is characterized by a network of three layers, viz. an input layer, one or more hidden layers and an output layer. The processing elements in each layer are connected to those in the immediate next layer by acyclic links. It has been proved by Cybenko [34] that a network with one hidden layer can approximate any continuous function with desired precision, if no restrictions are placed on the number of hidden and input nodes. Due to this result, we employed a single hidden layer feed forward architecture in our study.

After defining the network structure, the output $y_t$ of the time series is computed using the following expression:

$$y_t = G\left(\alpha_0 + \sum_{j=1}^{q} \alpha_j F\left(\beta_{0j} + \sum_{i=1}^{p} \beta_{ij} y_{t-i}\right)\right) \ . \tag{1}$$

Here $y_{t-i}\,(i = 1, 2, \ldots, p)$ are the $p$ inputs; the integers $p, q$ are the number of input and hidden nodes, respectively; the constants $\alpha_j\,(j = 1, 2, \ldots, q)$ and $\beta_{ij}\,(i = 1, 2, \ldots, p; j = 1, 2, \ldots, q)$ are the connection weights with $\alpha_0$ and $\beta_0$ as the bias terms. $F$ and $G$ are respectively the hidden and output layer activation functions. Following other works [7, 10, 31], in this paper we used the logistic function for $F$ and the linear function for $G$, which are defined as:

$$F(x) = \frac{1}{1 + e^{-x}} \quad \text{and} \quad G(x) = x \ .$$

The model (1) is often referred as a $(p, q, 1)$ ANN model with $p$ input, $q$ hidden and one output node. One node in the output layer is typically used for one-step ahead forecasting. The number of network parameters is equal to the number of connections between the processing neurons and bias terms. Conceptually the ANN model (1) is a non-linear mapping of the $p$ past observations $y_{t-1}, y_{t-2}, \ldots, y_{t-p}$ to the future value $y_t$, i.e.

$$y_t = f\left(y_{t-1}, y_{t-2}, \ldots, y_{t-p}, \boldsymbol{w}\right) + \varepsilon_t \ . \tag{2}$$

Here $\boldsymbol{w}$ is a vector of network weights and $\varepsilon_t$ is the error occurred at time $t$. Thus the neural network is in fact similar to a non-linear autoregressive model. After determining the suitable network structure, the next task is to train the network, i.e. to estimate the network parameters. The training phase starts with a set of initial weights, which are iteratively modified according to the specific

learning rule. Learning in neural networks can be supervised or unsupervised [35]. In supervised learning, an input set is accompanied with a known target value, whereas in case of unsupervised learning, the target values are unknown. The most popular supervised learning rule is the standard backpropagation (BP) technique which is based on the minimization of a misfit function, known as the sum of squared error (SSE) and is defined as:

$$E(\psi) = \frac{1}{2} \sum_{t=p+1}^{N} (y_t - f_t)^2 \quad . \tag{3}$$

Here $N$ is the size of the training set, $y_t$ is the target value, $f_t$ is the predicted value and $\psi$ is the space of all connection weights and biases.

### 2.1 The Seasonal ANN(SANN) Model

The seasonal ANN (SANN) model was developed by C. Hamzacebi [23] for seasonal time series forecasting. He observed that in a seasonal time series some recognizable patterns repeat after a regular interval and so an appropriate network structure can learn the seasonal effect in the dataset without removing it. In his proposed model, the seasonal period $s$ of the series is used to represent the number of input and output neurons. Thus, the first $s$ observations $y_{t-i}\,(i = 0, 1, 2, \ldots, s-1)$ are the inputs to the neural network and the next $s$ observations $y_{t+j}\,(j = 1, 2, \ldots, s)$ are the corresponding targets and so on.

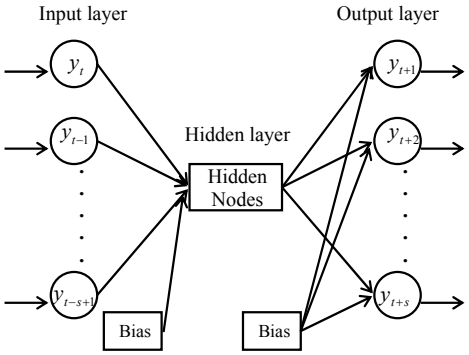


**Fig. 1.** Structure of a SANN model

Basically, the SANN model represents a $(s, h, s)$ neural network structure, where $h$ is the number of hidden nodes. The number of output nodes $s$ signifies a forecasting horizon of $s$-step ahead. For implementation, this model is quite simple as well as straight forward because the adequate network structure is already specified, which reduces a lot of burden for the user; only an appropriate number of

hidden nodes is to be determined, which can be done experimentally. Hamzacebi had empirically verified the superiority of SANN over other traditional statistical models, by applying it on four practical seasonal time series [23]. Due to its efficiency and simplicity as mentioned above, the $(s, h, s)$ structure is used as our base ANN in this paper. The proper number of hidden nodes for each dataset is determined by validation processes.

## 3 The Particle Swarm Optimization Algorithm

The Particle Swarm Optimization (PSO) [15, 16] is a population based search technique for optimizing continuous non-linear as well as multi-dimensional functions. It starts by randomly initializing the position and velocity of a number of particles in the population (also known as *swarm*). At the successive iterations, each particle is moved through the search space towards the location of its own personal best position achieved so far as well as the global best position discovered by any of the particles in the swarm [36]. In this manner the whole swarm is iteratively guided towards the location of the desired global optimal solution in the search space.

During the recent years PSO has been proficiently used to train neural network, i.e. to optimize the network parameters. Consider a $(s, h, s)$ neural network structure with $N$ particles in the swarm. Then, each particle represents an individual ANN. The dimension of every particle is $D = h(2s + 1) + s$, the total number of network parameters. Every particle $i\,(1 \leq i \leq N)$ in the swarm is associated with a current position $x_i$ and a current velocity $v_i$. The PSO algorithm begins by assigning randomized positions and velocities to the $N$ particles. The particles are moved through the $D$-dimensional search space until an average sum-squared error criterion per ANN processing unit is met. Each particle evaluates a fitness function (error function in this study) to find its own fitness value. The movement of the particles is based on two best positions, viz. the personal and global best. The personal best position of the $i^{\text{th}}$ particle is denoted by $p_i$ which indicates the best fitness achieved by the particle till now. The global best position, denoted by $p_g$ indicates the best fitness achieved so far across the whole population. Using these two values, the velocity and position for the $d^{\text{th}}$ dimension $(1 \leq d \leq D)$ of the $i^{\text{th}}$ particle are updated as follows:

$$v_{id}(t+1) = a v_{id}(t) + b_1 r_1 \left(p_{id} - x_{id}(t)\right) + b_2 r_2 \left(p_{gd} - x_{id}(t)\right) \ . \tag{4}$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \ . \tag{5}$$

Here $x_{id}$ and $v_{id}$ are respectively the position and velocity of the $i^{\text{th}}$ particle at the $d^{\text{th}}$ dimension; $p_{id}$ and $p_{gd}$ are respectively the personal and global best positions, achieved at the $d^{\text{th}}$ dimension; $b_1$ and $b_2$ are the acceleration coefficients to control the movement of the particle; $r_1$ and $r_2$ are two uniform random numbers in the interval $[0, 1]$ and $a$ is the inertia weight. The process of updating velocity and position of swarm elements continues, until some predefined

stopping criterion, such as the maximum number of iterations or the maximum increase in the validation error is reached [21].

The tuning parameters associated with the PSO algorithm greatly influence its performance, commonly stated as the exploration-exploitation tradeoff [37]. Efforts have been made by researchers for proper selection of the PSO parameters; as a result, several modified versions of the algorithm have been developed in literature. In this paper, we employed the deterministic PSO algorithm, proposed by I. Trelea [37]. Following the work of Trelea, we have:

$$r_1 = r_2 = \frac{1}{2};\ b = \frac{b_1 + b_2}{2};\ p_d = \frac{b_1}{b_1 + b_2} p_{id} + \frac{b_2}{b_1 + b_2} p_{gd} \ .$$

Using these co-efficients, (4) and (5) can be rewritten as:

$$v_{id}(t+1) = a v_{id}(t) + b\,(p_d - x_{id}(t)) \ . \tag{6}$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \ . \tag{7}$$

Equations (6) and (7) are the expressions for updated velocity and position in the deterministic PSO algorithm. After a lot of simulation experiments, Trelea suggested two sets of parameters for $a$ and $b$, which are commonly known as PSO Trelea1 ($a = 0.6, b = 1.7$) and PSO Trelea2 ($a = 0.729, b = 1.494$). These two types of PSO algorithms are applied in our study.

## 4 Experimental Results and Discussions

Three well-known seasonal time series from literature were used to empirically examine the forecasting ability of PSO based neural networks. These are the monthly airline passenger time series, the quarterly sales time series and the quarterly USA beer production time series. Following other studies [21, 23], we divided each dataset into three parts, viz. training, validation and test set. The validation set is used to select the best network model from several ANN structures; the training set is used for parameter estimation and the test set is kept for assessing the out-of-sample forecast performance of the fitted model. In this study, all experiments were performed using MATLAB. The neural network models were fitted through the ANN toolbox [38] while the PSO algorithms were implemented through the PSO toolbox, developed by Birge [39]. The RPROP method, proposed by Reidmiller and Braun [12] was used as the standard back-propagation algorithm for training SANN models, due to its computational simplicity, robustness and fast convergence rate. Thus, in this study we used three ANN training algorithms, viz. the standard BP, PSO-Trelea1 and PSO-Trelea2.

Each model was trained for a period of 2000 epochs. To reduce the likelihood of network overfitting, MacKay's regularization function *msereg* [23, 35, 38] was used as the performance criterion for each training algorithm. The three series are normalized in the range $[-1, 1]$, before fitting PSO based ANN models. The swarm size is kept in the range of 24 to 30, as it was observed that increasing the number of particles above this range does not significantly improve the PSO

performance [10, 37]. For each dataset, the number of particles was varied within this range and subsequent PSO performances were noted on the validation set before confirming on the exact swarm size. Three common error measures are used to compare forecasting ability of all the fitted models. These are the Mean Absolute Error (MAE), the Mean Squared Error (MSE) and the Mean Absolute Percentage Error (MAPE), which are defined as below [10, 23]:

$$\text{MAE} = \frac{1}{n}\sum_{t=1}^{n}|y_t - f_t| \quad .$$

$$\text{MSE} = \frac{1}{n}\sum_{t=1}^{n}(y_t - f_t)^2 \quad .$$

$$\text{MAPE} = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{y_t - f_t}{y_t}\right| \times 100 \quad .$$

In all the three above definitions, $y_t$ and $f_t$ are respectively the actual and forecasted values at time $t$ and $n$ is the size of the test set.

As each time the compiler randomly initializes the weights, positions, velocities etc. for a training algorithm, so the obtained best error measures alone cannot correctly reflect the forecasting performance of a fitted model. Keeping this fact in mind, every experiment is executed 50 times and the best as well as the average values for the error measures are recorded. Also, we used the term *Forecast Diagram* for the graph showing the test and forecasted observations. In each forecast diagram, solid and dotted line respectively represents the test and forecasted time series.

### 4.1 The Airline Passenger Time Series (APTS)

The airline passenger dataset represents the monthly total number of international airline passengers (in thousands) from January 1949 to December 1960. It is the well-known series G, as listed by Box and Jenkins [1]. This time series is affected by a multiplicative seasonality with an upward trend. SARIMA$(0, 1, 1) \times (0, 1, 1)^{12}$ was found to be the best stochastic model by Box and Jenkins for the airline passenger series. They made logarithmic transformation followed by a seasonal and non-seasonal differencing to make the series stationary. Faraway and Chatfield [31] made an extensive comparison by fitting thirteen different neural network structures to the airline data. Recently, Hamzacebi [23] empirically demonstrated that SANN can recognize the seasonal effect in the airline data, much better than the Box-Jenkins and traditional ANN models.

In this study, we used the observations for 1949 to 1958 for training, 1959 for validation and 1960 for testing. After validation experiments, we agreed on 2 hidden nodes and 25 swarm particles for the airline series. Each particle is of dimension 62. Following previous works [23, 31] we divided the observations by 100 for rescaling. The comparison of best and mean error measures obtained by the three training algorithms for APTS is presented in Table 1.

**Table 1.** Forecasting comparison for APTS

| Error Measures | | Standard BP | PSO-Trelea1 | PSO-Trelea2 |
|---|---|---|---|---|
| MSE $\times 10^{-4}$ | Best | 0.0329 | 0.0155 | 0.0242 |
| | Mean | 0.2420 | 0.2054 | 0.2310 |
| MAE $\times 10^{-2}$ | Best | 0.1547 | 0.1005 | 0.1244 |
| | Mean | 0.4407 | 0.3849 | 0.4006 |
| MAPE | Best | 3.261% | 2.253% | 2.635% |
| | Mean | 9.244% | 8.126% | 8.412% |

From Table 1, it can be seen that both types of PSO based training algorithms performed better than the standard BP. In particular with PSO-Trelea1 much less values of MSE, MAE and MAPE are obtained than the BP algorithm. Moreover, the mean values of the error measures indicate the throughout consistent performances of PSO-Trelea1 and PSO-Trelea2 in forecasting the APTS observations. The airline passenger time series and the PSO-Trelea1 forecast diagram for APTS are shown in Fig. 2(a) and Fig. 2(b) respectively.

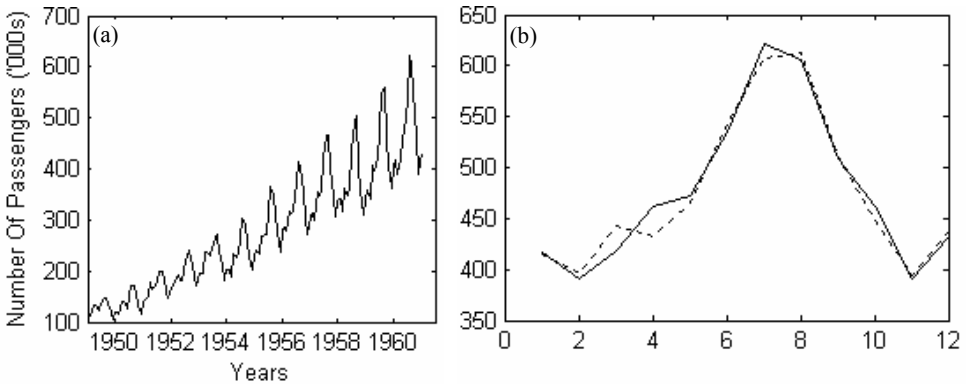


**Fig. 2.** (a) Airline passenger time series. (b) PSO-Trelea1 forecast diagram for APTS.

### 4.2 The quarterly sales time series (QSTS)

This dataset represents the quarterly export values of a French firm for 6 years and is taken from Makridakis et al. [40]. Following previous works [23, 40], we used the first 4 year data of QSTS for training, the fifth year data for validation and the sixth year data for testing. After validation processes, 2 hidden nodes and a swarm size of 27 was found appropriate for the dataset. Each swarm particle is of dimension 22. The original QSTS observations were divided by 100,

before applying the neural network models. The comparison of best and mean error measures obtained by the three network training algorithms for QSTS is presented in Table 2.

**Table 2.** Forecasting comparison for QSTS

| Error Measures | | Standard BP | PSO-Trelea1 | PSO-Trelea2 |
|---|---|---|---|---|
| MSE $\times 10^{-4}$ | Best | 0.0940 | 0.0827 | 0.0879 |
| | Mean | 0.2277 | 0.0992 | 0.1014 |
| MAE $\times 10^{-2}$ | Best | 0.2876 | 0.2754 | 0.2705 |
| | Mean | 0.3877 | 0.2993 | 0.3031 |
| MAPE | Best | 4.085% | 3.909% | 3.738% |
| | Mean | 5.270% | 4.269% | 4.318% |

From Table 2, it can be seen that the PSO based training algorithms outperformed the standard BP rule, in terms of both best and mean error measures. Moreover, it can be reasonably stated that training with PSO-Trelea1 and PSO-Trelea2 generated consistent forecasts for QSTS, as the obtained best error measures are not much deviated from their mean values. The original QSTS observations and the PSO-Trelea1 forecast diagram for QSTS are shown in Fig. 3(a) and Fig. 3(b) respectively.

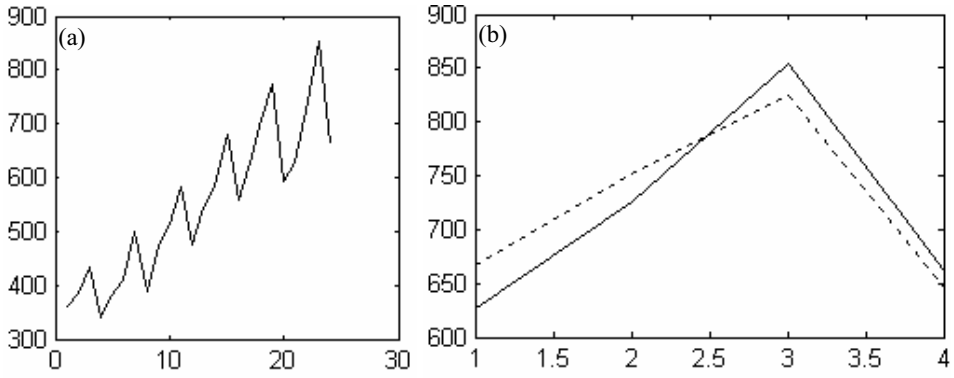


**Fig. 3.** (a) Quarterly sales time series. (b) PSO-Trelea1 forecast diagram for QSTS.

### 4.3 The quarterly USA beer production time series (QBPTS)

The data for quarterly beer production in USA from 1975 to 1982 is taken from William W. S. Wei [41]. This dataset shows strong trends and seasonal patterns.

In this study, we have used the data for 1975 to 1979 for neural network training, the data for 1980 for validation and the data for 1981 and 1982 for testing. After validation processes, we determined that 4 hidden nodes and 24 swarm particles are appropriate for this dataset. Each swarm particle is of dimension 40. A data preprocessing was performed in which the original observations were divided by 10. The comparison of best and mean error measures obtained by the three training algorithms for QBPTS is presented in Table 3.

**Table 3.** Forecasting comparison for QBPTS

| Error Measures | | Standard BP | PSO-Trelea1 | PSO-Trelea2 |
|---|---|---|---|---|
| MSE | Best | 1.8400 | 1.2436 | 1.2333 |
| | Mean | 3.0827 | 1.9079 | 2.1020 |
| MAE | Best | 1.1303 | 0.9893 | 0.9756 |
| | Mean | 1.3551 | 1.1971 | 1.2793 |
| MAPE | Best | 2.305% | 2.109% | 2.052% |
| | Mean | 2.703% | 2.536% | 2.686% |

Table 3 depicts that the best and mean forecast errors obtained by both PSO training algorithms are less than those obtained by the BP training rule. In particular, training with PSO-Trelea2 produced least error measures for this dataset. The QBPTS observations as well as the PSO-Trelea2 forecast diagram for QBPTS are shown in Fig. 4(a) and Fig. 4(b) respectively.

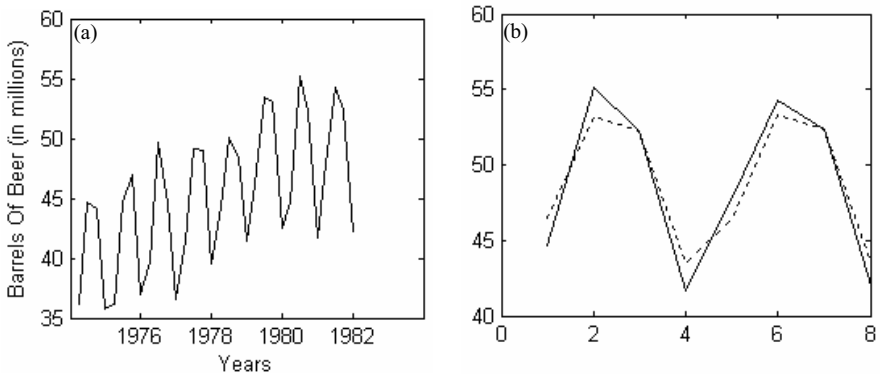


**Fig. 4.** (a) Quarterly USA beer production time series. (b) PSO-Trelea2 forecast diagram for QBPTS.

## 5 Conclusions

Particle swarm optimization is a population based evolutionary computation method with a high search power for the global optimal solution in the state space. During recent years, it has been efficiently used for training neural networks. However, not much research work has been done on time series forecasting by PSO trained neural networks.

In this paper, we applied two variants of the basic PSO algorithm, viz. Trelea1 and Trelea2 along with a standard BP (the RPROP) algorithm to train neural networks for seasonal time series forecasting. The SANN model, developed by Hamzacebi was used, as it has been shown that this model can efficiently track the seasonal effect in a time series without removing it. We implemented all the three training algorithms on three real world seasonal time series, one monthly and two quarterly. Each experiment was executed 50 times with different random initial values for weights, biases and swarm particles. The obtained best MSE, MAE and MAPE values showed that both the PSO based training algorithms produced much better forecasting results in comparison with the common BP technique. Moreover, the mean values of the error measures indicated throughout consistent performances of PSO-Trelea1 and PSO-Trelea2. Thus from this study, we can suggest that the ANN forecasting ability for seasonal time series can be significantly improved by using a suitable network structure (like SANN) together with one of the PSO training algorithms.

It is to be noted that all three time series used in our study are affected by strong seasonal pattern. So the forecasting ability of PSO based ANN for time series containing weak seasonality is yet to be examined. We shall try to explore this issue in future works.

**Acknowledgments.** The first author would like to thank Council of Scientific and Industrial Research (CSIR) for the obtained financial assistance, which helped a lot while performing this work.

## References

1. Box, G.E.P., Jenkins, G.M., Reinsel, G.C.: Time Series Analysis: Forecasting and Control. 3rd ed. Prentice Hall, New Jersey (1994)
2. Parrelli, R.: Introduction to ARCH & GARCH models. Optional TA Handouts. Econ 472 Department of Economics, University of Illinois (2001)
3. Engle, R.: Autoregressive conditional heteroskedasticity, with estimates of the variance of UK inflation. J. Econometrica 50 (1982)
4. Park, H.: Forecasting Three-Month Treasury Bills Using ARIMA and GARCH Models. Econ 930, Department of Economics, Kansas State University (1999)
5. Tong, H.: Threshold Models in Non-Linear Time Series Analysis. Springer-Verlag, New York (1983)
6. Zhang, G.P.: A neural network ensemble method with jittered training data for time series forecasting. Information Sciences 177, 5329–5346 (2007)

7. Zhang, G.P.: Time series forecasting using a hybrid ARIMA and neural network model. Neurocomputing 50, 159–175 (2003)
8. Zhang, G., Patuwo, B.E., Hu, M.Y.: Forecasting with artificial neural networks: The state of the art. J. Forecasting 14, 35–62(1998)
9. Adya, M., Collopy, F.: How effective are neural networks at forecasting and prediction? A review and evaluation. J. Forecasting 17, 481–495 (1998)
10. Jha, G.K., Thulasiraman, P.,Thulasiram, R.K.: PSO Based Neural Network for Time Series Forecasting. In: Proceedings of the IEEE International Joint Conference on Neural Networks, June 1419, pp. 1422–1427. Atlanta, Georgia, USA (2009)
11. Hagan, M., Menhaj, M.: Training feedforward networks with the marquardt algorithm. IEEE Transactions on Neural Networks 5(6), 989–993 (1994)
12. Reidmiller, M., Braun, H.: A direct adaptive method for faster backpropagation learning: The rprop algorithm. In: Proceedings of the IEEE Int. Conference on Neural Networks (ICNN), pp. 586–591. San Francisco (1993)
13. Moller, M.F.: A scaled conjugate gradient algorithm for fast supervised learning. J. Neural Networks. 6, 525–533(1993)
14. Battiti, R.: One step secant conjugate gradient. Neural Computation 4, 141–166 (1992).
15. Kennedy, J., Eberhart, R.C.: Particle Swarm Optimization. In: Proceedings of the IEEE International Conference on Neural Networks, IV, pp. 1942–1948. Piscataway, NJ (1995)
16. Kennedy, J., Eberhart, R.C., Shi, Y.: Swarm Intelligence. Morgan Kaufmann, San Francisco (2001)
17. Wilson, E.O.: Sociobiology: The new synthesis. Belknap Press, Cambridge (1975)
18. Leung, F.H.F., Lam, H.K., Ling, S.H., Tam, P.K.S.: Tuning of the structure and parameters of the neural network using an improved genetic algorithm. IEEE Transactions on Neural Networks. 14(1), 79–88 (2003)
19. Eberhart, R.C., Hu, X.: Human tremor analysis using particle swarm optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1927–1930. Washington, USA (1999)
20. Chen, Y., Abraham, A.: Hybrid Neurocomputing for Breast Cancer Detection. In: 4th IEEE International Workshop on Soft Computing as Transdisciplinary Science and Technology (WSTST05),pp. 884–892. Muroran, Japan (2005)
21. Paulo, S.G., Neto, de M., Gustavo G.P., Aranildo Rodrigues, L.J., Tiago, A.E.F.: Combining Artificial Neural Network and Particle Swarm System for Time Series Forecasting. In: Proceedings of the IEEE International Joint Conference on Neural Networks, June 1419, pp. 2230–2237. Atlanta, Georgia, USA (2009)
22. Chen, Y., Yanga, B., Dong, J.: Time-series prediction using a local linear wavelet neural network. Neurocomputing 69, 449–465 (2006)
23. Hamzacebi, C.: Improving artificial neural networks performance in seasonal time series forecasting. Information Sciences 178, 4550–4559 (2008)
24. Chatfield, C.: The Analysis of Time Series: An Introduction. Chapman Hall/CRC, Washington D.C. (1996)
25. Nelson, M., Hill, T., Remus, W., OConnor, M.: Can neural networks applied to time series forecasting learn seasonal patterns: an empirical investigation. In: Proceedings of the 27th IEEE Annual Hawaii International Conference on System Sciences, vol. 3, pp. 649–655. Maui, Hawaii (1994)
26. Hill, T., OConnor, M., Remus, W.: Neural networks models for time series forecasts. J. Management Sciences 42(7), 1082–1092 (1996)

27. Zhang, G., Qi, M.: Neural network forecasting for seasonal and trend time series. European Journal of Operational Research 160, 501–514 (2005)
28. Tang, Z., Almeida, C., Fishwick, P.A.: Time series forecasting using neural networks vs BoxJenkins methodology. J. Simulation 57(5), 303–310 (1991)
29. Franses, P.H., Draisma, G.: Recognizing changing seasonal patterns using artificial neural networks. J. Econometrics 81, 273–280 (1997)
30. Alon, I., Qi, M., Sadowski, R.J.: Forecasting aggregate retail sales: a comparison of artificial neural networks and traditional methods. J. Retailing and Consumer Services 8(3), 147–156 (2001)
31. Faraway, J., Chatfield, C.: Time series forecasting with neural networks: a comparative study using the airline data. J. Applied Statistics 47, 231–250(1998)
32. Kihoro, J.M., Otieno, R.O., Wafula, C.: Seasonal Time Series Forecasting: A Comparative Study of ARIMA and ANN Models. African Journal of Science and Technology (AJST) 5(2), 41–49 (2004)
33. White, H.: Learning in artificial neural networks: A statistical perspective. Neural Computation 1, 425-464 (1989)
34. Cybenko, G.: Approximation by superpositions of a sigmoidal function. Mathematics of Control Signals and Systems 2, 303-314 (1989)
35. Kamruzzaman, J., Begg, R., Sarker, R.: Artificial Neural Networks in Finance and Manufacturing. Idea Group Publishing (2006)
36. VandenBergh, F., Engelbrecht, A.P.: A cooperative approach to particle swarm optimization. IEEE Transaction on Evolutionary Computation 8(3), 225-239 (2004)
37. Trelea, I.: The particle swarm optimization algorithm: convergence analysis and parameter selection. Information Processing Letters 85, 317–325 (2003)
38. Demuth, M., Beale, M., Hagan, M.: Neural Network Toolbox User's Guide. Natic, MA: the MathWorks (2010)
39. Birge, B.: PSOt-A Particle Swarm Optimization Toolbox for use with Matlab. In: Proceedings of the IEEE Swarm Intelligence Symposium, pp. 182-186. Indianapolis, Indiana, USA (2003)
40. Makridakis, S., Wheelwright, S., Hyndman, R.J.: Forecasting Methods and Applications. John Wiley & Sons, New York (1998)
41. Wei, William, W.S.: Time Series Analysis: Univariate and Multivariate Methods, 2nd ed. Addison Wesley (2005)